2024 33rd International Conference on Computer Communications and Networks (ICCCN) | 979-8-3503-8461-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ICCCN61486.2024.10637636

Exploring Similarity-Based Graph Compression for Efficient Network Analysis and Embedding

Hamdi Selim Akin
Department of Computer Science
Georgia State University
Atlanta, GA, USA
hakin2@student.gsu.edu

Mehmet Emin Aktas
Institute for Insight
Georgia State University
Atlanta, GA, USA
maktas@gsu.edu

Muhammed Ifte Islam
Department of Computer Science
Georgia State University
Atlanta, GA, USA
mislam29@student.gsu.edu

Tanvir Hossain
Department of Computer Science
Georgia State University
Atlanta, GA, USA
thossain5@student.gsu.edu

Esra Akbas
Department of Computer Science
Georgia State University
Atlanta,GA, USA
eakbas1@gsu.edu

***Abstract*—Network analysis is an emerging field with a wide spectrum of applications across many disciplines such as social networks, computer networks, and healthcare. However, the ever-increasing size of real-world networks is a major challenge for network analysis due to their high computational and space costs. In this paper, we utilize a node similarity-based graph compression method, `SGC`, and investigate the effect of various node similarity measures on graph compression. `SGC` compresses the input graph to a smaller graph without losing any/much information about its global structure and the local proximity of its vertices. We apply our compression method to the network embedding problem to study its effectiveness and efficiency. Our experimental results on four real-world networks show that each similarity measure has a different effect on graph compression and embedding, where some yield an improvement up to 70% network embedding time without decreasing classification accuracy as evaluated on single and multi-label classification tasks.**

***Index Terms*—graph compression, network embedding, node similarity, node classification**

I. Introduction

Network analysis is an emerging field with a wide spectrum of applications across many disciplines, such as social networks [1], computer networks [2] and healthcare [3]. Examples of applications include finding groups of users in social networks (e.g., Facebook), which is useful for a personalized recommendation [1], and detecting drug-drug interaction, which may cause dangerous side effects on health [3]. In network mining, many useful methods can be applied to small graphs effectively. However, the ever-increasing size of real-world networks is a major challenge for these methods due to their high computational and space costs [4]. This challenge has motivated researchers to develop various graph compression (summarization) algorithms to reduce the complexity and size of large graphs. Such reductions are essential to scale up or scale out existing algorithms to better manage, query, store, and display them [5]. Graph compression aims to create a smaller supergraph from a massive graph such that the crucial information of the original graph will be maintained in the supergraph.

In this paper, we study graph compression by preserving similarity between nodes. Similarity is used in many graph mining problems, such as node classification, community detection, and link prediction. For example, in a social network (e.g., Facebook), if two people are similar, they are expected to have similar interests, backgrounds, or friends. That is why similarity can be used for friend recommendation (i.e., link prediction) [6], and group recommendation [7].

Based on this observation, in our preliminary work [8], we employed one neighborhood-based similarity, namely Jaccard similarity, for graph compression. The core concept involves assessing the similarities between nodes and compressing those whose similarity falls below a predefined threshold. In this paper, we leverage various node similarity metrics, each emphasizing distinct structural aspects of the nodes. Consequently, the compressed graphs produced by each metric will vary, leading to diverse outcomes in learning. Our project aims to explore how different metrics influence compression.

To study the effectiveness and efficiency of the proposed compression methods, we further employ the compressed graph in the network embedding problem. Network embedding is defined as encoding structural information of graphs, such as characteristics or role of vertices, into a low-dimensional vector space on their connections [9]. Instead of embedding the original large graph to bring down the embedding cost, we embed the compressed graph and use this embedding to get the embedding of the original graph. Hence, network embedding will become a lot more efficient thanks to graph compression. Next, to address the effectiveness of the proposed compression methods, we use these node embeddings in the single-label and multi-label node classification problem on four different real-world networks. Our experimental results show that the

This work has been supported partially by the National Science Foundation under Grant No 2308206.

proposed graph compression methods have better efficiency with similar accuracy than the baseline methods in different classification tasks on several real-world graphs.

II. Background

A. Preliminaries

1) Graph Compression: In this paper, we consider an undirected, simple graph $G = (V; E)$ where V is the set of vertices, and $E \subseteq \{V \times V\}$ is the set of edges. The set of neighbors for given a vertex $v \in V$ is denoted as $N(v)$, where $N(v) = \{u | u \in V : (u, v) \in E\}$.

Graph compression aims to reduce the size of a graph to improve our ability to analyze them [10], [11]. While compressed graphs can be used directly by many graph mining algorithms, they can also be used with decompressing parts of it on the fly when needed. A general formal definition of a compressed graph is given as follows:

Definition 1 (Compressed graph)**.** *For a given graph $G = (V; E)$, a compressed graph is represented as $CG = (\mathcal{G}; \mathcal{M})$ where $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the **supergraph** with supernodes $\mathcal{V}$ and superedges $\mathcal{E}$ and $\mathcal{M}$ is a mapping from each vertex $v \in V$ to its supernode $v \in \mathcal{V}$.*

There are several approaches for graph compression (see Section II-B). In this paper, our focus is to use proximity-based similarities for compression. Hence, next, we define the similarities that we use in the paper.

2) Proximity-based Node Similarities: The proximity-based similarities are defined using the relations between nodes and edges, such as the number of shared edges between nodes. Definitions of the proximity-based similarities utilized in this paper are listed below.

i. Common Neighbors is calculated based on the number of common neighbors, with an additional consideration for a direct edge between them. The formula is defined as

$$\text{CN}(i, j) = \frac{2 \times (|N(i) \cap N(j)| + \mathbf{1}_{\{j \in N(i)\}})}{|N(i)| + |N(j)|}$$

where $N(i)$ and $N(j)$ are the sets of neighbors of nodes i and j, respectively. $\mathbf{1}_{\{j \in N(i)\}}$ is an indicator function that is 1 if there is a direct edge from i to j, and 0 otherwise.

ii. Jaccard similarity is defined based on the number of common neighbors between two nodes i and j as follows

$$Jaccard(i, j) = \frac{|N(i) \cap N(j)|}{|N(i)| + |N(j)|}$$

where $N(i)$ is the set of neighbors of ith vertex.

iii. Overlap similarity is a variation of Jaccard similarity that measures the relative overlap between the neighborhoods of two nodes. It normalizes the number of common neighbors by the size of the smaller neighborhood, giving more weight to nodes with smaller neighborhoods. It is defined as

$$\text{Overlap}(i, j) = \frac{|N(i) \cap N(j)|}{\min(|N(i)|, |N(j)|)}$$

iV. Wasserstein similarity quantifies the dissimilarity between two probability distributions. It measures the minimum amount of work required to transform one distribution into another. Given two nodes i and j with probability distributions p_i and p_j over their respective neighborhoods, Wasserstein similarity is defined as

$$\text{Wasserstein}(i, j) = \inf_{\gamma \in \Gamma(p_i, p_j)} \sum_{u,v} c(u, v)\gamma(u, v)$$

where $\Gamma(p_i, p_j)$ is the set of all joint probability distributions with marginals p_i and p_j, and $c(u, v)$ is the cost of transporting mass from node u to node v.

V. Resource Allocation Index measures the amount of a "resource" that a node can allocate to another through their common neighbors. It is defined as

$$\text{RAI}(i, j) = \sum_{z \in N(i) \cap N(j)} \frac{1}{|N(z)|}.$$

3) Network Embedding: Network embedding is a technique used in network analysis to represent nodes and edges in a network as low-dimensional vectors in a continuous vector space. The goal of network embedding is to capture the structural and relational information of the network in these vector representations, enabling downstream machine learning tasks such as node classification, link prediction, and community detection.

In general, the embedding process includes 2 steps. As the first step, a set of neighbors is sampled from a graph. Then, the learning process leverages the co-occurrence probability of the vertices that appear within a window in a sampled neighborhood [12], [13]. Node pairs with high co-occurrence probability are regarded as neighbors. We define network embedding as follows.

Definition 2 (Network embedding)**.** *Network embedding is a mapping $\phi : V \to \mathbb{R}^d, d << |V|$ which represents each vertex $v \in V$ as a point in a low dimensional space $\mathbb{R}^d$.*

Here d is a parameter specifying the number of dimensions of our feature representation. For every source node $u \in V$, we define $N_S(u) \subset V$ as a network neighborhood of node u generated through a neighborhood sampling strategy S. We seek to optimize the following objective function, which maximizes the log-probability of observing a network neighborhood $N_S(u)$ for a node u conditioned on its feature representation, given by ϕ

$$\max_f \sum_{u \in V} logPr(N_S(u)|\phi(u)). \tag{1}$$

B. Related Work

In the context of network embedding, graph compression is employed as a preprocessing step in many research works. GraphZoom [14], MILE [15], HARP [16] simplify the large graphs utilizing multi-level graph clustering, and at each level, they learn and refine nodes' representations.

NECL [17] uses meta-strategy by applying neighborhood similarity (Jaccard) between nodes where nodes with similar neighbors are grouped into supernodes. EnD [18] employs degree-based compression and iterative dedensification for efficient representation learning. Mostly, graph compression models combine graph components for efficient representation learning, while the dedicated function provides a comparable test performance to the original graph.

III. Methodology

A. Random walk based sampling

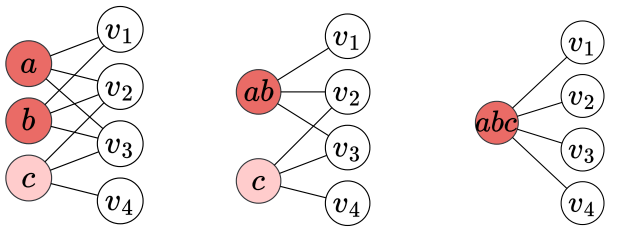
The neighborhoods in the network embedding are not restricted to just immediate neighbors but can have vastly different structures depending on the sampling strategy. There are many possible neighborhood sampling strategies for vertices as a form of local search. Different neighborhoods coming from different strategies result in different learned feature representations. For scalability of learning, random walk-based methods are used to capture the structural relationships of vertices and proximity-based similarities in the graph. They maximize the co-occurrence probability of subsequent vertices within a fixed-length window of random walks to preserve higher-order proximity between vertices. With random walks, networks are represented as a collection of vertex sequences. In this section, we take a deeper look at the network neighborhood sampling strategy based on random walks and the proximity captured by random walks.

The co-occurrence probability of node pairs depends on the transition probabilities of vertices during the random walk. Considering a graph G with adjacency matrix A, we define the diagonal matrix, known as degree matrix, as $D_{ij} = \sum_k A_{ik}$ if $i = j$ and $D_{ij} = 0$ otherwise. In a random walk, the transition probability from one node to another depends on the degree of the vertices. The probability of leaving a node from one of its edges is split uniformly among the edges. We define this 1 step transition probability as T: $T = D^{-1}A$ where T_{ij} is the probability of a transition from vertex v_i to vertex v_j within one step.

We observe here that if two vertices, v_i, v_j, of a graph have many common neighbors, they also have similar transition probabilities to other vertices. This means that if A_i and A_j are similar, transition probability vectors of them, $T_i = A_i * D_{ii}^{-1}$ and $T_j = A_j * D_{jj}^{-1}$, will be similar as well. Hence they have similar neighborhoods and get similar neighborhood sets from random walks, and as a result, they get similar representations from the learning process. Therefore, the random walk-based neighborhood sampling strategy captures the higher-order proximity within the neighborhood of the vertices.

B. Graph compression based on node similarity

Based on the observation given in the previous section, if two vertices have similar neighborhoods on the graph, random walks starting from these vertices would produce similar paths and similar representations. As a result, instead of learning representations for these vertices individually, which is time-consuming, we can merge these two vertices into a supernode and use the representation of the supernode to learn the representation of the original vertices.



((a)) Original ((b)) 1st comp. ((c)) 2nd comp.

Fig. 1: Example of graph compression

As an example, we provide an illustration of our compression idea on a toy graph in Figure 1. As seen in Figure 1-a nodes a and b share identical neighbor sets. Hence, they have a high proximity-based similarity as a result of very similar sampled paths. From identical neighbor sets, their transition probabilities to other neighboring vertices also become identical, denoted as $p(v_i|a) = p(v_i|b) = 1/3$ for all $i \in 1, 2, 3$. Initiating a walk from either a or b results in the same/similar paths. Hence, rather than conducting walks and deriving representations for a and b separately, learning from just one of them suffices. This motivates the merging of the node pair (a, b) into a single super-node ab (see in Figure 1-b). The transition probabilities of this supernode to the neighbors of a and b remain identical to those of a and b, respectively, expressed as $p(v_i|ab) = 1/3$ for all $i \in 1, 2, 3$.

Moreover, compression may alter the transition probabilities of vertices due to reductions in their neighbor counts. As a result, the transition probabilities of each neighbor are subject to change. For instance, in the toy graph depicted in Figure 1-a, while the transition probability from v_1 to its neighbors is initially $\frac{1}{|N(v_1)|}$, it becomes $\frac{1}{|N(v_1)|-1}$ post-compression, as the number of neighbors decreases by one. To address this issue, we assign weights to the edges of super-nodes based on the number of merged edges during compression. For instance, the super-edge between super-node ab and v_1 encompasses 2 merged edges, namely (a, v_1) and (b, v_1). Consequently, the weight of the super-edge (ab, v_1) should be 2.

Furthermore, in real-world graphs, it is uncommon to find many vertices with entirely identical neighbor sets. Therefore, if the neighbors of two vertices are similar (though not identical), rather than individually learning representations for each node, we can still merge them into a super-node and derive a single representation for all of them. For example, in Figure 1-b, nodes ab and c do not share identical neighbors, yet their 2 out of 4 neighbors are in common. Hence, we may still want to merge them into the supernode abc as in Figure 1-c.

Based on this observation, we integrate multiple node similarity measures within our graph compression framework. The existing literature provides a diverse array of such measures. Specifically detailed in Section II-A2, we investigate five distinct proximity-based similarity measures. Each of these measures sheds light on different structural characteristics of

Algorithm 1: Graph Compressing(G, λ)

Input: $G(V_G, E_G)$, similarity threshold λ
Output: $S(V_{G'}, E_{G'}, W_E)$, mapping **M**
M is a mapping from super-node to original node

$S \leftarrow G$
$NSQ \leftarrow \emptyset$
for $v \in V_G$ **do**
  for $u \in N_G(v)$ **do**
    for $k \in N_G(u)$ **do**
      Compute Proximity-based similarity between v and k as $PSim(v, k)$
      $NSQ \leftarrow NSQ \cup (v, k)$
    end
  end
end
for $(v, k) \in NSQ$ **do**
  if $PSim(v, k) > \lambda$ **then**
    Merge them into a super-node $s_{v,k}$
    $M(s_{v,k}) \leftarrow v$; $M(s_{v,k}) \leftarrow k$
    Delete v and k from S and add $s_{v,k}$ into S.
  end
  for $ng \in N_S(v)$ **do**
    add edge between $s_{v,k}$ and ng
    $w(ng, s_{vk}) = w(ng, v)$
  end
  for $ng \in N_S(k)$ **do**
    add edge between $s_{v,k}$ and ng if there is no
    $w(ng, s_{vk}) = w(ng, s_{vk}) + w(ng, k)$
  end
end

the nodes. Consequently, the resulting compressed graphs vary across measures, leading to diverse learning outcomes. Following similarity calculations between node pairs, we determine compression based on a given threshold; we compress nodes as a supernode whose similarities are larger than the given threshold. Our project aims to examine how these measures affect compression and representation learning. The generic compressing algorithm is provided in Algorithm 1.

C. Network embedding on compressed graph

After obtaining the compressed graph, we employ a network embedding method to derive embeddings for the compressed graph. Specifically, we utilize random walk-based network embedding techniques such as `DeepWalk` and `Node2vec` to learn representations for the super nodes within the compressed graph. It's worth noting that our model remains adaptable to any network embedding method. Given that the size of the compressed graph is smaller than the original graph, obtaining embeddings for super nodes is more efficient compared to individual vertices in the original graph. Subsequently, we assign the embeddings of super nodes to vertices based on the mapping M acquired from the compression process.

TABLE I: Graph statistics

Network	|V|	|E|	class #	Multi-label
Wiki	2405	23192	17	No
Cora	2708	10858	7	No
DBLP	27199	133664	4	Yes
BlogCatalog	10312	667966	39	Yes

IV. Experiments

A. Datasets

We consider four real-world graphs[1], which have been widely adopted in the studies of network embedding. The general statistics of the datasets used for experiments are reported in Table I.

B. Baseline methods

For the performance evaluation, we use `DeepWalk` and `Node2vec` as baseline embedding methods in our model and compare our model with them. We combine each baseline method with our method and compare their performance.

Parameter Settings: For `DeepWalk`, `Node2vec`, and `SGC(DW)`, `SGC(N2V)`, we set the following parameters: the number of random walks γ, walk length t, window size w for the Skip-gram model and representation size d. The parameter setting for all models is $\gamma = 40$, $t = 10$, $w = 10$, $d = 128$. The initial learning rate and final learning rate are set to 0.025 and 0.001 respectively in all models.

C. Classification

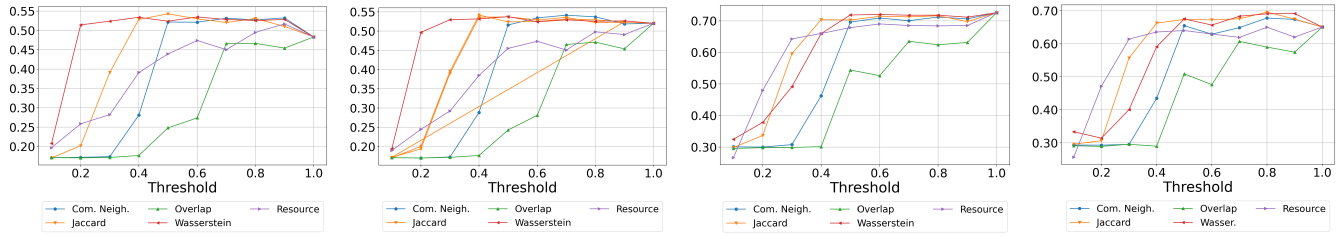
Our experiments aimed to evaluate the effectiveness of various proximity-based node similarity measures in the context of graph compression and their subsequent impact on network embedding. We compare the different proximity-based node similarity measures on two different node classification tasks, namely single-label and multi-label classification. At first, we get the node embedding for all nodes in the network using all similarity measures for `Node2vec` and `DeepWalk` methods. Then we randomly select a portion of the labeled nodes to train the classification method and the rest of them are used for testing. For a fair comparison, we do the classification process 10 times and average the micro F1 scores.

1) Single-label classification:: For single-label classification experiments, we used Cora and Wiki datasets where nodes have a single label, i.e., a node belongs to only one class. We use the SVM model as the classifier for this experiment. Table II shows node classification accuracy results for Cora and Wiki datasets. The experiments were conducted using 5% labeled vertices for training the classifier. We select the similarity threshold values, denoted as λ, for each similarity measure based on significant accuracy improvements observed in Figure 2. Specifically, we set the thresholds λ to 0.5, 0.4, 0.7, 0.4, 0.9, 0.3, and 0.9 for the respective similarities, as they appear in the table. The table indicates that among all five proximity-based similarity measures, the Jaccard similarity measure yields the highest accuracy for both datasets. Our model with the Jaccard similarity measure demonstrates a noteworthy enhancement of 64.41% and 64.58% in efficiency as embedding time without experiencing significant loss, or even showcasing better improvements, in effectiveness, as evidenced by the micro F1 score on the Cora dataset for the `DeepWalk` and `Node2vec` methods respectively. For

[1]https://linqs.soe.ucsc.edu/data

TABLE II: Performance comparison of the single-label and multi-label classification tasks for the training ratio 5% for Cora, Wiki, BlogCatalog, and DBLP. Here, we select the similarity threshold λ for each similarity based on their performance.

		Cora				Wiki				BlogCatalog				DBLP			
		DeepWalk		Node2vec		DeepWalk		Node2vec		DeepWalk		Node2vec		DeepWalk		Node2vec	
Similarity		Mi. F_1	Time	Mi. F_1	Time	Mi. F_1	Time	Mi. F_1	Time	Mi. F_1	Time	Mi. F_1	Time	Mi. F_1	Time	Mi. F_1	Time
CN	Comp.	0.692	4.78	0.703	4.91	0.500	3.16	0.508	5.66	0.290	235.06	0.292	2154.54	0.658	120.13	0.657	121.93
	Orig.	0.726	15.2	0.710	15.6	0.521	12.01	0.521	19.93	0.309	278.74	0.316	2395.05	0.654	423.85	0.653	484.80
	Gain %	-4.68	**68.55**	-0.99	**67.61**	-4.03	**73.69**	-2.50	**71.60**	-6.15	**15.67**	-7.59	**10.04**	0.61	**71.66**	0.61	**74.85**
J	Comp.	0.711	5.41	0.715	5.37	0.522	4.78	0.533	8.00	0.297	264.36	0.305	2234.58	0.658	211.04	0.661	275.71
	Orig.	0.726	15.2	0.710	15.16	0.521	12.01	0.521	19.93	0.309	278.74	0.316	2395.05	0.654	423.85	0.653	484.80
	Gain %	-2.07	**64.41**	0.70	**64.58**	0.19	**60.20**	2.30	**40.41**	-3.88	**5.16**	-3.48	**6.70**	0.61	**50.21**	1.22	**43.13**
O	Comp.	0.637	7.77	0.622	8.55	0.467	5.65	0.481	11.46	0.282	96.88	0.279	1446.84	0.622	177.11	0.622	223.51
	Orig.	0.726	15.2	0.710	15.16	0.521	12.01	0.521	19.93	0.309	278.74	0.316	2395.05	0.654	423.85	0.653	484.80
	Gain %	-12.26	**48.88**	-12.39	**43.60**	-10.36	**52.96**	-7.68	**42.50**	-8.74	**34.76**	-11.71	**39.59**	-4.89	**58.21**	-4.75	**53.90**
W	Comp.	0.649	5.04	0.646	7.09	0.526	7.55	0.529	11.5	0.313	232.09	0.315	2168.47	0.455	34.32	0.653	311.36
	Orig.	0.726	15.2	0.710	15.16	0.521	12.01	0.521	19.93	0.309	278.74	0.316	2395.05	0.654	423.85	0.653	484.80
	Gain %	-10.61	**66.84**	-9.01	**53.23**	0.96	**37.14**	1.54	**42.30**	1.29	**16.74**	-0.32	**9.46**	-30.43	**91.90**	0.00	**35.78**
R	Comp.	0.638	9.82	0.650	19.63	0.293	10.93	0.299	9.57	0.249	237.09	0.237	1914.12	0.582	360.86	0.588	374.60
	Orig.	0.726	15.2	0.710	15.16	0.521	12.01	0.521	10.69	0.309	278.74	0.316	2395.05	0.654	423.85	0.653	484.80
	Gain %	-11.12	**35.39**	-8.45	**29.49**	-43.76	**8.99**	-42.61	**10.41**	-19.42	**14.94**	-25.00	**20.08**	-11.01	**14.86**	-9.10	**22.73**



((a)) SGC(DeepWalk) - Wiki ((b)) SGC(Node2vec) - Wiki ((c)) SGC(DeepWalk) - Cora ((d)) SGC(Node2vec) - Cora

Fig. 2: Detailed classification results (Micro F1) for Wiki and Cora datasets with different similarity threshold λ.

the Wiki dataset, our model shows 60.20% and 40.41% improvements in efficiency compared to the baseline models.

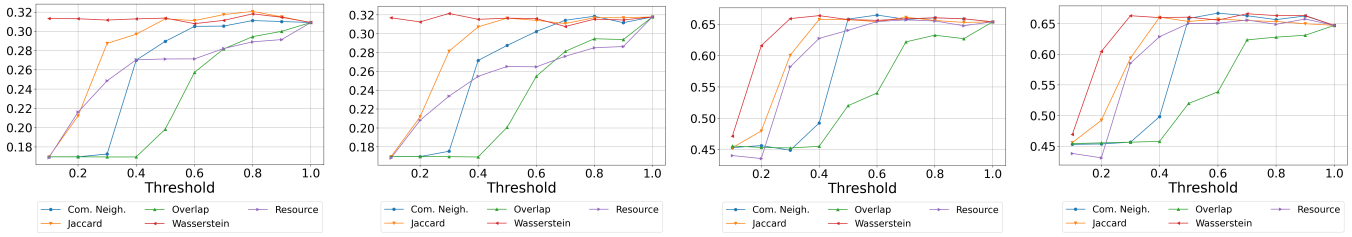
Figure 2 shows the detailed result for the Cora and wiki datasets for different similarity threshold values. From the figures, we can see that at the beginning when the threshold is $\lambda = 0.1$ the accuracy is very low. When we increase the threshold value λ the accuracy is also changing dramatically till the threshold value 0.5. After the threshold value of 0.5, the accuracy increases very slowly or remains constant for most of the similarity measures. One possible reason for this result is that graph sizes are very small when the threshold value $\lambda < 0.5$ which loses lots of important information. For the Cora dataset, for low threshold values, the Resource similarity metric performs better than the other similarity metrics. While the Jaccard similarity metric gives better accuracy when the threshold value is more than 0.4. On the other hand, the Overlap similarity measure gives the lowest accuracy. Similarly for the Wiki dataset, Wasserstein, Common Neighbor, Resource, and Jaccard similarity metrics show almost the same results when the threshold value $\lambda > 0.5$.

2) Multi-label classification:: A vertex may belong to more than one class in the multilabel classification task. To test the effectiveness of our compressed graphs, we utilize the logistic regression model in rest vs one manner as the classifier for the classification task. The model fits with the ridge regression l_2 and is implemented by LibLinear. Table II demonstrates the Multi-label classification accuracy for DBLP and BlogCatalog datasets. From the table, we can see that for the BlogCatalog dataset, the Wasserstein similarity measure gives the best results among all proximity-based measures for the DeepWalk method. Where it improves 1.29% micro F1 score and 16.74% gain on embedding time. For the Node2vec model, Jaccard similarity measures give better results than other similarity measures. For the DBLP dataset, the Common Neighbors similarity measure gives the best result for both embedding methods among all similarity measures. Our model shows 71.66% and 74.85% efficiency improvements compared to baseline models for the Common Neighbors similarity measure.

Figures 3 demonstrate the impact of the node classification accuracy on the DBLP and BlogCatalog datasets for changing the threshold. For both the DeepWalk and Node2vec embedding methods, we observe an almost similar pattern in node classification accuracy. Notably, the accuracy with Wasserstein distance similarity achieves stable accuracy for incrementing the threshold λ. One of the reasons is that the method does not greatly compress the network. In contrast, with the Jaccard similarity and resource allocation index, all datasets archive stable results when the λ value reaches near 0.4. In the case of the typical neighborhood, overlapping similarities gradually attain their best accuracies for incrementing threshold.

D. Graph compression

In this section, we present how the graph size, i.e., the number of vertices and edges, is decreasing by graph com-



((a)) SGC(DeepWalk) - BlogC ((b)) SGC(Node2vec) - BlogC ((c)) SGC(DeepWalk) - DBLP ((d)) SGC(Node2vec) - DBLP

Fig. 3: Detailed classification results (Micro F1) for BlogCatalog and DBLP datasets with different similarity threshold λ.

pression with different similarity measures. Table III shows the Percentage statistics of compression on nodes and edges in the compressed graph for different proximity-based similarity metrics on all four datasets. For the Cora dataset, the Common Neighbor similarity metric reduces the highest number of nodes and edges from the original graph. It reduces 47.3% and 51.74% of the original node and edge sizes respectively. For the Wiki dataset, the Common Neighbor similarity measure shows the highest gain on graph compression. For the DBLP dataset, the Common Neighbor similarity measure again shows the highest gain on graph compression whereas Overlap similarity gets the highest graph compression gain for the BlogCatalog dataset.

TABLE III: Graph Compression Statistics

Graph		CN	J	O	W	R
Cora	$\|V\|$	47.3%	27.1%	38.2%	33.6%	12.6%
	$\|E\|$	51.7%	28.5%	37.8%	30.5%	26.2%
Wiki	$\|V\|$	55.9%	35.8%	52.9%	7.9%	24.7%
	$\|E\|$	63.0%	46.3%	59.8%	6.9%	67.2%
DBLP	$\|V\|$	67.6%	45.2%	53.2%	37.5%	25.3%
	$\|E\|$	75.3%	55.4%	63.6%	42.9%	54.1%
BC	$\|V\|$	17.5%	12.7%	26.3%	2.0%	6.5%
	$\|E\|$	18.4%	4.4%	28.1%	0.1%	78.8%

V. Conclusion

In this paper, we propose a novel graph compression method, SGC, for an efficient network analysis and network embedding method preserving the local structural features of the vertices. Based on different node similarities, we compress related vertices of a network into supernodes that preserve the neighborhood information of the vertices. Then, we use the compressed graph to learn the representation of the vertices in the original graph. We leverage the utility of this paradigm by integrating SGC with two commonly used network embedding techniques, DeepWalk and Node2vec. Extensive experiments on various real-world graphs show distinct effects of each similarity measure on graph compression and embedding. Some measures lead to enhancements of up to 70% in network embedding time without sacrificing classification accuracy, as assessed across single and multi-label classification tasks. This study underscores the promising potential of similarity-based graph compression for efficient network analysis and network embedding.

References

[1] P. Bedi and C. Sharma, "Community detection in social networks," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 6, no. 3, pp. 115–135, 2016.

[2] P. Fang and T. Wolf, "Value trees: Multi-hop and asynchronous economic transactions in distributed systems," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*. IEEE, 2023, pp. 1–10.

[3] M. Zitnik, M. Agrawal, and J. Leskovec, "Modeling polypharmacy side effects with graph convolutional networks," *Bioinformatics*, vol. 34, no. 13, pp. i457–i466, 2018.

[4] A. Labrinidis and H. V. Jagadish, "Challenges and opportunities with big data," *Proceedings of the VLDB Endowment*, vol. 5, no. 12, pp. 2032–2033, 2012.

[5] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *ACM Comput. Surv.*, vol. 51, no. 3, pp. 62:1–62:34, 2018.

[6] M. Rahman and M. Al Hasan, "Link prediction in dynamic networks using graphlet," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 394–409.

[7] J. Yang, J. McAuley, and J. Leskovec, "Community detection in networks with node attributes," *Proceedings - IEEE International Conference on Data Mining, ICDM*, pp. 1151–1156, Jan. 2013. [Online]. Available: http://arxiv.org/abs/1401.7267

[8] E. Akbas and M. E. Aktas, "Network embedding: on compression and learning," in *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 2019, pp. 4763–4772.

[9] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the SIGKDD'14*. ACM, 2014, pp. 701–710.

[10] A. Loukas, "Graph reduction with spectral and cut guarantees," *Journal of Machine Learning Research*, vol. 20, no. 116, pp. 1–42, 2019.

[11] M. E. Aktas, T. Nguyen, and E. Akbas, "Homology preserving graph compression," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2021, pp. 930–935.

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *Proceedings of Workshop at ICLR, 2013.*, 2013.

[13] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[14] Y. W. Z. Z. Z. F. Chenhui Deng, Zhiqiang Zhao, "Graphzoom: A multi-level spectral approach for accurate and scalable graph embedding," in *In ICLR 2020*, 2020.

[15] J. Liang, S. Gurukar, and S. Parthasarathy, "Mile: A multi-level framework for scalable graph embedding," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 15, 2021, pp. 361–372.

[16] H. Chen, B. Perozzi, Y. Hu, and S. Skiena, "Harp: Hierarchical representation learning for networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[17] M. I. Islam, F. Tanvir, G. Johnson, E. Akbas, and M. E. Aktas, "Proximity-based compression for network embedding," *Frontiers in big Data*, vol. 3, p. 608043, 2021.

[18] T. Hossain, E. Akbas, and M. I. K. Islam, "End: Enhanced dedensification for graph compressing and embedding," in *2022 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 2022, pp. 674–681.