



Topological Anonymous Walk Embedding: A New Structural Node Embedding Approach

Yuchen Yan*
yucheny5@illinois.edu
University of Illinois at
Urbana-Champaign
Urbana, IL, USA

Yongyi Hu*
huyongyi0903@sjtu.edu.cn
Shanghai Jiao Tong University
Minhang, Shanghai, China

Qinghai Zhou
qinghai2@illinois.edu
University of Illinois at
Urbana-Champaign
Urbana, IL, USA

Shurang Wu
wu_sr@mail.ustc.edu.cn
University of Science and Technology
of China
Hefei, Anhui, China

Dingsu Wang
dingsuw2@illinois.edu
University of Illinois at
Urbana-Champaign
Urbana, IL, USA

Hanghang Tong
htong@illinois.edu
University of Illinois at
Urbana-Champaign
Urbana, IL, USA

Abstract

Network embedding is a commonly used technique in graph mining and plays an important role in a variety of applications. Most network embedding works can be categorized into positional node embedding methods and target at capturing the proximity/relative position of node pairs. Recently, structural node embedding has attracted tremendous research interest, which is intended to perceive the local structural information of node, i.e., nodes can share similar local structures in different positions of graphs. Although numerous structural node embedding methods are designed to encode such structural information, most, if not all, of these methods cannot simultaneously achieve the following three desired properties: (1) bijective mapping between embedding and local structure of node; (2) inductive capability; and (3) good interpretability of node embedding. To address this challenge, in this paper, we propose a novel structural node embedding algorithm named topological anonymous walk embedding (TAW). Specifically, TAW creatively integrates *anonymous walk* and *breadth-first search (BFS)* to construct the bijective mapping between node embedding and local structure of node. In addition, TAW possesses inductive capability and good interpretability of node embedding. Experimental results on both synthetic and real-world datasets demonstrate the effectiveness of the proposed TAW algorithm in both structural node classification task and structural node clustering task.

CCS Concepts

• Information systems → Data mining.

Keywords

Structural node embedding, Network embedding, Bijective mapping

Both authors contributed equally to the paper.



This work is licensed under a Creative Commons Attribution International 4.0 License.

CIKM '24, October 21–25, 2024, Boise, ID, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0436-9/24/10
<https://doi.org/10.1145/3627673.3679565>

ACM Reference Format:

Yuchen Yan*, Yongyi Hu*, Qinghai Zhou, Shurang Wu, Dingsu Wang, and Hanghang Tong. 2024. Topological Anonymous Walk Embedding: A New Structural Node Embedding Approach. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24)*, October 21–25, 2024, Boise, ID, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3627673.3679565>

1 Introduction

In the era of big data and AI [3, 4, 26, 42, 67, 70], network¹ embedding [6, 25, 32, 62, 63] aims to map each node to a real-valued vector such that information in the graph could be preserved, which plays an important role in many real-world applications such as node classification [33, 58, 60, 61], social network alignment [13, 64–66, 69, 71], knowledge graph reasoning [54, 55] and traffic flow prediction [22, 53]. Most, if not all, of the existing network embedding methods [7, 11, 14, 15, 39, 40, 46, 47] can be categorized into positional node embedding methods, which target at capturing relative positional information/proximity score of node pairs. This direction of methods forces nodes that are directly connected [47] or sampled in the same random walk [15, 39] to obtain similar embeddings in the latent space.

Recently, a different category of network embedding methods, i.e. structural node embedding method has attracted tremendous research interest. Structural node embedding focuses on preserving structural similarity, which makes nodes with similar neighborhood degree distributions or k -hop neighborhood topologies, to possess similar embeddings. We illustrate the difference between positional node embedding methods and structural node embedding methods in Figure 1. In positional node embedding methods, node v_1 and node v_3 are close to each other, i.e. with similar positions in the graph, embeddings of them will be optimized to be similar. On the contrary, node v_1 and node v_2 are far away from each other but their local structures are similar, which leads to analogous embeddings in structural node embedding methods. Numerous structural node embedding methods have been proposed to perceive the local structural information of node. To name a few, Struc2vec [41] and xNetMF [18] use degree distribution to measure the structural similarity between nodes and generate node embeddings via matrix

¹In this paper, we use the term network and graph interchangeably.

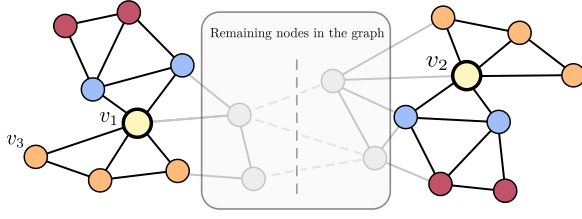


Figure 1: Illustration of positional node embedding methods and structural node embedding methods.

factorization or the skip-gram framework [35]. DRNE [49] utilizes degree distribution by ordering the neighbors of a node according to their degrees and feeding them into a layer-normalized long short-term memory (LSTM) model [21]. To grasp more complex structural features, HONE [43] and MCN [31] select specific motifs and conduct motif counting. In addition, graph kernel [17] can function as a measure of structural similarity in GraphWave [12] and SEGK [37].

Despite the remarkable progress achieved by structural node embedding, most, if not all, of existing structural node embedding methods are faced with the challenge that the following three desired properties cannot be fulfilled simultaneously: (1) bijective mapping between embedding and local structure of node; (2) inductive capability; and (3) good interpretability of node embedding. First, bijective mapping between node embedding and local structure of node means that nodes with same local structures can earn identical node embeddings and identical node embeddings refers to the fact that these two nodes own same local structures. This is not satisfied in existing methods [18, 31, 41], where identical node embeddings do not necessarily mean two nodes own same local structures. As the toy example shown in Figure 2(a), all nodes share same degree. Simply relying on the degree distribution leads to identical embeddings for all nodes, which can not discriminate the true structural equivalence in Figure 2(b). Second, the inductive capability requires that the designed structural node embedding algorithm can efficiently learn the embeddings of unseen/newly arrived nodes without re-training embeddings of all nodes from scratch. However, most existing structural node embedding [12, 43, 49] methods are transductive, which follow the assumption that the topology of the graph is fixed and can not be generalized to unseen/newly arrived nodes in inductive setting. Third, good interpretability of node embedding means that the meaning of each dimension in the node embedding should be clear and can be explained directly. In existing methods [12, 37, 41, 44], node embeddings are optimized in the multi-dimensional latent space and the meaning of each dimension of the latent space is unclear.

In this paper, to address the above challenge, we propose a simple yet novel structural node embedding algorithm named topological anonymous walk embedding (TAW), which creatively integrates anonymous walk [23] and breadth-first search (BFS) [9]. Specifically, anonymous walk (AW) removes the node’s identifier during random walk in order to preserve structural features while discarding positional information. However, anonymous walk has some drawbacks in capturing local structure of node. For instance, the anonymous walk “ $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ ” only indicates that the

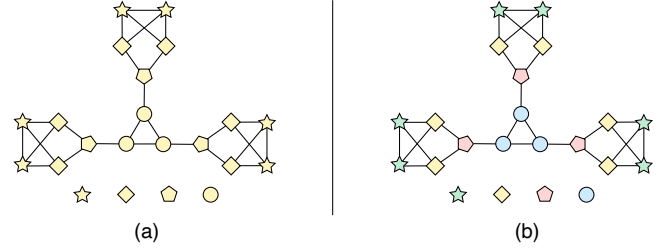


Figure 2: Limitation of degree-based embedding methods.

walk passes through four distinct nodes, without providing any other structural information. In addition, anonymous walk needs to sample a lot of random walks, which bears high uncertainty in the sampling process. In this case, different structures of nodes may contribute to same set of anonymous walks (e.g., “ $1 \rightarrow 2 \rightarrow 3 \rightarrow 4$ ”), which fails to satisfy bijective mapping between node embedding and local structure of node. To improve the capability of anonymous walk to capture local structure of node, we first propose topological anonymous walk (TAW). Each TAW is a two dimension sequence (e.g., “ $(1, 0) \rightarrow (2, 1) \rightarrow (3, 2) \rightarrow (4, 3)$ ”) and encodes two aspects of information: (1) anonymous walk; and (2) shortest path distance. Then, for a k -hop subgraph starting from the root node, we collect all TAWs via BFS to build a group of basic TAWs. We prove that the group of basic TAWs together with TAWs’ frequencies (e.g., “ $(1, 0) \rightarrow (2, 1) \rightarrow (3, 2) \rightarrow (4, 3) : 2$ ”) has a *bijective mapping* to the structure of a k -hop subgraph (*bijective mapping*). Based on the group of basic TAWs, TAW generates the structural node embedding via selecting frequency of each basic TAW as the entry in each dimension of the embedding (*interpretability*). Furthermore, we demonstrate that TAW can be easily generated for unseen node with the help of the group of basic TAWs. At last, the effectiveness of the proposed TAW is corroborated through extensive experiments on both synthetic and real-world datasets in the following two tasks: (1) structural node classification; and (2) structural node clustering.

The contributions of this work as follows:

- **Problem.** To the best of our knowledge, we are the first to study how to fulfill all three desired properties for structural node embedding problem simultaneously: (1) bijective mapping between embedding and local structure of node; (2) inductive capability; and (3) good interpretability of node embedding.
- **Model.** We propose a simple yet novel structural node embedding algorithm TAW, which integrates the anonymous walk and breadth-first search together to propose the new concept named topological anonymous walk.
- **Experiments.** Extensive experiments on both synthetic and real-world datasets demonstrate the effectiveness of TAW in (1) structural node classification task and (2) structural node clustering task.

2 Problem Definition

In this section, we will introduce some preliminary knowledge of structural equivalence and the definition of structural node embedding problem. We first summarize the main symbols used in this

paper. We adopt bold uppercase letters for matrices (e.g., $\mathbf{A}$), bold lowercase letters for vectors (e.g., $\mathbf{v}$), calligraphic letters for sets (e.g., $\mathcal{A}$) and lowercase letters for scalars (e.g., a). In addition, we follow the convention to represent the u -th row of matrix $\mathbf{A}$ as $\mathbf{A}(u, :)$, the v -th column as $\mathbf{A}(:, v)$ and the (u, v) -th entry as $\mathbf{A}(u, v)$. Next we present the definitions of graph isomorphism, node set-induced subgraph and k -hop local topological equivalence, followed by the formal definition of structural node embedding problem.

For two graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$, where $\mathcal{V}_1$ and $\mathcal{V}_2$ are node sets and $\mathcal{E}_1$ and $\mathcal{E}_2$ are edge sets, graph isomorphism can be defined as follows:

DEFINITION 1 (GRAPH ISOMORPHISM). *Two graphs $G_1 = (\mathcal{V}_1, \mathcal{E}_1)$ and $G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ are isomorphic if there exists a bijective mapping m between node sets $\mathcal{V}_1$ and $\mathcal{V}_2$, such that:*

$$\forall u, v \in \mathcal{V}_1 \quad (u, v) \in \mathcal{E}_1 \iff (m(u), m(v)) \in \mathcal{E}_2$$

Usually, structural node embedding algorithms pay more attention to local topological equivalence (i.e., subgraph isomorphism) rather than the isomorphism problem for the whole graph. The node set-induced subgraph can be defined as:

DEFINITION 2 (NODE SET-INDUCED SUBGRAPH [10]). *Given a graph $G = (\mathcal{V}, \mathcal{E})$, let $\mathcal{S} \subset \mathcal{V}$ be a subset of nodes of G . Then the $\mathcal{S}$ -induced subgraph $G[\mathcal{S}]$ is the graph whose node set is $\mathcal{S}$ and whose edge set consists of all of the edges in $\mathcal{E}$ that have both endpoints in $\mathcal{S}$.*

With the definition of node set-induced subgraph, given a root node v in G , the k -hop subgraph rooted in v can be denoted as $G[\mathcal{N}_k(v)]$, where $\mathcal{N}_k(v)$ is the node set within k -hops of v . For any two nodes v_i and v_j , the k -hop local topological equivalence (k -LTE) can be defined as:

DEFINITION 3 (k -HOP LOCAL TOPOLOGICAL EQUIVALENCE). *Given a graph $G = (\mathcal{V}, \mathcal{E})$, two nodes $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$ are k -hop local topological equivalent if there is a bijective mapping m between $\mathcal{N}_k(v_i)$ and $\mathcal{N}_k(v_j)$ such that $m(v_i) = v_j$ and $G[\mathcal{N}_k(v_i)]$ is isomorphic to $G[\mathcal{N}_k(v_j)]$ under the mapping $m(\cdot)$, where $\mathcal{N}_k(v_i)$ denotes the node set within k -hop scale of node v_i .*

Based on the definitions of graph isomorphism, node set-induced subgraph and k -hop local topological equivalence, we formally define the structural node embedding problem:

PROBLEM 1. Structural Node Embedding.

Given: (1) an undirected graph $G = (\mathcal{V}, \mathcal{E})^2$, (2) the hop number k .

Output: an embedding matrix $\mathbf{F} \in \mathbb{R}^{|\mathcal{V}| \times d}$ such that for any node pair $v_i \in \mathcal{V}$ and $v_j \in \mathcal{V}$:

(v_i, v_j) are k -hop local topological equivalent $\iff \mathbf{F}(v_i, :) = \mathbf{F}(v_j, :)$, where d is the dimension of node embeddings.

In real-world applications, the hop k can be chosen as an appropriate number according to down-stream tasks.

²Without loss of generality, we do not allow self loops and duplicated edges in G throughout the whole paper.

3 Model

In this section, we present our proposed topological anonymous walk embedding (TAWE) model, which creatively integrates anonymous walk [23] and breadth-first search (BFS) [9] together to fulfill all three desired properties: (1) bijective mapping between node embedding and local structure of node; (2) inductive capability; and (3) good interpretability of node embedding. Specifically, we first introduce anonymous walk [23] and conduct an analysis on its strengths and drawbacks. We identify that anonymous walk suffers from repeatedly sampling random walk, which leads to its failure of satisfying the bijective mapping property. This motivates us to integrate *additional topological information* into the anonymous walk and propose the general definition of *topological anonymous walk* (TAW). (Subsection 3.1) Then, we adopt the breadth-first search (BFS) [9] to obtain a simple yet effective topological information: shortest path distance (i.e., hop number). By integrating the hop number in anonymous walk, we propose the topological anonymous walk used in TAWE and introduce the details of TAWE. Concretely, we construct a group of basic TAWs and prove that the TAW group together with each basic TAW's frequency possesses a bijective mapping with the k -hop structure of node (i.e., the bijective mapping property). Accordingly, the normalized frequency of each basic TAW naturally serves as the entry in each dimension of a structural node embedding. (Subsection 3.2). At last, we also propose a pruning algorithm (TAWE-P) for TAWE and have a brief discussion about (1) the inductive capability of TAWE/TAWE-P and the interpretability of node embedding in TAWE/TAWE-P and (2) how to generalize TAWE/TAWE-P to heterogeneous graphs [11, 14] and attributed graphs [16, 28, 51].

3.1 Topological Anonymous Walk

In this subsection, we analyze the strength and drawbacks of anonymous walk [23], which encourages us to propose the general form of *topological anonymous walk* (TAW) with additional topological information. Furthermore, we show that TAW can reconstruct the corresponding anonymous walk and enjoys at least same capability of capturing local structure of node as anonymous walk.

In anonymous walk, random walk is the key component, which is a sequence of nodes sampled according to certain probability distribution defined on the neighbors of the latest sampled node. Different from the random walk utilized in positional node embedding methods [15, 39], anonymous walk discards the original node identity information to seize the local structural information of node. To be more specific, anonymous walk embraces *position function* to describe the node sequence and removes the original node identity. The *position function* can be defined as follows:

DEFINITION 4 (POSITIONAL FUNCTION). *Given a sequence $q = \{v_i\}_{i=1}^l$ (i.e., $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_l$), where l is the length of the sequence. The position function for u appearing in this sequence is defined as $p_q(u) = \{i \mid v_i = u\}$.*

Position function and the sequence are two different but equivalent ways to present the same mathematical object. For example, given a sequence $q = a \rightarrow b \rightarrow c \rightarrow a \rightarrow c$, the position functions for a , b and c in q are as the following: $p_q(a) = \{1, 4\}$, $p_q(b) = \{2\}$, $p_q(c) = \{3, 5\}$.

With the help of position function, the anonymous walk can be defined as:

DEFINITION 5 (ANONYMOUS WALK (AW)). Given a random walk $q = \{v_i\}_{i=1}^l$ in graph $G = (\mathcal{V}, \mathcal{E})$, then its corresponding anonymous walk is a sequence of positive integers $aw = \{a_i\}_{i=1}^l$, where $a_i = \min p_q(v_i)$ is the first position of node v_i in the random walk q .

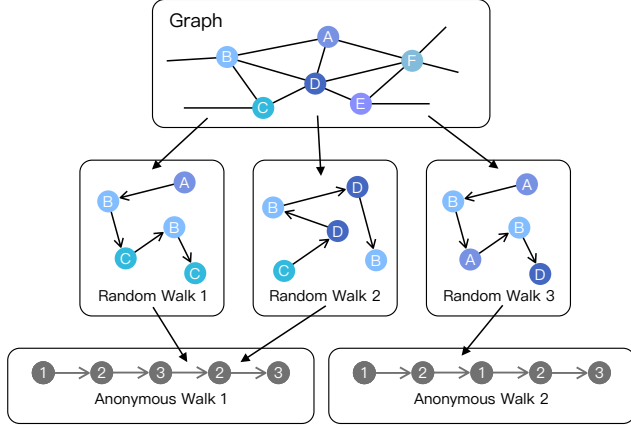


Figure 3: Illustration of anonymous walk with examples.

As shown in Figure 3, three random walks are sampled from the graph. Random walk 1 ($A \rightarrow B \rightarrow C \rightarrow B \rightarrow C$) and random walk 2 ($C \rightarrow D \rightarrow B \rightarrow D \rightarrow B$) share the same anonymous walk $1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3$, where same node may own different a_i s in different random walks. For example, node B 's first position in random walk 1 is 2 and its first position in random walk 2 is 3. From this example, the strength of anonymous walk can be illustrated: nodes' original identities are removed and only the structural pattern of random walk will be kept.

However, in [23], a large number of anonymous walks are sampled repeatedly to approximate the local structure of the root (i.e., starting) node. The repeated sampling process causes that same random walk may be sampled more than one time. In addition, different local structures may generate same anonymous walks. As shown in Figure 4, the blue part and the green part possess different structures but same anonymous walks ($1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 3$) are generated. These two drawbacks lead to the failure of satisfying the bijective mapping property between embedding and local structure of node.

To this end, we identify that additional topological information should be integrated in anonymous walk to improve its capability to capture local structure of node. Thus, we propose the *topological anonymous walk* (TAW) and its general definition is as follows:

DEFINITION 6 (TOPOLOGICAL ANONYMOUS WALK (TAW)). Given a specific node $u \in G$ and the k -hop subgraph rooted in u denoted as $G[\mathcal{N}_k(u)]$, a random walk $q = \{v_i\}_{i=1}^l$ on $G[\mathcal{N}_k(u)]$, where l is the length of the random walk, the topological anonymous walk can be defined as $taw = \{(a_i, f(G[\mathcal{N}_k(u)], v_i))\}_{i=1}^l$, where $a_i = \min p_q(v_i)$ is the first position of node v_i in the random walk q and $f(G[\mathcal{N}_k(u)], v_i)$ can be any topological information defined on $G[\mathcal{N}_k(u)]$ for node v_i .

Algorithm 1 Construction of the group of basic TAWs

Input: (1) Graph $G = (\mathcal{V}, \mathcal{E})$; (2) maximal hop k ; and (3) longest length for TAW L .
Output: $\mathcal{B}_u$ s for all nodes u in G .
for node $u \in \mathcal{V}$ **do** ▷ Parallel Computing
 Build $G[\mathcal{N}_k(u)]$ via BFS
 Compute $hop(v_i, u)$ for any v_i in $G[\mathcal{N}_k(u)]$ via BFS
 Initialize $\mathcal{B}_u$ as empty
 Initialize random walk queue Q with u
 while Q is not empty **do**
 Pop random walk q from Q
 Compute the TAW taw for q according to Eq.(1)
 if taw not in $\mathcal{B}_u$ **then**
 Add taw : 1 into $\mathcal{B}_u$
 else
 Increase $C_u(taw)$ by 1 in $\mathcal{B}_u$
 end if
 if the length of q equal to L **then**
 Continue
 else
 Pick the last node v_{last} in q
 for $v \in G[\mathcal{N}_k(u)]$ and $(v_{last}, v) \in \mathcal{E}$ **do**
 Add random walk $q_{new} = q + v$ into Q
 end for
 end if
 end while
end for
return $\mathcal{B}_u$ s for all nodes u s in G .

In Definition 6, the topological information $f(\cdot, \cdot)$ can be defined arbitrarily such as the original node identity ($f(G[\mathcal{N}_k(u)], v_i) = v_i$) or a_i defined in Definition 5 ($f(G[\mathcal{N}_k(u)], v_i) = a_i$). Actually, for any choice of the topological information $f(\cdot, \cdot)$, we have the following proposition:

PROPOSITION 1. For any choice of $f(\cdot, \cdot)$, the set of topological anonymous walks can reconstruct the **corresponding set** of anonymous walks.

From Proposition 1, since the set of anonymous walks can be reconstructed from the set of topological anonymous walks, TAW with any choice of $f(\cdot, \cdot)$ owns at least the same capability of capturing local structure of node as AW. Then, one question naturally arises: How to select an appropriate $f(\cdot, \cdot)$ to (1) avoid repeatedly sampling in AW; and (2) make the mapping between local structure of node and the final node embedding bijective?

3.2 TAW E

In this subsection, we present the details of topological anonymous walk embedding (TAW E), which chooses the hop number (i.e., shortest path distance from the root node u) as the additional topological information (i.e., $f(G[\mathcal{N}_k(u)], v_i) = hop(u, v_i)$) to solve the question raised in the end of Subsection 3.1. Specifically, we adopt breadth-first search (BFS) [9] algorithm to compute the hop number for each node v_i in $G[\mathcal{N}_k(u)]$ from the root node u . Then, a group of basic TAWs is built for $G[\mathcal{N}_k(u)]$ and we prove that the TAW group together with each TAW's frequency possesses

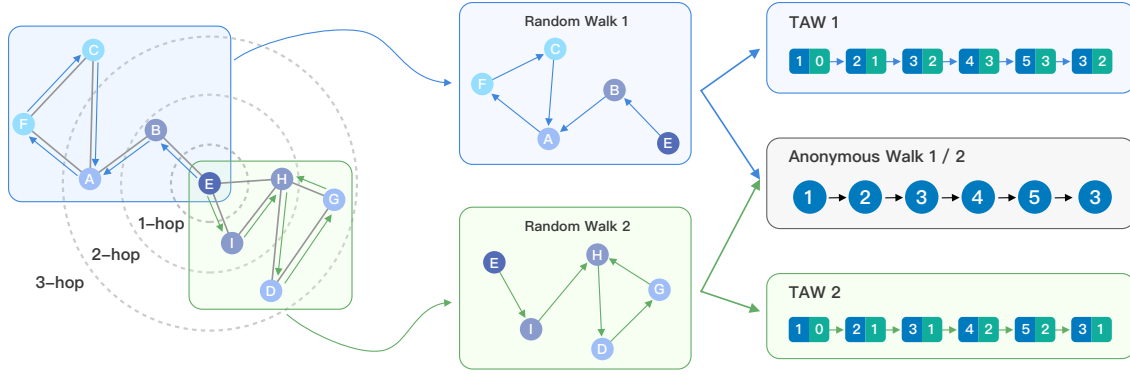


Figure 4: Illustration of topological anonymous walk.

a bijective mapping with the k -hop structure of the root node u . Finally, we describe how the normalized frequency of each basic TAW naturally forms each dimension of node embedding of u .

The first step of TAW is to extract $G[\mathcal{N}_k(u)]$ for the root node u and calculate the hop number $\text{hop}(v_i, u)$ for each node v_i in $G[\mathcal{N}_k(u)]$ from u , which can be easily obtained via BFS³. With the hop number $\text{hop}(v_i, u)$, the topological anonymous walk for a random walk $q = \{v_i\}_{i=1}^l$ in TAW can be built as:

$$\text{taw} = \{(a_i, \text{hop}(v_i, u))\}_{i=1}^l, \quad (1)$$

where $a_i = \min p_q(v_i)$ is defined in Definition 5. We can observe that TAW has two values for each node v_i in the random walk: (1) the first position a_i inherited from anonymous walk and (2) the hop number for v_i from the root node u . One example of TAW can be found in Figure 4, where random walk 1 and random walk 2 have different TAWs to discriminate the topologies in the blue part and the green part.

The second step of TAW is to construct a group of basic TAWs together with the frequency of each TAW, which is the core component of TAW. Given the root node u and a maximal length L of TAW, we denote the target of this step as $\mathcal{B}_u = \{\text{taw}_1 : C_u(\text{taw}_1), \dots, \text{taw}_l : C_u(\text{taw}_l), \dots\}$, where $C_u(\text{taw}_i)$ is the frequency of a basic TAW taw_i starting from the root node u with the length l no larger than L in $G[\mathcal{N}_k(u)]$. $\mathcal{B}_u$ can be regarded as a *dictionary*, where the indexes are TAWs and the values are TAWs' frequencies. The process of constructing $\mathcal{B}_u$ is shown in Algorithm 1. One thing worth noticing is that $\mathcal{B}_u$ for each node u in G can be built in a parallel way to save time in implementation. Regarding the details of Algorithm 1, we utilize a *first-in-first-out queue* (Q) to store all random walks starting from the root node u . After computing the hop numbers for any v_i in $G[\mathcal{N}_k(u)]$, Q is initialized with the root node u , which is also a random walk with length $l = 1$. Then, while Q is not empty, we pop the first random walk q out and build its corresponding TAW according to Eq. (1). The TAW is added into/counted in the TAW base $\mathcal{B}_u$. If the random walk q already has a length $l = L$, we continue popping random walks from

Q . Otherwise, we follow BFS to consider all nodes (vs) connected to the last node of the random walk q in $G[\mathcal{N}_k(u)]$. We place each v at the end of q to form a new random walk q_{new} and add q_{new} in Q . In this way, $\mathcal{B}_u$ for the root node u can be constructed after Q is empty.

In fact, integrating BFS⁴ in TAW solves the question raised at the end of Subsection 3.1. First, BFS can avoid sampling one random walk repeatedly with the help of Q . Second, we identify that selecting the hop number as additional topological information $f(\cdot, \cdot)$ enables the TAW base $\mathcal{B}_u$ to enjoy the desired *bijective mapping* property with the local structure rooted in u (i.e., $G[\mathcal{N}_k(u)]$) and have the following proposition:

PROPOSITION 2 (BIJECTIVE MAPPING PROPERTY). *The mapping between $\mathcal{B}_u$ and $G[\mathcal{N}_k(u)]$ is bijective, when the maximal $L \rightarrow \infty$ ⁵.*

PROOF. See Appendix. $\square$

Finally, we describe how to use the constructed TAW group $\mathcal{B}_u$ to generate structural node embedding in TAW. After constructing $\mathcal{B}_u$ s for all nodes in graph G , we merge different TAWs in all $\mathcal{B}_u$ s to a merged group $\mathcal{B} = \{\text{taw}_1, \dots, \text{taw}_i, \dots\}$. Given $\mathcal{B}$, we can generate a structural node embedding with a $|\mathcal{B}|$ dimension. The i -th dimension of the structural node embedding for node u is the normalized frequency of taw_i in $\mathcal{B}_u$. Concretely, for the i -th basic TAW taw_i in $\mathcal{B}$, we can obtain the mean μ_{taw_i} and the standard deviation σ_{taw_i} of taw_i in $[\mathcal{B}_{u_1}[\text{taw}_i], \dots, \mathcal{B}_{u_{|V|}}[\text{taw}_i]]$. The i -th dimension for the structural embedding of node u is as follows:

$$F(u, i) = \frac{(\mathcal{B}_u[\text{taw}_i] - \mu_{\text{taw}_i})}{\sigma_{\text{taw}_i}}. \quad (2)$$

3.3 TAW-P and Discussion

In this subsection, we further propose a *pruning* algorithm of TAW named TAW-P and have a brief discussion about (1) the inductive capability of TAW/TAW-P and the interpretability of node embedding in TAW/TAW-P and (2) the generalization of TAW/TAW-P to heterogeneous graphs and attributed graphs.

³Actually, there is no need to extract $G[\mathcal{N}_k(u)]$ and compute hop numbers explicitly in the first step, both of which can be conducted during the process of building the group of basic TAWs in implementation. We make them the first step here to better present the TAW algorithm for the purpose of readability.

⁴Notice that BFS in TAW during the process of constructing $\mathcal{B}_u$ allows visiting one node multiple times to collect different random walks/TAWs.

⁵Actually, L does not need to be ∞ for real-world graphs. Usually, $L = 2k$ or $L = 3k$ is enough in our experiments for real-world datasets.

In TAWE, the longest length L can be used to limit the collections of TAWs. However, a small L will do damage to TAWE in that short TAWs can not capture sophisticated structures. Thus, we propose a pruning algorithm TAWE-P, whose core part is to limit the expansion of BFS. We have added two pruning hyperparameters to control the expansion of BFS: (1) the maximal branch N_{max} ; and (2) the maximal times of visit T_{max} . We consider the neighbor nodes v of the last node v_{last} in the random walk q : if node v_{last} has too many neighbors, TAWE-P only allows a maximal N_{max} different neighbor nodes to be added after v_{last} to build a new random walk q_{new} . Furthermore, TAWE-P embraces the assumption that if the neighbor node v has been visited too many times during the process of constructing $\mathcal{B}_u$, its topological information has been well captured. So, TAWE-P sets a maximal times of visit T_{max} and records the times of visit of each node in $G[\mathcal{N}_k(u)]$. If the time of visit of the neighbor node v (i.e., T_v) exceeds T_{max} , it will not be added after v_{last} in random walk q to build q_{new} . Via such pruning strategy, TAWE-P can save much time compared to TAWE in real-world graphs.

Discussion. At last, we discuss about (1) the inductive capability of TAWE/TAWE-P and the interpretability of node embedding in TAWE/TAWE-P and (2) the generalization of TAWE/TAWE-P to heterogeneous graphs and attributed graphs. For the inductive capability, it is unnecessary to repeat the entire embedding process of TAWE/TAWE-P. Taking the structural node classification as an example, after obtaining $\mathcal{B}_u$ for each node u in the original graph G and the merged $\mathcal{B}$, the inductive learning pipeline for an unseen/new node v_{new} has two steps: (1) build $\mathcal{B}_{v_{new}}$ and update $\mathcal{B}_{v_i}$ s for other nodes in $G[\mathcal{N}_k(v_{new})]$; and (2) use the original merged $\mathcal{B}$ to construct $F(v_{new}, :)$ ⁶ and apply the original trained classifier on $F(v_{new}, :)$. We have experimental results on the inductive capability in Subsection 4.4. For the interpretability of node embedding, based on Eq. (2), we can see that the i -th dimension of the structural node embedding ($F(u : i)$) corresponds to the normalized frequency of $taw_i \in \mathcal{B}$ in $G[\mathcal{N}_k(u)]$. If $F(u : i)$ is large, it means the k -hop subgraph rooted in u possesses more taw_i s compared to k -hop subgraphs rooted in other nodes of G . For the generalization of TAWE/TAWE-P, since the definition of additional topological information $f(\cdot, \cdot)$ is general, rather than the hop number ($hop(v_i, u)$), we can design special $f(\cdot, \cdot)$ s for different types of graphs. To name a few, we can integrate metapath [11, 14] into $f(\cdot, \cdot)$ for heterogeneous graphs and add node attributes into $f(\cdot, \cdot)$ for attributed graphs [16, 28, 51]. This means that our proposed TAWE/TAWE-P can be easily generalized to other types of graphs.

4 Experiment

In this section, we evaluate TAWE/TAWE-P on both synthetic datasets and real-world datasets in the following two tasks: (1) structural node classification; and (2) structural node clustering. We also conduct experiments to validate the inductive capability of TAWE.

⁶Usually, for real-world graphs, if $|\mathcal{B}|$ is too large, we can generate a subset of $\mathcal{B}$ to replace it.

4.1 Experimental Setup

Datasets. We use 7 synthetic graphs to evaluate TAWE and 3 real-world datasets to evaluate TAWE-P. All 7 synthetic datasets are generated in [24]. The information of 7 synthetic graphs are presented in Table 1 and the 5 base graphs they are generated from are shown in Figure 5. We also use 2 real-world air-traffic networks Brazil and Europe same as Struct2vec [41] for structural node classification and the same Enron dataset [29] as GraphWave [12] for structural node clustering task. For node classification, we randomly split every dataset into 33/33/34% for training, validation, and testing. Brazil has 131 nodes, 1,038 edges and 4 classes of nodes labelled in [24]. Europe has 399 nodes, 5,995 edges and 4 classes of nodes. Enron has 184 nodes, 3,089 edges and 8 different clusters.

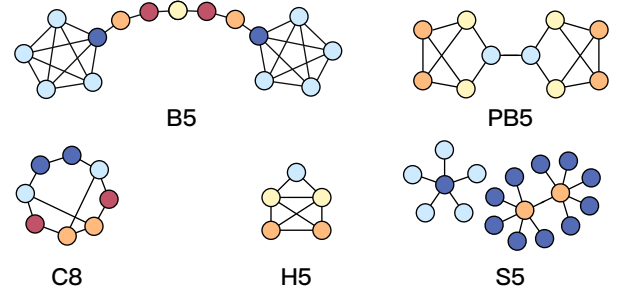


Figure 5: Illustration of base graphs in synthetic datasets.

Metrics. For structural node classification, we adopt accuracy and F1-macro as metrics. For structural node clustering, we choose normalized mutual information (NMI) and purity as metrics.

Baselines. We compare TAWE/TAWE-P with 5 baselines: Node2vec [15] is a classical positional embedding method and the remaining 4 are representative or latest structural node embedding algorithms: xNetMF [18], GraphWave [12], Role2vec [1] and SEGK [37].

Parameter Settings. The dimensions of structural node embeddings in all baselines and TAWE/TAWE-P are presented in Table 2. The dimensions in all baselines are 16 for synthetic graphs and 128 for real-world graphs. Since the dimension of embeddings generated in TAWE/TAWE-P is equal to the size of the merged TAW group (i.e., $|\mathcal{B}|$), the dimension of embeddings varies across different datasets. One thing we want to emphasize is that the dimensions in TAWE/TAWE-P on all datasets are consistently smaller than those of baselines. For example, the dimension of embeddings in TAWE on CoS is 3, which is smaller than 16 in all baselines. This means that if TAWE/TAWE-P can achieve better performances than baselines, the capability of TAWE/TAWE-P to capture local structure of node is much stronger than baselines given the smaller dimension of node embeddings in TAWE/TAWE-P. All other hyperparameters of baselines are set as default in their papers. We evaluate TAWE on synthetic datasets and run TAWE-P on real-world datasets. The hyperparameters are as the following: (1) the maximal hop $k = 3$ and the longest length for TAW $L = 4$ in TAWE; and (2) in TAWE-P, $k = 3$, $L = 9$, the maximal times of visit $T_{max} = 100$ and $N_{max} = 10$.

Table 1: Information of synthetic graphs.

Name	# Node	# Edge	Base Graph	Generation Descriptions from [24]
B-A	90	150	B5	Connecting the out-most nodes on the chain of B5 into a circle.
B-B	140	260	B5	Connecting the out-most nodes on the chain of B5 into a circle. Additional 5-clique at each connector.
FW	80	100	C8	Enlarged version of C8 with similar perturbation.
H-S	80	120	H5	10 H5 on a circle with 2 circular nodes between each connecting circular node with house's side.
H-T	80	120	H5	10 H5 on a circle with 2 circular nodes between each connecting circular node with house's roof.
PB-L	60	90	PB5	10 half-sided PB5 connected to each node of a 10-node circular graph. All the node degrees are 3.
CoS	170	155	S5	10 normal stars and 5 binary stars as in S5.

Table 2: The dimensions of structural node embeddings in baselines and TAWE/TAWE-P.

Datasets	B-A	B-B	FW	H-S	H-T	PB-L	CoS	Brazil	Europe	Enron
Baselines	16	16	16	16	16	16	16	128	128	128
TAWE/TAWE-P	6	6	4	7	7	7	3	91	127	70

Table 3: Accuracy ($\pm std$) of node classification on synthetic dataset(%).

Methods	B-A	B-B	FW	H-S	H-T	PB-L	CoS
Node2vec	37.8 $\pm$ 9.8	50.1 $\pm$ 8.1	14.4 $\pm$ 6.3	18.3 $\pm$ 8.1	24.0 $\pm$ 8.9	20.7 $\pm$ 9.1	84.8 $\pm$ 5.3
xNetMF	81.0 $\pm$ 9.8	72.3 $\pm$ 6.8	89.7 $\pm$ 15.5	72.8 $\pm$ 13.7	81.4 $\pm$ 10.4	32.5 $\pm$ 13.8	93.3 $\pm$ 4.0
GraphWave	44.5 $\pm$ 7.4	56.7 $\pm$ 5.2	27.4 $\pm$ 13.6	28.2 $\pm$ 13.2	29.1 $\pm$ 15.1	30.8 $\pm$ 9.9	59.3 $\pm$ 6.7
Role2vec	82.5 $\pm$ 11.1	81.4 $\pm$ 7.3	56.9 $\pm$ 12.1	61.7 $\pm$ 10.4	65.0 $\pm$ 12.8	33.0 $\pm$ 10.9	89.6 $\pm$ 4.0
SEGK	<u>95.9$\pm$8.5</u>	<u>94.7$\pm$6.8</u>	<u>97.9$\pm$8.5</u>	<u>89.7$\pm$11.4</u>	<u>94.1$\pm$8.4</u>	<u>91.3$\pm$11.7</u>	99.5$\pm$2.1
TAWE	98.5$\pm$4.6	99.0$\pm$3.2	99.6$\pm$4.1	96.1$\pm$7.7	99.0$\pm$4.6	99.4$\pm$4.6	<u>99.3$\pm$2.4</u>

Table 4: F1-macro ($\pm std$) of node classification on synthetic dataset(%).

Methods	B-A	B-B	FW	H-S	H-T	PB-L	CoS
Node2vec	17.9 $\pm$ 5.3	18.0 $\pm$ 3.6	12.3 $\pm$ 5.7	10.4 $\pm$ 4.0	17.5 $\pm$ 6.4	15.9 $\pm$ 8.5	45.3 $\pm$ 3.6
xNetMF	62.3 $\pm$ 14.8	39.1 $\pm$ 12.6	87.9 $\pm$ 17.7	62.1 $\pm$ 16.1	67.6 $\pm$ 13.3	14.7 $\pm$ 10.6	71.1 $\pm$ 9.8
GraphWave	12.4 $\pm$ 1.6	14.5 $\pm$ 1.0	14.8 $\pm$ 12.4	10.7 $\pm$ 8.4	13.0 $\pm$ 11.4	14.0 $\pm$ 8.1	19.3 $\pm$ 5.6
Role2vec	67.5 $\pm$ 17.1	60.2 $\pm$ 14.2	48.8 $\pm$ 13.9	46.8 $\pm$ 12.9	55.3 $\pm$ 15.3	16.3 $\pm$ 8.3	55.9 $\pm$ 10.1
SEGK	<u>92.4$\pm$13.8</u>	<u>86.9$\pm$15.5</u>	<u>97.9$\pm$8.4</u>	<u>85.6$\pm$14.3</u>	<u>90.4$\pm$13.3</u>	<u>85.8$\pm$17.9</u>	98.4$\pm$7.3
TAWE	97.2$\pm$8.0	97.4$\pm$7.9	99.6$\pm$3.7	94.7$\pm$10.1	98.7$\pm$5.6	99.2$\pm$5.5	<u>97.7$\pm$8.0</u>

4.2 Structural Node Classification

For TAWE, the results of structural node classification on 7 synthetic graphs are shown in Table 3 and Table 4. First, for 6 out of 7 synthetic datasets, TAWE outperforms baselines by a large margin in both Accuracy and F1-macro. For example, TAWE has an 13.4% increase in F1-macro compared to the second best performance obtained by SEGK [37] on the PB-L dataset. Second, on the CoS dataset, TAWE's F1-macro is only 0.7% lower than that of SEGK. Given the fact that the dimension of node embeddings in TAWE is only 3 compared to 16 in SEGK, TAWE also performs quite well in such an unfair comparison.

For two real-world datasets Brazil and Europe, the results of TAWE-P and 5 baselines are presented in Table 5. We can observe that TAWE-P beats all baselines on these two real-world datasets. Taking the Brazil dataset as an example, the F1-macro of TAWE-P (74.3%) is about 10% higher than that of SEGK [37] (64.2%),

which is the best among all baselines. The results on both synthetic datasets and real-world datasets demonstrate the effectiveness of TAWE/TAWE-P in structural node classification task.

Table 5: Accuracy and F1-macro ($\pm std$) of node classification on airport datasets (%).

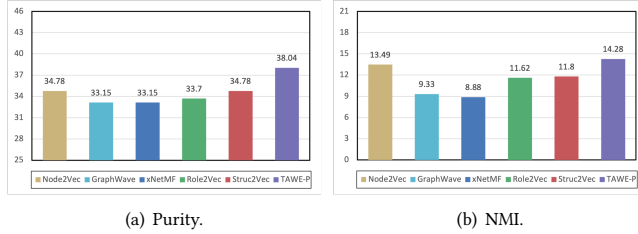
Method	Brazil		Europe	
	Accuracy	F1-macro	Accuracy	F1-macro
Node2vec	61.3 $\pm$ 6.0	25.3 $\pm$ 1.5	43.5 $\pm$ 4.3	24.7 $\pm$ 2.6
xNetMF	63.8 $\pm$ 6.6	37.0 $\pm$ 12.3	60.2 $\pm$ 4.2	44.5 $\pm$ 7.5
GraphWave	<u>76.5$\pm$6.3</u>	55.6 $\pm$ 7.6	66.1$\pm$3.7	<u>48.8$\pm$5.6</u>
Role2vec	58.0 $\pm$ 6.7	41.1 $\pm$ 10.0	43.6 $\pm$ 4.3	28.8 $\pm$ 3.9
SEGK	75.9 $\pm$ 8.9	<u>64.2$\pm$15.1</u>	61.4 $\pm$ 4.1	43.6 $\pm$ 6.0
TAWE-P	79.8$\pm$6.0	74.3$\pm$7.3	<u>63.2$\pm$4.2</u>	58.2$\pm$4.4

Table 6: Purity and NMI of node clustering on synthetic datasets.

Method	B-A		B-B		FW		H-S		H-T		PB-L		CoS	
	Purity	NMI	Purity	NMI	Purity	NMI	Purity	NMI	Purity	NMI	Purity	NMI	Purity	NMI
Node2vec	0.44	0.01	0.57	0.00	0.28	0.00	0.25	0.00	0.28	0.00	0.33	0.00	0.59	0.02
xNetMF	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
GraphWave	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Role2vec	1.00	1.00	0.97	0.93	0.81	0.66	0.93	0.89	0.85	0.77	0.62	0.34	0.92	0.72
SEGK	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
TAWC	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

4.3 Structural Node Clustering

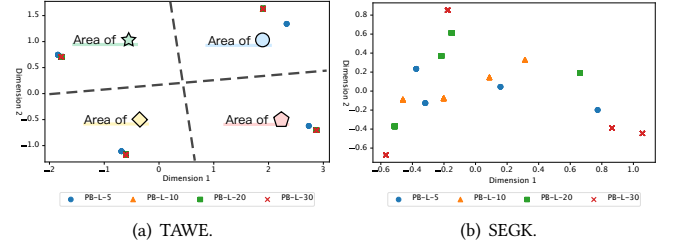
For structural node clustering, we apply k-means++ [2] for clustering on embeddings obtained by different methods. The Purity and NMI of all baselines and our proposed TAWC on 7 synthetic datasets are shown in Table 6. We can see that most algorithms have achieved a perfect clustering performance on simple synthetic datasets (Purity=1 and NMI=1), including our TAWC. This reflects that structural node clustering is a much less challenging task than structural node classification on these simple synthetic graphs. However, for the real-world graph Enron, the proposed TAWC-P consistently outperforms all other baselines in both Purity and NMI, which is presented in Figure 6. The results on synthetic datasets and the real-world Enron graph validate TAWC-P’s capability of grasping local structure of node.

**Figure 6: The purity and NMI on Enron (%).**

4.4 Inductive Capability

In this subsection, we conduct a study on the inductive capability of the proposed TAWC. We launch the inductive experiments on the synthetic PB-L dataset, which is constructed by connecting L numbers of H5 base graph. An illustrative example of PB-L is PB-3 shown in Figure 2 in Section 1, which has 3 H5 base graphs. We build a series of PB-L datasets with 4 steps: PB-5, PB-10, PB-20, PB-30. In each step, some H5 base graphs are added into the previous PB-L graph. The nodes in these new H5 base graphs are unseen nodes in previous step and TAWC needs to generate the structural embeddings for these new nodes without re-running from scratch. The details of the inductive experiments are as the following: First, we run TAWC on PB-5 to construct the merged TAW group $\mathcal{B}$. Then, for each new node u added in PB-10/PB-20/PB-30, we extract the k -hop subgraph $G[\mathcal{N}_k(u)]$ rooted in u and build $\mathcal{B}_u$ and obtain its embedding $F(u, :)$. During this process we do not need to rebuild

the groups of basic TAWs for old nodes appearing in previous step. Taking structural node classification as an example, the classifier trained on PB-5 can still be used for PB-10/PB-20/PB-30. The visualizations of node embeddings obtained via principle component analysis (PCA) [27] from PB-5 to PB-30 are shown in Figure 7. From Figure 7(a), we can observe that node embeddings obtained from TAWC are more concentrate and the classification boundaries for different structural equivalences of nodes (e.g., different shapes of nodes in Figure 2(b)) are clear from PB-5 to PB-30, which verifies the good inductive capability of TAWC.

**Figure 7: Visualization of TAWC’s embeddings compared to those of SEGK.**

5 Related Work

Our paper is related to network embedding and structural node embedding.

Network embedding. Network embedding is to represent nodes in the graph as real-value vectors. Traditional network embedding methods usually take the adjacency matrix as input and project each node into a low-dimensional latent space [5, 8, 38, 45, 48, 52, 72, 73]. Then, DeepWalk [39] first samples the random walks and takes advantage of the skip-gram architecture [36] to maximize the probability of nodes appearing in the same random walk via optimizing node embeddings. Based on DeepWalk [39], node2vec [15] introduces two hyper-parameters to strike a balance between the breadth and depth during search. Different from the previous two approaches based on random walk, LINE [47] explicitly constructs a loss function with 1-hop and 2-hop neighbors using KL-divergence [30]. To unify DeepWalk [39], node2vec [15] and LINE [47], NetMF [40] proves that all these node embedding algorithms can be regarded as some specific forms of matrix factorization. Recently, graph neural networks (GNNs) [16, 28, 51, 59, 68] have become a novel toolkit for

network embedding. The general message-passing mechanism [28] of GNN is simple: at each layer, each node aggregates the information from its neighbors and transforms the input information as the output for next layer via trainable parameters. Based on the message-passing mechanism, GraphSAGE [16] proposes various aggregation strategies. With the purpose of assigning different weights to different neighbors, graph attention network (GAT) [51] integrates the attention mechanism [50] into GNN. A comprehensive survey of different architecture design and taxonomy of GNNs can be found in [57].

Structural node embedding. Structural node embedding methods focus on perceiving the local structural information of node. Struc2vec [41] is one of the most popular approaches of structural node embedding. It identifies that the structural similarity between nodes can be evaluated using the ordered degree sequence of nodes and has been applied to various applications (e.g., influence maximization [56]). Following Struc2vec, xNetMF [18] adopts node degree as structural feature and aggregates the node degree distributions among different hops of neighborhood with a weighted sum. DRNE [49] is another degree-based structural node embedding method, which utilizes node degree as a criterion to sort neighbors into an ordered sequence and feeds the sequence into a layer-normalized long short-term memory (LSTM) model [21]. To capture more sophisticated structures, DeepGL [44] extracts motif-based structural features from the original network. HONE [43] proposes a framework based on a weighted motif graph and leverages the attribute diffusion mechanism to learn node embeddings. Similarly, MCN [31] constructs a weighted motif graph and integrates it in GNN framework. Some other structural node embedding methods include RolX [19], GraphWave [12], Role2vec [1], RiWalk [34] and SEGK [37]. RolX [19] is a structural feature-based matrix factorization algorithm relying on structural feature discovery [20]. GraphWave [12] conducts the structural node embedding with the help of graph kernel [17]. Role2vec [1] and RiWalk [34] sample the feature-based random walk as corpus and learn node embedding via the skip-gram architecture [36]. SEGK [37] learns the structural node embedding in a two-step framework, which first constructs a structural similarity matrix based on graph kernels, and then derives node embeddings with matrix factorization.

6 Conclusion

In this paper, we study the structural node embedding problem. To satisfy the three desired properties: (1) bijective mapping between embedding and local structure of node; (2) inductive capability; and (3) good interpretability of node embedding, we propose a simple yet novel structural node embedding approach named TAW. In particular, TAW creatively integrates anonymous walk and breadth-first search to propose the topological anonymous walk. Based on the merged group of basic TAWs, the normalized frequency of each basic TAW naturally becomes the entry in the corresponding dimension of structural node embedding. Furthermore, we propose a pruning algorithm TAW-P of TAW to handle real-world graphs. Extensive experiments on both synthetic and real-world datasets demonstrate the effectiveness of our proposed TAW/TAW-P in capturing local structure of node.

7 Appendix

PROPOSITION 2 (BIJECTIVE MAPPING PROPERTY). *The mapping between $G[\mathcal{N}_k(u)]$ and $\mathcal{B}_u$ is bijective, when the maximal $L \rightarrow \infty$.*

PROOF. The key to prove this property is the *deterministic process* of constructing random walks, which means that each random walk q with length $l \leq L$ will be visited and more importantly visited only *once*. We need to prove two directions: the first direction is that one $G[\mathcal{N}_k(u)]$ can not generate two different $\mathcal{B}_u$ s, which has already been satisfied in existing methods (e.g., [41]) and is obvious because by adopting the random walk queue Q and BFS, the process to build random walk is deterministic and only one $\mathcal{B}_u$ can be generated for $G[\mathcal{N}_k(u)]$.

The other direction that one $\mathcal{B}_u$ only corresponds to one topology of $G[\mathcal{N}_k(u)]$ is not trivial to prove. Here we prove it by induction on $k_{iso} = \{0, 1, \dots, k\}$ and contradiction. Assume that $\mathcal{B}_u$ corresponds to $G[\mathcal{N}_k(u)]$ and $G[\mathcal{N}_k(v)]$ with different topologies but $\mathcal{B}_u = \mathcal{B}_v = \{taw_1 : C(taw_1), \dots, taw_i : C(taw_i), \dots\}$. We prove that $G[\mathcal{N}_{k_{iso}}(u)]$ and $G[\mathcal{N}_{k_{iso}}(v)]$ are isomorphic by induction for $k_{iso} = \{0, 1, \dots, k\}$. For $k_{iso} = 0$, $taw_1 = (1, 0)$ for both u and v and they are isomorphic with only one node. For $k_{iso} = 2$, since the $taw_2 = (1, 0) \rightarrow (2, 1) : C(taw_2)$ is same for $G[\mathcal{N}_1(u)]$ and $G[\mathcal{N}_1(v)]$, u and v have same number of one-hop neighbors. The only reason that $G[\mathcal{N}_1(u)]$ and $G[\mathcal{N}_1(v)]$ are not isomorphic is that the number of edges between 1-hop nodes for u and v are different⁷. Without loss of generality, let us assume that $G[\mathcal{N}_1(u)]$ has fewer edges between 1-hop nodes than $G[\mathcal{N}_1(v)]$. This will result in the number of $taw_3 = (1, 0) \rightarrow (2, 1) \rightarrow (3, 1)$ different for $G[\mathcal{N}_1(u)]$ and $G[\mathcal{N}_1(v)]$, which leads to contradiction. Then, let us assume that $G[\mathcal{N}_{k_{iso}}(u)]$ and $G[\mathcal{N}_{k_{iso}}(v)]$ are isomorphic. For $k_{iso} = k_{iso} + 1$, the step 1 is that we remove the edges between the $k_{iso} + 1$ -hop nodes. If the remaining two graphs are not isomorphic (i.e., different connection structures between k_{iso} -hop nodes and $k_{iso} + 1$ -hop nodes), this will lead to the number of at least one of the following two kinds TAWs to be different: (1) $(1, 0) \rightarrow (2, 1) \rightarrow \dots \rightarrow (k_{iso} + 1, k_{iso})$ (i.e., random walk from the start node to the k_{iso} -hop end node). (2) $(1, 0) \rightarrow (2, 1) \rightarrow \dots \rightarrow (k_{iso} + 1, k_{iso}) \rightarrow (k_{iso}, k_{iso} - 1) \rightarrow (k_{iso} + 1, k_{iso}) \rightarrow (k_{iso}, k_{iso} - 1) \dots$ (i.e., random walk repeatedly from the $k_{iso} + 1$ -hop node back to the k_{iso} -hop node and returning to the $k_{iso} + 1$ -hop node again). This causes contradiction. Then, we added the edges between the edges between the $k_{iso} + 1$ -hop nodes back to these two graphs, the contradiction can be proved same as taw_3 for $G[\mathcal{N}_1(u)]$ and $G[\mathcal{N}_1(v)]$. So, the bijective mapping property proposition is proved to be true. $\square$

Acknowledgments

This work is supported by NSF (2324770), DARPA (HR001121C0165), and AFOSR (FA9550-24-1-0002). The content of the information in this document does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

⁷This is because we have clearly stated that self loops and duplicated edges are not allowed in G .

References

- [1] Nesreen K. Ahmed, Ryan A. Rossi, John Boaz Lee, Theodore L. Willke, Rong Zhou, Xiangnan Kong, and Hoda Eldardiry. 2022. Role-Based Graph Embeddings. *IEEE Trans. Knowl. Data Eng.* 34, 5 (2022), 2401–2415.
- [2] David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7–9, 2007*. SIAM, 1027–1035.
- [3] Yikun Ban, Yuchen Yan, Arindam Banerjee, and Jingrui He. 2021. Ee-net: Exploitation-exploration neural networks in contextual bandits. *arXiv preprint arXiv:2110.03177* (2021).
- [4] Yikun Ban, Yuchen Yan, Arindam Banerjee, and Jingrui He. 2023. Neural exploitation and exploration of contextual bandits. *arXiv preprint arXiv:2305.03784* (2023).
- [5] Ingwer Borg and Patrick JF Groenen. 2005. *Modern multidimensional scaling: Theory and applications*. Springer Science & Business Media.
- [6] Huiyuan Chen, Lan Wang, Yusan Lin, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2021. Structured graph convolutional networks with stochastic masks for recommender systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 614–623.
- [7] Huiyuan Chen, Zhe Xu, Chin-Chia Michael Yeh, Vivian Lai, Yan Zheng, Minghua Xu, and Hanghang Tong. 2024. Masked Graph Transformer for Large-Scale Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2502–2506.
- [8] Huiyuan Chen, Chin-Chia Michael Yeh, Fei Wang, and Hao Yang. 2022. Graph neural transport networks with non-local attentions for recommender systems. In *Proceedings of the ACM Web Conference 2022*. 1955–1964.
- [9] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.
- [10] R. Diestel. 2006. *Graph Theory*. Springer.
- [11] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. 135–144.
- [12] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings via Diffusion Wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19–23, 2018*, Yike Guo and Faisal Farooq (Eds.). ACM, 1320–1329.
- [13] Boxin Du, Si Zhang, Yuchen Yan, and Hanghang Tong. 2021. New frontiers of multi-network mining: Recent developments and future trend. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 4038–4039.
- [14] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 1797–1806.
- [15] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13–17, 2016*. ACM, 855–864.
- [16] William L. Hamilton, Zitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4–9, 2017, Long Beach, CA, USA*. 1024–1034.
- [17] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. 2011. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis* 30, 2 (2011), 129–150.
- [18] Mark Heimann, Haoming Shen, Tara Safavi, and Danai Koutra. 2018. REGAL: Representation Learning-based Graph Alignment. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22–26, 2018*. ACM, 117–126.
- [19] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. RoLX: structural role extraction & mining in large graphs. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12–16, 2012*. ACM, 1231–1239.
- [20] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: graph mining using recursive structural features. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21–24, 2011*. ACM, 663–671.
- [21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [22] Xiaohui Huang, Yuming Ye, Xiaofei Yang, and Liyan Xiong. 2022. Multistep Coupled Graph Convolution With Temporal-Attention for Traffic Flow Prediction. *IEEE Access* 10 (2022), 48179–48192.
- [23] Sergey Ivanov and Evgeny Burnaev. 2018. Anonymous Walk Embeddings. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholm, Sweden, July 10–15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 2191–2200.
- [24] Junchen Jin, Mark Heimann, Di Jin, and Danai Koutra. 2022. Toward Understanding and Evaluating Structural Node Embeddings. *ACM Trans. Knowl. Discov. Data* 16, 3 (2022), 58:1–58:32.
- [25] Baoyu Jing, Yuchen Yan, Kaize Ding, Chanyoung Park, Yada Zhu, Huan Liu, and Hanghang Tong. 2024. Sterling: Synergistic representation learning on bipartite graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 12976–12984.
- [26] Baoyu Jing, Yuchen Yan, Yada Zhu, and Hanghang Tong. 2022. Coin: Co-cluster infomax for bipartite graphs. In *NeurIPS 2022 Workshop: New Frontiers in Graph Learning*.
- [27] Ian T Jolliffe. 2002. *Principal component analysis for special types of data*. Springer.
- [28] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24–26, 2017, Conference Track Proceedings*. OpenReview.net.
- [29] Bryan Klimt and Yiming Yang. 2004. Introducing the Enron corpus.. In *CEAS*, Vol. 45. 92–96.
- [30] Solomon Kullback and Richard A Leibler. 1951. On information and sufficiency. *The annals of mathematical statistics* 22, 1 (1951), 79–86.
- [31] John Boaz Lee, Ryan A. Rossi, Xiangnan Kong, Sungchul Kim, Eunye Koh, and Anup Rao. 2019. Graph Convolutional Networks with Motif-based Attention. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3–7, 2019*. ACM, 499–508.
- [32] Jinning Li, Huajie Shao, Dachun Sun, Ruijie Wang, Yuchen Yan, Jinyang Li, Shengzhong Liu, Hanghang Tong, and Tarek Abdelzaher. 2022. Unsupervised belief representation learning with information-theoretic variational graph auto-encoders. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1728–1738.
- [33] Zhining Liu, Ruizhong Qiu, Zhichen Zeng, Hyunsik Yoo, David Zhou, Zhe Xu, Yada Zhu, Kommy Weldemariam, Jingrui He, and Hanghang Tong. 2024. Class-Imbalanced Graph Learning without Class Rebalancing. In *Forty-first International Conference on Machine Learning*.
- [34] Xuewei Ma, Geng Qin, Zhiyang Qiu, Mingxin Zheng, and Zhe Wang. 2019. RiWalk: Fast Structural Node Embedding via Role Identification. In *2019 IEEE International Conference on Data Mining, ICDM 2019, Beijing, China, November 8–11, 2019*. IEEE, 478–487.
- [35] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2–4, 2013, Workshop Track Proceedings*.
- [36] Tomáš Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5–8, 2013, Lake Tahoe, Nevada, United States*. 3111–3119.
- [37] Giannis Nikolentzos and Michalis Vazirgiannis. 2021. Learning Structural Node Representations Using Graph Kernels. *IEEE Trans. Knowl. Data Eng.* 33, 5 (2021), 2045–2056.
- [38] Karl Pearson. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2, 11 (1901), 559–572.
- [39] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, Sofus A. Macskassy, Claudia Perlich, Jure Leskovec, Wei Wang, and Rayid Ghani (Eds.). ACM, 701–710.
- [40] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. 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*. 459–467.
- [41] Leonardo Filipe Rodrigues Ribeiro, Pedro H. P. Saverese, and Daniel R. Figueiredo. 2017. struc2vec: Learning Node Representations from Structural Identity. *ACM*, 385–394.
- [42] Shane Roach, Connie Ni, Alexei Kopylov, Tsai-Ching Lu, Jiejun Xu, Si Zhang, Boxin Du, Dawei Zhou, Jun Wu, Lihui Liu, et al. 2020. CANON: Complex Analytics of Network of Networks for Modeling Adversarial Activities. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 1634–1643.
- [43] Ryan A. Rossi, Nesreen K. Ahmed, and Eunye Koh. 2018. HONE: Higher-Order Network Embeddings. *CoRR abs/1801.09303* (2018). <http://arxiv.org/abs/1801.09303>
- [44] Ryan A. Rossi, Rong Zhou, and Nesreen K. Ahmed. 2017. Deep Feature Learning for Graphs. *CoRR abs/1704.08829* (2017).
- [45] Sam T Roweis and Lawrence K Saul. 2000. Nonlinear dimensionality reduction by locally linear embedding. *science* 290, 5500 (2000), 2323–2326.

- [46] Jian Tang, Meng Qu, and Qiaozhu Mei. 2015. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*. 1165–1174.
- [47] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.
- [48] Joshua B Tenenbaum, Vin de Silva, and John C Langford. 2000. A global geometric framework for nonlinear dimensionality reduction. *science* 290, 5500 (2000), 2319–2323.
- [49] Ke Tu, Peng Cui, Xiao Wang, Philip S. Yu, and Wenwu Zhu. 2018. Deep Recursive Network Embedding with Regular Equivalence. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19–23, 2018*. ACM, 2357–2366.
- [50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).
- [51] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.
- [52] Ulrike Von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and computing* 17, 4 (2007), 395–416.
- [53] Dingsu Wang, Yuchen Yan, Ruizhong Qiu, Yada Zhu, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2023. Networked time series imputation via position-aware graph enhanced variational autoencoders. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2256–2268.
- [54] Ruijie Wang, Baoyu Li, Yichen Lu, Dachun Sun, Jinning Li, Yuchen Yan, Shengzhong Liu, Hanghang Tong, and Tarek F Abdelzaher. 2023. Noisy positive-unlabeled learning with self-training for speculative knowledge graph reasoning. *arXiv preprint arXiv:2306.07512* (2023).
- [55] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. 2018. Acekg: A large-scale knowledge graph for academic data mining. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 1487–1490.
- [56] Wei Wang, Haili Yang, Yuanfu Lu, Yuanhang Zou, Xu Zhang, Shuting Guo, and Leyu Lin. 2021. Influence Maximization in Multi-Relational Social Networks. In *CIKM '21: The 30th ACM International Conference on Information and Knowledge Management, Virtual Event, Queensland, Australia, November 1 - 5, 2021*. ACM, 4193–4202.
- [57] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.
- [58] Haobo Xu, Yuchen Yan, Dingsu Wang, Zhe Xu, Zhichen Zeng, Tarek F Abdelzaher, Jiawei Han, and Hanghang Tong. 2024. SLOG: An Inductive Spectral Graph Neural Network Beyond Polynomial Filter. In *Forty-first International Conference on Machine Learning*.
- [59] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How Powerful are Graph Neural Networks? *CoRR* abs/1810.00826 (2018).
- [60] Yuchen Yan, Yuzhong Chen, Huiyuan Chen, Minghua Xu, Mahashweta Das, Hao Yang, and Hanghang Tong. 2024. From trainable negative depth to edge heterophily in graphs. *Advances in Neural Information Processing Systems* 36 (2024).
- [61] Yuchen Yan, Yuzhong Chen, Mahashweta Das, Hao Yang, and Hanghang Tong. 2022. ReD-GCN: Revisit the Depth of Graph Convolutional Network. (2022).
- [62] Yuchen Yan, Yongyi Hu, Qinghai Zhou, Lihui Liu, Zhichen Zeng, Yuzhong Chen, Menghai Pan, Huiyuan Chen, Mahashweta Das, and Hanghang Tong. 2024. Pacer: Network embedding from positional to structural. In *Proceedings of the ACM on Web Conference 2024*. 2485–2496.
- [63] Yuchen Yan, Baoyu Jing, Lihui Liu, Ruijie Wang, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2024. Reconciling competing sampling strategies of network embedding. *Advances in Neural Information Processing Systems* 36 (2024).
- [64] Yuchen Yan, Lihui Liu, Yikun Ban, Baoyu Jing, and Hanghang Tong. 2021. Dynamic knowledge graph alignment. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 4564–4572.
- [65] Yuchen Yan, Si Zhang, and Hanghang Tong. 2021. Bright: A bridging algorithm for network alignment. In *Proceedings of the web conference 2021*. 3907–3917.
- [66] Yuchen Yan, Qinghai Zhou, Jinning Li, Tarek Abdelzaher, and Hanghang Tong. 2022. Dissecting cross-layer dependency inference on multi-layered interdependent networks. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*. 2341–2351.
- [67] Xiaodong Yang, Huiyuan Chen, Yuchen Yan, Yuxin Tang, Yuying Zhao, Eric Xu, Yiwei Cai, and Hanghang Tong. 2024. SimCE: Simplifying Cross-Entropy Loss for Collaborative Filtering. *arXiv preprint arXiv:2406.16170* (2024).
- [68] Chin-Chia Michael Yeh, Mengting Gu, Yan Zheng, Huiyuan Chen, Javid Ebrahimi, Zhongfang Zhuang, Junpeng Wang, Liang Wang, and Wei Zhang. 2022. Embedding compression with hashing for efficient representation learning in large-scale graph. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 4391–4401.
- [69] Zhichen Zeng, Boxin Du, Si Zhang, Yinglong Xia, Zhining Liu, and Hanghang Tong. 2024. Hierarchical multi-marginal optimal transport for network alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 16660–16668.
- [70] Zhichen Zeng, Ruizhong Qiu, Zhe Xu, Zhining Liu, Yuchen Yan, Tianxin Wei, Lei Ying, Jingrui He, and Hanghang Tong. 2024. Graph Mixup on Approximate Gromov–Wasserstein Geodesics. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 58387–58406.
- [71] Zhichen Zeng, Si Zhang, Yinglong Xia, and Hanghang Tong. 2023. Parrot: Position-aware regularized optimal transport for network alignment. In *Proceedings of the ACM Web Conference 2023*. 372–382.
- [72] Zhichen Zeng, Ruike Zhu, Yinglong Xia, Hanqing Zeng, and Hanghang Tong. 2023. Generative graph dictionary learning. In *International Conference on Machine Learning*. PMLR, 40749–40769.
- [73] Yuying Zhao, Minghua Xu, Huiyuan Chen, Yuzhong Chen, Yiwei Cai, Rashidul Islam, Yu Wang, and Tyler Derr. 2024. Can One Embedding Fit All? A Multi-Interest Learning Paradigm Towards Improving User Interest Diversity Fairness. In *Proceedings of the ACM on Web Conference 2024*. 1237–1248.