TÜBİTAK

Turkish Journal of Electrical Engineering & Computer Sciences

http://journals.tubitak.gov.tr/elektrik/

Research Article

Turk J Elec Eng & Comp Sci
(2022) 30: 1868 – 1881
© TÜBİTAK
doi:10.55730/1300-0632.3910

Degree-based random walk approach for graph embedding

Sarmad N. MOHAMMED[1,2,*], **Semra GÜNDÜÇ**[2]
[1]Department of Computer Science, College of Computer Science and Information Technology, University of Kirkuk, Kirkuk, Iraq
[2]Department of Computer Engineering, Faculty of Engineering, Ankara University, Ankara, Turkey

Received: 08.12.2021 • **Accepted/Published Online:** 26.04.2022 • **Final Version:** 22.07.2022

Abstract: Graph embedding, representing local and global neighbourhood information by numerical vectors, is a crucial part of the mathematical modeling of a wide range of real-world systems. Among the embedding algorithms, random walk-based algorithms have proven to be very successful. These algorithms collect information by creating numerous random walks with a predefined number of steps. Creating random walks is the most demanding part of the embedding process. The computation demand increases with the size of the network. Moreover, for real-world networks, considering all nodes on the same footing, the abundance of low-degree nodes creates an imbalanced data problem. In this work, a computationally less intensive and node connectivity aware uniform sampling method is proposed. In the proposed method, the number of random walks is created proportionally with the degree of the node. The advantages of the proposed algorithm become more enhanced when the algorithm is applied to large graphs. A comparative study using two networks, namely CORA and CiteSeer, is presented. Compared with the fixed number of walks case, the proposed method requires approximately 50% less computational effort to reach the same accuracy for node classification and link prediction calculations.

Key words: Graph representation learning, node embedding, feature learning, random walk

1. Introduction

Networks are ubiquitous and are the main infrastructure for modeling biological, physical as well as social systems. In any particular event, the determination of the roles of the nodes becomes crucial for both better understanding the dynamics and, if necessary, taking the prevention measures (for example in the epidemic case who spreads the disease faster, in the social network who influences the others, etc.). Hence the networks are crucial tools for modeling and understanding the real-world systems by the introduction of a device to map elements of the graph into a low-dimensional representation. To achieve this goal, it is important to preserve the local and global properties of individual nodes in a very wide range of graph structures that are seen in real-world and artificial networks.

Mathematical modeling of such a wide range of phenomena requires a good description of the underlying connectivity structure among the nodes. The connectivity of the nodes determines the pattern of the interactions and dynamics. For example in social networks, to recommend a new friend or to predict the role of a person needs different approaches mainly sourced from the topology, like considering the number of common neighbors, strength of the link, or being in the same community. Main tasks include node classification [1, 2], link prediction [3, 4], anomaly detection [5, 6], and community detection [7]. Many machine learning algorithms [8–10] have

*Correspondence: sarmad_mohammed@uokirkuk.edu.iq shown to be very successful in prediction, classification, reconstruction, and various more complicated tasks. For the success of each task specially prepared input data is necessary. Graphs are different data structures. They have richer features than usual inputs of machine learning algorithms such as pictures or categorical items. Hence, an algorithm to capture the information behind this high dimensional data to introduce the machine learning algorithms as the input vector is necessary.

Traditional approaches for extracting the structural features of graphs use statistical techniques [11–15] (number of neighbors, clustering coefficient, centrality measures, etc.). Traditional methods are time-consuming and they are short of fully exploiting the hidden features of graphs. Recently a new idea has been employed to embed the nodes into a low dimensional vector space by using machine learning techniques, namely representation learning. In the literature, a wide range of approaches has been presented ([16] for a survey). The method imported from natural-language processing has been employed in embedding the connectivity structure of graphs into n-dimensional vectors. This approach uses the similarities between the structure of natural languages and the connectivity structure of graphs.

Some of the successful methods use random walks to collect the nodes' local and global connectivity structures. These models have attracted much attention [17–21]. The idea behind the use of random walk statistics is to enlist the cooccurrence of nodes. The neighbors of a given node are visited on the repeated random walks. The cooccurrence of the nodes in the same random walk ensures the connectivity of the group of nodes. The information collected during random walks is used in analogy with the natural language processing algorithms. Well-known random walk-based embedding methods are DeepWalk [17] and node2vec [18]. Both methods are shallow embedding and follow an optimization strategy by using cooccurrence statistics.

Many recent studies are using the random walk method. These studies perform various network analysis tasks by reducing the number of training samples and computational cost with various random walk approaches. In [22, 23], nodes are embedded based on specified characteristics of random walks that begin at one node and proceed to the next. Role2vec [22] samples a corpus using attributed random walks, which preserves each node's structural type as well as increases the efficiency of available space. Based on the structural patterns of nodes in each surrounding, RiWalk [23] learns an embedding for each node by building a relabeled subgraph for each node and performs random walks on the subgraph created. As a scalable graph embedding approach, DiaRW [24] has recently been proposed as one that uses a degree-biased random walk and variable lengths policy. DiaRW method creates random walks depending on the source node's centrality in reducing sampling redundancy. To modify the node preferences when walking, BalNode2Vec [25] defined the concept of a node's network neighborhood and developed a balanced random walk process that adapts to the graph's structure. Fazaeli and Momtazi [26] presented GuidedWalk, the idea behind this method is to increase the probability of visiting the local structure of nodes in the same class by using the label information of the nodes in the random walk phase. Wang et al. [27] proposed the HashWalk method to preserve the network topology and improve node embedding. This method first compresses the cliques in the network into single nodes and then obtains the compressed clique sequences using the random walk method. However, even though the state-of-the-art random walk-based embedding methods have accomplished some benefits, they cannot disregard the restrictions imposed by randomness in the walking process. For example, eliminating oversampling (number of random walks being chosen proportional to the degree of the node).

The embedding strategy guarantees that the similarities of the nodes are related to the similarities of corresponding embedding vectors. Both node2vec and DeepWalk algorithms use the same random walk approach to collect information from the graph. The difference between DeepWalk and node2vec is that they use different optimizations and approximations to compute the embeddings. Besides that, the two methods use the same user-defined initial values for the number of walks, and the walk length to capture the features. These values are set by the user at the beginning of the data collection stage and optimized values can only be achieved with trial and error.

Among the real-world networks, scale-free networks have a special place since they represent the majority class among the real-world networks [28]. The characteristic power-law degree distribution of the scale-free networks is an indication of many nodes with a low degree and a few nodes with a high degree. For this reason, the fixed number of random walks oversamples the low degree while high degree nodes cannot be sampled as necessary. In case all nodes are considered on equal footing, to increase the sampling rate of high degree nodes, it is necessary to increase the number of walks that start from every node. This situation corresponds to an increase in the computation time proportional to the size of the network. Meantime it results in an excessive number of walks for low degree nodes. This situation results in the creation of imbalanced data during the embedding stage.

The present work aims to propose a modification in the feature extracting algorithm. To eliminate oversampling, the number of random walks is chosen proportional to the degree of the node. Low degree nodes are sampled relatively less than higher degree nodes. Considering the proportions of the low and high degree nodes in scale-free networks, this choice reduces the total number of walks considerably while increasing the sampling rate of the high degree nodes. This modification introduces remedies for excessive computational time with the growing network size and it also helps to overcome imbalanced data problems. In the proposed model the number of walks that start from each node, tuned to be proportional with the degree of the related node instead of a fixed number, while for the rest of embedding the node2vec strategy is employed. The paper is organized as follows: the next chapter is devoted to the explanation of the method. The results of the comparative study of the proposed algorithm are presented in the third section. The final section consists of the discussions over the results and conclusions.

2. Method

The successful implementation of representation learning in natural language processing domains (e.g., word2vec [29, 30]) paved the way for new directions in network representation learning by optimizing the neighborhood preserving likelihood concept. A document is a collection of words. In word2vec the features of the words are extracted from the relative occurrence with the related, or in other words, nearby words (being in the same window size). A text document consists of already existing sentences. The sentences are the natural constructs representing the relations among the words. There are no such sequences or grouping for the nodes of the networks. The difficulty of using representation learning techniques on the graphs appears to be constructing sentence-like structures. To obtain such structured constructions which introduce an order for nodes and reveals hidden features, random walks are the best candidates and often employed.

Graphs, (G) essentially consist of two components; nodes (N) and edges (E). Nodes are entities that are connected by edges. There is a neighbor set for each node (n_i) in the graph. These neighbors will be determined by a set of random walks with some strategy. This strategy must guarantee that local and global neighborhood information is integrated into the embedding. As shown in Figure 1, third node has 5 neighbors, $N_3 = \{n_1, n_2, n_4, n_5, n_6\}$, and if we start two random walks from the node n_3, following the same strategy (the walk length is 4), then the first walk (W_1) can be $(n_3 \rightarrow n_1 \rightarrow n_2 \rightarrow n_3 \rightarrow n_5)$ and the second walk (W_2) can be ($n_3 \rightarrow n_5 \rightarrow n_6 \rightarrow n_7 \rightarrow n_8$). These walks will give sentence-like sequences:

$$S_1 : \quad n_3\ n_1\ n_2\ n_3\ n_5.$$
$$S_2 : \quad n_3\ n_5\ n_6\ n_7\ n_8.$$

The idea behind creating sentence-like node sequences is to represent the similarities between the nodes with the corresponding embedding vectors. Every random walk produces a sequence that possesses the inherent hidden traits between the nodes. The walk length enables to penetrate deeper into the graph while the number of walks adds different connectivity patterns. Hence, more accurate embedding vectors can be obtained. So there must be enough walks with sufficient length to map the different paths start from each node.

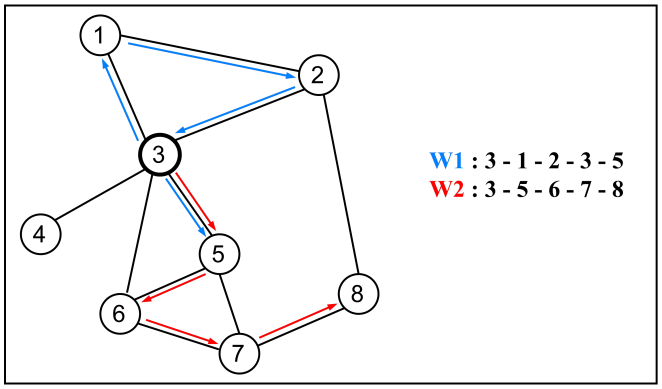


Figure 1. A representation of 2 random walks starting from node n_3 on a sample graph.

To detect the similarity of the nodes in a graph both connectivity and structural resembles must be taken into account. In the node2vec model, the parameters p and q allow sampling local and global neighborhoods of nodes in the walks. The parameter p controls the probability of returning to the previous node while q value makes visiting the far-away neighbors possible. For low p values, it results in revisiting already visited neighbors for through the graph.

In a fixed number of walk cases, the same number of walks start from each node, respectively. In the case where the number of walks proportional with the degree of the node, the number of walks started from a node is given as,

$$NW_i = NWPD \times k_i, \tag{1}$$

where NW_i and k_i are the number of walks started and the degree of the i^{th} node. $NWPD$ is the number of walks per degree. Figure 2 presents an illustration of the difference in random walk strategy between degree-based node2vec and node2vec in a sample network. Algorithm 1 shows the pseudocode for creating random walks per node using degree-based weighted sampling approach. This approach provides more even sampling the network, since the nodes have varying number of neighbors. Moreover, for large networks, it increases the sampling efficiency. For large scale-free networks, the probability distribution is given by

$$P(k) = (\gamma - 1) \times k_{min}^{\gamma-1} \times k^{-\gamma}, \tag{2}$$

where k_{min} and γ are the degree of least connected node and degree exponent, respectively. Hence, the average degree, $< k >$ is

$$\begin{aligned} \langle k \rangle &= \int_{k_{min}}^{k_{max}} \times k \times p(k) \\ &= (\gamma - 1) \times k_{min}^{\gamma-1} \times \frac{k_{max}^{2-\gamma} - k_{min}^{2-\gamma}}{2-\gamma}, \end{aligned} \tag{3}$$

where k_{min} and k_{max} are the minimum and maximum degrees exist among the nodes, respectively. For scale-free networks, the relation between minimum, and maximum degrees can be given as

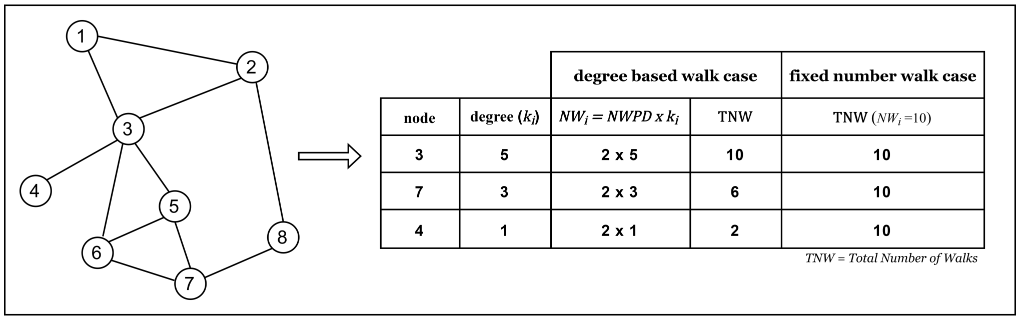
$$k_{max} = k_{min} \times N^{1/(\gamma-1)}, \tag{4}$$

where N is the number of nodes. Using the maximum degree relation (Eq. 4) in Equation 3, the average degree is given by

$$\begin{aligned} \langle k \rangle &= \frac{\gamma-1}{2-\gamma} \times \left(N^{-\frac{\gamma-2}{\gamma-1}} - 1\right) \times k_{min} \\ lim_{N\to\infty} \langle k \rangle &\sim \frac{\gamma-1}{\gamma-2} \times k_{min}, \end{aligned} \tag{5}$$

where $2 < \gamma < 3$.

For large scale-free networks the average degree is proportional with the minimum degree, k_{min}. Hence, for the degree-based approach, the total number of random walks, *TNW*, is given by the relation, $NWPD \times N \times < k >$, where *NWPD* is the number of random walks per degree. Even though the total number of random walks grows with the network size, N, the minimum degree proportionality (Eq. 5) ensures the quality of embedding with minimum possible computational effort. In this approach, each edge becomes starting edge for random walks with equal probability which eliminates oversampling of the fixed number of random walk approach.



		degree based walk case		fixed number walk case
node	degree (k_i)	$NW_i = NWPD \times k_i$	TNW	TNW (NW_i=10)
3	5	2 x 5	10	10
7	3	2 x 3	6	10
4	1	2 x 1	2	10

TNW = Total Number of Walks

Figure 2. An illustration of counting differences between degree-based and fixed number of random walks.

For a fixed number of random walk cases, the number of random walks at each node must increase with the network size since the maximum degree increases with the size (Eq. 4). Unless this is done, hub nodes cannot be sampled accurately. To achieve the same accuracy, number of walks must increase as the size increases. Hence, the computational effort does not increase linearly with size. Moreover, starting an equal number of random walks from each node may result in creating very similar sequences for low degree nodes. This oversampling is redundant information and becomes the source of imbalanced data at the embedding stage.

Algorithm 1 Degree-based sampling.

```
Input: Network $G = (V, E)$
Initialization:
  Number of nodes $N$;
  Array of node degrees $D$;
  Length of the walk $WL$;
  Number of walks per degree $WD$;
  Return-weight parameter $p$;
  Neighbor weight parameter $q$;

*Section to create degree based random walks per node:
    for node in $N$ do                                                    ▷ Loop over the nodes
        StartNode = node                                                  ▷ Starting node
        epochs = $WD$ * $D$[node]                ▷ How many times to start a walk from the node
        Walk[node] = Random Walk($G$, StartNode, $WL$, epochs, $p$, $q$)
    endfor
```

In the natural language case, the natural flow of the sentences determines the word connectivity structure. While some words occur frequently, less common words rarely take place in the sentences. Hence an embedding algorithm, that processes meaningful sentences, naturally distinguishes common words (central nodes, nodes with high degree) from the less common words (nodes with low degree). In the graph representation algorithms, such a natural distinction can only be imposed by the connectivity structure of the graph. Simplest and the most apparent choice is the degree of the nodes. If the number of walks which start from any node can be proportional with its degree, the appearance of these nodes will be proportional to the connectivity structure of the node. This approach has three advantages:

1. The nodes with a high degree will have more emphasis on the sentence-like structures, which create weighted embedded vectors. Such a degree-based approach combined with the stochastic nature of the random walk corresponds to importance sampling that improves the efficiency.

2. The abundance of low degree nodes creates an imbalanced data problem. The high contribution of low degree nodes shadows some features of the graph; by choosing degree proportionality in the number of random walks, the data imbalance problem will be eliminated.

3. Choosing degree-based random walks grossly reduces the computation time. So it can be possible to work with large-size graphs.

The proposed model uses the node2vec strategy to visit the neighbors, with the difference that the number of walks that start from each node is proportional with the degree of the node instead of being a fixed number.

3. Experimental results and analysis

The effect of the fixed and degree-based number of random walks is tested by using three different networks. The first one of these networks is Zachary's karate club network [31]. This small and well-studied network is used as a test case and also a platform to discuss the implications of degree-based and the fixed number of random walks on a larger network. Two real-world networks, CORA [32] and CiteSeer [33] are chosen as test environments for the real-world networks. Node classification and link prediction are used as application areas. Table 1 presents the details of the datasets used for the experimental analysis.

Table 1. Overview of Zachary's karate club and two citation networks datasets.

Dataset	$\|N\|$	$\|E\|$	Classes
Zachary's karate club	34	78	2
CORA	2,708	5,429	7
CiteSeer	3,327	4,732	6

3.1. Zachary's karate club network

Fixed and degree-based random walk approaches are employed for calculating the embedding vectors of the Zachary's karate club network. Apart from some detailed studies [34], the Zachary's karate club network is commonly considered as a two community network. For comparison of the quality of embedding, with the least number of random walks, two criteria are considered: i) cosine similarity among the embedding vectors and ii) identification of the communities in the network using dimensional reduction techniques. For both cases, the number of walks is increased until comparable results are achieved. When the target is reached the computational efforts are compared.

3.1.1. Cosine similarity studies

One way of checking the embedding quality is the identification of the most similar node by using the cosine similarity. Embedding vectors are calculated by keeping the window size ($WS = 5$) and embedding vector dimensionality (32) fixed. For each node, the cosine similarities with the rest of the nodes are calculated regardless of being connected or not. The nodes with maximum cosine similarity created a topologically similar set of nodes. Cosine similarity results are presented by Figure 3. As it is seen, the similarity calculations indicate two distinct communities. Moreover, the communities consist of the nodes that are shown as the communities of the Zachary's karate club network [31]. In the degree based case 5 walks per degree (total number of 780 walks) has been sufficient for such separation of the communities. The same result is achieved 40 walks per node (total of 1360 walks) for the fixed random of walks case. The difference in the number of walks is a clear indication of the gain in the computational time for larger networks.

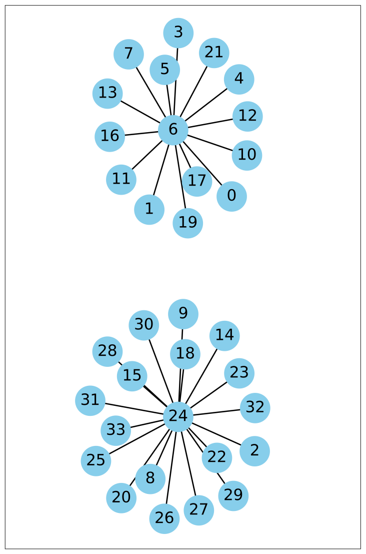
3.1.2. Community detection by using dimensional reduction

Figure 3 shows that even a simple similarity study reveals the communities in accord with the well-established community structure of the network. Multidimensional scaling (MDS) and k-means algorithms of the python NetworkX[1] package are used for dimensional reduction and clustering. Embedding vector space is mapped onto 2-dimensional vectors where the K-Means algorithm is used for community identification. The quality of the embedding is the assurance of the resemblance of high dimensional and low dimensional vector spaces. Figure 4 shows communities and their constituent nodes. For a fixed and degree-based number of walk cases, embedding vector dimension, the walk length and window size are 32, 10, and 5, respectively. For similar community identification, 1360 walks are realized for the fixed number of random walks case while for the degree-based approach total of $5 \times \mathrm{k_i} = 780$ walks are observed to be sufficient.

Figures 3 and 4 show that even for small graphs, the degree-based random walk approach is less computationally intensive. As the size increases, the power-law behavior of scale-free networks becomes an

[1]NetworkX developers (2021). NetworkX: Network Analysis in Python [online]. Website https://networkx.org/ [accessed 17 September 2021].

indicative factor for the increase of the low-degree nodes. Hence, the advantage of the degree-based approach becomes more emphasized for large networks.


Degree-based: 5 walks per degree.

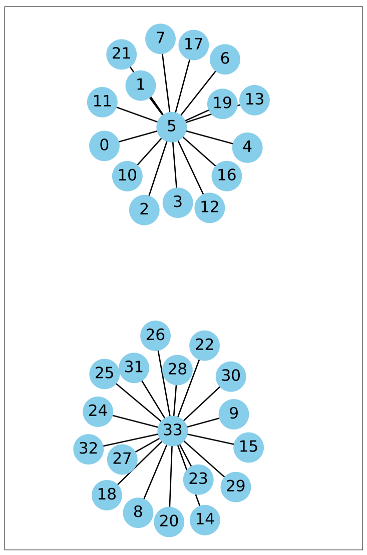
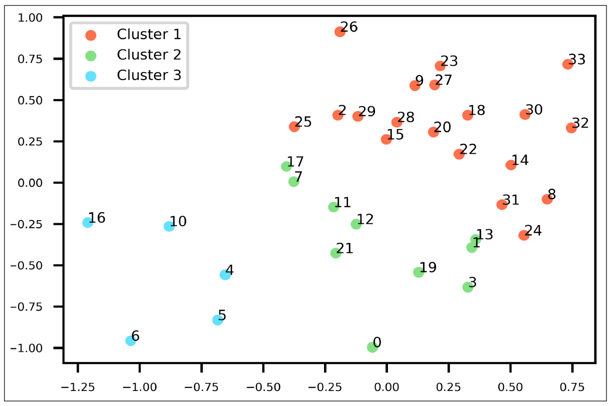

Fixed-number: 40 walks per node.

Figure 3. Cosine similarity values (a) using $5 \times$ degree and (b) 40 random walks started at each node.


Fixed number of walks.

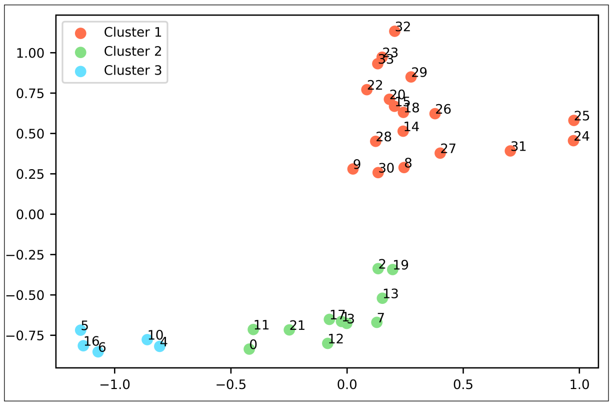

Degree based walks

Figure 4. Graph representation of the Zachary's karate club in two dimensional representations of the nodes. The distance and colors in the figure reflect the clustering. Here (a) fixed 40 and (b) degree-based $5 \times$ degree number of random walks started at each node.

3.2. CORA and CiteSeer datasets

CORA dataset consists of $2,708$ nodes and $5,429$ edges. Similarly, CiteSeer dataset consists of $3,327$ nodes and $4,732$ edges (Table 1). By using abovementioned datasets, embedding vectors are created using both degree-based and fixed numbers of random walks. The embedding vectors are used for i) node classification and ii) link prediction. For node classifications, dimensional reduction and K-Means algorithms are employed similar to method explained in subsection 3.1.2. LogisticRegressionCV program of the scikit-learn python package [35] is employed for predictions. The link embedding vectors are constructed from the node embedding vectors by using Hadamard, L1, L2, and Average algorithms [18]. During the test runs, it is observed that the Hadamard technique gives the best link prediction score. Hadamard operator is used in all link prediction calculations presented in this work. In this experimental study, the main focus has been to observe the correlations between accuracy, the number and lengths of random walks.

Two parameters, the number, and length of the walks play a crucial role in mapping the topological structure of the network onto embedding. First, for fixed walk length of 30, the variation of accuracy values with changing number of total walks are presented in Tables 2 and 3. Tables 2 and 3 present node classification accuracy values, relative percentage of total number of walks and accuracy gain with respect to 20 runs per node case, obtained on CORA and CiteSeer datasets, respectively. Tables 2 and 3 show the improvement of accuracy values with the number of walks. In the degree-based case, accuracy values increases starting from 1 run per degree reaches a plateau at 3 times per degree. The plateau value is also the value of the accuracy obtained by using a fixed number of walks (20) per node.

Table 2. Summary of results in terms of node classification for degree-based node2vec and original node2vec. Number of new walks started each node depends on the degree of the node in the node based case, while for the original approach 20 new runs (fixed) for each node performed for comparison (first column of the table). For each new random walk, walk continued until a fixed walk length of 30 steps is reached. The values are obtained by using CORA dataset.

Number of walks	Total number of walks (TNW)	Decrease in number of walks [%]	Accuracy	Accuracy Loss / Gain
$1 \times k_i$	10,138	79.6	80.3	-4.6
$2 \times k_i$	20,276	59	83.0	-1.9
$3 \times k_i$	30,414	38.8	85.7	+0.8
$4 \times k_i$	40,552	18.4	85.0	+0.1
Fixed $(20 \times n_i)$	49,700		84.9	

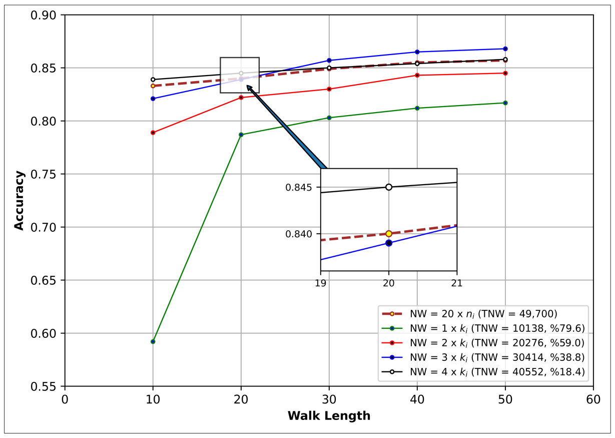
Table 3. Summary of results in terms of node classification for degree-based node2vec and original node2vec for the CiteSeer dataset. The number of initiated random walks and length of the random walks are the same as the CORA dataset.

Number of walks	Total number of walks (TNW)	Decrease in number of walks [%]	Accuracy	Accuracy Loss / Gain
$1 \times k_i$	7,388	82.49	72.6	-3.4
$2 \times k_i$	14,776	64.98	75.5	-0.5
$3 \times k_i$	22,164	47.47	76.5	+0.5
$4 \times k_i$	29,552	29.97	76.9	+0.9
$5 \times k_i$	36,940	12.46	76.7	+0.7
Fixed $(20 \times n_i)$	42,200		76.0	

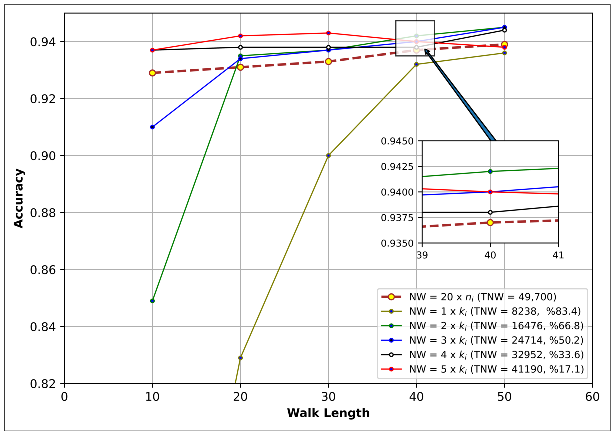
When Tables 2 and 3 are examined, it can be seen that the number of walks per degree is continued to increase even after accuracy is reached to a comparable value with the original algorithm. For example, for the CORA dataset Table 2, the original algorithm (with a fixed number (20) of walks per node) reaches 84.9% at accuracy. The degree-based approach reaches the same accuracy between 2- and 3- walk per degree per node. This corresponds to a 59% to 38.8% decrease in the number of walks. Similarly, for the CiteSeer dataset Table 3, the original approach with fixed 20 runs per node reaches 76% accuracy. The degree-based approach has the same accuracy between 2- and 3- walk per degree per node. This approximately corresponds to a 64.98% to 47.74% gain in the computational time. In order to show that accuracies level off around the obtained values, the number of random walks has been increased, which reduces the gain in the computational efficiency. The computational gains and accuracies are dependent on the structure of the network and which network parameters are measured by using the obtained random walks. In the light of the above discussions, the computational gain can be predicted as an average of 50% for both CORA and CiteSeer datasets.

Figure 5 shows the relation between the number of walks, walk length, and accuracy for node classification (Figure 5a), and link prediction (Figure 5b) for the CORA dataset. CORA network is used as a testing ground for accuracy versus walk length (10 to 50 steps). Each subfigure (Figures 5a and 5b) contains lines representing an increasing number of random walks per node and a fixed number (20) of walks per node. From bottom to top lines indicate the increasing number of walks per degree. At a certain point accuracies of the degree-based approach and the fixed number of walks become equal. The observed general trend for a given number of walks per degree is that as the length of the walk increases, the accuracy increases. For relatively long runs a plateau can be seen. As the number of walks increases, even for short random walks results in accuracy values near the plateau values. For the CiteSeer dataset, the observed behaviors are very similar.

In creating a sentence-like structure using random walks, two parameters play a crucial role: Number of walks and walk length (number of steps for each new walk). For the method's efficiency, both parameters must be optimized to reach the highest accuracy with a minimal computational expense. Figure 5 is devoted to such a comparative discussion. Figure 5 compares degree-based and original node2vec algorithms on node classification (Figure 5a) and link prediction (Figure 5b).



Node classification.



Link prediction.

Figure 5. The effects of the number and length of random walks on the accuracy of node classification and link prediction in CORA dataset.

Although the identical sentence-like node sequences obtained by using the nodes of the CORA dataset are used, node classification and link prediction require different processing techniques. In the node classification case, the degree-based approach reaches and overtakes the accuracy of the original algorithm by using shorter random walks. For the CORA dataset, around 20 steps, with a number of walks higher than 2 per degree, is sufficient to reach and overpass the accuracy obtained by the original algorithm. This point can be considered as the optimized number of walk-number of steps point. For link prediction (Figure 5b), the same situation is reached by using 40 step long random walks. To indicate these facts, subfigures were added into Figures 5a and 5b.

Tables 4 and 5 present node classification and link prediction data for CiteSeer dataset. Starting from 20 steps of walk length, the degree-based approach reaches near plateau value accuracies as early as 3 to 4 walks per degree. The ratio between the total number of walks done for degree-based and fixed approaches indicates the efficiency of the degree-based random walk approach. Using the values presented in Tables 4 and 5, comparisons between the total number of walks done for a degree-based and fixed number of walks per node indicate that approximately 50% less effort is sufficient to obtain similar accuracies.

Table 4. CiteSeer dataset accuracy values in terms of node classification for degree-based node2vec and original node2vec.

	Walk length				
Number of walks	10	20	30	40	50
$1 \times k_i$	52.3	67.0	72.6	73.3	73.7
$2 \times k_i$	69.9	74.5	75.5	75.5	76.8
$3 \times k_i$	72.8	76.0	76.5	76.9	77.1
$4 \times k_i$	73.2	76.3	76.9	77.5	78.6
$5 \times k_i$	74.6	76.2	76.7	76.7	77.2
Fixed $(20 \times n_i)$	75.1	75.8	76.0	76.7	77.2

Table 5. CiteSeer dataset accuracy values in terms of link prediction for degree-based node2vec and original node2vec.

	Walk length				
Number of walks	10	20	30	40	50
$1 \times k_i$	76.1	81.7	88.3	92.0	94.0
$2 \times k_i$	84.9	93.6	95.1	95.1	95.5
$3 \times k_i$	93.5	94.9	95.6	96.4	96.6
$4 \times k_i$	95.4	96.4	96.3	96.5	96.3
$5 \times k_i$	96.2	96.4	96.5	96.2	96.1
$6 \times k_i$	96.2	96.2	96.2	96.2	95.9
Fixed $(20 \times n_i)$	94.7	95.9	96.0	96.0	96.2

4. Conclusion

Graphs provide an effective abstraction for the local and global neighborhood information on the pairwise relationships. The main drawback of graph representations is that the very valuable information cannot be directly embedded into numerical vectors that can be used for further processing. There exist a large literature on the embedding methods, with different efficiencies. Matrix methods and random walk provide a basis for the majority of the proposed solutions. Among the random-walk-based solutions two of the best embedding methods, DeepWalk and node2vec make use of shallow neural network techniques which are imported from natural language processing. The sentence-like structures are created by using random walk techniques which is a computationally intensive part of the embedding process. In both of these methods, a fixed number of random walks are started from each node. The number of walks is determined by optimizing the value of the control parameter.

In the proposed method, the number of walks is chosen proportional with the degree of the node, instead of a fixed number for each node. The degree-based number walk choice also has three advantages: i) High degree nodes will have more emphasis on the sentence-like structures, which will correspond to importance sampling, ii) by choosing degree proportionality in the number of random walks, the contribution of low degree-nodes will be suppressed which eliminates the data imbalance problem, iii) degree-based random walks grossly reduces the computation time. Hence, the proposed approach is most efficient on scale-free networks.

For large scale-free networks, the degree-based sampling approach will be more efficient since the total number of walks is proportional to the average degree (Eq. 3) which is a multiple of the minimum degree of the network (Eq. 5). The method realizes correct sampling of the neighborhoods while keeping the total number of walks as small as possible. Considering the majority of real-wold networks are large and scale-free, the advantage of choosing the number of walks proportional to the degree of the node becomes more apparent. For reasonably small size lattices the proposed method is tested. Three networks, Zachary Karate Club, CORA, and CiteSeer data sets are used for node classification and link prediction. It is observed that for CORA and CiteSeer data sets, approximately 50% less computational effort than the fixed number of walks case is sufficient to reach the same accuracy for node classification and link prediction.

Many node classification and link prediction algorithms that perform successfully are based on random walks in the literature. The idea of anonymous random walks is also based on random walks. The main future direction of our work will be to carry the idea of degree-based random walks to anonymous random walks.

References

[1] Bhagat S, Cormode G, Muthukrishnan S. Node classification in social networks. Social Network Data Analytics 2011; 115-148. doi: 10.1007/978-1-4419-8462-3_5

[2] Kumar S, Chaudhary S, Kumar S, Yadav R. Node Classification in Complex Networks using Network Embedding Techniques. In: Proceedings Of The 5th International Conference On Communication And Electronics Systems (ICCES); 2020. 369-374. doi: 10.1109/ICCES48766.2020.9138094

[3] Liben-Nowell D, Kleinberg J. The link-prediction problem for social networks. Journal Of The American Society For Information Science And Technology 2007; 58 (7): 1019-1031. doi: 10.1002/asi.20591

[4] Martinez V, Berzal F, Cubero J. A survey of link prediction in complex networks. ACM Computing Surveys (CSUR) 2016; 49 (4): 1-33. doi: 10.1145/3012704

[5] Chandola V, Banerjee A, Kumar V. Anomaly detection: A survey. ACM Computing Surveys (CSUR) 2009; 41 (3): 1-58. doi: 10.1145/1541880.1541882

[6] Ma X, Wu J, Xue S, Yang J, Zhou C et al. A comprehensive survey on graph anomaly detection with deep learning. IEEE Transactions On Knowledge And Data Engineering 2021. doi: 10.1109/TKDE.2021.3118815

[7] Fortunato S. Community detection in graphs. Physics Reports 2010; 486 (3-5): 75-174. doi: 10.1016/j.physrep.2009.11.002

[8] Witten I, Frank E, Hall M, Pal C. Practical machine learning tools and techniques. Morgan Kaufmann 2005; 2, 1.

[9] Kotsiantis S, Zaharakis I, Pintelas P. Supervised machine learning: A review of classification techniques. Emerging Artificial Intelligence Applications In Computer Engineering 2007; 160 (1): 3-24.

[10] Jordan M, Mitchell T. Machine learning: Trends, perspectives, and prospects. Science 2015; 349 (6245): 255-260. doi: 10.1126/science.aaa8415

[11] Huang Z. Link prediction based on graph topology: The predictive value of generalized clustering coefficient. Available At SSRN 1634014 2010. doi: 10.2139/ssrn.1634014

[12] Fortunato S, Latora V, Marchiori M. Method to find community structures based on information centrality. Physical Review E 2004; 70 (5): 056104. doi: 10.1103/PhysRevE.70.056104

[13] Gallagher B, Eliassi-Rad T. Leveraging label-independent features for classification in sparsely labeled networks: An empirical study. International Workshop On Social Network Mining And Analysis 2008; 1-19. doi: 10.1007/978-3-642-14929-0_1

[14] Henderson K, Gallagher B, Li L, Akoglu L, Eliassi-Rad T et al. It's who you know: graph mining using recursive structural features. In: Proceedings Of The 17th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining; 2011. 63-671. doi: 10.1145/2020408.2020512

[15] Murata T, Moriyasu S. Link prediction based on structural properties of online social networks. New Generation Computing 2008; 26 (3): 245-257. doi: 10.1007/s00354-008-0043-y

[16] Makarov I, Kiselev D, Nikitinsky N, Subelj L. Survey on graph embeddings and their applications to machine learning problems on graphs. PeerJ Computer Science 2021; 7. doi: 10.7717/peerj-cs.357

[17] Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: Proceedings Of The 20th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining; 2014. 701-710. doi: 10.1145/2623330.2623732

[18] Grover A, Leskovec J. node2vec: Scalable feature learning for networks. In: Proceedings Of The 22nd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining; 2016. 855-864. doi: 10.1145/2939672.2939754

[19] Tang J, Qu M, Wang M, Zhang M, Yan J et al. Line: Large-scale information network embedding. In: Proceedings Of The 24th International Conference On World Wide Web; 2015. 1067-1077. doi: 10.1145/2736277.2741093

[20] Ribeiro L, Saverese P, Figueiredo D. struc2vec: Learning node representations from structural identity. In: Proceedings Of The 23rd ACM SIGKDD International Conference On Knowledge Discovery And Data Mining; 2017. 385-394. doi: 10.1145/3097983.3098061

[21] Nguyen D, Malliaros F. BiasedWalk: Biased sampling for representation learning on graphs. In: Proceedings Of The IEEE International Conference On Big Data (Big Data); 2018. 4045-4053. doi: 10.1109/BigData.2018.8621872

[22] Ahmed N, Rossi R, Lee J, Willke T, Zhou R et al. role2vec: Role-based network embeddings. In: Proceedings Of The First International Workshop On Deep Learning for Graphs (DLG'19); 2019. pp. 1-7.

[23] Ma X, Qin G, Qiu Z, Zheng M, Wang Z. RiWalk: Fast structural node embedding via role identification. In: Proceedings Of The IEEE International Conference On Data Mining (ICDM); 2019. 478-487. doi: 10.1109/ICDM.2019.00058

[24] Zhang Y, Shi Z, Feng D, Zhan X. Degree-biased random walk for large-scale network embedding. Future Generation Computer Systems 2019; 100: 198-209. doi: 10.1016/j.future.2019.05.033

[25] Salamat A, Luo X, Jafari A. BalNode2Vec: Balanced Random Walk based Versatile Feature Learning for Networks. In: Proceedings Of The International Joint Conference On Neural Networks (IJCNN); 2020. pp. 1-8. doi: 10.1109/IJCNN48605.2020.9206737

[26] Fazaeli M, Momtazi S. GuidedWalk. World Wide Web 2022; 1-23. doi: 10.1007/s11280-021-00999-9

[27] Wang S, Qin X, Chi L. HashWalk: An efficient node classification method based on clique-compressed graph embedding. Pattern Recognition Letters 2022. doi: 10.1016/j.patrec.2022.02.001

[28] Barabási A, Bonabeau E. Scale-free networks. Scientific American 2003; 288 (5): 60-69.

[29] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. Advances In Neural Information Processing Systems 2013; 26.

[30] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. In: Proceedings Of Workshop At International Conference On Learning Representations (ICLR); 2013.

[31] Girvan M, Newman M. Community structure in social and biological networks. Proceedings Of The National Academy Of Sciences 2002; 99 (12): 7821-7826. doi: 10.1073/pnas.122653799

[32] McCallum A, Nigam K, Rennie J, Seymore K. Automating the construction of internet portals with machine learning. Information Retrieval 2000; 3 (2): 127-163. doi: 10.1023/A:1009953814988

[33] Giles C, Bollacker K, Lawrence S. CiteSeer: An automatic citation indexing system. In: Proceedings Of The Third ACM Conference On Digital Libraries; 1998. 89-98. doi: 10.1145/276675.276685

[34] Perry M. On the detection of transitive clusters in undirected networks. Journal Of Applied Statistics 2019; 46 (2): 364-384. doi: 10.1080/02664763.2018.1491535

[35] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B et al. Scikit-learn: Machine learning in Python. The Journal Of Machine Learning Research 2011; 12, 2825-2830.