struc2vec: Learning Node Representations from Structural Identity

Leonardo F. R. Ribeiro
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
leo@land.ufrj.br

Pedro H. P. Saverese
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
savarese@land.ufrj.br

Daniel R. Figueiredo
Federal University of Rio de Janeiro
Systems Eng. and Comp. Science Dep.
daniel@land.ufrj.br

ABSTRACT

Structural identity is a concept of symmetry in which network nodes are identified according to the network structure and their relationship to other nodes. Structural identity has been studied in theory and practice over the past decades, but only recently has it been addressed with representational learning techniques. This work presents *struc2vec*, a novel and flexible framework for learning latent representations for the structural identity of nodes. *struc2vec* uses a hierarchy to measure node similarity at different scales, and constructs a multilayer graph to encode structural similarities and generate structural context for nodes. Numerical experiments indicate that state-of-the-art techniques for learning node representations fail in capturing stronger notions of structural identity, while *struc2vec* exhibits much superior performance in this task, as it overcomes limitations of prior approaches. As a consequence, numerical experiments indicate that *struc2vec* improves performance on classification tasks that depend more on structural identity.

CCS CONCEPTS

•**Computing methodologies** → **Unsupervised learning;** *Learning latent representations;* •**Artificial Intelligence** → Learning;

KEYWORDS

feature learning; node embeddings; structural identity

1 INTRODUCTION

In almost all networks, nodes tend to have one or more functions that greatly determine their role in the system. For example, individuals in a social network have a social role or social position [11, 19], while proteins in a protein-protein interaction (PPI) network exert specific functions [1, 22]. Intuitively, different nodes in such networks may perform similar functions, such as interns in the social network of a corporation or catalysts in the PPI network of a cell. Thus, nodes can often be partitioned into equivalent classes with respect to their function in the network.

Although identification of such functions often leverage node and edge attributes, a more challenging and interesting scenario emerges when node function is defined solely by the network structure. In this context, not even the labels of the nodes matter but just their relationship to other nodes (edges). Indeed, mathematical sociologists have worked on this problem since the 1970s, defining and computing *structural identity* of individuals in social networks [11, 17, 19]. Beyond sociology, the role of webpages in the webgraph is another example of identity (in this case, hubs and authorities) emerging from the network structure, as defined by the celebrated work of Kleinberg [8].

Figure 1: An example of two nodes (u and v) that are structurally similar (degrees 5 and 4, connected to 3 and 2 triangles, connected to the rest of the network by two nodes), but very far apart in the network.

The most common practical approaches to determine the structural identity of nodes are based on distances or recursions. In the former, a distance function that leverages the neighborhood of the nodes is used to measure the distance between all node pairs, with clustering or matching then performed to place nodes into equivalent classes [5, 9]. In the later, a recursion with respect to neighboring nodes is constructed and then iteratively unfolded until convergence, with final values used to determine the equivalent classes [3, 8, 26]. While such approaches have advantages and disadvantages, we provide an alternative methodology, one based on unsupervised learning of representations for the structural identity of nodes (to be presented). Recent efforts in learning latent representations for nodes in networks have been quite successful in performing classification and prediction tasks [6, 14, 16, 23]. In particular, these efforts encode nodes using as context a generalized notion of their neighborhood (e.g., w steps of a random walk, or nodes with neighbors in common). In a nutshell, nodes that have neighborhoods with similar sets of nodes should have similar latent representations. But neighborhood is a local concept defined by some notion of proximity in the network. Thus, two nodes with neighborhoods that are structurally similar but that are far apart will not have similar latent representations. Figure 1 illustrates the problem, where nodes u and v play similar roles (i.e., have similar local structures) but are very far apart in the network. Since their neighborhoods have no common nodes, recent approaches cannot capture their structural similarity (as we soon show).

KDD '17, August 13-17, 2017, Halifax, NS, Canada

DOI: 10.1145/3097983.3098061

It is worth noting why recent approaches for learning node representations such as *DeepWalk* [16] and *node2vec* [6] succeed in classification tasks but tend to fail in structural equivalence tasks. The key point is that many node features in most real networks exhibit a strong homophily (e.g., two blogs with the same political inclination are much more likely to be connected than at random). Neighbors of nodes with a given feature are more likely to have the same feature. Thus, nodes that are "close" in the network and in the latent representation will tend to share features. Likewise, two nodes that are "far" in the network will tend to be separated in the latent representation, independent of their local structure. Thus, structural equivalence will not properly be captured in the latent representation. However, if classification is performed on features that depend more on structural identity and less on homophily, then such recent approaches are likely to be outperformed by latent representations that better capture structural equivalence (as we soon show).

Our main contribution is a flexible framework for learning latent representations for the structural identity of nodes, called *struc2vec*. It is an alternative and powerful tool to the study of structural identity through latent representations. The key ideas of *struc2vec* are:

- Assess structural similarity between nodes independently of node and edge attributes as well as their position in the network. Thus, two nodes that have a similar local structure will be considered so, independent of network position and node labels in their neighborhoods. Our approach also does not require the network to be connected, and identifies structurally similar nodes in different connected components.
- Establish a hierarchy to measure structural similarity, allowing progressively more stringent notions of what it means to be structurally similar. In particular, at the bottom of the hierarchy, structural similarity between nodes depend only on their degrees, while at the top of the hierarchy similarity depends on the entire network (from the viewpoint of the node).
- Generates random *contexts* for nodes, which are sequences of structurally similar nodes as observed by a weighted random walk traversing a multilayer graph (and *not* the original network). Thus, two nodes that frequently appear with similar contexts will likely have similar structure. Such context can be leveraged by language models to learn latent representation for the nodes.

We implement an instance of *struc2vec* and show its potential through numerical experiments on toy examples and real networks, comparing its performance with *DeepWalk* [16] and *node2vec* [6] – two state-of-the-art techniques for learning latent representations for nodes, and with *RolX* [7] – a recent approach to identify roles of nodes. Our results indicate that while *DeepWalk* and *node2vec* fail to capture the notion of structural identity, *struc2vec* excels on this task – even when the original network is subject to strong random noise (random edge removal). We also show that *struc2vec* is superior in a classification task where node labels depends more on structural identity (i.e., air-traffic networks with labels representing airport activity).

The remainder of this paper is organized as follows. Section 2 briefly overviews the recent related work on learning latent representations of nodes in networks. Section 3 presents the *struc2vec* framework in detail. Experimental evaluation and comparison to other methods are shown in Section 4. Finally, Section 5 concludes the paper with a brief discussion.

2 RELATED WORK

Embedding network nodes in (Euclidean) space has received much attention over the past decades from different communities. The technique is instrumental for Machine Learning applications that leverage network data, as node embeddings can be directly used in tasks such as classification and clustering.

In Natural Language Processing [2], generating dense embeddings for sparse data has a long history. Recently, Skip-Gram [12, 13] was proposed as an efficient technique to learn embeddings for text data (e.g., sentences). Among other properties, the learned language model places semantically similar words near each other in space.

Learning a language model from a network was first proposed by *DeepWalk* [16]. It uses random walks to generate sequences of nodes from the network, which are then treated as sentences by Skip-Gram. Intuitively, nodes close in the network will tend to have similar contexts (sequences) and thus have embeddings that are near one another. This idea was later extended by *node2vec* [6]. By proposing a biased second order random walk model, *node2vec* provides more flexibility when generating the context of a vertex. In particular, the edge weights driving the biased random walks can be designed in an attempt to capture both vertex homophily and structural equivalence. However, a fundamental limitation is that structurally similar nodes will never share the same context if their distance (hop count) is larger than the Skip-Gram window.

subgraph2vec [14] is another recent approach for learning embeddings for rooted subgraphs, and unlike the previous techniques it does not use random walks to generate context. Alternatively, the context of a node is simply defined by its neighbors. Additionally, *subgraph2vec* captures structural equivalence by embedding nodes with the same local structure to the same point in space. Nonetheless, the notion of structural equivalence is very rigid since it is defined as a binary property dictated by the Weisfeiler-Lehman isomorphism test [21]. Thus, two nodes that are structurally very similar (but fail the test) and have non-overlapping neighbors may not be close in space.

Similarly to *subgraph2vec*, considerable effort has recently been made on learning richer representations for network nodes [4, 24]. However, building representations that explicitly capture structural identity is a relative orthogonal problem that has not received much attention. This is the focus of *struc2vec*.

A recent approach to explicitly identify the role of nodes using just the network structure is *RolX* [7]. This unsupervised approach is based on enumerating various structural features for nodes, finding the more suited basis vector for this joint feature space, and then assigning for every node a distribution over the identified roles (basis), allowing for mixed membership across the roles. Without explicitly considering node similarity or node context (in terms of

structure), *RolX* is likely to miss node pairs that are structurally equivalent (to be shown).

3 *STRUC2VEC*

Consider the problem of learning representations that capture the structural identity of nodes in the network. A successful approach should exhibit two desired properties:

- The distance between the latent representation of nodes should be strongly correlated to their structural similarity. Thus, two nodes that have identical local network structures should have the same latent representation, while nodes with different structural identities should be far apart.
- The latent representation should not depend on any node or edge attribute, including the node labels. Thus, structurally similar nodes should have close latent representation, independent of node and edge attributes in their neighborhood. The structural identity of nodes must be independent of its "position" in the network.

Given these two properties, we propose *struct2vec*, a general framework for learning latent representations for nodes composed of four main steps, as follows:

(1) Determine the structural similarity between each vertex pair in the graph for different neighborhood sizes. This induces a hierarchy in the measure for structural similarity between nodes, providing more information to assess structural similarity at each level of the hierarchy.
(2) Construct a weighted multilayer graph where all nodes in the network are present in every layer, and each layer corresponds to a level of the hierarchy in measuring structural similarity. Moreover, edge weights among every node pair within each layer are inversely proportional to their structural similarity.
(3) Use the multilayer graph to generate context for each node. In particular, a biased random walk on the multilayer graph is used to generate node sequences. These sequences are likely to include nodes that are more structurally similar.
(4) Apply a technique to learn latent representation from a context given by the sequence of nodes, for example, Skip-Gram.

Note that *struct2vec* is quite flexible as it does not mandates any particular structural similarity measure or representational learning framework. In what follows, we explain in detail each step of *struct2vec* and provide a rigorous approach to a hierarchical measure of structural similarity.

3.1 Measuring structural similarity

The first step of *struct2vec* is to determine a structural similarity between two nodes without using any node or edge attributes. Moreover, this similarity metric should be hierarchical and cope with increasing neighborhood sizes, capturing more refined notions of structural similarity. Intuitively, two nodes that have the same degree are structurally similar, but if their neighbors also have the same degree, then they are even more structurally similar.

Let $G = (V, E)$ denote the undirected, unweighted network under consideration with vertex set V and edge set E, where $n = |V|$ denotes the number of nodes in the network and k^* its diameter. Let $R_k(u)$ denote the set of nodes at distance (hop count) exactly $k \geq 0$ from u in G. Note that $R_1(u)$ denotes the set of neighbors of u and in general, $R_k(u)$ denotes the ring of nodes at distance k. Let $s(S)$ denote the ordered degree sequence of a set $S \subset V$ of nodes.

By comparing the ordered degree sequences of the rings at distance k from both u and v we can impose a hierarchy to measure structural similarity. In particular, let $f_k(u, v)$ denote the *structural distance* between u and v when considering their k-hop neighborhoods (all nodes at distance less than or equal to k and all edges among them). In particular, we define:

$$f_k(u, v) = f_{k-1}(u, v) + g(s(R_k(u)), s(R_k(v))), \quad k \geq 0 \text{ and } |R_k(u)|, |R_k(v)| > 0 \tag{1}$$

where $g(D_1, D_2) \geq 0$ measures the distance between the ordered degree sequences D_1 and D_2 and $f_{-1} = 0$. Note that by definition $f_k(u, v)$ is non-decreasing in k and is defined only when both u or v have nodes at distance k. Moreover, using the ring at distance k in the definition of $f_k(u, v)$ forces the comparison between the degree sequences of nodes that are at the same distance from u and v. Finally, note that if the k-hop neighborhoods of u and v are isomorphic, and maps u onto v, then $f_{k-1}(u, v) = 0$.

A final step is determining the function that compares two degree sequences. Note that $s(R_k(u))$ and $s(R_k(v))$ can be of different sizes and its elements are arbitrary integers in the range $[0, n-1]$ with possible repetitions. We adopt Dynamic Time Warping (DTW) to measure the distance between two ordered degree sequences, a technique that can cope better with sequences of different sizes and loosely compares sequence patterns [18, 20].

Informally, DTW finds the optimal alignment between two sequences A and B. Given a distance function $d(a, b)$ for the elements of the sequence, DTW matches each element $a \in A$ to $b \in B$, such that the sum of the distances between matched elements is minimized. Since elements of sequence A and B are degrees of nodes, we adopt the following distance function:

$$d(a, b) = \frac{\max(a, b)}{\min(a, b)} - 1 \tag{2}$$

Note that when $a = b$ then $d(a, b) = 0$. Thus, two identical ordered degree sequences will have zero distance. Also note that by taking the ratio between the maximum and the minimum, the degrees 1 and 2 are much more different than degrees 101 and 102, a desired property when measuring the distance between node degrees. Finally, while we use DTW to assess the similarity between two ordered degree sequences, any other cost function could be adopted by our framework.

3.2 Constructing the context graph

We construct a multilayer weighted graph that encodes the structural similarity between nodes. Recall that $G = (V, E)$ denotes the original network (possibly not connected) and k^* its diameter. Let M denote the multilayer graph where layer k is defined using the k-hop neighborhoods of the nodes.

Each layer $k = 0, \ldots, k^*$ is formed by a weighted undirected complete graph with node set V, and thus, $\binom{n}{2}$ edges. The edge weight between two nodes in a layer is given by:

$$w_k(u, v) = e^{-f_k(u,v)}, \quad k = 0, \ldots, k^* \tag{3}$$

Note that edges are defined only if $f_k(u,v)$ is defined and that weights are inversely proportional to structural distance, and assume values smaller than or equal to 1, being equal to 1 only if $f_k(u,v) = 0$. Note that nodes that are structurally similar to u will have larger weights across various layers of M.

We connect the layers using directed edges as follows. Each vertex is connected to its corresponding vertex in the layer above and below (layer permitting). Thus, every vertex $u \in V$ in layer k is connected to the corresponding vertex u in layer $k+1$ and $k-1$. The edge weight between layers are as follows:

$$\begin{aligned} w(u_k, u_{k+1}) &= \log(\Gamma_k(u) + e), \quad k = 0, \ldots, k^* - 1 \\ w(u_k, u_{k-1}) &= 1, \quad k = 1, \ldots, k^* \end{aligned} \tag{4}$$

where $\Gamma_k(u)$ is number of edges incident to u that have weight larger than the average edge weight of the complete graph in layer k. In particular:

$$\Gamma_k(u) = \sum_{v \in V} \mathbb{1}(w_k(u,v) > \overline{w_k}) \tag{5}$$

where $\overline{w_k} = \sum_{(u,v)\in\binom{V}{2}} w_k(u,v)/\binom{n}{2}$. Thus, $\Gamma_k(u)$ measures the similarity of node u to other nodes in layer k. Note that if u has many similar nodes in the current layer, then it should change layers to obtain a more refined context. Note that by moving up one layer the number of similar nodes can only decrease. Last, the log function simply reduces the magnitude of the potentially large number of nodes that are similar to u in a given layer.

Finally, note that M has nk^* vertices and at most $k^*\binom{n}{2} + 2n(k^* - 1)$ weighted edges. In Section 3.5 we discuss how to reduce the complexity of generating and storing M.

3.3 Generating context for nodes

The multilayer graph M is used to generate structural context for each node $u \in V$. Note that M captures the structure of structural similarities between nodes in G using absolutely no label information. As in previous works, *struct2vec* uses random walks to generate sequence of nodes to determine the context of a given node. In particular, we consider a biased random walk that moves around M making random choices according to the weights of M. Before each step, the random walk first decides if it will change layers or walk on the current layer (with probability $q > 0$ the random walk stays in the current layer).

Given that it will stay in the current layer, the probability of stepping from node u to node v in layer k is given by:

$$p_k(u,v) = \frac{e^{-f_k(u,v)}}{Z_k(u)} \tag{6}$$

where $Z_k(u)$ is the normalization factor for vertex u in layer k, simply given by:

$$Z_k(u) = \sum_{\substack{v \in V \\ v \neq u}} e^{-f_k(u,v)} \tag{7}$$

Note that the random walk will prefer to step onto nodes that are structurally more similar to the current vertex, avoiding nodes that have very little structural similarity with it. Thus, the context of a node $u \in V$ is likely to have structurally similar nodes, independent of their labels and position in the original network G.

With probability $1-q$, the random walk decides to change layers, and moves to the corresponding node either in layer $k+1$ or layer $k-1$ (layer permitting) with probability proportional to the edge weights. In particular:

$$\begin{aligned} p_k(u_k, u_{k+1}) &= \frac{w(u_k, u_{k+1})}{w(u_k, u_{k+1}) + w(u_k, u_{k-1})} \\ p_k(u_k, u_{k-1}) &= 1 - p_k(u_k, u_{k+1}) \end{aligned} \tag{8}$$

Note that every time the walker steps within a layer it includes the current vertex as part of its context, independent of the layer. Thus, a vertex u may have a given context in layer k (determined by the structural similarity of this layer), but have a subset of this context at layer $k+1$, as the structural similarity cannot increase as we move to higher layers. This notion of a hierarchical context across the layers is a fundamental aspect of the proposed methodology.

Finally, for each node $u \in V$, we start a random walk in its corresponding vertex in layer 0. Random walks have a fixed and relatively short length (number of steps), and the process is repeated a certain number of times, giving rise to multiple independent walks (i.e., the multiple contexts of node u).

3.4 Learning a language model

Recent language modeling techniques have been extensively used to learn word embeddings, and only require sets of sentences in order to generate meaningful representations. Informally, the task can be defined as learning word probabilities given a context. In particular, Skip-Gram [12] has proven to be effective at learning meaningful representations for a variety of data. In order to apply it to networks, it suffices to use artificially generated node sequences instead of word sentences. In our framework, these sequences are generated by biased random walks on the multilayer graph M. Given a node, Skip-Gram aims to maximize the likelihood of its context in a sequence, where a node's context is given by a window of size w centered on it.

For this work we use Hierarchical Softmax, where conditional symbol probabilities are calculated using a tree of binary classifiers. For each node $v_j \in V$, Hierarchical Softmax assigns a specific path in the classification tree, defined by a set of tree nodes $n(v_j, 1), n(v_j, 2), \ldots, n(v_j, h)$, where $n(v_j, h) = v_j$. In this setting, we have:

$$P(v_j | v_i) = \prod_{k=1}^{h} C(n(v_j, k), v_i) \tag{9}$$

where C is a binary classifier present in every node in the tree. Note that since Hierarchical Softmax operates on a binary tree, we have that $h = O(\log |V|)$.

We train Skip-Gram according to its optimization problem given by equation (9). Note that while we use Skip-Gram to learn node embeddings, any other technique to learn latent representations for text data could be used in our framework.

3.5 Complexity and optimizations

In order to construct M, the structural distance between every node pair for every layer must be computed, namely, $f_k(u,v)$ for $u, v \in V$, and $0 \leq k \leq k^*$. However, $f_k(u,v)$ uses the result of a DTW calculation between two degree sequences. While classic implementation of DTW has complexity $O(\ell^2)$, fast techniques have

complexity $O(\ell)$, where ℓ is the size of the largest sequence [20]. Let $d_{\max}$ denote the largest degree in the network. Then, the size of the degree sequence $|s(R_k(u))| \leq \min(d_{\max}^k, n)$, for any node u and layer k. Since in each layer there are $\binom{n}{2}$ pairs, the complexity of computing all distances for layer k is $O(n^2 \min(d_{\max}^k, n))$. The final complexity is then $O(k^* n^3)$. In what follows we describe a series of optimizations that will significantly reduce the computation and memory requirements of the framework.

Reducing the length of degree sequences (OPT1). Although degree sequences at layer k have lengths bounded by $\min(d_{\max}^k, n)$, for some networks this can be quite large even for small k (e.g., for $k = 3$ the sequences are already $O(n)$). To reduce the cost of comparing large sequences, we propose compressing the ordered degree sequence as follows. For each degree in the sequence, we count the number of occurrences of that degree. The compressed ordered degree sequence is a tuple with the degree and the number of occurrences. Since many nodes in a network tend to have the same degree, in practice the compressed ordered degree sequence can be an order of magnitude smaller than the original.

Let A' and B' denote the compressed degree sequences of A and B, respectively. Since the elements of A' and B' are tuples, we adapt the DTW pairwise distance function as follows:

$$\text{dist}(\boldsymbol{a}, \boldsymbol{b}) = \left(\frac{\max(a_0, b_0)}{\min(a_0, b_0)} - 1 \right) \max(a_1, b_1) \tag{10}$$

where $\boldsymbol{a} = (a_0, a_1)$ and $\boldsymbol{b} = (b_0, b_1)$ are tuples in A' and B', respectively; a_0 and b_0 are the degrees; a_1 and b_1 are the number of occurrences. Note that using the compressed degree sequence leads to comparisons between pieces of the original sequences that have the same degree (as opposed to comparing every degree). Thus, equation (10) leads to an approximation of the DTW on the original degree sequences, as given by equation (2). However, DTW now operates on A' and B', which are much shorter than A and B, respectively.

Reducing the number of pairwise similarity calculations (OPT2). While the original framework assesses the similarity between every node pair at every layer k, clearly this seems unnecessary. Consider two nodes with very different degrees (eg., 2 and 20). Their structural distance even for $k = 0$ will be large, and consequently the edge between them in M will have a very small weight. Thus, when generating context for these nodes, the random walk is unlikely to traverse this edge. Consequently, not having this edge in M will not significantly change the model.

We limit the number of pairwise similarity calculations to $\Theta(\log n)$ per node, for every level k. Let J_u denote the set of nodes that will be neighbors of u in M, which will be the same for every level. J_u should have the nodes most structurally similar to u. In order to determine J_u, we take the nodes that have degrees most similar to u. This can be computed efficiently by performing a binary search on the ordered degree sequence of all nodes in the network (for the degree of node u), and taking $\log n$ consecutive nodes on each direction after the search completes. Thus, computing J_u has complexity $\Theta(\log n)$. Computing J_u for all nodes has complexity $\Theta(n \log n)$ which is also needed for sorting the degrees of the network. As for memory requirements, each layer of M will now have $\Theta(n \log n)$ edges as opposed to $\Theta(n^2)$.

Reducing the number of layers (OPT3). The number of layers in M is given by the diameter of the network, k^*. However, for many networks the diameter can be much larger than the average distance. Moreover, the importance of assessing the structural similarity between two nodes diminishes with arbitrarily large values for k. In particular, when k is near k^* the length of the degree sequences of the rings become relatively short, and thus $f_k(u, v)$ is not much different from $f_{k-1}(u, v)$. Therefore, we cap the number the layers in M to a fixed constant $k' < k^*$, capturing the most important layers for assessing structural similarity. This significantly reduces the computational and memory requirements for constructing M.

Although the combination of the above optimizations affects the capacity of the framework in generating good representations for nodes that are structurally similar, we will show that their impact is marginal and sometimes even beneficial. Thus, the benefits in reducing computational and memory requirements of the framework greatly outweighs its drawbacks. Last, we make *struc2vec* available at: https://github.com/leoribeiro/struc2vec

4 EXPERIMENTAL EVALUATION

In what follows we evaluate *struct2vec* in different scenarios in order to illustrate its potential in capturing the structural identity of nodes, also in light of state-of-the-art techniques for learning node representations.

4.1 Barbell graph

We denote $B(h, k)$ as the (h, k)-barbell graph which consists of two complete graphs K_1 and K_2 (each having h nodes) connected by a path graph P of length k. Two nodes $b_1 \in V(K_1)$ and $b_2 \in V(K_2)$ act as the bridges. Using $\{p_1, \ldots, p_k\}$ to denote $V(P)$, we connect b_1 to p_1 and b_2 to p_k, thus connecting the three graphs.

The barbell graph has a significant number of nodes with the same structural identity. Let $C_1 = V(K_1) \setminus \{b_1\}$ and $C_2 = V(K_2) \setminus \{b_2\}$. Note that all nodes $v \in \{C_1 \cup C_2\}$ are structurally equivalent, in the strong sense that there exists an automorphism between any pair of such nodes. Additionally, we also have that all node pairs $\{p_i, p_{k-i}\}$, for $1 \leq i \leq k-1$, along with the pair $\{b_1, b_2\}$, are structurally equivalent in the same strong sense. Figure 2a illustrates a $B(10, 10)$ graph, where structurally equivalent nodes have the same color.

Thus, we expect *struct2vec* to learn vertex representations that capture the structural equivalence mentioned above. Every node pair that is structurally equivalent should have similar latent representation. Moreover, the learned representations should also capture structural hierarchies: while the node p_1 is not equivalent to neither nodes p_2 or b_1, we can clearly see that from a structural point of view it is more similar to p_2 (it suffices to compare their degrees).

Figure 2 shows the latent representations learned by *DeepWalk*, *node2vec* and *struct2vec* for $B(10, 10)$. *DeepWalk* fails to capture structural equivalences, which is expected since it was not designed to consider structural identities. As illustrated, *node2vec* does not capture structural identities even with different variations of its parameters p and q. In fact, it learns mostly graph distances, placing closer in the latent space nodes that are closer (in hops) in the graph. Another limitation of *node2vec* is that Skip-Gram's window size

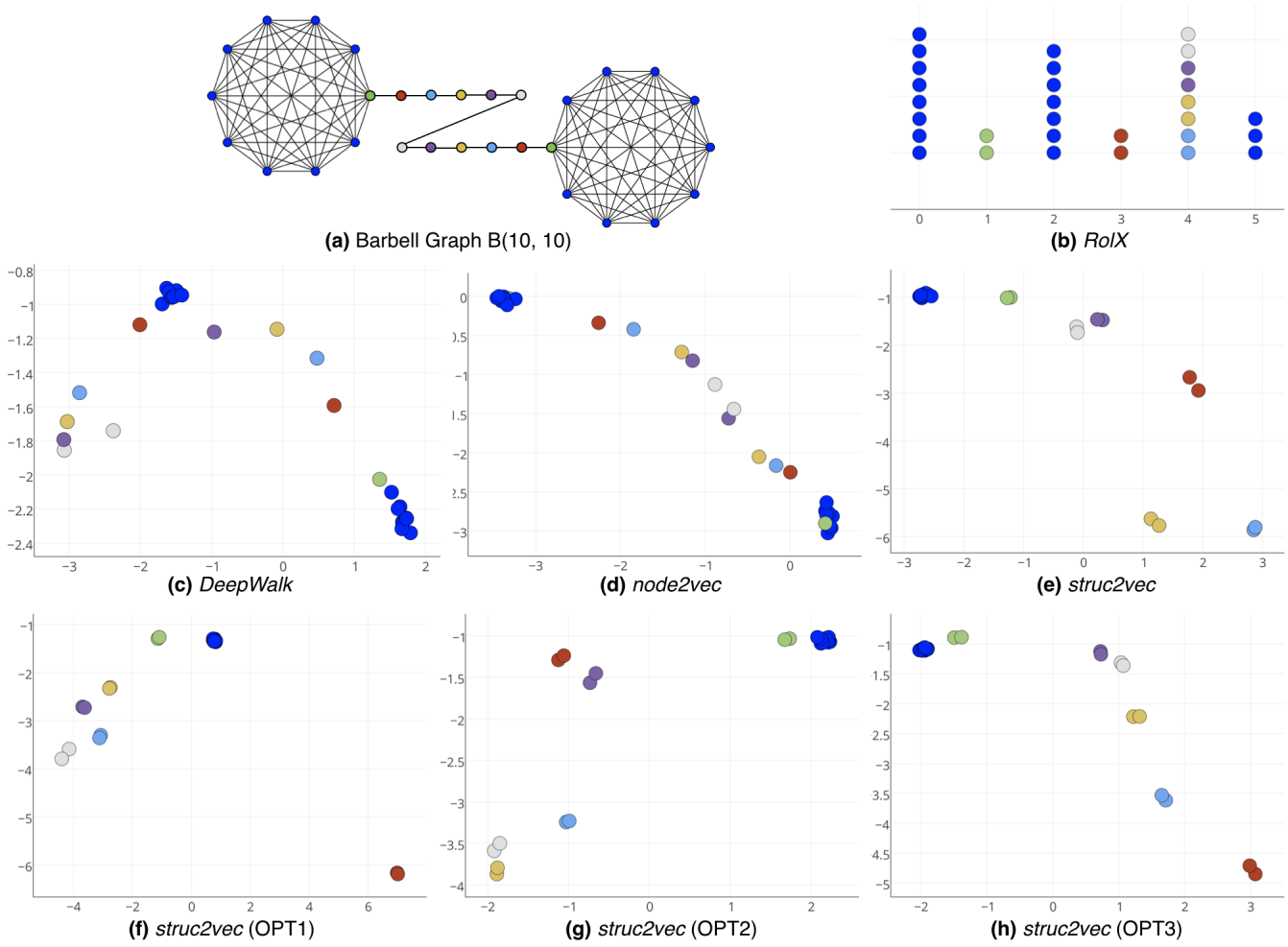

Figure 2: (a) Barbell graph $B(10, 10)$. (b) Roles identified by *RolX*. Latent representations in $\mathbb{R}^2$ learned by (c) DeepWalk, (d) *node2vec* and (e,f,g,h) *struc2vec*. Parameters used for all methods: number of walks per node: 20, walk length: 80, skip-gram window size: 5. For *node2vec*: $p = 1$ and $q = 2$.

makes it impossible for nodes in K_1 and K_2 to appear in the same context.

struct2vec, on the other hand, learns representations that properly separate the equivalent classes, placing structurally equivalent nodes near one another in the latent space. Note that nodes of the same color are tightly grouped together. Moreover, p_1 and p_{10} are placed close to representations for nodes in K_1 and K_2, as they are the bridges. Finally, note that none of the three optimizations have any significant effect on the quality of the representations. In fact, structurally equivalent nodes are even closer to one another in the latent representations under OPT1.

Last, we apply *RolX* to the barbell graph (results in Figure 2(b)). A total of six roles were identified and some roles indeed precisely captured structural equivalence (roles 1 and 3). However, structurally equivalent nodes (in K_1 and K_2) were placed in three different roles (role 0, 2, and 5) while role 4 contains all remaining nodes in the path. Thus, although *RolX* does capture some notion of structural equivalence when assigning roles to nodes, *struct2vec* better identifies and separates structural equivalence.

4.2 Karate network

The Zachary's Karate Club [25] is a network composed of 34 nodes and 78 edges, where each node represents a club member and edges denote if two members have interacted outside the club. In this network, edges are commonly interpreted as indications of friendship between members.

We construct a network composed of two copies G_1 and G_2 of the Karate Club network, where each node $v \in V(G_1)$ has a mirror node $u \in V(G_2)$. We also connect the two networks by adding an edge between mirrored node pairs 1 and 37. Although this is not necessary for our framework, *DeepWalk* and *node2vec* cannot place in the same context nodes in different connected components of the graph. Thus, we add the edge for a more fair comparison

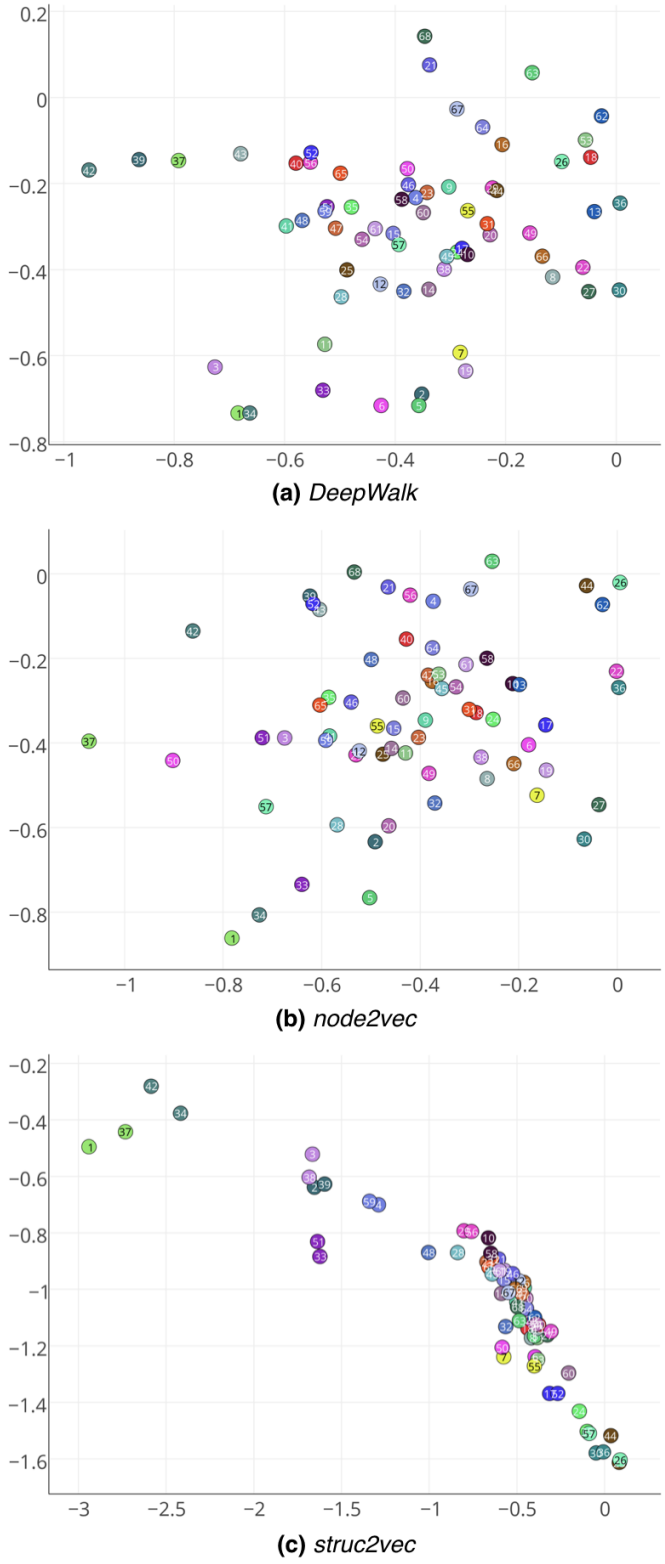

Figure 3: Node representations for the mirrored Karate network created by (a) *DeepWalk*, (b) *node2vec* and (c) *struc2vec*. Parameters used for all methods: number of walks per node: 5, walk length: 15, Skip-Gram window size: 3. For *node2vec*: $p = 1$ and $q = 2$.

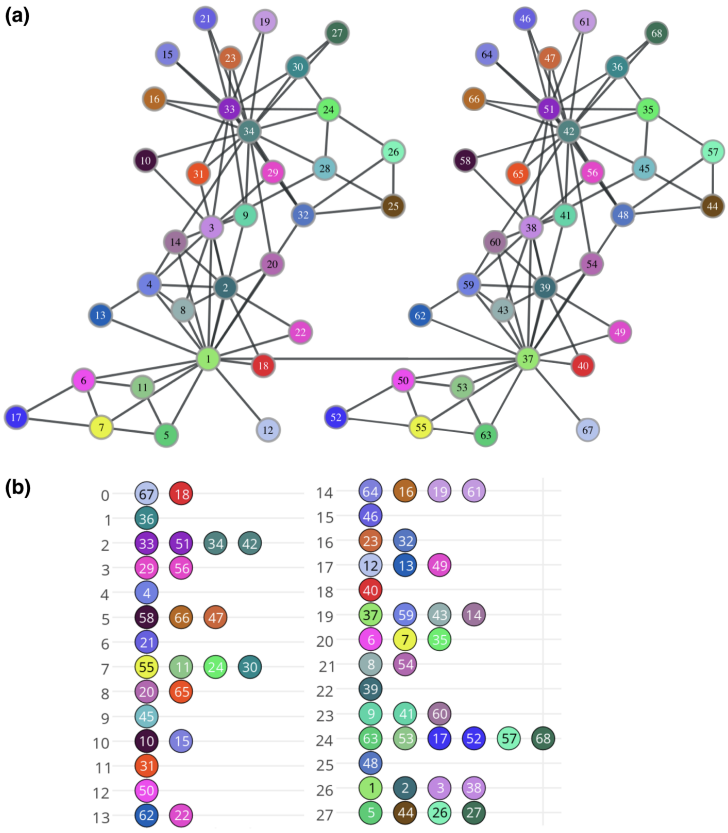

Figure 4: (a) Mirrored Karate network. Identical colors correspond to mirrored nodes. (b) Roles identified by *RolX*

with the two baselines. Figure 4a shows mirrored network with corresponding pairs having the same color.

Figure 3 shows the representations learned by *DeepWalk*, *node2vec* and *struct2vec*. Clearly, *Deepwalk* and *node2vec* fail to group in the latent space structurally equivalent nodes, including mirrored nodes.

Once again, *struct2vec* manages to learn features that properly capture the structural identity of nodes. Mirrored pairs – that is, nodes with the same color – stay close together in the latent space, and there is a complex structural hierarchy in the way the representations are grouped together.

As an example, note that nodes 1, 34 and their correspondent mirrors (37 and 42) are in a separate cluster in the latent space. Interestingly, these are exactly the nodes that represent the club instructor Mr. Hi and his administrator John A. The network was gathered after a conflict between the two split the members of the club into two groups – centered on either Mr. Hi or John A. Therefore, nodes 1 and 34 have the specific and similar role of leaders in the network. Note that *struct2vec* captures their function even though there is no edge between them.

Another visible cluster in the latent space is composed of nodes 2, 3, 4 and 33, also along with their mirrors. These nodes also have a specific structural identity in the network: all of them have high

degrees and are also connected to at least one of the leaders. Lastly, nodes 26 and 25 (far right in the latent space) have extremely close representations, which agrees with their structural role: both have low degree and are 2 hops away from leader 34.

struct2vec also captures non-trivial structural equivalences. Note that nodes 7 and 50 (pink and yellow) are mapped to close points in the latent space. Surprisingly, these two nodes are structurally equivalent – there exists an automorphism in the graph that maps one into the other. This can be more easily seen once we note that nodes 6 and 7 are also structurally equivalent, and 50 is the mirrored version of node 6 (therefore also structurally equivalent).

Last, Figure 4b shows the roles identified by *RolX* in the mirrored Karate network (28 roles were identified). Note that leaders 1 and 34 were placed in different roles. The mirror for 1 (node 37) was also placed in a different role, while the mirror for 34 (node 42) was placed in the same role as 34. A total of 7 corresponding pairs (out of 34) were placed in the same role. However, some other structural similarities were also identified – e.g., nodes 6 and 7 are structurally equivalent and were assigned the same role. Again, *RolX* seems to capture some notion of structural similarities among network nodes but *struct2vec* can better identify and separate structural equivalences using latent representations.

Consider the distance between the latent representation for nodes. We measure the distance distribution between pairs corresponding to mirrored nodes and among all node pairs (using the representation shown in Figure 3). Figure 5 shows the two distance distributions for the representations learned by *node2vec* and *struc2vec*. For *node2vec* the two distributions are practically identical, indicating that distances between mirrored pairs blend well with all pairs. In contrast, *struc2vec* exhibits two very different distributions: 94% of mirrored node pairs have distance smaller than 0.25 while 68% of all node pairs have distance larger than 0.25. Moreover, the average distance between all node pairs is 5.6 times larger than that of mirrored pairs, while this ratio is about slightly *smaller* than 1 for *node2vec*.

To better characterize the relationship between structural distance and distances in the latent representation learned by *struc2vec*, we compute the correlation between the two distances for all node pairs. In particular, for each layer k we compute the Spearman and Pearson correlation coefficients between $f_k(u, v)$, as given by equation (1), and the euclidean distance between u and v in the learned representation. Results shown in Table 1 for the mirrored Karate network indeed corroborate that there is a very strong correlation between the two distances, for every layer, captured by both coefficients. This suggests that *struc2vec* indeed captures in the latent space the measure for structural similarity adopted by the methodology.

4.3 Robustness to edge removal

We illustrate the potential of the framework in effectively representing structural identity in the presence of noise. In particular, we randomly remove edges from the network, directly changing its structure. We adopt the parsimonious *edge sampling model* to instantiate two structurally correlated networks [15].

The model works by taking a fixed graph $G = (V, E)$ and generating a graph G_1 by sampling each edge $e \in E$ with probability s,

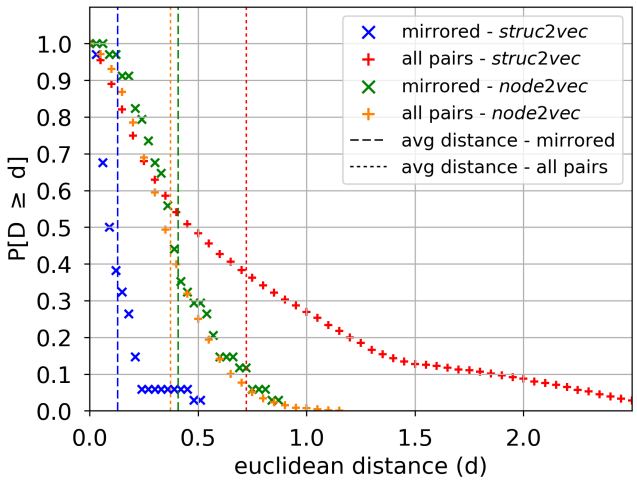

Figure 5: Distance distributions between node pairs (mirrored pairs and all pairs) in the latent space, for the mirrored Karate network learned by *node2vec* and *struc2vec* (as shown in Figure 3). Curves marked with × correspond to distances between mirrored pairs while + corresponds to all pairs; corresponding averages indicated by vertical lines.

Table 1: Pearson and Spearman correlation coefficients between structural distance and euclidean distance in latent space for all node pairs in the mirrored Karate network.

Layer	Pearson (p-value)	Spearman (p-value)
0	0.83 (0.0)	0.74 (0.0)
2	0.71 (0.0)	0.65 (0.0)
4	0.70 (0.0)	0.57 (0.0)
6	0.74 (0.0)	0.57 (2.37)

independently. Thus, each edge of G is present in G_1 with probability s. Repeat the process again using G to generate another graph G_2. Thus, G_1 and G_2 are structurally correlated through G, and s controls the amount of structural correlation. Note that when $s = 1$, G_1 and G_2 are isomorphic, while when $s = 0$ all structural identity is lost.

We apply the edge sampling model to an *egonet* extracted from Facebook (224 nodes, 3192 edges, max degree 99, min degree 1) [10] to generate G_1 and G_2 with different values for s. We relabel the nodes in G_2 (to avoid identical labels) and consider the union of the two graphs as the input network to our framework. Note that this graph has at least two connected components (corresponding to G_1 and G_2) and every node (in G_1) has a corresponding pair (in G_2).

Figure 6 shows the distance (latent space with 2 dimensions) distribution between corresponding node pairs and all node pairs for various values for s (corresponding averages are shown in Table 2). For $s = 1$, the two distance distributions are strikingly different, with the average distance for all pairs being 21 times larger than that for corresponding pairs. More interestingly, when $s = 0.9$ the two distributions are still very different. Note that while further

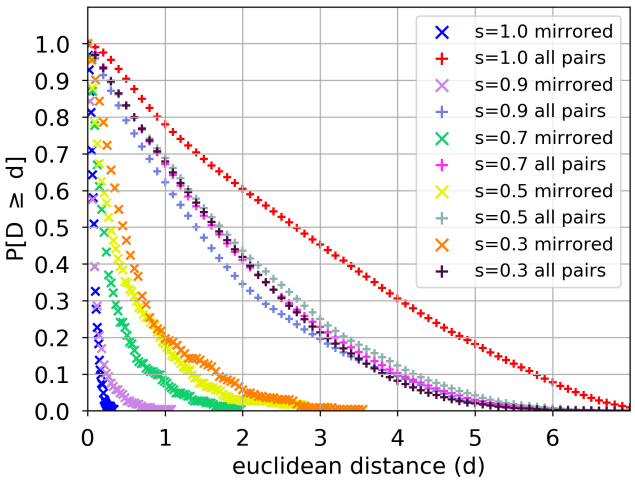

Figure 6: Distance distribution between node pairs in latent space representation (2 dimensions) under the edge sampling model (different values for s). Bottom curves (marked with ×) are for corresponding node pairs; top curves (marked with +) are for all node pairs.

decreasing s does not significantly affect the distance distribution of all pairs, it slowly increases the distribution of corresponding pairs. However, even when $s = 0.3$ (which means that the probability that an original edge appears both in G_1 and G_2 is 0.09, s^2), the framework still places corresponding nodes closer in the latent space.

This experiment indicates the robustness of the framework in uncovering the structural identity of nodes even in the presence of structural noise, modeled here through edge removals.

Table 2: Average and standard deviation for distances between node pairs in the latent space representation (see corresponding distributions in Figure 6).

s	Corresponding - avg (std)	All nodes - avg (std)
1.0	0.083 (0.05)	1.780 (1.354)
0.9	0.117 (0.142)	1.769 (1.395)
0.3	0.674 (0.662)	1.962 (1.445)

4.4 Classification

A common application of latent representations for network nodes is classification. *struc2vec* can be leveraged for this task when labels for nodes are more related to their structural identity than to the labels of their neighbors. To illustrate this potential, we consider air-traffic networks: unweighted, undirected networks where nodes correspond to airports and edges indicate the existence of commercial flights. Airports will be assigned a label corresponding to their level of activity, measured in flights or people (discussed below). We consider the following datasets (collected for this study):

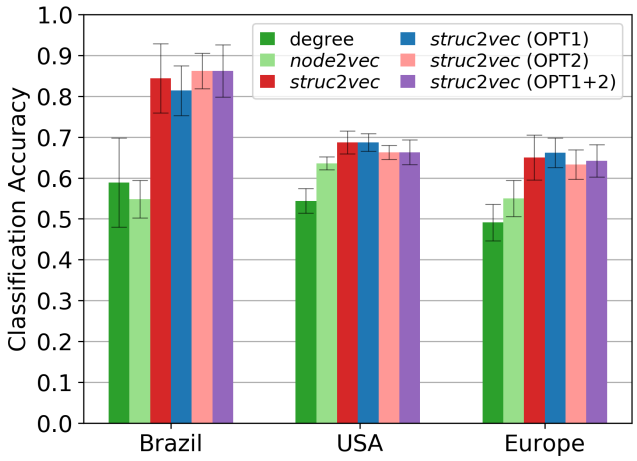

Figure 7: Average accuracy for multi-class node classification in air-traffic networks of Brazil, USA and Europe for different node features used in supervised learning.

- *Brazilian air-traffic network:* Data collected from the National Civil Aviation Agency (ANAC)[1] from January to December 2016. The network has 131 nodes, 1,038 edges (diameter is 5). Airport activity is measured by the total number of landings plus takeoffs in the corresponding year.
- *American air-traffic network:* Data collected from the Bureau of Transportation Statistics[2] from January to October, 2016. The network has 1,190 nodes, 13,599 edges (diameter is 8). Airport activity is measured by the total number of people that passed (arrived plus departed) the airport in the corresponding period.
- *European air-traffic network:* Data collected from the Statistical Office of the European Union (Eurostat)[3] from January to November 2016. The network has 399 nodes, 5,995 edges (diameter is 5). Airport activity is measured by the total number of landings plus takeoffs in the corresponding period.

For each airport, we assign one of four possible labels corresponding to their activity. In particular, for each dataset, we use the quartiles obtained from the empirical activity distribution to split the dataset in four groups, assigning a different label for each group. Thus, label 1 is given to the 25% less active airports, and so on. Note that all classes (labels) have the same size (number of airports). Moreover, classes are related more to the role played by the airport.

We learn latent representations for nodes of each air-traffic network using *struc2vec* and *node2vec* using a grid search to select the best hyperparameters for each case. Note that this step does not use any node label information. The latent representation for each node becomes the feature that is then used to train a supervised classifier (one-vs-rest logistic regression with L2 regularization). We also consider just the node degree as a feature since it captures a very basic notion of structural identity. Last, since classes have identical

[1] http://www.anac.gov.br/
[2] https://transtats.bts.gov/
[3] http://ec.europa.eu/

sizes, we use just the accuracy to assess performance. Experiments are repeated 10 times using random samples to train the classifier (80% of the nodes used for training) and we report on the average performance.

Figure 7 shows the classification performance of the different features for all air-traffic networks. Clearly, *struc2vec* outperforms the other approaches, and its optimizations have little influence. For the Brazilian network, *struc2vec* improves classification accuracy by 50% in comparison to *node2vec*. Interestingly, for this network *node2vec* has average performance (slightly) inferior to node degree, indicating the importance played by the structural identity of the nodes in classification.

4.5 Scalability

In order to illustrate its scalability, we apply *struc2vec* with the first two optimizations to instances of the Erdös-Rényi random graph model (using 128 dimensions, 10 walks per node, walk length 80, Skip-Gram window 10). We compute the average execution time for 10 independent runs on graphs with sizes from 100 to 1,000,000 nodes and average degree of 10. In order to speed up training the language model, we use Skip-Gram with Negative Sampling [13]. Figure 8 shows the execution time (in log-log scale) indicating that *struc2vec* scales super-linearly but closer to linear than to $n^{1.5}$ (dashed lines). Thus, despite its unfavorable worst case time and space complexity, in practice *struc2vec* can be applied to very large networks.

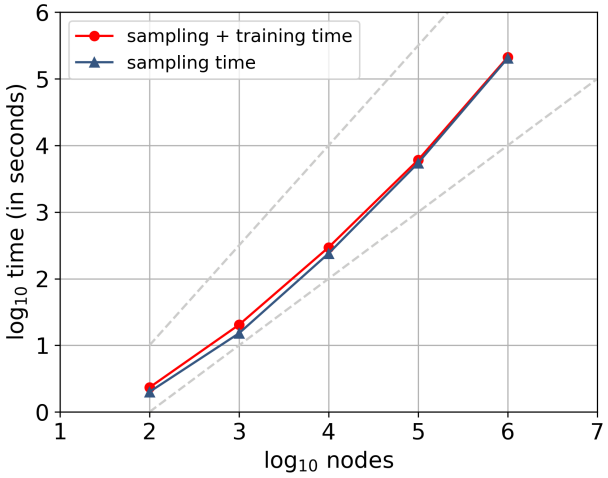

Figure 8: Average execution time of *struc2vec* on Erdös-Rényi graphs with average degree of 10. Training time refers to the additional time required by Skip-Gram.

5 CONCLUSION

Structural identity is a concept of symmetry in networks in which nodes are identified based on the network structure. The concept is strongly related to functions or roles played by nodes in the network, an important problem in social and hard sciences.

We propose *struc2vec*, a novel and flexible framework to learn representations that capture the structural identity of nodes in a network. *struc2vec* assesses the structural similarity of node pairs by considering a hierarchical metric defined by the ordered degree sequence of nodes and uses a weighted multilayer graph to generate context.

We have shown that *struc2vec* excels in capturing the structural identity of nodes, in comparison to state-of-the-art techniques such as *DeepWalk*, *node2vec* and *RolX*. It overcomes their limitation by focusing explicitly on structural identity. Not surprising, we also show that *struc2vec* is superior in classification tasks where node labels are more dependent on their role or structural identity. Last, different models to generate representations tend to capture different properties, and we argue that structural identity is clearly important when considering possible node representations.

REFERENCES

[1] Nir Atias and Roded Sharan. 2012. Comparative analysis of protein networks: hard problems, practical solutions. *Commun. ACM* 55 (2012).
[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *JMLR* (2003).
[3] V Blondel, A Gajardo, M Heymans, P Senellart, and P Van Dooren. 2004. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review* (2004).
[4] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2016. Deep Neural Networks for Learning Graph Representations. In *AAAI*.
[5] F Fouss, A Pirotte, J Renders, and M Saerens. 2007. Random-Walk Computation of Similarities Between Nodes of a Graph with Application to Collaborative Recommendation. *IEEE Trans. on Knowl. and Data Eng.* (2007).
[6] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *ACM SIGKDD*.
[7] K Henderson, B Gallagher, T Eliassi-Rad, H Tong, S Basu, L Akoglu, D Koutra, C Faloutsos, and L Li. 2012. Rolx: structural role extraction & mining in large graphs. In *ACM SIGKDD*.
[8] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* (1999).
[9] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. 2006. Vertex similarity in networks. *Physical Review E* 73 (2006).
[10] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *NIPS*.
[11] Francois Lorrain and Harrison C White. 1971. Structural equivalence of individuals in social networks. *The Journal of mathematical sociology* 1 (1971).
[12] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR Workshop*.
[13] T Mikolov, I Sutskever, K Chen, G Corrado, and J Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *NIPS*.
[14] A Narayanan, M Chandramohan, L Chen, Y Liu, and S Saminathan. 2016. subgraph2vec: Learning Distributed Representations of Rooted Sub-graphs from Large Graphs. In *Workshop on Mining and Learning with Graphs*.
[15] Pedram Pedarsani and Matthias Grossglauser. 2011. On the privacy of anonymized networks. In *ACM SIGKDD*.
[16] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *ACM SIGKDD*.
[17] Narciso Pizarro. 2007. Structural Identity and Equivalence of Individuals in Social Networks Beyond Duality. *International Sociology* 22 (2007).
[18] T Rakthanmanon, B Campana, A Mueen, G Batista, B Westover, Q Zhu, J Zakaria, and E Keogh. 2013. Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM TKDD* (2013).
[19] Lee Douglas Sailer. 1978. Structural equivalence: Meaning and definition, computation and application. *Social Networks* (1978).
[20] S Salvador and P Chan. 2004. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Workshop on Min. Temp. and Seq. Data, ACM SIGKDD*.
[21] N Shervashidze, P Schweitzer, E van Leeuwen, K Mehlhorn, and K Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *JMLR* (2011).
[22] R Singh, J Xu, and B Berger. 2008. Global alignment of multiple protein interaction networks with application to functional orthology detection. *PNAS* (2008).
[23] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. LINE: Large-scale Information Network Embedding. In *WWW*.
[24] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural Deep Network Embedding. In *ACM SIGKDD*.
[25] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* (1977).
[26] Laura A Zager and George C Verghese. 2008. Graph similarity scoring and matching. *Applied mathematics letters* (2008).