Federation University ResearchOnline
https://researchonline.federation.edu.au
Copyright Notice

This is the peer-reviewed version of the following article:

See this record in Federation ResearchOnline at:
https://researchonline.federation.edu.au/vital/access/manager/Index

Graph Force Learning

Ke Sun[1], Jiaying Liu[1], Shuo Yu[1], Bo Xu[1], and Feng Xia[2]

[1]School of Software, Dalian University of Technology, Dalian 116620, China

[2]School of Engineering, IT and Physical Sciences, Federation University Australia, VIC 3353, Australia

{kern.sun, jiaying_liu, y_shuo}@outlook.com, boxu@dlut.edu.cn, f.xia@ieee.org

Abstract—Features representation leverages the great power in network analysis tasks. However, most features are discrete which poses tremendous challenges to effective use. Recently, increasing attention has been paid on network feature learning, which could map discrete features to continued space. Unfortunately, current studies fail to fully preserve the structural information in the feature space due to random negative sampling strategy during training. To tackle this problem, we study the problem of feature learning and novelty propose a force-based graph learning model named GForce inspired by the spring-electrical model. GForce assumes that nodes are in attractive forces and repulsive forces, thus leading to the same representation with the original structural information in feature learning. Comprehensive experiments on benchmark datasets demonstrate the effectiveness of the proposed framework. Furthermore, GForce opens up opportunities to use physics models to model node interaction for graph learning.

Index Terms—Network feature learning, Spring-electrical model, Label prediction, Graph visualization

I. INTRODUCTION

A network is composed of a set of nodes and edges, which is commonly employed to represent data in real world. The effective representation of network features can significantly improve the performance of downstream tasks [1]. For example, in knowledge graph research, effective entity feature representation can promote the establishment of more accurate relationship between entities [2]; in online social recommendation systems, capable user feature representation is conducive to accurate recommendation [3]. Therefore, it is necessary to develop algorithms that could automatically extract features representation from the original graph, thus freeing people from tedious feature extraction works.

Network feature learning [4] is critical for network analysis tasks, such as node clustering [5], label prediction [6], and link prediction [7]. It could map desired information of vertices or edges to low dense feature vectors in a continuous feature space. The features extracted from the network simultaneously preserve network structural information, node attributes information, and promote users to deeply understand hidden characteristics of networked data. What calls for special attention is that most existing network feature learning algorithms could not fully preserve original features of network structure due to the random sampling of negative nodes [6]. Specifically, unconnected nodes have similar representation in the feature space due to the mechanism of most current methods. For example, Node2vec [6] and DeepWalk [8] algorithms are based on skip-gram model [9]–[11]. They rely on negative sampling to calculate nodes similarity. Therefore, negative
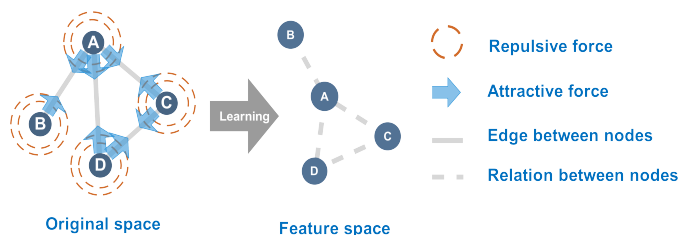

Fig. 1. The principle of GForce. The dash circle represents the repulsive force between nodes. Two nodes will move far away with the force. The vector arrow represents an attractive force between two nodes. Finally, nodes will learn the right location (vector) in the feature space.

samples that are not selected might be very close to the target nodes in the feature space. Matrix factorization based methods, such as GraRep [12], HOPE [13], cannot get rid of the disadvantages of matrix decomposition, e.g., computational complexity. Recent advancements in network feature learning mainly focus on deep learning, such as graph convolutional networks [14], graph attention networks [15], and graph capsule networks [16]. In these studies, the total loss function and stochastic gradient descent (SGD) are employed to optimize weight matrices for minimizing the loss function. A large amount of parameter matrices of perceptrons are required to be updated, but still cannot resolve the problem of gradient explosion.

In this work, we propose a non-neural network feature learning algorithm, namely GForce, that can fully preserve the structure in the feature space. GForce is inspired by the spring-electrical model [17], which simulates the motion of particles under force. Such idea has been successfully applied to graph visualization [18]–[20] in 2D and 3D space, aiming at presenting the original network topology of graph. The similar and connected nodes tend to be located in the similar position or belonging to the same cluster. Therefore, we believe that the positions of nodes in 2D or 3D space can be regarded as vectors of nodes in feature space, so similar embedding vectors can represent similar information of nodes. We adopt the idea of spring-electrical model, and extend embedded dimensions to any dimensions. Our goal is to fully extract structural features from the observed network rather than using any hand-engineered features. We assume that each node is repulsive to other nodes but attractive to neighbors. The two connected nodes will be relatively close to each other, otherwise, nodes will be far from each other in feature space. The core idea of GForce can be simply illustrated as Fig 1. In addition, GForce

can directly learn the vectors of nodes without using SGD. Every node calculates its own vector, and they move through attractive relations and repulsive relations in the feature space. Since every node computes its own vector, the algorithm could run in a parallel model with multi-threads and multi-processes.

We evaluate the performance of GForce against several baselines on five real-world datasets (i.e., Cora[1], Citeseer[2], Wikipedia[3], WebKB[4], 20newsgroups[5]). Experimental results show that our algorithm achieves better performance in label prediction and graph visualization in comparison with the state-of-the-art baselines (i.e., DeepWalk [8], LINE [5], Node2vec [6]). Our major contributions can be summarized as follows:

- We propose a novel graph force learning algorithm based on the spring-electrical model. The proposed GForce can sufficiently preserve structural information, which fills the gap of network feature learning.
- GForce could run in parallel with threads or multi-processes, that make full use of computing resources. The representation process of GForce depends on network structure information. Therefore, GForce owns the advantage of expandability in graph structure based application scenarios.
- We evaluate the GForce on real-world datasets in the tasks of label prediction and graph visualization. Experimental results show the effectiveness and efficiency of our proposed algorithm.

The rest of this paper is organized as follows. We first discuss related works in Section II. Then, we present related notations and definitions of graph feature learning in Section III. And then, we introduce the GForce in Section IV. In the following, we outline the experiments and present the results in Section V. Finally, we conduct our works in Section VI.

II. RELATED WORK

Network feature learning is always an important research problem in network science. Amount of previous studies have proved its validity for applications, such as link prediction, label prediction, and network visualization. There have been a large amount of network embedding models proposed in recent years, such as linear methods PCA [21], and non-linear methods including IsoMap [22], multidimensional scaling (MDS) [23], and LLE [24]. These methods are mainly applied to network embedding and dimensionality reduction. However, these methods have disadvantages in both computational and statistical performance.

In recent years, some representative studies have been devoted to other aspects of this field. For example, GraRep [12], HOPE [13], TADW [25] are matrix factorization methods. They utilize matrix factorization to obtain nodes embeddings.

DeepWalk [8], Node2vec [6] are established based on the Skip-gram model [9]–[11], which is a kind of natural language processing model. DeepWalk utilizes a truncated random walk to capture the network structure information. Node2vec improves the sampling strategy with walking preference parameters to maintain the local or global characteristics of the graph. Besides, some work focuses on edge modeling, such as LINE [5] and TLINE [26].

Taking advantage of the ability to extract hidden features in deep learning, recent advancements in network feature learning are mainly focused on deep learning based models. These deep learning technologies such as convolutional neural networks [27], autoencoder [28], attention neural networks [29], capsule neural networks [30], and reinforcement learning [31] are leveraged to network embedding [32], such as VGAE [33], Shifu2 [34], graph convolutional networks [35], [36], graph attention networks [15], [37], graph capsule networks [38], [39], graph generative network [40] and graph reinforcement learning [41]. These deep learning based models rely on a total loss function and utilize SGD to optimize weight matrices to minimize the loss function. Hence, these methods need to update a large number of weight matrices by using SGD.

Different from all these graph feature learning algorithms, we abandon the way of constructing complex graph neural-based model and the mechanism of message propagating between nodes, which are often utilized by recent advancements in network feature learning. Most of them rely on SGD to learn the feature vectors. We novelly involve physical model to graph feature learning, and propose the GForce. GForce enables each node to have the learning ability to calculate similarity or heterogeneity to others. This is why GForce can be computed in parallel without depending on a single loss function and SGD. Nodes can independently learn the feature vector through direct and concise operations according to the structure information while it could fully preserve original structure information in feature space.

III. PRELIMINARIES

We focus on network (graph) structure data. In this section, we formally give the related definitions of graph, network embedding, label prediction, etc.

DEFINITION 1: (Graph). A graph is defined as a set of nodes and edges, $G = (V, E, Y)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of nodes. Each node is usually associated with labels and attributes Y. The notation $E \subseteq (V \times V)$ is a set of edges between nodes. Each edge $e \in E$ connects a pair of nodes $e = (i, j)$. Edges are associated with weight $w_{ij} > 0$, which represents the strength of relations between two nodes. For undirected graph G, $e_{ij} = e_{ji}$. Otherwise, $e_{ij} \neq e_{ji}$.

In practice, graph is a kind of complex structure data. A graph can be directed (e.g., citation networks), undirected (e.g., social network), homogeneous (e.g., protein network), and heterogeneous (e.g., webpage network). Weights between nodes can be real or binary. This paper mainly studies
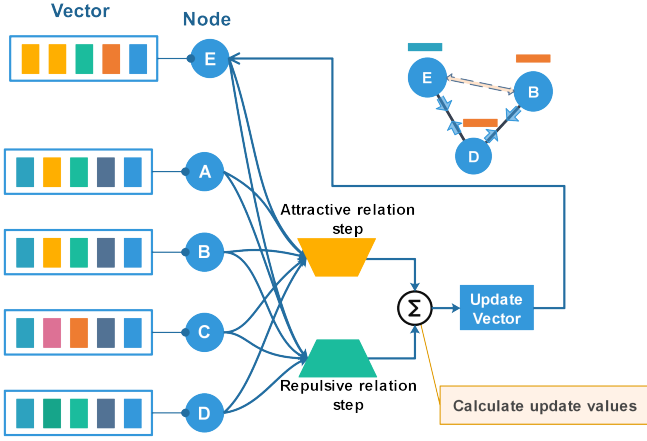
Fig. 2. The framework of GForce.

isomorphic graphs.

DEFINITION 2: (K-order Proximity). The K-order proximity is used to weigh the relation between two nodes in network, where $K \subseteq N+$ is a set of positive integers. For instance, K=1 or K=2 is corresponding to the first-order proximity and second-order proximity of nodes. The first-order (K=1) proximity is evaluated by the weight of the edge connected by two nodes. The second-order (K=2) proximity is determined by directed neighborhoods of two compared nodes. Specifically, given neighbors $N_i = \{s_i 1, \ldots, s_i |V|\}$ of node i and $N_j = \{s_j 1, \ldots, s_j |V|\}$ of node j, the second-order proximity between i and j is calculated as the similarity between N_i and N_j.

K-order proximity is based on the assumption that nodes tend to have similar information and attributes if they have a direct or indirect connection. For example, people who are friends with each other are likely to have similar interests in social networks. Two persons who have similar friends may also have the same preferences in shopping or other habits.

DEFINITION 3: (Network Embedding). Given a network G, the aim of network embedding is to learn a function that can map nodes to a continuous feature space $f : V \mapsto \mathbf{R}^d$, where $d \ll |V|$. In the feature space, the distance between nodes approximates to the real distribution of the original space, maintaining first-order proximity, second-order proximity or high-order proximity.

DEFINITION 4: (Label Prediction in Feature Space). Given feature vectors $\mathbf{U}$ generated from feature mapping function f by graph $G = (V, E, Y)$, we aim to learn a hypothesis H, that could aggregate labels to unseen data $H : \mathbf{U} \mapsto \mathbf{Y}$.

In the following, we introduce a force-based graph learning model that could fully preserves structure information.

IV. GRAPH FORCE LEARNING

The algorithm contains two main steps: attractive relation step and repulsive relation step similar to spring-electrical model that has attractive and repulsive components. Attractive relation step could make similar nodes move to closer positions. Repulsive relation component makes different nodes stay away from each other. The input of GForce is an undirected (un)weighted graph. Given a graph $G = \{V, E\}$, the extracted feature vectors of G is $\mathbf{U} \in \mathbf{R}^{|V| \times n}$, where the dimension of feature space: $n \gg 3$. The update process of feature vectors is expressed as the following equation:

$$\mathbf{U} = \mathbf{U} + h * (\nabla F_a + \nabla F_r), \tag{1}$$

where ∇F_a is the update of attractive relation step, ∇F_r is the update of repulsive relation step. The notation h is a parameter to control the learning speed. It can be a constant or a self-adjusting variable. In this paper, we adopt the calculation method proposed by Weinred et al. [42], which could adapt to both global and local learning speed on the graph. The feature vectors $\mathbf{U}$ update in each iteration. We define the distance between nodes i and j is $\mathbf{d}_{ij} = \mathbf{u}_i - \mathbf{u}_j$. The update of attractive relation of node i is expressed as following:

$$\nabla f_a(i) = -p * \hat{w}_{ij} * (\mathbf{u}_i - \mathbf{u}_j), \tag{2}$$

where $\mathbf{u}_i$ is an feature vector of nodes i. p is the learning rate, which controls the learning speed. It can be set to a constant (such as $p = 1$). $\hat{w}_{ij} \in \hat{\mathbf{W}}$ is a positive link weight between nodes i and j from a dataset. The repulsive relation strength is inversely proportional to Euler distance $\mathbf{d}$. However, when $\mathbf{d}$ approaches zero ($\mathbf{d} \to \mathbf{0}$), ∇f_r approaches infinite ($\nabla f_r \to \infty$). To overcome this weakness, we design a bias repulsive relation method. The update of it is expressed as:

$$\nabla f_r(i) = q * \tilde{w}_{ij} * \frac{(\mathbf{u}_i - \mathbf{u}_j)}{||\mathbf{u}_i - \mathbf{u}_j| + \mathbf{b}|^2}, \tag{3}$$

where $\mathbf{b}$ is a distance bias, which can be set to a small positive value, for example, $\mathbf{b} = \{0.01\}_1^n$. q has similar function to p, which controls the learning speed. $\tilde{w}_{ij}$ is a negative weight between two nodes divergence. We can regard the distance bias as the diameter of nodes. It means that two nodes cannot be located too close to avoid the infinite repulsive relation value between them. We define that node has attractive relations between its neighbours. By the Eq. 2, the update of the attractive relation of node k between neighbours is expressed as:

$$\nabla f_a(k) = -p * \sum_{i=1}^{Nei_k} \hat{w}_{ki} * \mathbf{d}_{ki}, i \in Nei_k, \tag{4}$$

where $\hat{w}_{ki} \in \hat{\mathbf{W}}$ is the weight between node k and node $i \in Nei_k$. Nei_k is the neighbors set of k. $\mathbf{d}_{ki}$ is one element

of matrix $\mathbf{D}_k = \{\mathbf{d}_{k1}, \mathbf{d}_{k2}, \ldots, \mathbf{d}_{k|V|-1}\}$. $\mathbf{D}_k \in \mathbf{R}^{\{|V|-1\}\times n}$ is defined as:

$$\mathbf{D}_k = \begin{bmatrix} \mathbf{u}_k - \mathbf{u}_1 \\ \cdots \\ \mathbf{u}_k - \mathbf{u}_{k-1} \\ \mathbf{u}_k - \mathbf{u}_{k+1} \\ \cdots \\ \mathbf{u}_k - \mathbf{u}_{|V|} \end{bmatrix}. \tag{5}$$

The $\nabla f_a(k)$ can be reformed as:

$$\nabla f_a(k) = -p * \sum_{i=1}^{Nei_k} \hat{w}_{ki} * (\mathbf{u}_k - \mathbf{u}_i). \tag{6}$$

By the Eq. 3, the update of repulsive relation between k and other nodes is expressed as:

$$\nabla f_r(k) = q * \sum_{i=1}^{|V|-1} \tilde{w}_{ki} * \frac{\mathbf{d}_{ki}}{\hat{d}_{ki}}, \tag{7}$$

$\hat{d}_{ki}$ is the ith element of $\hat{\mathbf{D}}_k$, which is formulated as:

$$\hat{\mathbf{D}}_k = \mathbf{S} \cdot (\tilde{\mathbf{D}}_k^\top \circ \tilde{\mathbf{D}}_k^\top), \tag{8}$$

where $\circ$ is the Hadamard product of matrix. The matrix $\mathbf{S} = [1_1, 1_2, \ldots, 1_{n-1}, 1_n]$, which is used to sum columns of $\tilde{\mathbf{D}}_k^\top \circ \tilde{\mathbf{D}}_k^\top$. The notation $\tilde{\mathbf{D}}_k$ is the bias distance matrix, whose element is represented as:

$$\tilde{\mathbf{d}}_{ki} = |\mathbf{d}_{ki}| + \mathbf{b}, \tag{9}$$

where $\mathbf{d}_{ki}$ is an element of $\mathbf{D}_k$. So, $\nabla f_r(k)$ can be reformed as:

$$\nabla f_r(k) = q * \sum_{i=1}^{|V|-1} \tilde{w}_{ki} * \frac{\mathbf{u}_k - \mathbf{u}_i}{||\mathbf{u}_k - \mathbf{u}_i| + \mathbf{b}|^2}. \tag{10}$$

GForce directly updates the vectors through the nodes. In other words, each node has its own feature vector and updates itself, so the algorithm can learn and operate in parallel. We can assign the designated nodes to different CPUs, and each CPU calculates features of nodes independently. Nodes will reach a position where the forces are balanced after independent calculations according to nodes' relations. To measure whether all nodes are in equilibrium, we define the energy E of the graph:

$$E = \sum_{k=1}^{|V|} ||\nabla f_a(k) + \nabla f_r(k)||_2^2. \tag{11}$$

Symbol E can be used to stop the iteration of learning progress. When E is stable, it means that the node has reached its equilibrium position. In the proposed framework, each node retains its own feature vector $\mathbf{u}_k$, and updates it through the influence of relations imitating force in each iteration. The framework of GForce is shown in Fig 2. The pseudo-code of GForce is presented in Algorithm 1.

Algorithm 1 GForce Algorithm
___
Input: graph $G = \{V, E\}$, randomly initial feature vectors $\mathbf{U} \in \mathbf{R}^{|V|\times n}$ of nodes;
Output: feature vectors $\mathbf{U}$;
repeat
    for $k \in |V|$ **do**
        Calculate $\mathbf{D}_k$ by Eq. 5 and $\hat{\mathbf{D}}_k$ by Eq. 8
        $\nabla f_r(k) = q * \sum_{i=1}^{|V|-1} \frac{\mathbf{d}_{ki}}{\hat{d}_{ki}} * \tilde{w}_{ki}$;
        $\nabla f_a(k) = -p * \sum_{i=1}^{N_k} \mathbf{d}_{ki} * \hat{w}_{ki}$;
        $\nabla f(k) = \nabla f_a(k) + \nabla f_r(k)$;
        Calculate learning speed h;
        $\mathbf{u}_k = \mathbf{u}_k + h * \nabla f(k)$;
    end for
    Move nodes in the feature space;
until Reached minimal energy state;
___

TABLE I
DATASETS USED IN OUR EXPERIMENTS

Dataset	Cora	Citeseer	Wikipedia	WebKB	20newsgroups
|V|	2,708	3,312	2,405	877	1,727
|E|	5,429	4,732	17,981	1,605	2,982,529
|Y|	7	6	20	5	20
Vertex attr	Yes	Yes	Yes	No	No

V. EXPERIMENTS

In this section, we present a series of experimental analyses on GForce. We thoroughly evaluate the performance of our model by conducting extensive experiments analysis from three aspects including embedding analysis, label prediction, and graph visualization.

A. Experimental Setup

GForce could run in parallel. We implement a multi-processes version of GForce, for which, each process could handle operations of a certain number of nodes. In the following experiments, we set the parameters $p = 1$ and $q = 5$. We use energy E to determine whether the learning process is complete. If E is stable, which means node has reached its equilibrium position and learning process is finished.

Before conducing experiments with real-world datasets, we first simply verify whether GForce follows the embedding results, that is, the two connected nodes in the feature space will be relatively close to each other, otherwise, nodes maintain far from each other. We learn feature vectors from a 15×15 grid graph, and then map the dimensionality reduction vector to nodes in 2D space to observe whether nodes follow the embedding results. Then we conduct an experiment on the dataset to verify the relationship between the embedding dimension and structure embedding accuracy.

To test the feature learning performance of GForce, we compare it with baselines on benchmark network datasets. These baselines are all graph structure-based embedding algorithm focusing on first-order proximity and second-order proximity. In the label prediction experiment, we train our algorithm on the citation network, and webpage network. We

TABLE II
SUMMARY OF BASELINES

Methods	First-order	Second-order	Higher-order
DeepWalk		✓	✓
LINE	✓	✓	
Node2vec		✓	✓
GraRep		✓	✓
HOPE		✓	✓
SDNE	✓	✓	

randomly sample a portion of the labeled nodes as training samples, and the rest as the test set. In each group of controlled experiments, we randomly select data from 40% to 80% of a set of labels and repeat 5 times. We compute the average Micro-F1 socre and Macro-F1 score for label prediction task. In the graph visualization experiment, we utilize t-SNE toolkit to reduce feature vectors extracted from the 20newsgroups dataset. Nodes of the same type are assigned the same color. Therefore, clustering the same nodes together can represent a better representation.

1) Datasets: Five different benchmark datasets including Cora, Citeseer, Wikipedia, WebKB, and 20newsgroups are selected to implement experiments. The overview of datasets is presented in Table I.

- Cora dataset is a citation network, which is one of the most commonly used benchmark dataset to verify the performance of network feature learning. In the dataset, each vertex (publication) is represented by a unique number. Links between vertexes (publications) are stored in the form of adjacency matrix or edge lists. Labels and attributes of vertexes are stored in two different dictionary files.
- Similar to Cora, Citeseer is a citation network that includes publications, publication classes, and links. Each attribute of the publication is extracted from the abstract. Attribute is represented by a 0/1-valued sequence. Each single value of attribute indicates whether the keyword exists in the abstract.
- Wikipedia dataset is a co-occurrence network extracted from Wikipedia website, containing keywords, their links and labels. This dataset has been pre-processed for ease of use. All nodes and labels are replaced with data in digital string format.
- WebKB (World Wide Knowledge Base) is a dataset of webpages. WebKB contains original hyperlinks of web pages. The data in the dataset are organized in a similar way to other datasets.
- 20newsgroups dataset is is one of the international standard data sets for text classification and text extraction. The dataset has about 20,000 newsgroup documents, which were evenly divided into 20 newsgroups with different topics.

2) Baseline Methods: We validate the performance of GForce by comparing it with graph structure-based baselines including DeepWalk [8], LINE [5], Node2vec [6], GraRep [12], HOPE [13] and SDNE [43]. DeepWalk,
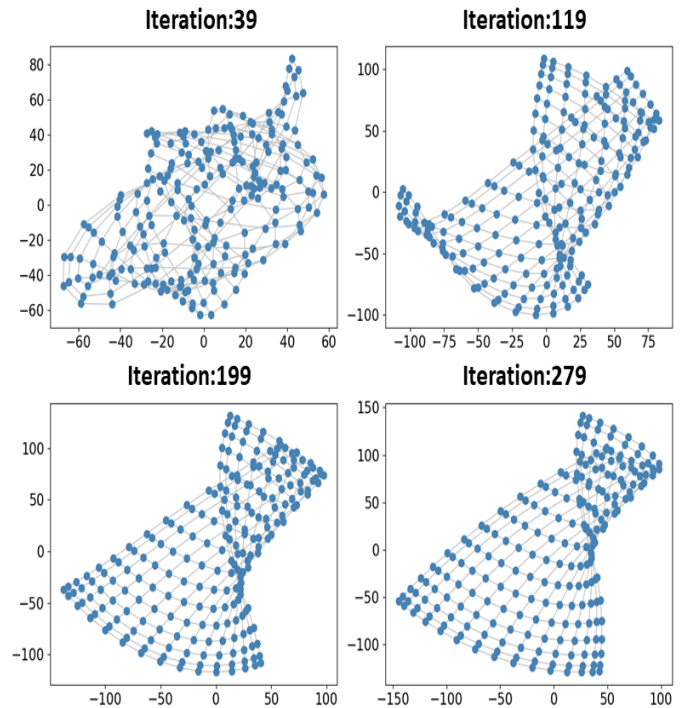

Fig. 3. Verification embedding results of GForce on a grid graph.

Node2vec, GraRep, and HOPE focus on global graph characteristics including second-order and higher-order proximities. LINE can simultaneously maintain the first-order and second-order proximity of the graph. The properties of them are summarized in Table II. The detailed descriptions of these methods are listed as follows.

- DeepWalk [8] algorithm is established on word2vec model. The algorithm utilizes random walk to sample graph and then generates nodes sequence, that is similar to word sequence. DeepWalk could keep the local and global graph characteristics.
- LINE [5] is a kind of network embedding learning algorithm for large graphs, which can preserve the first-order and second-order proximity of the graph.
- Node2vec [6] is an improved algorithm based on Deep-Walk. The algorithm is combined with biased sampling strategy with parameters p and q so as to keep local or global characteristics of the graph.
- GraRep [12] also leverages the skip-gram model [10] but keeps the high-order proximity of graph. The main idea is that nodes with k-steps neighbors should have similar representations in the embedding space.
- HOPE [13] could capture the asymmetric high-order proximity in a directed graph, which is a kind of matrix factorization based network embedding algorithm.
- SDNE [43] could preserve first-order and seconder-order proximity of nodes. It is a network representation learning model based on the deep autoencoder approach.

TABLE III
RESULTS OF LABEL PREDICTION ON THREE BENCHMARK DATASETS

| Datasets		Cora					Citeseer					Wikipedia				
Maetric	Algorithm	40%	50%	60%	70%	80%	40%	50%	60%	70%	80%	40%	50%	60%	70%	80%
Micro-F1	DeepWalk	0.761	0.770	0.780	0.787	0.789	0.576	0.578	0.589	0.590	0.594	**0.672**	0.680	0.683	0.687	0.688
	Node2vec	0.810	0.811	0.815	0.822	0.825	0.585	0.580	0.594	0.597	0.609	0.659	0.667	0.682	0.684	0.690
	LINE	0.755	0.773	0.778	0.797	0.797	0.493	0.500	0.501	0.507	0.527	0.645	0.666	0.652	0.650	0.652
	GraRep	0.768	0.769	0.781	0.803	0.785	0.548	0.545	0.542	0.545	0.545	0.647	0.651	0.656	0.662	0.659
	HOPE	0.644	0.652	0.654	0.676	0.656	0.452	0.448	0.453	0.457	0.452	0.591	0.606	0.611	0.599	0.609
	GForce	**0.817**	**0.832**	**0.837**	**0.838**	**0.865**	**0.643**	**0.648**	**0.649**	**0.648**	**0.647**	0.626	**0.685**	**0.689**	**0.699**	**0.700**
Macro-F1	DeepWalk	0.732	0.765	0.768	0.776	0.782	0.525	0.528	0.541	0.528	0.544	0.568	**0.580**	0.592	0.585	0.590
	Node2vec	0.776	0.806	0.808	0.812	0.817	0.545	0.531	0.543	0.554	0.572	**0.649**	0.547	0.548	0.561	0.559
	LINE	0.715	0.762	0.766	0.794	0.801	0.463	0.462	0.461	0.470	0.483	0.572	0.548	0.558	0.570	0.568
	GraRep	0.746	0.745	0.761	0.786	0.777	0.485	0.480	0.574	0.475	0.482	0.498	0.507	0.507	0.522	0.519
	HOPE	0.626	0.630	0.632	0.652	0.634	0.392	0.386	0.393	0.392	0.394	0.426	0.438	0.440	0.433	0.432
	GForce	**0.778**	**0.818**	**0.821**	**0.827**	**0.854**	**0.595**	**0.595**	**0.590**	**0.595**	**0.594**	0.457	0.570	**0.602**	**0.615**	**0.618**
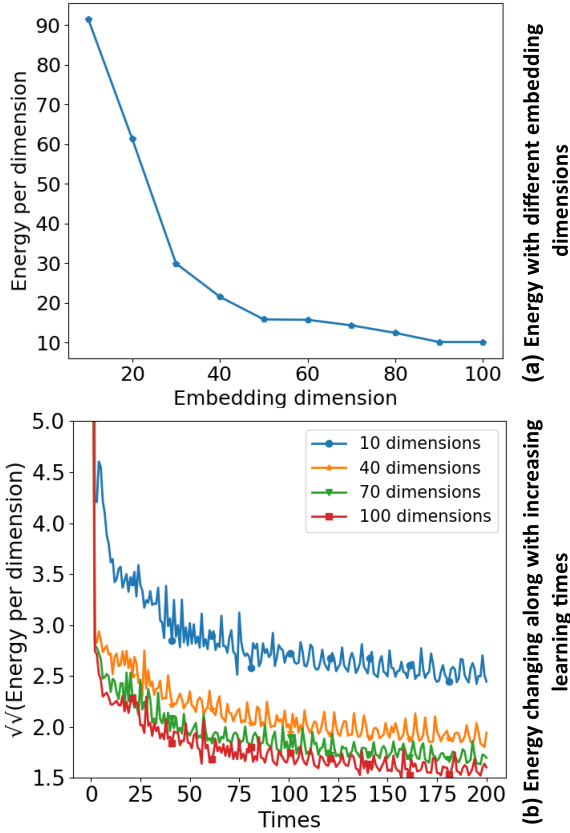
Fig. 4. Relations between the dimension of feature space and energy. As shown in the top figure, high embedding dimension can achieve low energy under the same conditions. As shown in the bottom figure, the red line of 100 dimensions is consistently lower than that of other dimensions with the increasing learning time. Low energy means the corresponding dimensions of embedding space could effectively embed original graph structure information.

B. Quantitative Results

The experimental results are summarized from Fig 3 to Fig 5 and Table III. We can make the following observations from the results.

1) Embedding Analysis: To verify that GForce is able to effectively embed structure information, we first conduct embedding analysis experiments. In the following, we present verification purposes and corresponding results.

Structure Embedding Verification. GForce follows the principle of spring-electrical model. To ensure the reliability of the structure embedding, we utilize GForce to extract structure information of 15×15 grid graph. To facilitate the observation of embedded results, we map nodes with vectors to 2D feature space with an increasing frame.

From the Fig 3, we can see that nodes are in chaos when the iteration time is small. As the number of learning iterations increases, unconnected nodes gradually move away from each other. Therefore, the nodes gradually learn the correct vector to become a grid graph. Experimental results show that GForce follows the aforementioned embedding hypothesis and can be used for structural feature extraction.

Embedding Accuracy with Dimensions. The feature learning of GForce maps nodes into a relatively high continuous feature space ($n \gg 3$), because high dimensional feature space could effectively represent features. To prove this conclusion, we verified it through the WebKB dataset. The results are shown in Fig 4.

From the Fig 4(a), the energy per dimension of node decreases with the increasing dimension of feature space, which means that the energy of nodes in high-dimensional feature space is lower. The low energy indicates that nodes have reached the equilibrium position in the feature space. From Fig 4(b), the energy of dimension in 100-dimensions feature space is consistently lower than others with increasing learning times. In conclusion, high-dimensional feature space can better preserve the original structure information.

2) Label Prediction: We conduct label prediction on three benchmark datasets. The results are shown in Table III.

From the Table III, we can see a significant improvement in GForce compared with baselines on the Cora dataset with both Micro-F1 and Macro-F1 score. As the training ratio increases, GForce obtains higher accuracy. Node2vec is based on DeepWalk and obtains the best performance among baseline algorithms. The prediction accuracy of GForce is consistently better than Node2vec on both Micro-F1 and Macro-F1. Compared with Node2vec, the best performance of GForce in Micro-F1 and Macro-F1 exceed 0.040 and 0.037 respectively. DeepWalk and LINE produce similar results, and can always improve prediction accuracy on HOPE.
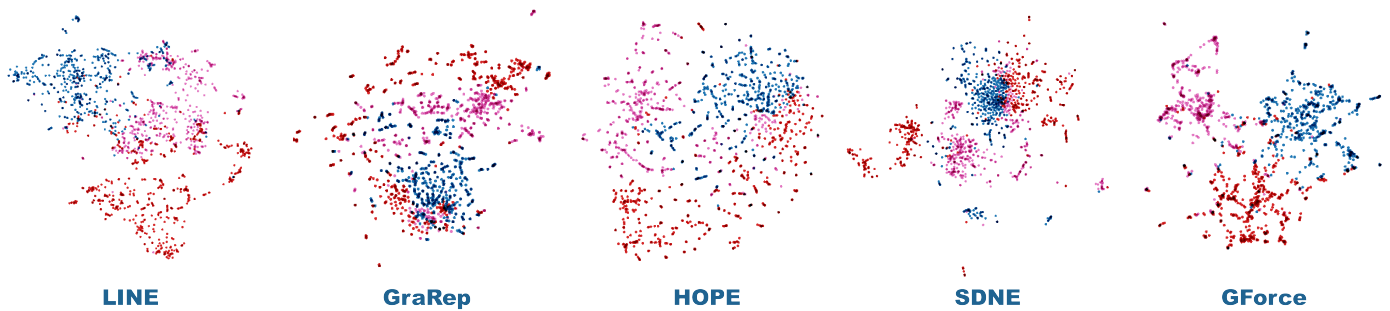
LINE **GraRep** **HOPE** **SDNE** **GForce**

Fig. 5. Visualization results of 20newsgroups dataset. Positions of vertexes are two-dimensional vectors reduced by t-SNE from the learned feature vectors. Color of a vertex indicates the category of the webpage.

From the Table III of label prediction results on Citeseer dataset, we can observe that the Micro-F1 score of GForce is consistently exceeding 0.6, but other baselines are not. In this experiment, the performance of baselines is similar to previous results. Node2vec performs best among the baselines. In addition, DeepWalk is slightly better than GraRep in both Micro-F1 and Macro-F1 scores.

From the Table III of label prediction results on Wikipedia dataset, DeepWalk achieved 0.672 on Micro-F1 score at 40%, which is over other baselines and GForce. However, node2vec achieved 0.649 on Macro-F1 score at 40%, that is better than other baselines and GForce. GForce obtained higher accuracy in Micro-F1 at 50% than others, and exceed baselines on both Micro-F1 and Macro-F1 with higher proportions of training set.

3) Graph Visualization: We select 3 classes of documents from the 20newsgroups dataset. Different colors in figures indicates different categories.

As can be seen from Fig 5, the layouts using SDNE and GraRep are not very effective because most different types of nodes are mixed. Visualization results of HOPE and LINE are better because nodes are clustered in separable groups with clear boundaries. However, most nodes of the same type are not close in the same group. For GForce, different types of nodes are relatively concentrated in distinguishable regions, and each group has a clear boundary. Therefore, from the visualization results, GForce achieves better performance than other baselines.

VI. Conclusion

In this paper, we propose GForce algorithm, which can better represent the relations among nodes in network. The main two steps of GForce are attractive relation step and repulsive relation step, which can be a (non)linear method to capture hidden relationship between nodes. GForce is different from most network representation learning models that rely on global optimal solutions. It can learn the feature vector by the node itself. This feature makes the algorithm own the ability of parallel computing. The main advantage of GForce is that it can fully capture the feature of the graph. Empirically, we verify the performance of GForce through the tasks of label prediction and graph visualization on real-world network datasets. The results show that our algorithm achieves better performance than state-of-the-art baselines. In the future, gravity method will be taken into consideration in our proposed model, which could map the node towards a high degree node (a fix point) in the feature space. In order to further reduce computational complexity, edge-cutting strategy will also be considered. Learning each subgraph individually can effectively reduce learning complexity and shorten learning time.

References

[1] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[2] C. Shang, Y. Tang, J. Huang, J. Bi, X. He, and B. Zhou, "End-to-end structure-aware convolutional networks for knowledge base completion," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3060–3067.

[3] L. Wu, P. Sun, Y. Fu, R. Hong, X. Wang, and M. Wang, "A neural influence diffusion model for social recommendation," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2019, pp. 235–244.

[4] P. Cui, X. Wang, J. Pei, and W. Zhu, "A survey on network embedding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 5, pp. 833–852, 2018.

[5] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[6] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 855–864.

[7] L. Wang, J. Ren, B. Xu, J. Li, W. Luo, and F. Xia, "Model: Motif based network embedding for link prediction," *IEEE Transactions on Computational Social Systems*, vol. 7, no. 2, pp. 503–516, 2020.

[8] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2014, pp. 701–710.

[9] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model." in *In Proceedings of the International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 246–252.

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

[11] 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.

[12] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. ACM, 2015, pp. 891–900.

[13] M. Ou, P. Cui, J. Pei, Z. Zhang, and W. Zhu, "Asymmetric transitivity preserving graph embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 1105–1114.

[14] Y. Ma, S. Wang, C. C. Aggarwal, and J. Tang, "Graph convolutional networks with eigenpooling," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 723–731.

[15] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, "Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 137–146.

[16] T. Hahn, M. Pyeon, and G. Kim, "Self-routing capsule networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 7658–7667.

[17] M. E. Ko, C. M. Williams, K. I. Fread, S. M. Goggin, R. S. Rustagi, G. K. Fragiadakis, G. P. Nolan, and E. R. Zunder, "Flow-map: a graph-based, force-directed layout algorithm for trajectory mapping in single-cell time course datasets," *Nature Protocols*, pp. 1–23, 2020.

[18] A. Arleo, W. Didimo, G. Liotta, and F. Montecchiani, "A distributed multilevel force-directed algorithm," *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 4, pp. 754–765, 2018.

[19] H. Haleem, Y. Wang, A. Puri, S. Wadhwa, and H. Qu, "Evaluating the readability of force directed graph layouts: A deep learning approach," *IEEE Computer Graphics and Applications*, vol. 39, no. 4, pp. 40–53, 2019.

[20] B. Jenny, D. M. Stephen, I. Muehlenhaus, B. E. Marston, R. Sharma, E. Zhang, and H. Jenny, "Force-directed layout of origin-destination flow maps," *International Journal of Geographical Information Science*, vol. 31, no. 8, pp. 1521–1540, 2017.

[21] J. Lever, M. Krzywinski, and N. Altman, "Points of significance: Principal component analysis," *Nature Methods*, vol. 14, no. 7, pp. 641–643, 2017.

[22] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[23] N. Saeed, H. Nam, M. I. U. Haq, and D. B. Muhammad Saqib, "A survey on multidimensional scaling," *ACM Computing Surveys (CSUR)*, vol. 51, no. 3, pp. 1–25, 2018.

[24] H. Cao, H. Zhang, C. Wang, and M. Liu, "Supervised locally linear embedding for polarimetric sar image classification," in *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. IEEE, 2016, pp. 7561–7564.

[25] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Chang, "Network representation learning with rich text information," in *24th International Joint Conference on Artificial Intelligence*, 2015.

[26] X. Zhang, W. Chen, and H. Yan, "Tline: scalable transductive network embedding," in *Asia Information Retrieval Symposium*. Springer, 2016, pp. 98–110.

[27] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3075–3084.

[28] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4066–4075.

[29] Q. Kong, C. Yu, Y. Xu, T. Iqbal, W. Wang, and M. D. Plumbley, "Weakly labelled audioset tagging with attention neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 11, pp. 1791–1802, 2019.

[30] T. Vijayakumar, "Comparative study of capsule neural network in various applications," *Journal of Artificial Intelligence*, vol. 1, no. 01, pp. 19–27, 2019.

[31] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 1282–1289.

[32] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2020.

[33] H. Shi, H. Fan, and J. T. Kwok, "Effective decoding in graph auto-encoder using triadic closure," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 01, 2020, pp. 906–913.

[34] J. Liu, F. Xia, L. Wang, B. Xu, X. Kong, H. Tong, and I. King, "Shifu2: A network representation learning based model for advisor-advisee relationship mining," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[35] L. Yao, C. Mao, and Y. Luo, "Graph convolutional networks for text classification," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 7370–7377.

[36] J. Xu, S. Yu, K. Sun, J. Ren, I. Lee, S. Pan, and F. Xia, "Multivariate relations aggregation learning in social networks," in *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries in 2020*, 2020, pp. 77–86.

[37] W. Song, Z. Xiao, Y. Wang, L. Charlin, M. Zhang, and J. Tang, "Session-based social recommendation via dynamic graph attention networks," in *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, 2019, pp. 555–563.

[38] J. Ma, P. Cui, K. Kuang, X. Wang, and W. Zhu, "Disentangled graph convolutional networks," in *International Conference on Machine Learning*, 2019, pp. 4212–4221.

[39] K. Ahmed and L. Torresani, "Star-caps: Capsule networks with straight-through attentive routing," in *Advances in Neural Information Processing Systems*, 2019, pp. 9098–9107.

[40] A. Grover, A. Zweig, and S. Ermon, "Graphite: Iterative generative modeling of graphs," in *International Conference on Machine Learning*, 2019, pp. 2434–2444.

[41] Y. Wang, Y. Tong, C. Long, P. Xu, K. Xu, and W. Lv, "Adaptive dynamic bipartite graph matching: A reinforcement learning approach," in *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 2019, pp. 1478–1489.

[42] C. Weinreb, S. Wolock, and A. M. Klein, "Spring: a kinetic interface for visualizing high dimensional single-cell expression data," *Bioinformatics*, vol. 34, no. 7, pp. 1246–1248, 2018.

[43] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1225–1234.