struc2vec: Learning Node Representations from Structural Identity

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

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

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 has only recently been addressed with techniques from representational learning. This work presents *struc2vec*, a novel and flexible framework for learning latent representations of node's structural identity. *struc2vec* assesses structural similarity without using node or edge attributes, uses a hierarchy to measure similarity at different scales, and constructs a multilayer graph to encode the 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 techniques.

CCS CONCEPTS

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

KEYWORDS

Feature learning, Node embeddings, Structural Identity

ACM Reference format:
Daniel R. Figueiredo, Leonardo F. R. Ribeiro, and Pedro H. P. Saverese. 2017. *struc2vec*: Learning Node Representations from Structural Identity. In *Proceedings of 23rd SIGKDD Conference on Knowledge Discovery and Data Mining, Halifax, Nova Scotia, Canada, August 13 - 17, 2017 (KDD '17),* 9 pages.
DOI: 10.475/123_4

1 INTRODUCTION

In almost all networks, nodes tend to have one or more functions that greatly determines their role in the system. For example, individuals in a social network have a social role or social position [9, 17], while proteins in a protein-protein interaction (PPI) network exert specific functions [1, 20]. 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.

KDD '17, Halifax, Nova Scotia, Canada

DOI: 10.475/123_4

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.

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 [9, 15, 17]. 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 [6].

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 [4, 7]. 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, 6, 22]. While both approaches have advantages and disadvantages, we provide an alternative methodology, one based on unsupervised learning of representations that capture 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 [5, 12, 14]. In particular, these efforts encode nodes using as context a generalized notion of their neighborhood (e.g., w steps of a random walk). In a nutshell, nodes that have similar neighborhoods should have similar latent representations. But in all such works, neighborhood is a local concept defined by node labels! Thus, two nodes with neighborhoods that are structurally similar but that are far apart will not have similar latent representations, a fundamental requirement for structural equivalence. 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. Indeed, the aforementioned recent works fail to capture the notion of structural equivalence (as we soon show).

It is worth noting why recent approaches for learning node representations such as *DeepWalk* [14] and *node2vec* [5] 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.

Thus, our main contribution is to provide a flexible framework for learning latent representations for nodes' structural identity, called *struc2vec*. This framework offers an alternative and powerful tool to the study of structural identity through the latent space representation. The key ideas within this framework are:

- Assess structural similarity between nodes independently of node and edge attributes, including node labels. Thus, two nodes that are structurally similar will be considered so, independently of their position in the network and node labels in their vicinity. 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 (but *not* walking on the original network). Thus, two vertices that frequently appear in 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 our framework and show its potential through numerical experiments on toy examples and real networks, comparing its performance with *DeepWalk* [14] and *node2vec* [5] – two state-of-the-art techniques for learning latent representations of nodes in networks. Our results indicate that while both 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).

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 is shown in Section 4. Finally, we conclude the paper in Section 5 with a brief discussion.

2 RELATED WORK

Generating dense representations for sparse data has a long history in Natural Language Processing [2]. Traditional encodings, such as one-hot or bag of words, generate representations that have the same size as the vocabulary, which is commonly in the order of millions of words. The resultant sparse high-dimensional data pose an obstacle for many tasks, including text classification and clustering.

Recently, Skip-Gram [10, 11] was proposed as a technique to learn dense representations for text data, and provides an easy optimization problem where a word's context should be predicted given its latent features. Moreover, the embeddings capture word meanings, placing semantically similar words near each other in the latent space.

Due to the high-dimensional and often sparse nature of graph representations (e.g. the adjacency matrix), learning node embeddings is equally important for Machine Learning applications on network data. Since Skip-Gram (and most other language models) require temporal sequences as input, adapting it to learn representations for graphs is non-trivial as graph data is not linear.

Learning a language model from a network was first proposed by *DeepWalk* [14]. It uses random walks to generate sequences of nodes from a graph, which are then treated as sentences by Skip-Gram. Intuitively, since vertices in the same Skip-Gram window are close in the network, the learned representations capture mostly vertex homophily.

The idea was later extended by *node2vec* [5]. By proposing a biased 2nd order random walk model, it provides more flexibility when generating the context of a vertex. In its framework, biased random walks are designed to capture both vertex homophily and structural equivalence.

Although ablation experiments suggest that *node2vec* can capture structural equivalence, it is unclear how it would perform on larger graphs. More specifically, structurally equivalent vertices will never share the same context if their distance (hop count) is bigger than the Skip-Gram window.

subgraph2vec [12] is another recent approach that aims to learn latent features for rooted subgraphs, and unlike the previous techniques it does not use random walks to generate context for nodes. Alternatively, it proposes Radial Skip-Gram, a modification of the original Skip-Gram where the context of a node is simply defined by its neighbors. Additionally, *subgraph2vec* properly captures structural equivalence by anchoring equivalent vertices to the same point in the latent 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 [19].

3 *STRUC2VEC*

Consider the problem of learning latent representations for nodes that captures their structural identity 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 are identical from the network

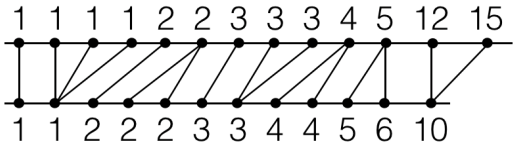


Figure 2: A warping between two ordered degree sequences using the function 2 to calculate the distances between the degrees. The distance between the two sequences, that is the sum of costs of matched elements, is 0.89.

structure point of view 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 methodology for learning latent representations for nodes. The methodology is composed of four main steps, informally defined as follows:

(1) Determine the structural similarity between each vertex pair in the graph considering 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 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 weighted random walk on the multilayer graph is used to generate vertex sequences. These sequences are likely to include nodes that have similar structure, and thus determine its context.
(4) Apply a technique to learn latent representation from a context given by sequences, for example, Skip-Gram.

Note that *struct2vec* is quite flexible as it does not determine 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 the structural similarity of node pairs without using any node or edge attributes. Moreover, we need a measure that can cope with increasing neighborhood sizes. While there are many ways to measure the *structural similarity* between two vertices, we would like a metric with the following property.

Let $G = (V, E)$ denote the 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 $N_k(u)$ denote the set of nodes with distance less than or equal to $k \geq 0$ from $u \in V$ (note that $N_0(u) = u$ and $N_1(u)$ are the neighbors of u and u itself). Let $G[S]$ denote the induced subgraph of G over the set of nodes $S \subset V$. Note that $G[N_1(u)]$ is often referred to as the *egonet* of node u.

Let $f(u, v) \geq 0$ denote a distance measure for the structural similarity between $u, v \in V$. A suitable f should satisfy the following property:

- $f(u, v) = 0$ if there exists an isomorphism between $G[N_k(u)]$ and $G[N_k(v)]$ for any $k > 0$, mapping u onto v.

Under this property, two nodes that have locally isomorphic neighborhoods should be considered identical to one another, and thus, have a structural distance of zero. Clearly, property is desired when considering the structural identity of nodes in networks.

However, isomorphisms cannot be used directly to measure structural similarity. For one reason, there are no polynomial time algorithm to determine if two arbitrary graphs are isomorphic, and second, isomorphism is a binary property. Thus, we consider the following approach.

Let $s(S)$ denote the ordered degree sequence of the set of vertices $S \subset V$. Note that if $G[N_k(u)]$ is isomorphic to $G[N_k(v)]$ for any $k > 0$, mapping u to v, then $s(N_{k-1}(u)) = s(N_{k-1}(v))$. Namely, the ordered degree sequences of nodes in the $(k-1)$-hop neighborhood of u and v must be identical. Thus, the ordered degree sequence can lead to a distance metric that satisfies the desired property. Moreover, the degree sequence avoids any label information associated to nodes or edges.

The ordered degree sequence is also a natural choice for inducing a hierarchy of distance functions. Let $R_k(u)$ denote the set of nodes at distance exactly $k \geq 0$ from u in G. Thus, $R_k(u) = N_k(u) \backslash N_{k-1}(u)$ for $k \geq 0$ (let $N_{-1}(u) = \emptyset$). By comparing the ordered degree sequences of the rings of nodes at distance k from both u and v we can impose a hierarchy in assessing their structural similarity, that becomes more stringent as k increases. In particular, let $f_k(u, v)$ denote the structural distance between u and v when considering their k-hop neighborhoods. In particular, we have:

$$f_k(u, v) = f_{k-1}(u, v) + g(s(R_k(u)), s(R_k(v))), \;\; k \geq 0 \qquad (1)$$

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. 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 $G[N_k(u)]$ and $G[N_k(v)]$ are isomorphic for some $k > 0$, mapping 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 $[0, n-1]$, where $n = |V|$ (i.e., any possible degree). 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 [16, 18].

Informally, DTW aims to find 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. Note that each element in one sequence can be matched to more than one element in the other, but crossings in the matching are not allowed, and all elements must be matched.

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. Figure 2 shows DTW applied to two ordered degree sequences.

Last, the function g in equation 1 is simply replaced by DTW. Note that k plays a key role in determining the structural distance between two nodes: $f_0(u,v) = 0$ if degrees of u and v are identical, while if $f_{k^*}(u,v) = 0$ then there is strong evidence that there exists an automorphism in G that maps u to v, since the degree sequence of all k-hop rings around u and v perfectly match. Note that if indeed there exists an automorphism in G that maps u to v, then $f_k(u,v) = 0$, for all k. Thus, structural similarity between u and v becomes more rigid as k increases.

3.2 Constructing the context graph

We construct a multilayer weighted graph that encodes 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, with layers going from 0 to k^*, corresponding to neighborhood hierarchy defined above. In particular, layer k will be defined using the k-hop neighborhoods of the nodes in V.

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 given layer is given by:

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

Note that edge weights are inversely proportional to structural distance, and assume values smaller than or equal to 1, being equal to 1 only if $f_k(u,v) = 0$.

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 $k^*\binom{n}{2} + 2n(k^* - 1)$ weighted edges.

3.3 Generating context for vertices

We will use the multilayer graph M to generate structural context for each node $u \in V$. The idea is 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 weighted 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. In particular, with probability $q > 0$ the random walk decides to stay 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 vertices that are structurally more similar to the current vertex, avoiding vertices that have very little structural similarity with the current vertex. Thus, the context of a node $u \in V$ is likely to have structurally similar nodes, independent of their labels and position on the original network G.

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

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

Also important, every time the walker steps within a layer it generates its current position as a vertex of V, 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. These node sequences generated by these walks form the context of node u.

3.4 Learning a language model

Recent language modeling techniques (CBOW, Skip-Gram, GloVe) 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, which can be the n precedent words or a centered window of size s, for example.

In particular, Skip-Gram 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 the node sequences generated by the random walk instead of word sentences. This can be viewed as adding an artificial ordering to the network, a property that is required by most language models.

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 s centered on it. Thus, we have the following objective function to be maximized:

$$P(v_{i-s},\ldots,v_{i-1},v_{i+1},\ldots,v_{i+s}|v_i) = \prod_{-s\leq t\leq s, t\neq 0} P(v_{i+t}|v_i) \tag{9}$$

where for the last step we assume that conditional independence. In the original formulation, $P(v_j|v_i)$ is defined as being proportional to the dot product of the latent features of v_j and v_i:

$$P(v_j|v_i) = \frac{\exp(\langle \Omega(v_j), \Phi(v_i)\rangle)}{Z_i} \tag{10}$$

Here, Φ and Ω map nodes to their correspondent latent features, and Z_i is the partition function for node v_i. Due to the computational complexity to calculate Z_i, alternative techniques such as Hierarchical Softmax and Negative Sampling have been proposed and widely used to speed up training.

For this work we use Hierarchical Softmax, where $P(v_j|v_i)$ is 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:

$$P(v_j|v_i) = \prod_{k=1}^{h} C(n(v_j,k), v_i) \tag{11}$$

where C is a binary classifier, commonly defined similarly to equation 10, with tree nodes having their own representations. 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 equations 11 and 9, using node sequences generated by the random walks as training data. Note that while we use Skip-Gram to learn node embeddings, virtually any technique to learn representations for text data could be used in its place.

3.5 Computational complexity

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, each value of $f_k(u,v)$ is the result of the 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 [18]. 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 $O(k^* n^3)$.

A practical optimization in constructing M is to consider all the edges in just the first few layers (e.g., 3 layers), but ignore some of the edges in the higher layers. The idea is that an edge with a large distance in layer k will necessarily have an equal or larger distance in layer $k+1$ (by monotonicity of $f_k(u,v)$). Note that large distances give rise to edges in M with very small weights, and thus are not very important to encode structural similarity. Thus, we compute $f_k(u,v)$ only if $f_{k-1}(u,v) \leq \tau$, for some fixed τ, otherwise we set $f_k(u,v) = \infty$. By controlling τ we can sparse out the distance calculations in the higher layers without compromising the framework.

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.

4.1 Barbell graph

We denote $B(h,k)$ as the (h,k)-barbell graph, which can be obtained by connecting two complete graphs K_1 and K_2 (each having h nodes) through a path graph P of length k. We choose two random nodes $b_1 \in V(K_1)$ and $b_2 \in V(K_2)$ to 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 joining the three graphs.

We use this specific network to illustrate how *struct2vec* works, since it 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 that maps one node to the other. 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 4 illustrates a $B(10,10)$ network, 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 be mapped to points that are close in the latent space. 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 p_2 (it suffices to compare their degrees).

Figure 3 shows the latent representations learned by *DeepWalk*, *node2vec* and *struct2vec* for the graph $B(10,10)$. *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,

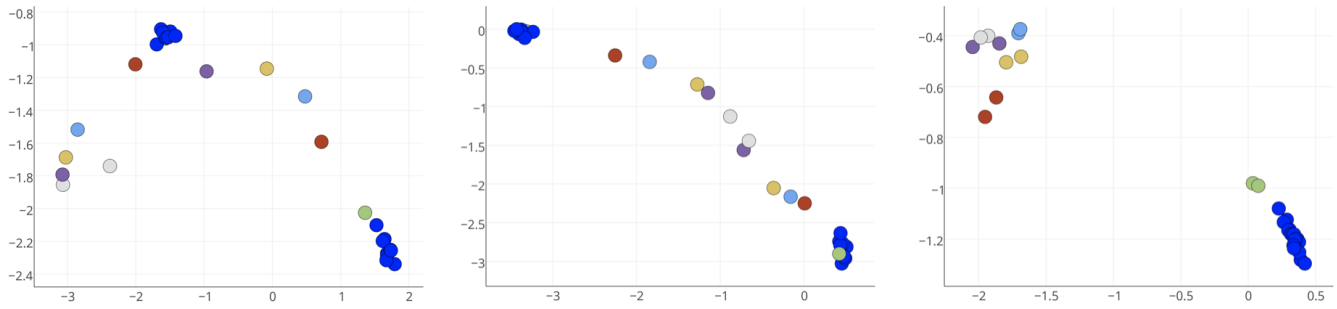


Figure 3: Latent representations in $\mathbb{R}^2$, for the Barball graph $B(10, 10)$, learned by (a) DeepWalk, (b) *node2vec* and (c) *struc2vec*. Parameters used for all methods: number of walks for node: 20, walk length: 80, window size of skip-gram: 5. For *node2vec*: $p = 1$ and $q = 2$.

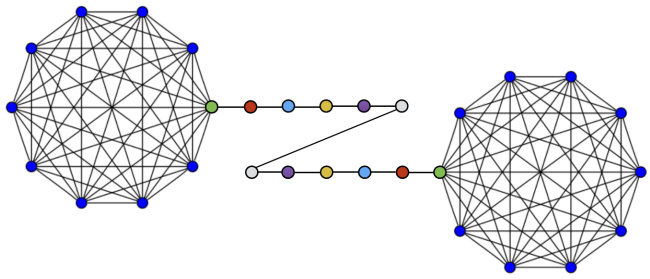


Figure 4: Barbell Graph $B(10, 10)$, composed of two complete graphs K with 10 nodes each and a path graph P of length 10.

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

struct2vec, on the other hand, learns features that properly separate the equivalent classes, and also maps structurally equivalent nodes (in the strong senses) to similar points in the latent space. Note that nodes of the same color are grouped together, and there is a clear distinction between two node groups: one composed of nodes $v \in V(P)$ (located to the left), and another of nodes $u \in V(K_1) \cup V(K_2)$ (to the right), with a small distinction for the endpoints of path P.

4.2 Karate network

The Zachary's Karate Club [21] is an unweighted undirected network composed of 34 nodes and 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 treated as indications of friendship between members.

We construct a graph composed of two copies G_1 and G_2 of the Karate Club network, where each node $v \in V(G_1)$ is a mirrored version of a 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 with the two baselines. Figure 6 shows the generated graph with mirrored node pairs exhibiting the same color.

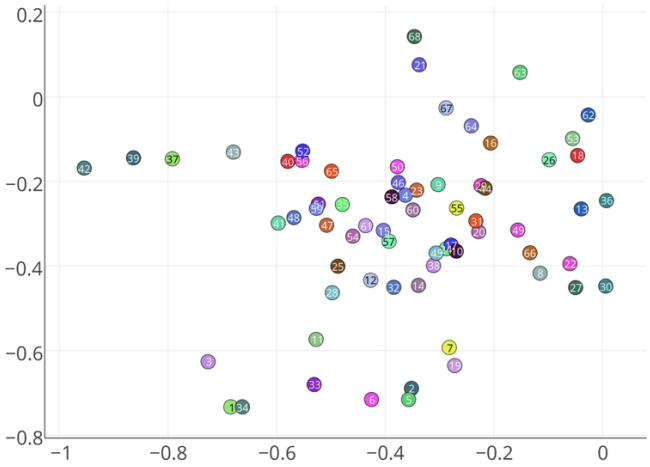
Figure 5 shows the representations learned by *DeepWalk*, *node2vec* and *struct2vec*. Clearly, *Deepwalk* and *node2vec* fail to group in the latent space structurally equivalent nodes, as was the case for the Barbell graph, including mirrored nodes.

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.

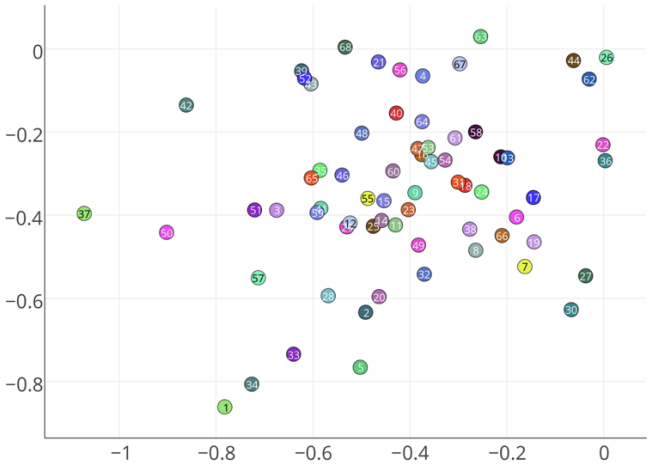
As an example, note that nodes 1, 34 and their correspondent mirrors (37 and 42) are in a separate cluster in the latent space. Interestingly, these are exactly the nodes that represent the club instructor Mr. Hi and his administrator John A. The network was gathered after a conflict between the two that split the members of the club which formed two groups – each centered on either Mr. Hi or John A. Therefore, nodes 1 and 34 have a truly specific – although similar – social role in the original network: they both act as leaders. Note that *struct2vec* captures this structural identity even though there are no edges connecting the two.

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.

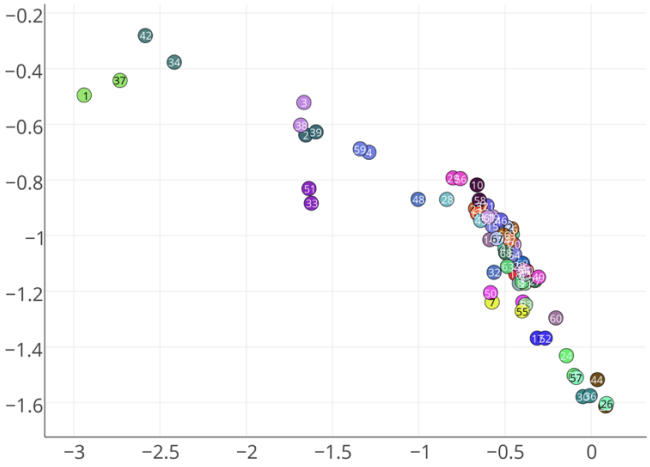
struct2vec also captures non-trivial structural equivalences. Note that nodes 7 and 50 (pink and yellow) are mapped to close points in the latent space. Surprisingly, these two nodes are structurally equivalent – there exists an automorphism in the graph that maps one into the other. This can be more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



(a) DeepWalk



(b) *node2vec*



(c) *struc2vec*

Figure 5: Mirrored Karate network representations created by (a) *DeepWalk*, (b) *node2vec* and (c) *struc2vec*. Parameters used for all methods: number of walks per node: 5, walk length: 15, window size of skip-gram: 3. For *node2vec* were used p=1 and q=2. *struc2vec* clearly identifies structurally equivalent nodes (mirrored nodes, with the same color) in the latent space.

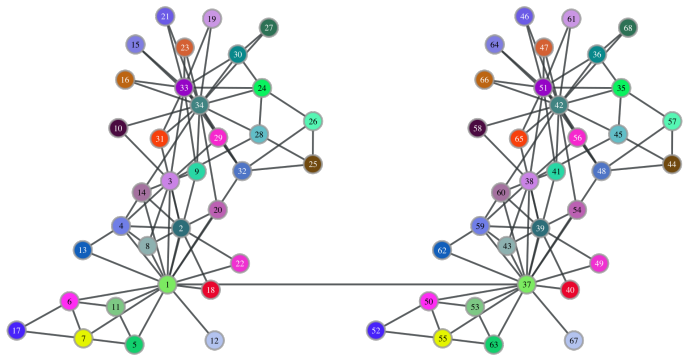


Figure 6: Mirrored Karate network. Colors correspond to mirrored nodes.

Analyzing how linear transformations in the latent space impact a node's structural identity is fundamental to further understand the learned manifold. Unlike *DeepWalk* and *node2vec*, our technique generates a latent space with a strongly dominant component: clearly, most nodes are spread among a line in the feature space. Note that linearity in this manifold has a direct correspondence to structural properties such as degree. For example, note that $\phi(42) - \phi(3) \approx \phi(3) - \phi(56)$ (where $\phi(i)$ is the latent representation of node i). This suggests that there is a *structural transformation* that maps node 56 to 3, and node 3 to 42. Indeed, it suffices to check each node's degree: $d(42) = 17, d(3) = 10, d(56) = 3$, and $d(42) - d(3) = 7 = d(3) - d(56)$. This is a strong indication that the latent space learned by *struc2vec* has fundamental aspects of the structural identity of nodes.

Table 1: Average and standard deviation for distances between node pairs in the latent space representation for the mirrored Karate network (see corresponding distributions in Figure 7).

Algorithms	Corresponding nodes avg (std)	All nodes avg (std)
DeepWalk	0.377 (0.184)	0.356 (0.195)
node2vec	0.407 (0.199)	0.372 (0.206)
struc2vec	0.129 (0.109)	0.722 (0.694)

Consider the distance between pairs of vertices in the latent representation. We measure the distance distribution between pairs corresponding to mirrored nodes and the distance distribution among all node pairs (using the representation shown in Figure 5). Figure 7 shows the two distance distributions for the representations learned by *node2vec* and *struc2vec*, with corresponding averages shown in Table 1. For *node2vec* the two distributions are practically identical, indicating no difference between distances among mirrored pairs and distances among all pairs. *DeepWalk* shows similar behavior (curves omitted for clarity) with averages shown in Table 1. In contrast, *struc2vec* exhibits two very different distributions: 94% of mirrored node pairs have distance smaller

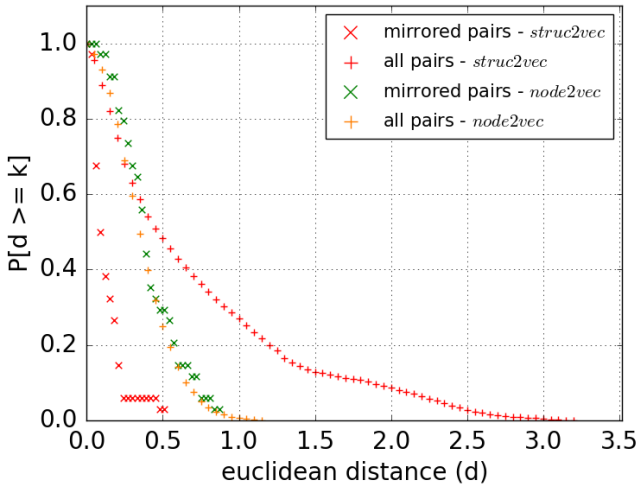


Figure 7: Distance distributions between pairs of nodes (mirrored pairs and all pairs) in the latent space, for the mirrored Karate network learned by *node2vec* and *struc2vec* (as shown in Figure 5). Curves marked with × correspond to distances between mirrored pairs while + corresponds to all pairs.

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 *DeepWalk* and *node2vec* (see Table 1).

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 latent representation. Results shown in Table 2 for the mirrored Karate network indeed corroborate that there is a very strong correlation between the two distances, for every layer, and 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 consider another scenario to illustrate the potential of the framework in effectively representing structural identity, even 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 that were subjected to random edge removal [13].

The model works as follows. Starting from a fixed graph $G = (V, E)$, we generate a graph G_1 by sampling each edge $e \in E$ with probability s, 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.

Table 2: 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 correlation (p-value)	Spearman correlation (p-value)
0	0.83 (0.0)	0.74 (0.0)
1	0.72 (0.0)	0.66 (0.0)
2	0.71 (0.0)	0.65 (0.0)
3	0.70 (0.0)	0.59 (0.0)
4	0.70 (0.0)	0.57 (0.0)
5	0.62 (0.0)	0.47 (2.40)
6	0.74 (0.0)	0.57 (2.37)
7	0.91 (0.0)	0.89 (2.45)

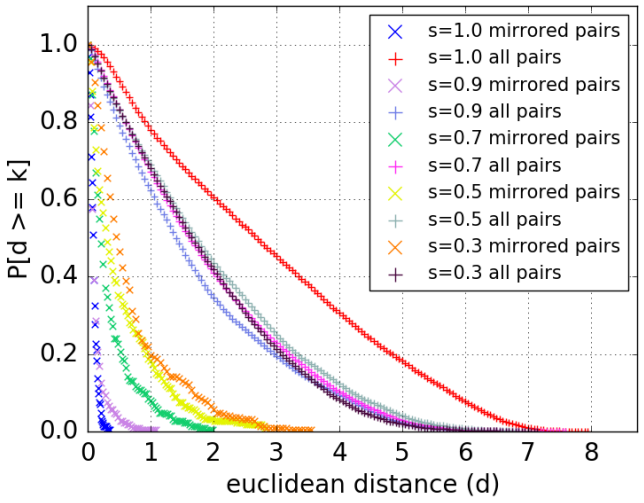


Figure 8: Distribution for distances between node pairs in latent space representation, under the edge sampling model (different values for s). Bottom curves (marked with ×) are distances between corresponding node pairs; top curves (marked with +) are distances between all node pairs.

We apply the edge sampling model to an *egonet* extracted from Facebook (224 nodes, 3192 edges, max degree 99, min degree 1) [8] to generate G_1 and G_2 with different values for s. We relabel the nodes in G_2 (as with the previous example), 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 node in G_2 (and vice-versa).

Figure 8 shows the distance distribution between node pairs in the latent space under various values for s (corresponding averages are shown in Table 3). In order to evaluate how well *struct2vec* captures structural identities in this setting, we compare the distance distributions between all node pairs and between only correspondent pairs.

For $s = 1$ (thus, G_1 is isomorphic to G_2), the two distance distributions are strikingly different, with the average distance for all pairs being 21 times larger than that for corresponding pairs (see Table 3). More interestingly, when $s = 0.9$ the two distributions are still very different. Note that while further 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 together corresponding nodes 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 3: Average and standard deviation for distances between node pairs in the latent space representation (see corresponding distributions in Figure 8).

s	Corresponding nodes 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)
0.7	0.338 (0.374)	1.975 (1.438)
0.5	0.528 (0.588)	1.994 (1.480)
0.3	0.674 (0.662)	1.962 (1.445)

5 CONCLUSION

Structural identity is a concept of symmetry in networks in which nodes are identified using just the network structure, along with their relationship to other vertices. The concept is strongly related to the notion of function, in which nodes tend to play particular roles in the network. Thus, identifying nodes with similar identity has long been investigated, in social sciences 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 without leveraging node or edge attributes, including node labels. It also uses a hierarchy to measure structural similarity at different scales, using ordered node degree sequence within the k-hop neighborhood node pairs. These structural distances are used to construct a multilayer weighted graph that encodes structural similarities among all nodes in the network. Random walks on this graph are used to generate the structural context for every node.

Learning representations for network nodes is a topic recently explored, including representations that aim to capture homophily and structural identity, such as *DeepWalk* and *node2vec*. However, as we have shown, these state-of-the-art techniques fail to learn representations that can effectively capture structural identity. *struc2vec* overcomes their limitations and excels in this task, in comparison. In contrast, *struc2vec* was not designed to capture node *homophily*, a common property in networks that can be leveraged for solving the (supervised) classification task. Thus, we do not attempt to classify nodes using *struc2vec*, a task that *node2vec* shows good performance at.

Can structural identity and homophily of nodes be adequately captured by a latent representations? On the one hand, structural identity is a concept independent of network position, while on the other hand, homophily is a concept tied to network proximity. Reconciling these two fundamental aspects of network nodes is an open and active research question.

REFERENCES

[1] Nir Atias and Roded Sharan. 2012. Comparative analysis of protein networks: hard problems, practical solutions. *Commun. ACM* 55, 5 (2012), 88–97.

[2] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A Neural Probabilistic Language Model. *Journal of Machine Learning Research* 3 (2003), 1137–1155.

[3] Vincent D Blondel, Anahí Gajardo, Maureen Heymans, Pierre Senellart, and Paul Van Dooren. 2004. A measure of similarity between graph vertices: Applications to synonym extraction and web searching. *SIAM review* 46, 4 (2004), 647–666.

[4] Francois Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. 2007. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on knowledge and data engineering* 19, 3 (2007).

[5] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *ACM SIGKDD*.

[6] Jon M Kleinberg. 1999. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46, 5 (1999), 604–632.

[7] Elizabeth A Leicht, Petter Holme, and Mark EJ Newman. 2006. Vertex similarity in networks. *Physical Review E* 73, 2 (2006), 026120.

[8] Jure Leskovec and Julian J Mcauley. 2012. Learning to discover social circles in ego networks. In *Advances in neural information processing systems*. 539–547.

[9] Francois Lorrain and Harrison C White. 1971. Structural equivalence of individuals in social networks. *The Journal of mathematical sociology* 1, 1 (1971), 49–80.

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR Workshop*.

[11] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems 26*. 3111–3119.

[12] Annamalai Narayanan, Mahinthan Chandramohan, Lihui Chen, Yang Liu, and Santhoshkumar Saminathan. 2016. subgraph2vec: Learning Distributed Representations of Rooted Sub-graphs from Large Graphs. In *International Workshop on Mining and Learning with Graphs*.

[13] Pedram Pedarsani and Matthias Grossglauser. 2011. On the privacy of anonymized networks. In *ACM SIGKDD*.

[14] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: Online Learning of Social Representations. In *ACM SIGKDD*.

[15] Narciso Pizarro. 2007. Structural Identity and Equivalence of Individuals in Social Networks Beyond Duality. *International Sociology* 22, 6 (2007), 767–792.

[16] 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 Transactions on Knowledge Discovery from Data (TKDD)* 7, 3 (2013).

[17] Lee Douglas Sailer. 1978. Structural equivalence: Meaning and definition, computation and application. *Social Networks* 1, 1 (1978), 73–90.

[18] S Salvador and P Chan. 2004. FastDTW: Toward accurate dynamic time warping in linear time and space. In *Workshop on Mining Temporal and Sequential Data, ACM SIGKDD*.

[19] Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M. Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *J. Mach. Learn. Res.* 12 (Nov. 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* 105, 35 (2008), 12763–12768.

[21] Wayne W Zachary. 1977. An information flow model for conflict and fission in small groups. *Journal of anthropological research* 33, 4 (1977), 452–473.

[22] Laura A Zager and George C Verghese. 2008. Graph similarity scoring and matching. *Applied mathematics letters* 21, 1 (2008), 86–94.