Random-Walk Graph Embeddings and the Influence of Edge Weighting Strategies in Community Detection Tasks

ANDREAS KOSMATOPOULOS, KOSTAS LOUMPONIAS, DESPOINA CHATZAKOU, THEODORA TSIKRIKA, STEFANOS VROCHIDIS, and IOANNIS KOMPATSIARIS,
Information Technologies Institute, Centre for Research and Technology Hellas, Greece

Graph embedding methods have been developed over recent years with the goal of mapping graph data structures into low dimensional vector spaces so that conventional machine learning tasks can be efficiently evaluated. In particular, random walk based methods sample the graph using random walk sequences that capture a graph's structural properties. In this work, we study the influence of edge weighting strategies that bias the random walk process and we are able to demonstrate that under several settings the biased random walks enhance downstream community detection tasks.

CCS Concepts: • **Computing methodologies** → *Learning latent representations*; • **Networks** → Online social networks.

Additional Key Words and Phrases: latent representation, deepwalk, node2vec, edge weighting, community detection

ACM Reference Format:
Andreas Kosmatopoulos, Kostas Loumponias, Despoina Chatzakou,, Theodora Tsikrika, Stefanos Vrochidis, and Ioannis Kompatsiaris. 2021. Random-Walk Graph Embeddings and the Influence of Edge Weighting Strategies in Community Detection Tasks. In *Proceedings of the 2021 Workshop on Open Challenges in Online Social Networks (OASIS '21), August 30, 2021, Virtual Event, Ireland.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3472720.3483621

1 INTRODUCTION

Over the past few years, there has been a notable increase in the volume of data produced and exploited by applications and services that handle various types of networks. Most of these networks, such as citation networks, sensor networks and, most notably, social networks, can be naturally modelled through graph data structures, with the networks' entities and relationships being represented by a graph's nodes and edges respectively. Subsequently, by performing graph analytics tasks, such as node classification [1], link prediction [13], and community detection [8], we can discover inherent characteristics of the network's nature and gain additional insight regarding the relationships of its entities.

Recently, graph embedding methods that provide a latent representation of the graph data in a low-dimensional vector space have been developed. These methods employ the graph's components (nodes, edges, and features or attributes) and produce a mapping into an embedding space that targets to preserve the graph's topology and overall structural properties (such as the pairwise distance between nodes). The resultant graph embeddings can then be utilized for analytics tasks that are based on conventional machine learning mechanisms (e.g. executing the k-means algorithm to obtain a partition of the graph's nodes).

Graph embedding methods that map graph nodes to vector spaces can be typically categorized into three types [6]: (i) matrix factorization methods, (ii) deep learning methods, and (iii) methods based on random walks. Factorization methods attempt to accurately decompose the graph's adjacency matrix into eigenvectors and eigenvalues, while deep learning methods employ multi-layer architectures to capture structural similarity between nodes. Finally, random walk methods sample node sequences by executing random walks among the graph's nodes and adopting the intuition that similar nodes will tend to coexist in several of the sampled sequences.

The two most prominent random walk based methods are DeepWalk [17] and node2vec [7]. The DeepWalk method samples a number of fixed-length random walks from each graph node which are then supplied as input to the skip-gram model of the word2vec word embedding technique [15, 16]. The skip-gram model learns vector representations such that words with a similar meaning in a corpus will end up closer in the embedding space, while less similar words will end up further apart. DeepWalk intuitively uses a "corpus" of sampled sequences so that nodes that frequently appear together in a random walk (given a context window of a user-defined size) are characterized by a small distance in the final embedding.

Node2vec [7] builds upon the core idea of DeepWalk with the main difference being the induction of bias in the random walk process. In particular, in each transition during a random walk, node2vec adds bias to the transition probabilities of the node's neighbors according to two user-defined parameters p and q. Parameter p defines the tendency of a random walk to follow a Breadth-First-Search approach, while parameter q enables a Depth-First-Search approach to the random walk.

In this work, we focus on random walk methods and study the utilization of edge weighting strategies as a means of inducing bias to the random walk generation phase. Edge weighting strategies recalibrate and modify the edge weights of a graph with the end goal of enhancing a particular downstream analytics task. To the best of our knowledge, this work constitutes the first attempt at enhancing specifically the community detection downstream task by utilizing edge weighting strategies that attempt to guide the random walks into having predominantly members that belong in the same community. The experimental evaluation showcases that, for a variety of configurations, our approach yields more accurate and coherent community detection results than those executed on graph embeddings derived from state-of-the-art random walk embedding methods.

2 FRAMEWORK

We begin by providing an outline of the broad framework and the proposed methodology before discussing the individual edge weighting strategies and their overall rationale.

2.1 Outline

Given an unweighted graph $G = (V, E)$, where V and E correspond to the graph's node and edge set, respectively, the objective is to provide a graph embedding that enhances community detection tasks performed by typical machine learning techniques. Thus, we employ weighting strategies that reweight edges between nodes according to a perceived likelihood of the nodes belonging to the same community.

Algorithm 1 DetectCommunities(G, $\mathcal{S}$, k)

```
Input: unweighted graph $G$, weighting strategy $\mathcal{S}$, number of communities $k$
Output: community designations $\mathcal{CD}$ for all nodes in $G$
1: $G' \leftarrow$ WeighGraph($G$, $\mathcal{S}$)
2: $Em_{G'} \leftarrow$ node2vec($G'$)
3: $\mathcal{CD} \leftarrow k$-means($Em_{G'}$,$k$)
4: return $\mathcal{CD}$
```

The outline of our framework can be seen in Algorithm 1. Initially, we reweight the graph according to a weighting strategy $\mathcal{S}$ and obtain the weighted graph G'. After obtaining the embedding $Em_{G'}$ using the node2vec algorithm we execute the k-means algorithm on the embedding to obtain community designations for each node in G.

Note that Algorithm 1 is an indicative description of the overall framework and the implementation details such as the graph embedding technique (e.g. DeepWalk, node2vec, etc.) or the community detection algorithm (e.g. k-means, GMM [2], etc.) may vary depending on the graph domain or the application requirements.

2.2 Edge Weighting Strategies

The majority of the edge weighting strategies presented in this work focus on enhancing algorithms based on community detection through modularity maximization. Additionally, they attempt to handle the resolution limit problem [5] that exists in modularity maximization approaches. In the remaining of the section, we use an edge e_{ij} between two nodes i and j as a running example. The four well-established and effective methods presented in this work are:

EBC_CNR The "EBC_CNR" method [10] weights a graph's edges according to two measures: their edge betweenness centrality (EBC) and common neighbor ratio (CNR). EBC corresponds to the number of shortest paths that go through e_{ij} while CNR reflects the percentage of common neighbors shared between nodes i and j. The exact weight of the edge is contributed by both EBC and CNR through two parameters α and β that are defined in either a manner that attempts to maximize the variance of the weight distribution or through heuristics. Thus, the weight W_{ij} of e_{ij} is:

$$W_{ij} = \begin{cases} \frac{b_{ij}^{-\alpha} \cdot C_{ij}^{\beta}}{\sum\limits_{\substack{k,m \\ k \neq m}} b_{km}^{-\alpha} \cdot C_{km}^{\beta}} & \text{if } A_{ij} = 1 \\ 0 & \text{if } A_{ij} = 0 \end{cases}$$

where $\alpha, \beta > 0$, A is the adjacency matrix of the graph, b_{ij} is the normalized EBC of e_{ij}, and C_{ij} is the CNR between nodes i and j.

SimRank The "SimRank" approach is based on the SimRank similarity measure [9] which states that "two objects are similar if they are related to similar objects". SimRank measures each pair of nodes based on the structural functionality or purpose they exhibit in the whole graph. Conceptually, in its iterative form the SimRank measure $S_k(i, j)$ between two nodes i and j in the k-th iteration of computation is equal to:

$$S_k(i,j) = C \cdot \frac{\sum\limits_{l=1}^{|N(i)|} \sum\limits_{m=1}^{|N(j)|} S_{k-1}\left(N_l(i), N_m(j)\right)}{|N(i)|\,|N(j)|}$$

where $N(i)$ corresponds to the neighbor set of i, $N_l(i)$ refers to a particular neighbor l of i and C signifies a decay constant. Additionally, $S_0(i,j) = 1$ if $i = j$ and $S_0(i,j) = 0$ otherwise. In the SimRank method, the weight of an edge in the graph is equal to the SimRank score between the edge's two endpoints.

κ-path The "κ-path" method is based on the calculation of the κ-path edge centrality measure [4] along with additional operations [3]. The κ-path edge centrality measure assigns weights to the edges according to their centrality and is defined as:

$$L^{\kappa}\left(e_{ij}\right) = \sum_{s \in V} \frac{\pi_s^{\kappa}\left(e_{ij}\right)}{\pi_s^{\kappa}}$$

with $\pi_s^{\kappa}\left(e_{ij}\right)$ being the number of simple random paths of at most κ nodes initiating from s that pass through e_{ij} and π_s^{κ} being the number of simple random paths of at most κ nodes that originate from s. Finally, the weight between two nodes is set equal to the Euclidean distance of their κ-path centrality measures[1]:

$$W_{ij} = \sqrt{\sum_{k=1}^{n} \frac{\left(L^{\kappa}\left(e_{ik}\right) - L^{\kappa}\left(e_{kj}\right)\right)^2}{d\left(k\right)}}$$

where $d\left(k\right)$ is the degree of node k.

AdaptiveMM Finally, the "AdaptiveMM" approach [14] follows a three step approach to generating weights for an unweighted graph. At first, an artificial network is generated with ground truth communities and with topological characteristics that resemble the original graph. This graph is then used as a basis for extracting a selection of local topological features from each edge such as the difference in clustering coefficients of the edge's endpoints or the Adamic-Adar index. In the last step, the edge features are supplied as input to a regression model that weights the edges in a way that a modularity maximization approach would be able to efficiently detect the ground truth communities of the artificial network.

Even though our approach is not related to the problem of modularity maximization in its general form, the methods presented above can be used as intuitive heuristics for the purpose of assigning significant weights to nodes that could potentially exist in the same community.

3 EXPERIMENTAL EVALUATION

In this section, we conduct experimental evaluation on the framework presented in Section 2 against the baselines of DeepWalk and node2vec. Since node2vec performed better than DeepWalk in all the experiments, we regard node2vec as the highest performing baseline. We begin by discussing implementation details before presenting the results on both synthetic and real-world datasets.

3.1 Implementation Details

Initially, we begin with an unweighted graph that is assigned weights through an edge weighting strategy. Following that, the graph embedding is obtained using node2vec and the k-means algorithm is executed to obtain the final communities.

During the weighting process some existing edges may end up with a zero weight and this may affect the random walk sampling process in two ways. In the first case, edges with zero weight are assigned a small weight equal to the smallest weight of the neighbors of the node being traversed divided by their total count. In the second case, if all the edges of a node's neighbors have zero weight, then they are all equally probable to be selected during a random walk.

The parameters used in the node2vec technique are $dimensions = 128$, $walk\_length = 80$, $n_{walks} = 10$ (number of walks from each node), $window\_size = 10$. The parameters p and q were evaluated for each dataset using a grid search over $[0.25, 0.5, 1, 2, 4]$ as per the suggestions in [7]. In the "κ-path" method we set κ to 20 and in the "SimRank" method we set C to 0.8. In the case of "EBC_CNR" the parameters α and β were evaluated explicitly for each dataset using the heuristics described in [10]. All of the parameters selected above follow the suggestions of the authors in their respective original work.

3.2 Synthetic Datasets

We implemented a selection of LFR networks [11] with varying node counts and community sizes, and tested the performance of our framework for different values of the mixing parameter μ. Table 1 details the synthetic datasets used where n is the number of nodes, d_a and d_m are the average and maximum vertex degree, c_{min} and c_{max} are the minimum and maximum community sizes, and $\mu \in \{0.25, 0.35, 0.45, 0.55\}$. The exponent for the degree power law sequence was 2, while for the community size sequence was 3. In each experiment we measure

[1]This represents the pairwise proximity between two nodes, due to the definition of $L^{\kappa}\left(e_{ij}\right)$ [3].

Table 1. LFR Datasets

Name	**n**	**d_a**	**d_m**	**c_{min}**	**c_{max}**
LFR_1K_5_10_5_15	1K	5	10	5	15
LFR_1K_5_10_15_25	1K	5	10	15	25
LFR_5K_5_10_5_15	5K	5	10	5	15
LFR_5K_5_10_15_25	5K	5	10	15	25

Table 2. Highest improvement for each metric per dataset over node2vec (averaged over ten iterations). All results are obtained by the "AdaptiveMM" method (except those marked with † which are by the "κ-path" method) and their increase is statistically significant ($p < 0.05$) with respect to the results of node2vec.

Name	**ARI**	**NMI**	**Mod.**
LFR_1K_5_10_5_15	+9.5% (μ=0.45)	+1.7% (μ=0.45)	+3.5% (μ=0.55)†
LFR_1K_5_10_15_25	+5.5% (μ=0.35)	+2.2% (μ=0.35)	+3.7% (μ=0.55)†
LFR_5K_5_10_5_15	+13.0% (μ=0.45)	+1.6% (μ=0.45)	+2.5% (μ=0.45)
LFR_5K_5_10_15_25	+15.4% (μ=0.35)	+3.1% (μ=0.35)	+2.5% (μ=0.35)

the Adjusted Rand Index (ARI) and Normalized Mutual Information (NMI) measures, along with the graph's modularity on the final partition. The ARI and NMI measures are estimated after ten instances of the k-means algorithm with different centroid seeds and k being equal to the respective's datasets ground truth communities count.

Figure 1 presents our results where several observations can be made. "AdaptiveMM" consistently outperforms the rest of the methods and the baselines, while "DeepWalk" and "SimRank" achieve similar effectiveness, but are outperformed by the rest of the methods in the majority of the experiments across all measures. The effectiveness of our framework in the ARI measure increases for graphs with a higher node count. Finally, all methods, except "SimRank", achieve higher modularity than the baselines for $\mu < 0.5$, (i.e. communities with strong connections where a node has more neighbor nodes inside its' community than the rest of the graph), while "κ-path" achieves the highest modularity for $\mu = 0.55$ among all methods. Table 2 summarizes the best results depicted in Figure 1 for each metric in each dataset.

3.3 Real-world Datasets

Complementary to the experiments on synthetic datasets, we also performed experiments on real-world datasets equipped with ground-truth communities and, more specifically, a product network and two social networks originating from the SNAP Dataset Collection [12].

The "ego-Facebook" dataset represents a set of social circles in the Facebook social network. Nodes and edges in this network represent users and friendship relationships between them respectively. The "Amazon" dataset consists of products found in the Amazon website that are linked if they are frequently bought together. Products belong in the same ground-truth community if they are characterized by the same product category defined by Amazon. Similarly to "ego-Facebook", the "Youtube" dataset contains friendship links between users in the video-sharing website Youtube. Ground-truth communities correspond to user-formed group communities. Note

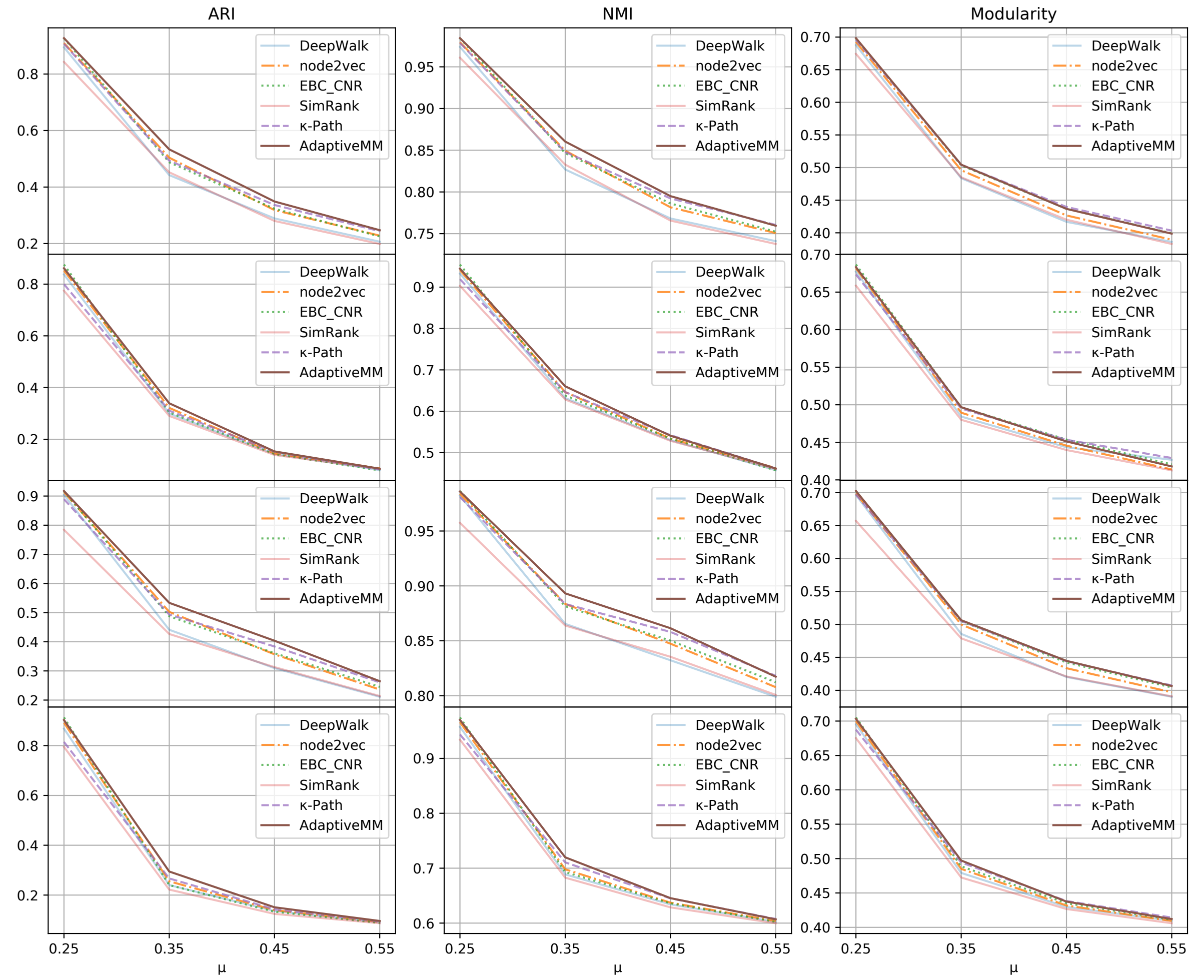

Fig. 1. Results on LFR networks (averaged over ten iterations). From top to bottom, each of the four rows corresponds, respectively, to: a) LFR_1K_5_10_5_15, b) LFR_1K_5_10_15_25, c) LFR_5K_5_10_5_15, d) LFR_5K_5_10_15_25.

.

that in all three datasets a node may belong to more than one ground-truth community so for the purposes of this experimental evaluation we restrict each node to one ground-truth community assignment and disregard the rest of the assignments.Similarly to the synthetic experiments, we set *k* equal to the respective's datasets ground truth communities count.

In the "ego-Facebook" dataset we omitted nodes without a community assignment and nodes without any edges. In the "Amazon" and "Youtube" datasets we focused on the top 5000 communities with highest quality [18] discarding nodes and edges that were not a member in any of the top 5000 communities while also removing duplicate communities with completely identical members. Table 3 showcases the resulting real-world datasets used in the evaluation.

Table 3. Real-world datasets

Datasets	Nodes	Edges	Communities
ego-Facebook	2.871	62.334	147
Amazon	16.716	48.739	1.229
Youtube	39.841	224.235	4.447

Table 4. Experimental evaluation on real-world datasets. The first number in each cell refers to the mean metric value (over ten iterations), while the second number to two standard deviations. The best performance in each metric for each dataset is denoted in bold. Results marked with “*” provide a statistically significant ($p < 0.05$) increase over the results of node2vec.

Dataset	**ego-Facebook**			**Amazon**			**Youtube**		
Metric	**ARI**	**NMI**	**Mod.**	**ARI**	**NMI**	**Mod.**	**ARI**	**NMI**	**Mod.**
DeepWalk	.311 ±.053	.738 ±.006	.402 ±.075	.530 ±.030	.955 ±.003	.965 ±.003	.066 ±.004	.762 ±.001	.217 ±.002
node2vec	.341 ±.036	.743 ±.005	.433 ±.044	.535 ±.024	.956 ±.001	**.967 ±.002**	.072 ±.022	.762 ±.001	.214 ±.001
EBC_CNR	**.361 ±.056***	**.746 ±.005***	**.455 ±.074***	.539 ±.018	.956 ±.002	.966 ±.003	**.088 ±.014***	**.767 ±.001***	**.232 ±.003***
SimRank	.332 ±.040	.729 ±.004	.350 ±.032	.540 ±.018	.956 ±.001	.965 ±.002	.078 ±.013	.755 ±.001	.183 ±.002
κ-path	.285 ±.026	.731 ±.004	.359 ±.037	.560 ±.011*	.957 ±.001*	**.967 ±.002**	.069 ±.003	.742 ±.001	.150 ±.001
AdaptiveMM	.306 ±.029	.739 ±.005	.387 ±.018	**.561 ±.013***	**.958 ±.001***	**.967 ±.001**	.076 ±.004	.760 ±.001	.184 ±.002

Table 4 details the results of the experimental evaluation on the real-world datasets. The two best performing approaches across all datasets and metrics are “EBC_CNR” and “AdaptiveMM” while in each dataset the best values for each metric are achieved by the same approach. With the exception of the modularity metric in the “Amazon” dataset, each metric is increased in a statistically significant ($p < 0.05$) improvement by at least one edge weighting strategy. The highest difference is on the “Youtube” dataset where “EBC_CNR” achieves +8.4% higher modularity than node2vec while the lowest difference is on the the “Amazon” dataset where “AdaptiveMM” and node2vec have nearly identical modularity.

4 CONCLUSIONS

The ubiquitous nature of networks and their ease of representation as graphs has led to several graph analytics tasks that seek to discern information about the characteristics of the network. By mapping graphs into vector spaces, classical machine learning algorithms can be efficiently applied to gain additional insight about a network’s features. In this work, we studied random walk graph embedding methods and the influence of edge weighting strategies in community detection. We used four intuitive state-of-the-art strategies and experimentally demonstrated that under several settings the utilization of edge weighting strategies can lead to improved performance according to the ARI, NMI and modularity measures.

ACKNOWLEDGMENTS

This research has received funding from the European Union's H2020 research and innovation programme as part of the CONNEXIONs (GA No 786731) and PREVISION (GA No 833115) projects.

REFERENCES

[1] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. 2011. Node classification in social networks. In *Social network data analytics*. Springer, 115–148.

[2] Christopher Bishop. 2006. Pattern Recognition and Machine Learning. *Pattern Recognition and Machine Learning* (2006).

[3] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Alessandro Provetti. 2011. Generalized louvain method for community detection in large networks. In *2011 11th international conference on intelligent systems design and applications*. IEEE, 88–93.

[4] Pasquale De Meo, Emilio Ferrara, Giacomo Fiumara, and Angela Ricciardello. 2012. A novel measure of edge centrality in social networks. *Knowledge-based systems* 30 (2012), 136–150.

[5] Santo Fortunato and Marc Barthelemy. 2007. Resolution limit in community detection. *Proceedings of the national academy of sciences* 104, 1 (2007), 36–41.

[6] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.

[7] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. ACM, 855–864.

[8] Muhammad Aqib Javed, Muhammad Shahzad Younis, Siddique Latif, Junaid Qadir, and Adeel Baig. 2018. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications* 108 (2018), 87–111.

[9] Glen Jeh and Jennifer Widom. 2002. SimRank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 538–543.

[10] Alireza Khadivi, Ali Ajdari Rad, and Martin Hasler. 2011. Network community-detection enhancement by proper weighting. *Physical Review E* 83, 4 (2011), 046104.

[11] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical review E* 78, 4 (2008), 046110.

[12] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[13] Linyuan Lü and Tao Zhou. 2011. Link prediction in complex networks: A survey. *Physica A: statistical mechanics and its applications* 390, 6 (2011), 1150–1170.

[14] Xiaoyan Lu, Konstantin Kuzmin, Mingming Chen, and Boleslaw K Szymanski. 2018. Adaptive modularity maximization via edge weighting scheme. *Information Sciences* 424 (2018), 55–68.

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[16] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems* 26 (2013), 3111–3119.

[17] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. DeepWalk: online learning of social representations. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*. ACM, 701–710.

[18] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (2015), 181–213.