MrMine: Multi-resolution Multi-network Embedding

Boxin Du*
University of Illinois at Urbana-Champaign
boxindu2@illinois.edu

Hanghang Tong*
University of Illinois at Urbana-Champaign
htong@illinois.edu

ABSTRACT

Network embedding has become the cornerstone of a variety of mining tasks, such as classification, link prediction, clustering, anomaly detection and many more, thanks to its superior ability to encode the intrinsic network characteristics in a compact low-dimensional space. Most of the existing methods focus on a single network and/or a single resolution, which generate embeddings of different network objects (node/subgraph/network) from different networks *separately*. A fundamental limitation with such methods is that the intrinsic relationship across different networks (e.g., two networks share same or similar subgraphs) and that across different resolutions (e.g., the node-subgraph membership) are ignored, resulting in *disparate* embeddings. Consequentially, it leads to sub-optimal performance or even becomes inapplicable for some downstream mining tasks (e.g., role classification, network alignment. etc.).

In this paper, we propose a unified framework (MrMine) to learn the representations of objects from multiple networks at three complementary resolutions (i.e., network, subgraph and node) simultaneously. The key idea is to construct the cross-resolution cross-network context for each object. The proposed method bears two distinctive features. First, it enables and/or boosts various multi-network downstream mining tasks by having embeddings at different resolutions from different networks in the same embedding space. Second, Our method is efficient and scalable, with a $O(nlog(n))$ time complexity for the base algorithm and a *linear* time complexity w.r.t. the number of nodes and edges of input networks for the accelerated version. Extensive experiments on real-world data show that our methods (1) are able to enable and enhance a variety of multi-network mining tasks, and (2) scale up to million-node networks.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; • **Theory of computation → Graph algorithms analysis**; • **Information systems** → *Information retrieval query processing*.

KEYWORDS

Network embedding; Cross-resolution cross-network context; Unified framework; Multi-network mining

*The work was partly done while the author was at Arizona State University.

ACM Reference Format:
Boxin Du and Hanghang Tong. 2019. MrMine: Multi-resolution Multi-network Embedding. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3357384.3357944

1 INTRODUCTION

Network mining is the cornerstone of many real-world applications, and has been receiving much research attention in recent years. It offers a powerful way to encode the underlying network characteristics (e.g., topology, attribute) into a compact low-dimensional space. As such, it has benefited a variety of downstream data mining tasks (e.g., node/network classification, link prediction, and clustering), often with a significantly boosted empirical performance. Despite much progress has been made (see Section 5 for a review), most of the existing work has not adequately, if at all, addressed two fundamental limitations, which we elaborate below.

First, Most of the existing work, with only a few exceptions (see Section 5), focuses on a *single network*. For multiple input networks, these methods will learn embeddings of different networks separately, and thus might result in disparate embedding space. To see this, let us take node embedding as an example. A dominant branch of node embedding (e.g., *deepwalk* and many of its follow-ups [5, 16, 18, 21]) relies on identifying appropriate node proximity/context based on truncated/short random walks. The identified node context will be then preserved in the embedding space, often through a language model (e.g., Continuous Bag of Words (CBOW) and SkipGram). However, any node pair across different networks are disconnected without auxiliary information (e.g., anchor links). In other words, nodes in one network will never be the context of nodes from another network and vice versa. Therefore, the node embeddings of different networks will be in different or disconnected space. This would render the inapplicability of some downstream mining tasks (e.g., cross-layer dependence inference in multi-layered network systems [11]) or add extra complexity for other mining tasks. For example, with the node embeddings from such methods as inputs, we would have to train an additional classifier for network alignment task, constrained by the availability of extra anchor links.

Second, most, if not all, of the existing work is designed to learn embeddings at *single resolution*. For example, the vast majority of network embedding focuses on *node* embedding (e.g., *node2vec* [5], *LINE* [21], *deepwalk* [16] and many more); at the coarser resolution, *subgraph2vec* [13] and *deep graph kernel* [25] learn the embeddings of subgraphs; at the coarsest resolution, *graph2vec* [14] focuses on learning vector representations for the entire networks. To the best of our knowledge, almost none of the existing methods support multi-resolution network embedding, although objects (e.g., nodes, subgraphs, and networks) across different resolutions are

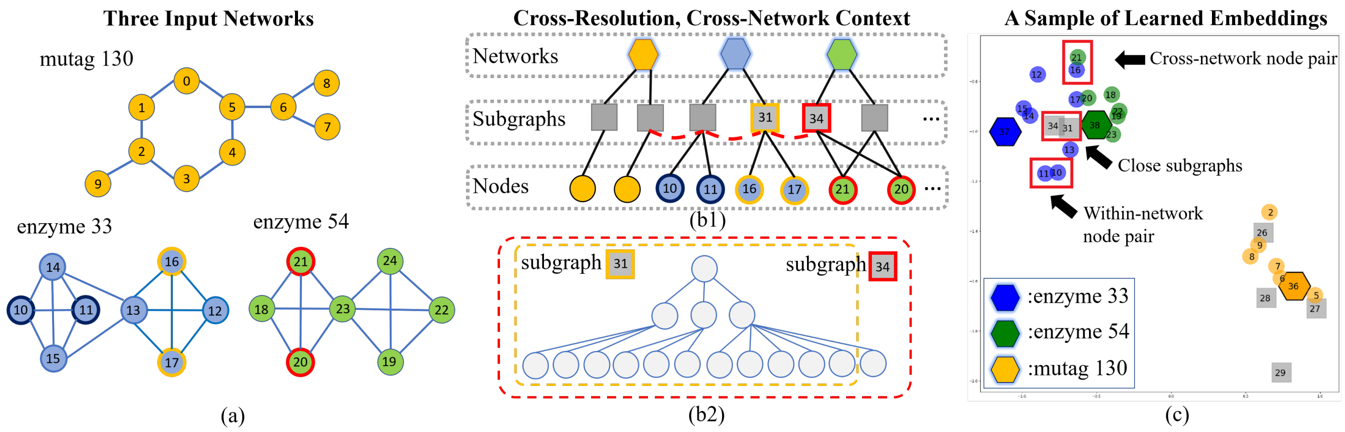


Figure 1: an illustrative example of multi-resolution multi-network embedding. (a) shows three small molecular graphs from bioinformatics dataset. The two lower graphs belong to the same category of enzyme. (b1) shows the Cross-Resolution Cross-Network (CRCN) relation network constructed by our method. Some subgraphs/nodes are omitted for brevity (see section 3 for detail). (b2) shows two similar subgraphs (WL subtrees here) extracted from enzyme 33 and enzyme 54, numbered in 31 and 34. (c) shows the learned 2-d representation for all network objects. Some embeddings are omitted to avoid overlap.

intrinsically correlated with each other. For example, networks with similar subgraph distributions tend to be similar to each other [25]; [18] indicates that, nodes inside similar subgraphs are likely to belong to the same category. Ignoring such cross-resolution correlation during the embedding learning process is likely to lead to sub-optimal results. On the contrary, if we could have objects at different resolutions in the same embedding space, it might greatly benefit certain downstream applications. For example, for network science of teams [12], by embedding both team members (nodes of the underlying person network) and teams (subgraphs) in the same space, it would immediately enable effective team member recommendation by calculating the similarity between a candidate team member and a given target team, say based on the cosine similarity between the member embedding and team embedding.

In order to address the above limitations, we propose a unified method (MrMine) to learn the multi-resolution multi-network representation simultaneously in a mutually beneficial way. The key idea is to construct the cross-resolution cross-network context for each object of each network at three complementary resolutions, including node, subgraph and network. The constructed corpus of such network context can be then fed into a variety of language models, such as CBOW, SkipGram, etc., to generate the embeddings for different network objects (nodes, subgraphs, networks) from different networks in the same space. The proposed method is highly efficient and scalable, with a time complexity of $O(Hnlog(n))$ (where n is the number of nodes of input networks and H is a constant much smaller than n) for the basic version. We further propose an accelerated version with a *linear* time complexity w.r.t. the number of nodes and edges of input networks.

Figure 1 presents an illustrative example of the simplified procedure of multi-resolution and multi-network embedding. Figure 1 (a) shows three input networks selected from bioinformatics dataset. (b1) represents the Cross-Resolution Cross-Network (CRCN) relation network in which we connect vertices of all three types of objects (i.e. nodes, subgraphs, and networks, represented as circles, squares, and hexagons respectively) in the same network, according to the node-subgraph, subgraph-network membership (vertical black solid links) and subgraph similarities (horizontal red dashed links). For example, enzyme 33 contains a subgraph (we use Weisfeiler-Lehman (WL) subtree [19] in this example; see more details in section 3.1) numbered in 31, which contains node 16 and 17. So the blue hexagon is connected to the gray square 31, and the gray square 31 is connected to the blue circle 16 and 17. Subgraph 31 and 34 are structurally similar with only one node difference (Fig. (b2)), so gray square 31 and 34 are connected. Context of objects from different resolutions and networks are then extracted from the CRCN relation network to learn the embeddings in (c). As we can observe from (c), the green network is close to the blue network (two enzymes of the same category) while both are far apart from the orange network (a mutag instance). Node 10 and 11 are close because they are connected by the same square node in (b1), which means they are rooted at the same WL subtree. Subgraph 31 and 34 are close in the embedding space although they exist in different networks. Also from different networks, node 16 and node 21 are close because they are structurally similar (connected to similar subgraph 31 and 34 respectively). Our method embeds multi-resolution multi-network objects into the same space, and it naturally enables downstream multi-network mining tasks. The main contributions of the paper are:

- **Problems**. To the best of our knowledge, we are the first to study the problem of learning multi-resolution multi-network embeddings.
- **Algorithms**. We propose effective and efficient algorithms for learning the embeddings of multi-resolution and multi-network objects simultaneously.
- **Empirical Evaluations**. We perform extensive experimental evaluations on a diverse set of real networks, which demonstrate that our methods (1) enable and enhance a

variety of graph mining tasks, such as collective network alignment, and (2) scale up to million-node graphs.

The rest of the paper is organized as follows. Section 2 formulates the problem of multi-resolution multi-network embedding. Section 3 presents our proposed base and improved method. Section 4 presents the experimental results. Section 5 provides a brief review of recent related work. The paper is concluded in Section 6.

2 PROBLEM DEFINITION

In this section we formally define the problem of multi-resolution multi-network embedding. The symbols and notations used in this paper are summarized in Table 1. Let $\mathcal{G} = \{G_1, G_2, ..., G_k\}$ represent a set of k input networks, and $\mathcal{S} = \{S_1, S_2, ...S_l\}$ represents the subgraph set which contains all the subgraphs extracted from each graph in $\mathcal{G}$ of a particular type. For example, $\mathcal{S}$ could be a set of all Breath First Search (BFS) subtrees or Weisfeiler-Lehman (WL) subtrees with height less than 3 exacted from all networks in $\mathcal{G}$. In this paper, we use WL subtrees for all the algorithms. With these notations, the problem of multi-resolution multi-network embedding can be defined as follows.

Problem 1. *Multi-resolution multi-network embedding*
Given: *(a) the inputs for constructing Cross-Resolution Cross-Network (CRCN) relation network: (a1) a set of networks $\mathcal{G}$, (a2) the dimension of embedding vectors p; (a3) the maximum height of WL subtrees H; and (b) the parameter set for corpus generation and SkipGram model (e.g. SkipGram window size w, random walk length l, etc.).*
Output: *the embedding matrices $\mathbf{F_g}$, $\mathbf{F_s}$, and $\mathbf{F_n}$ for (1) all input networks in $\mathcal{G}$, (2) all extracted subgraphs in $\mathcal{S}$, and (3) all nodes in $\mathcal{G}$, respectively, with all embeddings in the same space.*

We adopt the following terminologies and notations for simplicity of algorithm description. First, We use *vertex* and *vertices* to indicate the nodes in the CRCN relation network, and *nodes* to only indicate nodes in the original network set. Second, we use function p_n, p_s, p_g for the mappings from original network object to the vertices in the CRCN relation network, and q_n, q_s, q_g for the mapping from vertices in the relation network back to original network objects. For example, $p_s(L^n_G) = v$ maps the subgraph with label L^n_G (meaning WL subtree rooted at node $n \in G$; every unique WL subtree can be represented by a distinct label) to vertex v in the CRCN relation network, and $q_s(v) = l$ maps the vertex v from the CRCN relation network back to a subgraph label l. Similarly, p_n, q_n are used for node level mapping and p_g, q_g are used for network level mapping.

To illustrate the intuition of Problem 1, let us make an analogy between multi-resolution multi-network embedding and text embedding. The objects of multiple networks, subgraphs, and nodes in our problem setting can be seen as documents, sentences, and words in text embedding respectively. In this case, the problem studied by [10] embeds sentences/paragraphs with words in a document, which resembles our multi-resolution problem setting.

3 PROPOSED ALGORITHMS

In this section, we introduce our proposed method. We start with the preliminaries followed by challenges and key ideas. We then present the detailed demonstration of our basic algorithm and the accelerated algorithm.

Table 1: Major Notations and Definition

Symbols	Definition
$\mathcal{G} = \{G_1, G_2, ..., G_k\}$	a set of k graphs
$\mathcal{R}, \mathcal{E}_R$	cross-resolution cross-network relation network and edge set
$\mathcal{E}_{R_s}$	a set of edges within subgraph vertices
$\mathcal{E}_{R_g}$	a set of edges between subgraph vertices and network vertices
$\mathcal{E}_{R_n}$	a set of edges between subgraph vertices and node vertices
$\mathcal{L} = \{L^{n_1}_{G_1}, ...L^{n_k}_{G_k}\}$	a set of multi-set labels for all WL subtrees in the graph set $\mathcal{G}$
$\mathbf{F_g}, \mathbf{F_s}, \mathbf{F_n}$	embedding matrices for networks, subgraphs, and nodes
H	the maximum height of WL subtrees
L^n/L^n_G	the WL subtree label of node n/specifically $n \in G$
$(L^n_G)_i$	the WL label of node $n \in G$ at i-th WL iteration
Q_{S_i}, Q_{S_j}	the sorted degree sequence of subtree S_i, S_j
p	the dimension of embedding vectors
$\Phi(n)$	the embedding of vertex n
p_n, p_s, p_g	mapping functions from original objects to CRCN network
q_n, q_s, q_g	mapping functions from CRCN network to original objects

3.1 Preliminaries

Weisfeiler-Lehman (WL) subtree. WL subtree [19] is a subgraph with tree structure rooted at a designated node in a network. The height of the WL subtree is the maximum distance between the root node and any other nodes. Unlike the BFS subtree, the WL subtree treats the repetition of nodes in the node search of tree construction as distinct nodes. For example, in the upper right WL subtree in Figure 2, node 0 appears in both level 0 and level 2, because node 0 is the neighbor of node 1, 3 and 4 in level 1, and in level 2 it is regarded as a distinct node.
WL label transformation. Originated from the Weisfeiler-Lehman graph isomorphism test, the WL label transformation iteration [19] is an algorithm used for relabeling node labels. The time complexity is $O(hm)$, where h is the iteration number/height of WL subtrees, and m is the number of edges. The multi-set labels generated by the i-th WL iteration can be regarded as the identity of unique WL subtrees with height of i [13]. In each iteration of WL label transformation, a new set of labels is produced for all nodes in a network. Since each unique label corresponds to a particular WL subtree, we use the labels (e.g. L^n_G) as IDs for subtrees in this paper. Therefore, it can be used for efficiently generating subgraphs on each node, and building the vocabulary of subgraphs.

3.2 Challenges and Key Ideas

In order to learn the multi-resolution multi-network embedding on the same space, there are several challenges as follows. First, how to build cross-resolution cross-network context, so that objects at different resolutions from different networks could be in each other's context, which would in turn allow to find their embeddings in the same space? Our first key idea is to introduce Cross-Resolution Cross-Network (CRCN) relation network with vertices representing multi-resolution multi-network objects, and edges representing cross-resolution cross-network relations between objects. For example, in Figure 1 (b1), the CRCN relation network aims to capture the structural relationship among nodes, subgraphs, and three molecular networks. Second, how to construct the links of CRCN relation network? Our second key idea is to use WL subtrees for the subgraph resolution. It bears three advantages as follows. (1) the WL label transformation algorithm can be used to efficiently generate WL subtrees as subgraphs [19]; (2) the WL subtrees can act as bridges to build links between network/subgraph/node vertices (i.e. cross-resolution links); (3) the 'borderless' WL subtrees in conjunction with the similarity defined over subgraphs help to build cross-network links. Third, how to reduce the computation cost to build CRCN relation network (e.g. the number of subgraphs could be large, and it could be time-consuming to build subgraph-subgraph similarity links, etc.)? Our third idea is to explore a hierarchical structure of WL subtrees for the subgraph resolution in the CRCN relation network construction, for the sake of avoiding explicit cross-network links (e.g. red dashed links in Fig. 1 (b1)), but meanwhile still preserving the cross-network subgraph similarities (see section 3.4 for detail).

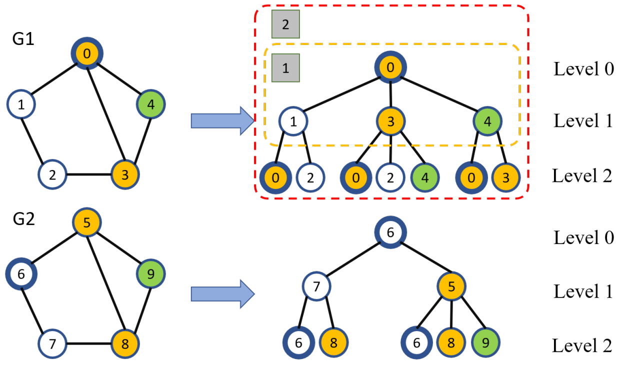


Figure 2: an illustrative example of WL subtree. The right trees are 2-level WL trees rooted at node 0 and 6, respectively. Best viewed in color.

3.3 Basic MrMine Method

First, we adopt H iterations of WL node label transformation to generate unique WL subtrees of up to height H [13, 19]. Specifically, in order to construct the cross-network links between subgraphs, we propose to use a function $f(S_i, S_j)$ to calculate the similarity between subgraph S_i and S_j. Similar to the selection of subtrees, the selection of function f is also not limited. The most intuitive method of calculating similarities between two graphs is using graph kernels. In this case, $f(S_i, S_j) = K(S_i, S_j)$, in which K can be any graph kernels such as random walk graph kernel [23], WL subtree kernel [19], etc. Although in common case, calculating graph kernel is costly ($O(n^3)$ in which n is the number of nodes [23]) without approximation, subgraphs are smaller-scaled (compared to networks) and can be calculated relatively efficient. We provide two more efficient f functions below.

We observe that, for WL subtrees, the structural characteristic is essentially preserved by the node degrees of each level. For example, in Figure 2, the structural characteristic of 2-level subtree can be simply represented by the degree sequence of node 0 concatenated with the sorted degrees of its neighboring node 1, 3, and 4, which gives level 1: 3, level 2: 2,2,3. Since node 3 is structurally equal to node 0 (belonging to isomorphic WL subtree), they both produce the same degree sequence. The degree sequence of WL subtree rooted at node 1 (i.e. level 1: 2, level 2: 2,3) is distinct from that of the WL subtree rooted at node 0 and 3. Therefore the subtree structures are also quite different. We adopt Dynamic Time Wrapping (DTW) [18] to measure the distance between two degree sequences at each level, and connect subgraphs with structural scores lower than a threshold. Formally, for sorted degree sequence Q_{S_i} and Q_{S_j} of subtrees S_i and S_j with length l_{S_i} and l_{S_j}:

$$f(Q_{S_i}, Q_{S_j}) = \sum_h DTW(Q^h_{S_i}, Q^h_{S_j}) \tag{1}$$

where $Q^h_{S_i}, Q^h_{S_j}$ are the sorted degree sequences of S_i and S_j at level h, respectively. However, DTW might fail to distinguish between high-level degree sequences of different length. For example, the sorted degree sequence for 2-level subtree of node 0 in G_1 (which is 2, 2, 3) has zero DTW value compared with the same level degree sequence of node 6 in G_2 (which is 2, 3). To address this issue, we propose to use a method similar to Spearman's footrule distance, which is commonly used in ranking list comparison.

$$f(Q_{S_i}, Q_{S_j}) = \sum_h \sum_t |\tilde{Q}^h_{S_i}(t) - \tilde{Q}^h_{S_j}(t)| \tag{2}$$

where $\tilde{Q}^h_{S_i}$ is the sorted degree sequence on level h after filling zeros to the front of the original list, $Q^h_{S_i}$, to make $\tilde{Q}^h_{S_i}$ and $\tilde{Q}^h_{S_j}$ have the same length (i.e. $max(l_{S_i}, l_{S_j})$). $t = 1, ..., max(l_{S_i}, l_{S_j})$.

Algorithm 1 CRCN Relation Network Builder

Input: Given input network set $\mathcal{G} = \{G_1, G_2, ...G_k\}$, maximum subtree height H.
Output: The CRCN relation network R.

```
1:  for $G \in \mathcal{G}$ do
2:      Conduct $H$ WL relabeling iterations to generate multi-set labels $\mathcal{L}$.
3:  end for
4:  Set edge set of $\mathcal{R}$: $\mathcal{E}_R = \Phi$.
5:  for each label $L^n_G \in \mathcal{L}$ do
6:      if $(p_s(L^n_G), p_n(n)) \notin \mathcal{E}_R$ then
7:          Add edge $(p_s(L^n_G), p_n(n))$ to $\mathcal{E}_R$.
8:      end if
9:      if $(p_g(G), p_s(L^n_G)) \notin \mathcal{E}_R$ then
10:         Add edge $(p_g(G), p_s(L^n_G))$ to $\mathcal{E}_R$.
11:     end if
12: end for
13: Return the CRCN relation network $\mathcal{R}$
```

The proposed CRCN relation network building algorithms are summarized in Algorithm 1 and Algorithm 2. Algorithm 1 builds the CRCN relation network without cross-network edges. Algorithm 2 generates the cross-network edges between subgraph vertices.

In line 2 of Algorithm 1, we conduct H WL relabeling iterations to generate multi-set node labels for the IDs of WL subtrees ($O(Hm)$). From line 5 to line 12 we add edges between node vertices and subgraph vertices, and edges between network vertices and subgraph vertices ($O(Hkn)$). Note that no edges between node vertices are added although edges exist between nodes in the original network.

After the CRCN relation network $\mathcal{R}$ is constructed, we conduct truncated random walk [16] on each vertex of $\mathcal{R}$ to build the corpus of multi-resolution multi-network objects, which is similar to build corpus for nodes in a single network. The corpus can be further fit

Algorithm 2 Cross-Network Subgraph Context Builder

```
Input: Given a multi-resolution relation network $\mathcal{R}$ (with subtree
    vertex list $V$ and edge set $\mathcal{E}_R$), threshold $\sigma$, window size $w$.
Output: The multi-resolution relation network with added cross-
    network links of subgraph vertices.
1: Sort $V$ of subtree vertices by the summation of degree sequence.
2: for each $v \in V$ do
3:     for each subtree $s$ in the window of size $w$ of $v$ do
4:         if $f(Q_s^h, Q_v^h) < \sigma$ then
5:             Add edge $(p_s(v), p_s(s))$ to $\mathcal{E}_R$
6:         end if
7:     end for
8: end for
9: Return $\mathcal{R}$
```

into a language model such as SkipGram with negative sampling or hierarchical softmax techniques. Here we use SkipGram with negative sampling. The overall model is summarized in Algorithm 3. Line 1 and 2 use Algorithm 1 and Algorithm 2 to construct the multi-resolution multi-network relation network for each network object. Line 3 builds corpus that preserves the structural information of the relation network. Line 4 to line 7 are SkipGram model that learns the embeddings of nodes, subgraphs, and networks simultaneously.

Algorithm 3 MrMine

```
Input: Given a set of networks $\mathcal{G} = \{G_1, G_2, ..., G_k\}$, the height of
    WL subtrees $H$, the embedding dimension $p$, the window size
    $w_1$, $w_2$ for adding cross-network edges of subgraph vertices
    and SkipGram model respectively, and the threshold $\sigma$.
Output: The network embedding matrix $\mathbf{F_g}$ for $\mathcal{G}$, the subgraph
    embedding matrix $\mathbf{F_s}$, and the node embedding matrix $\mathbf{F_n}$.
1: Construct multi-resolution relation network $\mathcal{R}$ by Algorithm 1.
2: Update $\mathcal{R}$ by Algorithm 2.
3: Construct corpus $W$ by applying truncated random walk on
    each vertex in $\mathcal{R}$.
4: for vertex $u_i$ in each random walks $r \in W$ do
5:     $J(\Phi) = -logPr(u_j|\Phi(u_i)), u_j \in r[i - w_2, i + w_2]$
6:     $\Phi = \Phi - \alpha \frac{\partial J}{\partial \Phi}$
7: end for
8: Return embedding matrices $\mathbf{F_n}$, $\mathbf{F_s}$, $\mathbf{F_g}$.
```

3.4 Complexity Analysis on MrMine

In Algorithm 2, instead of comparing every pair of subgraphs for adding cross-network links for subgraph vertices (with time complexity $O(H^2n^2)$), line 1 generates a sorted list of subtrees with the summation of their corresponding degree sequence for line 3 to only compares the structural similarity of the target vertex with other vertices within the window of size w in the list. Since sorting the list V only cost $O(nlogn)$, if the window size w is bounded by $O(nlogn)$, this method can lower the time complexity of Algorithm 2 from $O(H^2n^2)$ to $O(Hnlog(n))$. Line 4 can use either Eq. (1) or (2).

In Algorithm 3, since the constructed CRCN relation network has at most $(Hk+k)n+k$ vertices (linear w.r.t. n), and all the steps in MrMine have linear complexity except using Algorithm 1 in line 2, the overall time complexity of MrMine is $O(Hnlog(n))$ (assuming that n and m have the same order of magnitude).

3.5 Accelerated MrMine+ Method

The base MrMine model introduces both cross-resolution and cross-network links in the CRCN relation network to capture the network objects' relation both across different layers of resolutions and different networks. To further reduce the time complexity of MrMine, we propose the improved model MrMine+. The most intuitive way is to simply remove the cross-network edges between subgraph vertices (Algorithm 2), which immediately reduces the time complexity from $O(Hnlogn)$ to $O(Hm+Hkn)$. However, this method would not only reduce the ability of preserving cross-network similarities of the CRCN relation network, but also might lead to disconnected CRCN relation network. Instead, we explore the hierarchical structure of WL subtrees, and propose a hierarchical CRCN (H-CRCN) relation network. The idea is to preserve the structural characteristics across networks from different subgraph granularities (e.g. Fig. 3). The advantages are two-fold: (1) it largely reduces the time complexity of CRCN relation network construction; and (2) still preserves the cross-network context information without explicitly generating cross-network links. The details are as follows.

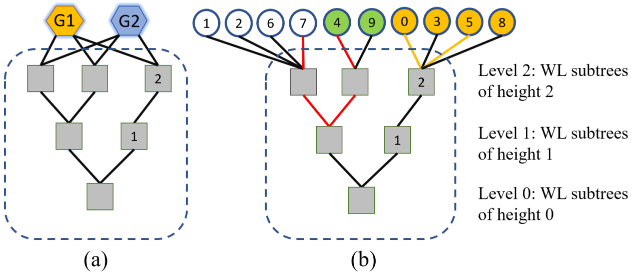

G1
G2
1
2
6
7
4
9
0
3
5
8
2
1
Level 2: WL subtrees of height 2
Level 1: WL subtrees of height 1
Level 0: WL subtrees of height 0
(a)
(b)

Figure 3: an example of H-CRCN relation network structure constructed from Figure 2. The reversed gray trees in the dashed rectangle are the hierarchical relation network of WL subtrees. (a) captures the relation of two networks (hexagons) with this hierarchical subgraph tree; (b) captures the relation of nodes (circles) with this hierarchical subgraph tree. Best viewed in color.

First, we construct a reversed hierarchical subgraph tree. The root of the tree is 0-level WL tree (i.e. node), and the vertices in level i represents the WL-subtrees of height i. Two vertices are connected if the WL subtree corresponding to a vertex from the higher level can be generated from a vertex's WL subtree of lower level. For example, in Figure 2, subtree 2 (level 2) can be generated by subtree 1 (level 1) by applying one more iteration of WL relabeling iteration, so subgraph vertex 1 and 2 are connected in Figure 3 (gray square 1 and 2). Next, network vertices or node vertices are connected to the last level of subgraph tree, based on the membership relation between the networks/nodes and the last level of WL subtrees. For example, in Figure 3 (b), node 0, 3, 5, 8 are connected to subtree vertex 2 because all these nodes can generate WL subtree 2 at level 2. The complete model is summarized in Algorithm 4. We use $\mathcal{E}_{R_s}$ to indicate edge set within the reversed relation tree of WL subtree vertices (e.g. edges within the dashed rectangle in Figure 3), $\mathcal{E}_{R_g}$ to

Algorithm 4 MrMine+

Input: Given $\mathcal{G} = \{G_1, G_2, ..., G_k\}$, the height of subgraph tree in H-CRCN relation network H, the embedding dimension p, the window size w_2 for SkipGram model.
Output: The network embedding matrix $\mathbf{F_g}$ for $\mathcal{G}$, the subgraph embedding matrix $\mathbf{F_s}$, and the node embedding matrix $\mathbf{F_n}$.

```
1:  for $G \in \mathcal{G}$ do
2:      Generate multi-set subtree labels $\mathcal{L}$ by $H$ WL iterations.
3:  end for
4:  Set $\mathcal{E}_{R_s} = \Phi, \mathcal{E}_{R_g} = \Phi, \mathcal{E}_{R_n} = \Phi$.
5:  for $(L_G^n)_i \in L_G$ do
6:      if $(p((L_G^n)_i), p((L_G^n)_i)) \notin \mathcal{E}_{R_s}$ then
7:          Add $(p((L_G^n)_i), p((L_G^n))_{i+1})$ to $\mathcal{E}_{R_s}$.
8:      end if
9:  end for
10: for vertex $v$ of level $H$ in $\mathcal{R}_s$ do
11:     Add $(v, p_g(G))$ to $\mathcal{E}_{R_g}$, if $q_g(v) \in G$.
12:     Add $(v, p_n(n))$ to $\mathcal{E}_{R_n}$, if $q_g(v) \in L^n$.
13: end for
14: Construct subtree vertices' corpus $P_s$ with $\mathcal{E}_{R_s}$.
15: Construct network vertices' corpus $P_g$ with $\mathcal{E}_{R_s} \cup \mathcal{E}_{R_g}$.
16: Construct node vertices' corpus $P_n$ with $\mathcal{E}_{R_s} \cup \mathcal{E}_{R_n}$.
17: for vertex $u_i$ in random walk $r \in P_g \cup P_n \cup P_s$ do
18:     $J(\Phi) = -logPr(u_j|\Phi(u_i)), u_j \in r[i - w_2, i + w_2]$
19:     $\Phi = \Phi - \alpha \frac{\partial J}{\partial \Phi}$
20: end for
21: Return embedding matrices $\mathbf{F_n}$, $\mathbf{F_s}$, $\mathbf{F_g}$.
```

indicate edges between network vertices and the highest level of WL subtree vertices (e.g. edges between hexagons and squares in Figure 3 (a)), and $\mathcal{E}_{R_n}$ to indicate edges between node vertices and the highest level of WL subtree vertices (e.g. edges between circles and squares in Figure 3 (b)).

With such a way of constructing H-CRCN relation network, similarities of network objects are captured in different subgraph granularities. For example, in Figure 3, node 0 and 5 are connected by the yellow short path while node 4 and 7 are connected by the red long path, which indicates that node 0 and 5 are closer (both connected to a subgraph of finer granularity).

In Algorithm 4, the $\mathcal{E}_{R_s}$ is constructed from line 5 to line 9. $\mathcal{E}_{R_g}$ and $\mathcal{E}_{R_n}$ are constructed from line 10 to line 13. Note that $\mathcal{E}_{R_g}$ and $\mathcal{E}_{R_n}$ are generated independently by the edges of (subgraph vertex, network vertex) and the edges of (subgraph vertex, network vertex) respectively. From line 14 to 16, we apply truncated random walks for building the corpus P_s, P_g and P_n with only $\mathcal{E}_{R_s}$, $\mathcal{E}_{R_s} \cup \mathcal{E}_{R_n}$ and $\mathcal{E}_{R_s} \cup \mathcal{E}_{R_n}$, respectively. The SkipGram model is applied on the union of P_g, P_n and P_s from line 17 to line 21.

3.6 Complexity Analysis on MrMine+

H iterations of WL relabeling processes cost $O(Hm)$. From line 5 to line 9, with the absence of cross-network edges between subgraph vertices, only subgraph vertices need to be traversed once, with the complexity of $O(Hkn)$. Since there are $O(n)$ WL subtrees in level H of $\mathcal{R}_s$, the complexity of generating $\mathcal{E}_{R_n}$ and $\mathcal{E}_{R_g}$ are $O(kn)$ and $O(n)$, respectively. There are at most $(Hk+1)n$ vertices in the CRCN relation network. Overall, the time complexity of MrMine+ is $O(Hm+cn)$, where $c = [H(k+1)+1]lr$ (l is the length of truncated random walks, and r is the number of walks sampled per vertex) is a constant much smaller than m, n.

4 EXPERIMENTAL RESULTS

In this section, we present the experimental results with extensive datasets and baseline methods, to evaluate the effectiveness of handling multi-network mining tasks, and the scalability of the proposed algorithms (MrMine and MrMine+).

4.1 Experimental Setup

Our proposed method is evaluated mainly on seven real-world datasets, which are summarized in Table 2. The brief description of each dataset and the experimental setup are presented as follows.

Table 2: Datasets Summary

Dataset Name	Category	# of Nodes	# of Edges
DBLP	Co-authorship	1.013	3,022
Flickr	User relationship	3,911	4,152
LastFm	User relationship	4,068	4,347
Douban	User relationship	1,118	3,022
MySpace	Social network	6,362	6,514
Aminer	Academic network	1,274,360	4,756,194
Bioinformatics	**Size (# of graphs)**	**Classes**	**Avg. nodes**
MUTAG	188	2	17.9
PTC	344	2	25.5
PROTEINS	1113	2	39.1
NCI1	4110	2	29.8
NCI109	4127	2	29.6

- *DBLP:* A co-authorship network with nodes representing authors and links representing co-authorship. The original dataset contains 42,252 nodes and 210,320 edges [17].
- *Flickr:* A network of friends on the image and video hosting website *Flickr* with each node representing a user and each edge reflecting friend relationship. It has 215,495 individual users and 9,114,557 friend relationships [30].
- *LastFm:* Collected in 2013, this is the following network of users on the music website *LastFm*. The network network contains 136,420 users and 1,685,524 following links [30].
- *Douban:* Collected in 2010, this data reflects the users' friend relationship in the offline and online activities of *Douban*, and contains 50k users and 5M edges. The offline and online activity communities share some overlaps of users, which makes it suitable for network alignment.
- *MySpace:* A social network which has a strong music emphasis. The links between nodes reflect the connections of users. It has 854,498 users and 6,489,736 relationship links.
- *AMiner:* An academic social network. Undirected edges represent co-authorship relationship [30]. The whole dataset contains 1,274,360 nodes and 4,756,194 edges.
- *Bioinformatics:* The bioinformatics dataset, including MUTAG, PTC, PROTEINS, etc. are small-scaled networks of chemical compound, proteins or enzymes, and are often used as benchmark datasets for graph classification.

Using the above datasets, we design the following five experimental scenarios for evaluating the effectiveness of our method.

S1. DBLP vs. Noisy DBLP Alignment. We extract a subnetwork with 1,013 nodes from the original *DBLP* dataset, and randomly add extra edges to the network to generate the second noisy network while keeping the node set unchanged. Note that our setting is different from that in [28] for that our noisy edges make two networks non-isomorphic while [28] only changes the edge weight.
S2. Douban-offline vs. Douban-online Alignment. We adopt a method introduced in [31] to construct the offline network according to users' co-occurrence in social gatherings. We treat people as contacts if they participate in the same offline events more than ten times. The constructed offline network has 1,118 users. We use the corresponding online social network for the same 1,118 users as the second network, and add extra nodes as noises in the alignment experiment.
S3. Cross-network Query Node Retrieval. As an complementary experiment for network alignment, we conduct cross-network query node retrieval, in which given a set of query nodes from one network, we aim to retrieve similar nodes from another network. We study the following network pairs and use their overlapping user set as groundtruth. DBLP and noisy *DBLP*, *Douban-offline* and *Douban-online*, *Flickr* and *LastFm*, *MySpace* and noisy *MySpace* with both node and edge noises.
S4. Collective Network Alignment. To further demonstrate the effectiveness of our method on multi-network mining, we adopt a novel colletive network alignment. Rather than conducting traditional two-network alignment, we use three input networks to collectively align nodes of three networks. To the best of our knowledge, we are the first to align more than two networks collectively. We use *Douban-offline* and *Douban-online* for this experimental scenario. More details will be elaborated in the next sub-section.
S5. Network Level Classification. We use bioinformatics dataset for graph level classification as shown in [13]. Since our method can learn network, subgraph and node embeddings simultaneously, we can either use network embeddings for classification directly or use node and subgraph embeddings combinatorially as shown in [25]. The results present the performance of the first one since it has better performance out of the two options. After we learn the embeddings, we use 80% of the bioinformatics networks for training and 20% for testing with a linear SVM model.

In all the above scenarios, we do not use network node/edge attributes as auxiliary information in the experiments.

Comparison methods. In total, we use nine comparison methods in our experiments. For (collective) network alignment and query node retrieval experiments, we use three traditional network alignment/matching methods (*FINAL* [28], *IsoRank* [20], and *UniAlign* [9]), and three network embedding methods (*deepwalk* [16], *node2vec* [5], and *struc2vec* [18]). For network classification experiment, we use three baselines including both traditional Weisfeiler-Lehman kernel method (*WL kernel* [19]) and two embedding-based methods (*Deep Graph Kernel* [25], and *subgraph2vec* [13]).

Repeatability. All of the datasets are public. All experiments are performed on a server with Intel(R) Xeon(R) CPU core with 2.00 GHz and 1.51 TB RAM. The operating system is Red Hat Enterprise Linux Server release 6.9. The algorithms are programmed with Python. The hyperparameters are set based on a grid search. We intend to release the source code after the paper is published.

4.2 Effectiveness

Visualization. To evaluate the effectiveness of the proposed method, we first use a widely used and small-scaled dataset, the Zachary's Karate Club [18] dataset for visualizing the embeddings in 2-d space for intuitively presenting the difference between our method and baseline embedding methods. We use two identical Karate Club networks in which the second one is permutated from the original network as shown in Figure 5. We apply three baseline embedding methods, *deepwalk*, *node2vec*, and *struc2vec* as well as MrMine on this dataset. Since all baseline methods only support single network embedding, we fit two networks as one single network into these models. The results are shown in Figure 4. As we can see, the embeddings of *deepwalk* clutter in four positions. Nodes from two networks are mixed together, and can not be distinguished. *node2vec* roughly embeds the nodes into two clusters by nodes' membership, but the nodes within each cluster are mixed up. Also nodes from different networks are incomparable. *struc2vec* embeds nodes based on structural roles, but it can not explicitly differentiate all roles as some nodes with different colors are embedded together, while some nodes with the same color are far apart. Our method pairs almost all nodes with the same colors together, and clusters structurally identical nodes (e.g. node 17, 19, 52, 54 with lime color). This visualization result matches our intuition that our model can preserve structural similarities of nodes across networks and shows great potential in many mining tasks as we will present shortly.

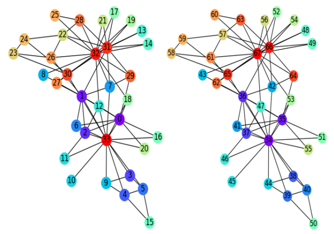
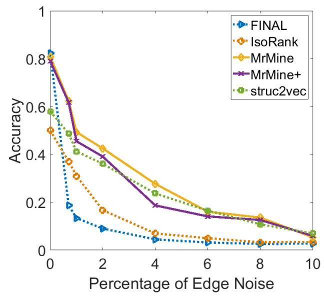
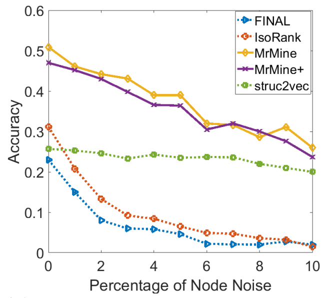


Figure 5: Original karate network and permutated karate network. Colors are set the same for identical nodes across two networks.



(a) Alignment result on original *DBLP* and noisy *DBLP* dataset.



(b) Alignment result on *Douban* offline and online network

Figure 6: Network alignment result of our methods compared with traditional alignment baseline methods and network embedding baseline methods. Best viewed in color.

Network Alignment. Next we perform two network alignment experiments (scenario *S1, S2*). For traditional network alignment baseline methods (e.g. *FINAL*, *IsoRank*), we calculate the cross-network similarity matrix and apply greedy match algorithm [28] to process the similarity matrix for one-to-one node alignment. For embedding methods, we first generate node embeddings, and calculate the

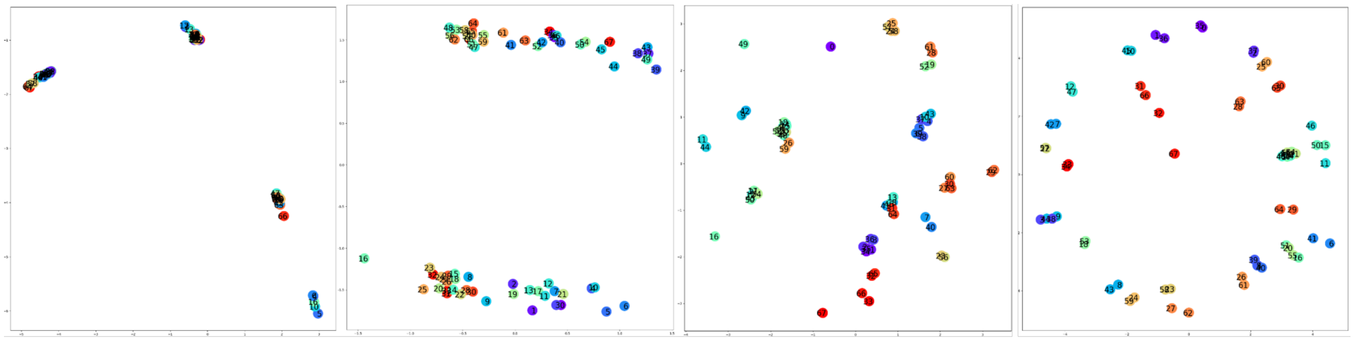
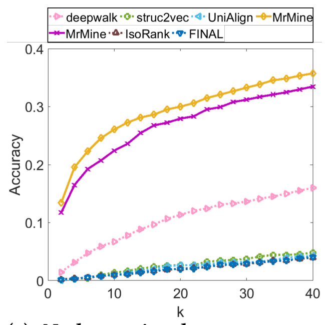
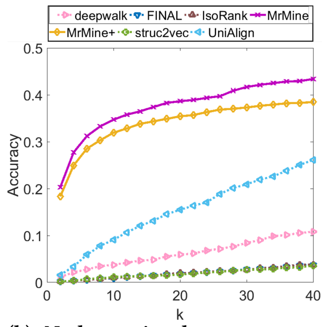

(a) *deepwalk* (b) *node2vec* (c) *struc2vec* (d) *MrMine*

Figure 4: 2-d visualization of embeddings learned on Zachary's Karate Club dataset by MrMine and three existing methods. The learned embeddings are 32-d and transformed to 2-d by Multi-Dimensional Scaling (MDS) to preserve the embedding distance. Nodes of same color represent that they are identical. Best viewed in color.

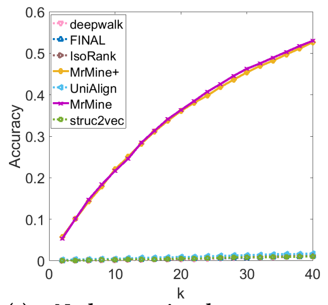
similarity matrix by the inner product of embedding vectors. The greedy match algorithm is then applied on the similarity matrix. The results are shown in Figure 6. As the percentage of edge/node noises increases, the accuracies of all methods decrease.



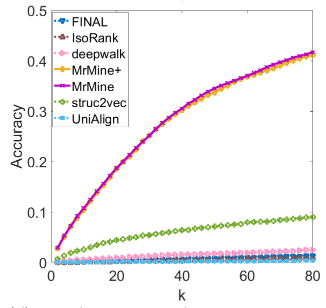
(a) Node retrieval accuracy on *DBLP* and noisy *DBLP* dataset. Higher is better



(b) Node retrieval accuracy on *Douban-online* and *Douban-offline* dataset. Higher is better.



(c) Node retrieval accuracy on *Flickr* and *LastFm* dataset. Higher is better.



(d) Node retrieval accuracy on *MySpace* dataset. Higher is better.

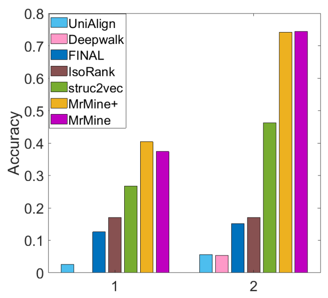
Figure 7: Node retrieval results on our methods with five baseline methods. Accuracy vs. top-k retrieved nodes. Best viewed in color.

On both datasets, our proposed methods MrMine and MrMine+ outperform all baselines. MrMine slightly outperforming MrMine+ indicates the usefulness of cross-network links in the CRCN relation network. Specifically, on *DBLP* dataset, our methods achieve close accuracy to *FINAL* when there is no edge noise, but *FINAL* is sensitive to edge noises and decreases rapidly on small percentage of extra edges. *struc2vec* also achieves close performance to MrMine+. On *Douban* dataset, our methods outperforms baselines at all node noise level by at most 19.71%.

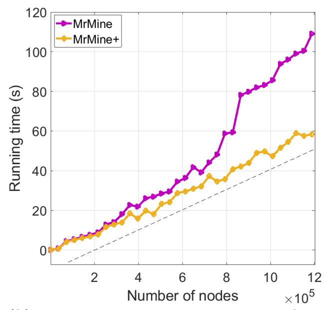
Query Node Retrieval. We then conduct query node retrieval experiments (i.e. scenario *S3*). We treat the nodes in one network as queries and the nodes in the other network as targets. For traditional methods (i.e. *FINAL*, *IsoRank* and *UniAlign*), we calculate the cross-network similarity matrix and sort the nodes in the targets based on their similarity values with query nodes. After sorting, top-k nodes are retrieved from targets for each query node. If the matching node exists in the top-k list, we consider it as one hit. The accuracy is calculated as (# of hits)/(# of query nodes). We calculate the accuracy of retrieval w.r.t. the k value and present the results in Figure 7. For (a) and (d) in Figure 7, we randomly add 2% edge noises and 3% node noises. We can observe that our proposed method, MrMine and MrMine+, outperform all baselines including both traditional network alignment/matching methods and network embedding methods on all four datasets.

Collective Network Alignment. We further conduct a novel collective network alignment experiment (scenario *S4*) to show the advantage of our methods on enabling difficult multi-network mining task. We collectively perform network alignment among all nodes in three networks (G_1: *Douban-offline*, G_2: *Douban-online*, and G_3: *Douban-online* with 3% edge and 5% node noises). Since traditional network alignment/matching methods only calculate similarity matrix between two networks, we need to calculate S_{12} for G_1, G_2, S_{13} for G_1, G_3, and S_{23} for G_2, G_3. The three-way similarity between G_1, G_2 and G_3 forms a 3-d similarity tensor S, with $S(i,j,k) = S_{12}(i,j) + S_{13}(i,k) + S_{23}(j,k)$. We implement a 3-d greedy match algorithm to produce a one-to-one-to-one alignment for nodes in three networks. For our method, since we simultaneously embed all nodes onto the same embedding space, we can directly calculate the similarity tensor S and apply 3-d greedy match algorithm. We use two metrics for this experiment. First, for each pair of three-node alignment (including three nodes from G_1, G_2 and G_3), we consider it a successful alignment if all nodes are aligned correctly (a strict metric, indicating the correct 3-way alignment).

Second, for each pair of three-node alignment, we consider it a successful alignment if two of the three nodes are aligned correctly (a relaxed metric, indicating at least 2-way correct alignment). The results are presented in Figure 8 (a). As we can see, our proposed methods MrMine and MrMine+ outperforms all baselines by both metrics. The largest accuracy improvement by the strict metric is 14.33%, and the largest accuracy improvement by the relaxed metric is 28.20%.



(a) 1 shows results by the first (strict) metric and 2 shows results by the second (relaxed) metric. Higher is better.



(b) Running time vs. number of nodes (SkipGram time is not included) of MrMine and MrMine+.

Figure 8: Collective alignment results on *Douban* offline and online dataset (a), and scalability study results on MrMine and MrMine+ (b). Best viewed in color.

Network level classification Lastly, we test the performance of the embeddings learned by our proposed methods on network classification. We use all five categories of datasets from bioinformatics dataset, and the results are presented in Table 3. The bold numbers show the best performance. Note that different from [13, 25], we do not use node or edge attributes in the embedding learning process. From the results, our methods outperform all baselines in four out of five categories of bioinformatics datasets, and perform very close to the best baseline in the proteins data, which indicates the the proposed methods' effectiveness of enhancing network level classification tasks. We also observe that MrMine+ performs consistently better than MrMine in this classification experiment, which denotes that the cross-network relation captured by H-CRCN relation network is more effective than the basic CRCN relation network in network classification task.

4.3 Scalability

We conduct scalability study of the proposed algorithms on the largest dataset *Aminer* which contains over 1M nodes. We use two subgraphs of *Aminer* as input networks, and conduct a series of running time test with the number of nodes ranging from 1,000 to 1.2M (which is close to the size of the entire *Aminer* dataset). Figure 8 (b) shows the results of the average running time on 5 runs. We can observe that the running time of both algorithms are less than 120s when applied on 1.2M-node networks. Particularly MrMine+ scales linearly while MrMine scales faster than (super-linearly) MrMine+ which is consistent with our analysis on the time complexity. Thus, the proposed methods can be applied to large networks.

Table 3: Comparison of Network classification Accuracy on Bioinformatics Datasets (± standard derivation)

	MUTAG	PTC	PROTEINS	NCI1	NCI109
MrMine+	**83.47 ± 2.01**	**62.00 ± 0.07**	71.22 ± 0.62	**68.50 ± 0.03**	**65.57 ± 0.02**
MrMine	82.19 ± 1.58	55.41 ± 2.52	70.88 ± 0.38	66.90 ± 0.05	64.53 ± 0.01
WL Kernel	80.66 ± 3.07	59.94 ± 2.79	64.45 ± 1.14	63.42 ± 0.22	62.94 ± 0.42
Deep WL Kernel	82.95 ± 1.96	53.29 ± 1.53	69.49 ± 0.26	62.83 ± 0.25	62.47 ± 0.15
subgraph2vec	79.33 ± 0.07	42.29 ± 0.09	**73.04 ± 0.04**	63.01 ± 0.01	49.20 ± 0.02

5 RELATED WORK

Massive studies haven been done recently on network embedding field. Traditional network embedding methods such as *deepwalk* [16], *node2vec* [5] and *LINE* [21] are some of the earliest works that propose to embed graph nodes into low-dimensional vectors by language model and using truncated random walks for node context collection. Proposed more recently, the structural role based node embedding methods such as *struc2vec* [18], *SDNE* [24], and *GraphWave* [2] are more related to our proposed method. Specifically, *struc2vec* explores the node structural proximity on single network and re-construct the original network by node structural score to use random walk based language model. *SDNE* applies the deep autoencoder to capture the non-linear vertex structural characteristics, while *GraphWave* unsupervisedly learns node embeddings by leveraging heat wavelet diffusion patterns.

Recently, the network embedding work becomes more diversified and focuses more on complex scenarios. A very recent work, *DMNE* [15] by J Ni et al. proposes to form many-to-many node mappings by coordinating multiple neural networks with a co-regularized loss function. But the method requires known links across networks. *NEST* [26] proposed by C Yang et al. studies a problem of hierarchical network embedding that combines motif filtering and convolutional neural networks. Its multi-resolution setting is close to our problem, yet only focuses on single network. *LinkNBed* by R Trivedi et al. [22] learns entity and relationship representations across multiple knowledge graphs. This work also has the potential to be applied to network alignment under method modification. *LATTE* by J Zhang et al. [27] proposes a task-oriented network embedding framework which uses diffusive proximity scores and learns the network representation vectors by extending the autoencoder model. A closely related work on multi-layered network embedding [11] by J Li et al. focuses on an inter-dependent multi-layered network where layers has dependencies on each other. For representation learning on different network resolutions, most methods are node embedding methods including the above ones. Inspired by the paragraph representation work [10] by Q Le, et al. and *deepwalk*, *graph2vec* [14] and *subgraph2vec* [13] by A Narayanan et al. provide graph-level and subgraph-level embedding methods, respectively. The Deep Graph Kernel method [25] by P Yanardag et al. also studies the subgraph embedding for the use in graph kernel computation. Embedding methods that also use the idea of Weisfeiler-Lehman subtree kernel/isomorphism test algorithm include *GraphSAGE* [6] by W Hamilton et al. and *GCN* [8] by Thomas N. Kipf et al.. Other network embeddings methods in the literature of recent years can be found in this survey [1].

The aforementioned works are all related to network embedding. Meanwhile, many traditional network mining tasks have been developed innovative methods including the multi-network mining tasks studied in this paper, such as attributed network alignment task [3, 28, 29], attributed subgraph matching task [4], and graph kernel [19]. Besides embedding methods, *ROLX* [7] by K Henderson et al. which uses matrix factorization method to capture vertex structural characteristics also shows competitive performance in many comparison tasks with network embedding methods.

6 CONCLUSION

In this paper, we study the multi-resolution multi-network embedding problem and develop efficient and effective algorithms (MrMine and MrMine+) to simultaneously learn the embeddings of multi-resolution multi-network objects in the same space. Specifically, we propose that the context of such objects can be captured by the Cross-Resolution Cross-Network (CRCN) relation network. We then propose a basic algorithm (MrMine) to construct such network, and an accelerated algorithm (MrMine+) which explores a more complex structure of the CRCN relation network (i.e. H-CRCN relation network) for reducing the time complexity while preserving cross-network context. Numerous experiments on real-world data show that (1) our methods lead up to 19.71%, 28.20%, and 5.08% increase of accuracy in traditional network alignment, collective network alignment, and network classification, respectively; (2) our method can scale up to over 1M-node networks. In the future, we will (1) explore the approaches to expand our method for more challenging multi-network mining tasks, such as subgraph matching, and (2) generalize it to handle different categories of network attributes.

7 ACKNOWLEDGEMENTS

This material is supported by the National Science Foundation under Grant No. IIS-1651203, IIS-1715385, IIS-1743040, and CNS-1629888, by DTRA under the grant number HDTRA1-16-0017, by the United States Air Force and DARPA under contract number FA8750-17-C-0153 , by the U.S. Department of Homeland Security under Grant Award Number 2017-ST-061-QA0001 and by Army Research Office under the contract number W911NF-16-1-0168. 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.

REFERENCES

[1] Peng Cui, Xiao Wang, Jian Pei, and Wenwu Zhu. 2018. A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering* (2018).

[2] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning Structural Node Embeddings via Diffusion Wavelets. (2018).

[3] Boxin Du and Hanghang Tong. 2018. FASTEN: Fast Sylvester Equation Solver for Graph Mining. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 1339–1347.

[4] Boxin Du, Si Zhang, Nan Cao, and Hanghang Tong. 2017. First: Fast interactive attributed subgraph matching. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1447–1456.

[5] 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*. ACM, 855–864.

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[7] 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 *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1231–1239.

[8] Thomas N Kipf and Max Welling. 2016. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907* (2016).

[9] Danai Koutra, Hanghang Tong, and David Lubensky. 2013. Big-align: Fast bipartite graph alignment. In *2013 IEEE 13th International Conference on Data Mining*. IEEE, 389–398.

[10] Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*. 1188–1196.

[11] Jundong Li, Chen Chen, Hanghang Tong, and Huan Liu. 2018. Multi-layered network embedding. In *Proceedings of the 2018 SIAM International Conference on Data Mining*. SIAM, 684–692.

[12] Liangyue Li, Hanghang Tong, Nan Cao, Kate Ehrlich, Yu-Ru Lin, and Norbou Buchler. 2015. Replacing the irreplaceable: Fast algorithms for team member recommendation. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.

[13] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. 2016. subgraph2vec: Learning Distributed Representations of Rooted Sub-graphs from Large Graphs. (2016).

[14] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. 2017. graph2vec: Learning Distributed Representations of Graphs. *13th International Workshop on Mining and Learning with Graphs* (2017).

[15] Jingchao Ni, Shiyu Chang, Xiao Liu, Wei Cheng, Haifeng Chen, Dongkuan Xu, and Xiang Zhang. 2018. Co-Regularized Deep Multi-Network Embedding. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 469–478.

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

[17] Adriana Prado, Marc Plantevit, Céline Robardet, and Jean-Francois Boulicaut. 2013. Mining graph topological patterns: Finding covariations among vertex descriptors. *IEEE Transactions on Knowledge and Data Engineering* 25, 9 (2013), 2090–2104.

[18] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 385–394.

[19] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research* 12, Sep (2011), 2539–2561.

[20] Rohit Singh, Jinbo Xu, and Bonnie Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proceedings of the National Academy of Sciences* (2008).

[21] 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*. International World Wide Web Conferences Steering Committee, 1067–1077.

[22] Rakshit Trivedi, Bunyamin Sisman, Jun Ma, Christos Faloutsos, Hongyuan Zha, and Xin Luna Dong. 2018. LinkNBed: Multi-Graph Representation Learning with Entity Linkage. (2018).

[23] S Vichy N Vishwanathan, Nicol N Schraudolph, Risi Kondor, and Karsten M Borgwardt. 2010. Graph kernels. *Journal of Machine Learning Research* 11, Apr (2010), 1201–1242.

[24] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1225–1234.

[25] Pinar Yanardag and SVN Vishwanathan. 2015. Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1365–1374.

[26] Carl Yang, Mengxiong Liu, Vincent W. Zheng, and Jiawei Han. 2018. Node, Motif and Subgraph: Leveraging Network Functional Blocks Through Structural Convolution. 47–52.

[27] Jiawei Zhang, Limeng Cui, and Yanjie Fu. 2017. LATTE: Application Oriented Network Embedding. (2017).

[28] Si Zhang and Hanghang Tong. 2016. Final: Fast attributed network alignment. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1345–1354.

[29] Si Zhang, Hanghang Tong, Jie Tang, Jiejun Xu, and Wei Fan. 2017. ineat: Incomplete network alignment. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 1189–1194.

[30] Yutao Zhang, Jie Tang, Zhilin Yang, Jian Pei, and Philip S Yu. 2015. Cosnet: Connecting heterogeneous social networks with local and global consistency. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 1485–1494.

[31] Erheng Zhong, Wei Fan, Junwei Wang, Lei Xiao, and Yong Li. 2012. Comsoc: adaptive transfer of user behaviors over composite social network. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 696–704.