Journal of Intelligent Information Systems (2021) 57:51–72
https://doi.org/10.1007/s10844-020-00625-6

Community detection in complex networks using network embedding and gravitational search algorithm

Sanjay Kumar[1,2] · B S Panda[2] · Deepanshu Aggarwal[1]

Received: 2 June 2020 / Revised: 17 August 2020 / Accepted: 11 October 2020 /
Published online: 5 November 2020

Abstract

The structural and functional characteristic features of nodes can be analyzed by visualizing community structure in complex networks. Community detection helps us to detect nodes having similar behavior in a system and organize the network into a network of closely connected groups or modules. Network embedding technique represents the nodes of the input graph into vector space and preserves their inherent and topological features and can contribute significantly to various applications in network analysis. In this paper, we propose a novel community detection method using network embedding technique. Firstly, nodes of the graph are embedded in feature space of d dimensions, and then low-rank approximation is applied to avoid the results from being affected by noise or outliers. Further, k-means clustering is employed to find the centroids of the clusters in the network and followed by a gravitational search algorithm to improve the results of centroids of clusters. Finally, we compute the effectiveness of detected communities using different performance measures. Our method serves as a universal framework towards applying and bench-marking various embedding techniques in graphs for performing community detection. We perform the test using various evaluation criteria on several real-life and synthetic networks and the obtained result reveals the utility of the proposed algorithm.

Keywords Community detection · Complex networks · GSA (gravitational search algorithm) · k-means clustering · Network embedding · Social networks

✉ Sanjay Kumar
sanjay.kumar@dtu.ac.in

B S Panda
bspanda@maths.iitd.ac.in

Deepanshu Aggarwal
deepanshu4929@gmail.com

[1] Department of Computer Science and Engineering, Delhi Technological University, New Delhi 110042, India

[2] Computer Science and Application Group, Department of Mathematics, Indian Institute of Technology Delhi, Hauz Khas, New Delhi 110016, India

1 Introduction

Most real-life networks, like social networks, biological networks, transportation networks, power networks, collaboration networks are complex networks composed of a large number of active entities (Newman 2001, 2012; Freeman 2004; Saleh et al. 2019). The study of complex networks helps us identify natural and social system dynamics. It has led to an interdisciplinary association between researchers from different fields like physics, mathematics, biology, sociology, computer-science to collaborate, analyze, and reveal interesting features (Mitchell 2006; Aggarwal 2011; Aggarwal and Subbian 2014). A complex network can be represented as a graph $G = (V, E)$, where V denotes nodes or entities in the system, and E corresponds to the relationship between the vertices. For example, in online social networks, nodes are the person, and edges are the virtual acquaintance like friendship, follow-followee. In the case of the airport network in the world, cities are nodes, and an edge between two cities means there is a direct flight between them.

The presence of community structure is one of the inherent features and quite common in real-life networks (Fortunato 2010; Girvan and Newman 2002). In general, a community is defined as a group of densely interconnected nodes and sparsely connected to the rest of the network (Chouchani and Abed 2020). Finding community structure in complex networks reveals many exciting features and association between entities. Community detection is a type of graph partitioning problem with proper optimization require to find the solution and lies under the NP-hard category (Fortunato and Hric 2016). The idea of community detection relies on the similarity between entities having similar properties or interests in the network and associating them into the same cluster or community. Practically this results in similar nodes forming small groups with high connectivity, connected by sparse edges with the nodes of other groups. Communities in social networks correspond to a set of people sharing similar hobbies, interests, backgrounds (Ramezani et al. 2018). In a protein-protein interaction network, a community may refer to modules with similar biological functions (Mahmoud et al. 2013). A community in collaboration networks represents a group of researchers to plan their work in associations to accomplish common goals (Battiston et al. 2016). Hence, community detection helps to comprehend and uncover the functional behavior of such complex systems. Finding communities in complex networks have a wide range of applications. Few examples of these applications include influence maximization and viral marketing (Kumar and Panda 2020; Huang et al. 2019), link prediction in complex networks (Yadav and Rai 2020), identifying opinion leaders (Bamakan et al. 2109), suggesting products or friends in recommendation system (Sahebi and Cohen 2011), and information propagation in networks (Danon et al. 2008).

Community detection is one of the widely researched topics in the field of network science. In the recent years, numerous community detection algorithms have been proposed to reveal community structures. Most of them are based on graph-theoretic approach (Clauset et al. 2004; Blondel et al. 2008) like modularity optimization (Newman 2006a), label propagation (Cordasco and Gargano 2010) and nature-inspired algorithms (Jaradat and Hamad 2018; Messaoudi and Kamel 2019; Honghao et al. 2013; Pattanayak et al. 2019) with certain limitations in each method, and scaling issues for applying in large networks. Most of these conventional techniques work well in the system with a clear community structure, but their performance may deteriorates in a hazy structure network.

A general graph representation like adjacency matrix and adjacency list can be used to represent a complex network. But, for a vast network having millions of nodes, the conventional representation poses several challenges like high computational complexity, low parallelizability, and inapplicability of machine learning methods. Recently, many graph embedding or network embedding methods are proposed to represent networks in low-level vector representation while preserving the topology and various features of nodes. We exploit such preservation of the topological features of nodes using the network embedding techniques and the notion of clustering in the feature space to uncover community structures. In this paper, we propose a community detection algorithm, which is a combination of network embedding and evolutionary-based methods. We use three state-of-art recently proposed network embedding methods, namely HOP, node2vec, and SDNE, in our algorithm. We obtain the low-dimensional vector representation of each node of the network by above said embedding methods then apply low-rank approximation to avoid the results from being affected by noise or outliers. Then we perform k-means clustering to obtain the centroids of clusters. The results of k-means may trap into local optima. The famous population optimization-based gravitational search algorithm is further employed to improve the results globally to detect community structures. Usually, swarm-based algorithms may suffer from the issue of random selection of the initial population and may fall into the problem of local optima. The output of k-means clustering as one of the initial population to GSA can serve as a valuable input, which is crucial for the GSA algorithm. We execute the proposed algorithm on many real-life and synthetic networks and estimate the effectiveness of identified communities using different performance measures like normalized mutual information (NMI), modularity, precision, and recall and obtained commendable results.

The main contributions of the proposed algorithm are:

– We devise a generic framework to apply various network embedding techniques on graph datasets to uncover community structure in complex networks.
– We introduce a community detection algorithm, which uses the notion of geometric space-based and evolutionary-based methods.
– The outcome of the k-means clustering as one of the input enhances the quality of the initial populations of GSA ensures a globally optimal solution.
– Experimental results obtained on various real-life and synthetic datasets using the different evaluation criteria reveal the effectiveness of the proposed algorithm.

The rest of the paper is organized as follows: Section 2 presents the the various community detection algorithms. Section 3 discusses the preliminaries of the work, including network embedding techniques, clustering methods, and gravitational search algorithm. The detailed descriptions of the proposed algorithm are described in Section 4. Section 5 lists the various real-life and synthetic datasets used for the experiments and the different evaluation criteria to judge the performance. Section 6 reports the results obtained and their analysis. Finally, we conclude the paper in Section 7.

2 Related work

Revealing the community structure manifested by real-life networks is a significant measure for an insight into complex systems, which go beyond the local structure of the members in the network. Due to the importance of community structure in the network analysis, numerous community detection algorithms are proposed by researchers of different disciplines. These methods are majorly partitioned into following three classes: graph theoretic based algorithms (Girvan and Newman 2002; Clauset et al. 2004; Blondel et al. 2008),evolutionary algorithms based techniques (Messaoudi and Kamel 2019; Hanghao et al. 2013; Pattanayak et al. 2019), and geometric space-based algorithms (Mahmood and Small 2015; Eustace et al. 2014; Ding et al. 2018). Graph-theoretic based algorithms exploit the various network topological features and construct like finding clique, clustering, label propagation, random walk, node centrality based methods, and others. Further such methods can be further subdivided into three categories -node-centric, and network-centric methods. In node-centric methods, nodes should satisfy some specific properties if they belong to a community. Some of these properties are - complete mutuality, reachability of members, node degrees, similarity between nodes. Usually, node centric method finds an initial subset of nodes as the seed nodes in the network and communities are extended from the seed set using techniques like label propagation (Gui et al. 2018; Ding et al. 2018). The node-centric Clique Percolation Method (CPM) rely on pre-defined substructures like finding cliques and k-cores within the network, which extracts all the maximal cliques in the network, to determine communities (Palla et al. 2005). CPM uses k-cliques to determine communities where a k-clique represents a complete subgraph with k nodes. Two k-cliques are said to be adjacent if they have an overlap of $k - 1$ nodes. Groups of adjacent k-cliques can form a network community. However, most node-centric methods often suffer from location and seed bias. To overcome such issue Ma et al. (2014) proposed a seed insensitive for local community detection. They measure the similarity among nodes by examining their neighborhoods and report a local community by maximizing internal similarity and minimizing external similarity simultaneously. Rosvall and Bergstrom (2008) proposed an information-theoretic approach known as infomap, a flow-based method that detects communities based on the map equation. Guo et al. (2020) proposed a local community detection algorithm using internal force. They obtain core members of communities as seeds in the network using local degree and Jaccard coefficient, and the fitness function uses the node with the highest degree among seeds every time.

Network-centric techniques rely on techniques like multidimensional scaling, spectral clustering, and modularity maximization (Newman 2006a; Zhou and Amini 2019). Newman (2006a) proposed a fast greedy approach for finding communities in the network which is based on the optimization of the quality function known as modularity. The objective of the partitioning of a network into communities is to maximize the intra-community interactions and minimize the inter-community interactions. In network-centric community detection, the connection of nodes are considered globally then partition the network into disjoint sets based on node similarity, latent space model, and modularity maximization. Based on modularity maximization, one of the popularly used methods for community detection is the Girvan and Newman (GN) Algorithm (Girvan and Newman 2002). It is a divisive hierarchical algorithm that removes links between vertices iteratively based on the value of betweenness of the links. This link removal process terminates upon reaching the maximum modularity of resulting partitions. The modularity formula of Newman and Girvan is a well-known quality function that estimates the goodness of a partition of the graph for community detection. However, the GN method experiences the issue of scalability and high time complexity. Recently, Arasteh and Alizadeh (2019) proposed a faster community detection technique based on the idea of edge degree betweenness centrality, and more than one edge can be removed to find the modular structure in the network using the idea of edge betweenness. The proposed method enhances the performance of the GN method.

In recent years, researchers explored various evolutionary methods to uncover community structure in networks and produced outstanding results. Honghao et al. (2013) introduced the Ant Colony Optimization (ACO) technique inspired by real ant colony formation heuristics for detecting community. Using a novel heuristic information and the locus-based adjacency presentation of the network, they adopted the max-min ant system (MMAS) framework to identify network community structure. Jaradat and Hamad (2018) proposed the firefly algorithm, where they randomly initialize the fireflies and move them using intensity equations and node connectivity for detecting communities. Authors in (Pattanayak et al. 2019) proposed a fire propagation-based community detection algorithm by relating the seed nodes in the community detection as the fire starting to form a source. They confined the search space by a two-radius neighborhood graph. Generally, evolutionary-based algorithms either set parameters in advance or uses a random method to produce the initial population leading to large search space and high computational complexity. Messaoudi and Kamel (2019) introduced multi-objective Bat Algorithm, which uses the Mean Shift algorithm to generate the initial population to control randomness in initial population choice, to offer better solutions. However, high complexity cost and non-scalability is the main challenge for these evolutionary algorithms.

Few geometric space-based has proposed in recent years has also gained popularity for finding communities. Mahmood and Small (2015) suggested a subspace-based community detection utilizing a sparse linear encoding using the least square method. The algorithm first computes the geodesic distance between every pair of nodes. After that, a similarity function is defined based on geodesic distance, and a similarity matrix is computed using similarity function and geodesic distance. Then a coefficient matrix is learned to represent a node(column) i as a linear combination of other nodes and store the linear coefficients and produce the linear encoding of the graph. High values of coefficients mean high similarity with that node(column), and lower values mean less similarity with the corresponding node. Finally, the algorithm applies spectral clustering to both adjacency matrix and linear encoding learned to find the communities. Ding et al. (2018) proposed a low-rank subspace learning-based community detection (LRSCD) algorithm. The algorithm performs low-rank decomposition to represent each node vector as a linear combination of the other nodes under the same subspace. After that, the proximity matrix is computed by the low-rank coefficient matrix, followed by spectral clustering to obtain the final communities. The above two methods learn low-dimensional vector representation or embedding features of nodes specific to community detection and then apply clustering algorithms to detect communities. In recent years, there are many remarkable network embedding methods have been devolved. In this paper, we use popular network embedding methods, first to find the low-dimensional vector representation of each node of the network followed by k-means clustering and then gravitational search algorithm to uncover the community structures. Hence, our proposed algorithm is a hybrid of geometric space-based and evolutionary-based methods.

3 Preliminaries

3.1 Network embedding

Many real-life complex systems are expressed as a graph such as online social networks, communication networks, biological networks. Conventional graph representation techniques may not be efficient in processing these voluminous network data to perform several analytic operations. Techniques that employ the representation of graph vertices in vector space so that different machine learning-based algorithms can be employed to perform various network analysis tasks. Such techniques are known as graph embedding or network embedding, which intend to learn a mapping of vertices to a low-dimensional space of features and preserving the inherent properties and topological characteristics of the network. Network embedding methods intend to preserve first-order proximity, second-order proximity, and higher-order proximity to preserve the network structure, where first-order proximity refers to the similarity between two nodes if an edge connects them. Similarly, second-order proximity refers to the similarity of two nodes, which are at a 2-hop distance, and if they have a similar neighborhood. Recently, researchers have come up with many network embedding algorithms, and these methods can be broadly classified into three categories- factorization based, random walk based, deep learning-based (Goyal and Ferrara 2018; Cui et al. 2018). In this manuscript, we utilize one popular recent method for embedding from each category- HOPE (higher-order proximity preserved embedding) (Ou et al. 2016) from factorization based methods, node2vec (Grover and Leskovec 2016) from random walk based techniques, and SDNE (structural deep network embedding) (Wang et al. 2016) from deep learning-based category. Their brief descriptions are as follows:

(i) HOPE:- Higher Order Proximity Preserved Embedding (HOPE) is a Factorization based methods, which represent the connections between nodes in the form of a matrix and achieve the embedding by factorizing the matrix. HOPE intends to preserve asymmetric transitivity in case of a directed graph. The algorithm obtains a general formulation of a class of high-order proximity measurements like Katz index and then utilize generalized SVD to find the overall solution.

(ii) node2vec:- It is a random walk based technique that uses random walks from a vertex (u) and calculates the probability of reaching a vertex (v) in that random walk. The more a vertex (v) occurs in random walks together with another vertex, the more is the chance that vertices u and v are closely related. The different random walks are fed into a skip-gram neural network to generate embedding of the graph. Node2vec has two controlling parameters p and q. The idea is to make a trade-off between local and global views of a network. The parameter p defines how probable is that the random walk would return to the previous node whereas q defines how probable is that the random walk would discover the undiscovered part of the graph Hence, p control discovery of the microscopic view around the node, and q controls the discovery of the broader neighborhood.

(iii) SDNE:- SDNE (structural deep network embedding) is a semi-supervised deep learning-based network embedding method to preserve the neighbor structures of nodes in the network with a high non-linearity and sparse structure. The method uses multiple layers of non-linear functions to preserve the first-order proximity and second-order proximity. SDNE uses the notion of Laplacian eigenmaps to preserve first-order proximity of vertices that captures the local structure of the network. The second-order proximity is used by the unsupervised component to capture the global network structure.

Hence, the embedding technique captures the geometric structure of the graph, its vertex-to-vertex connection, and all other related information in the form of low-dimensional vector representation so that various machine learning-based algorithms can be employed to perform network analysis tasks. Also, there is a trade-off between graph properties considered and the size of vectors in embedding. The embedding capturing more features requires more dimension d, results in better accuracy, but is computationally expensive. We need to maintain the proper balance between the size or dimension (d) of the vectors and the computational cost.

3.2 Gravitational search algorithm

The gravitational search algorithm (GSA) is a nature-inspired optimization algorithm that tries to search a solution using rules similar to the gravitational laws of nature (Rashedi et al. 2009), which requires polynomial time with respect to the problem size. The laws of gravity govern the force between the particles, their acceleration, and velocity are used to search the solution. The basic steps of GSA is outlined in Algorithm 1. Random initial solutions are generated by taking into account the limits of the solution. The values in the solution represent their position in different dimensions. Then these solutions referred to as particles that are assigned a fitness value. The fitness value is the measure of how good our solution is. Mass is assigned to each solution (particles), and force is calculated between them according to the law of gravitational force. The force then leads to acceleration, and that causes a particle to acquire velocity.

Algorithm 1 Gravitational search algorithm.

```
Input: $i$ : Search space dimensions,
    $S$: Number of initial particles
Output: Optimal Solution
 1: Select appropriate value for constants
 2: $Particles$ ← generate_solution(i,S)
 3: $Fitness$ ← calc_fitness(Particles)
 4: $Masses$ ← compute_mass(Fitness)
 5: $Acceleration$ ← calc_accelaration(Masses)
 6: $Velocity$ ← compute_velocity(Acceleration)
 7: $Particles$ ← get_new_position(Velocity,Particles)
 8: Repeat Step-3 to 7 until stopping condition is met
 9: Solution with best fitness value
```

Now, if we consider the time to be one unit, we can calculate the new position of the particles. We continue this until a terminating condition is met. The particles with better fitness will tend to attract the particles with lower fitness values towards themselves, so the fitness values of all the particles will improve after many iterations. GSA depends heavily on the choice of the initial population. A randomly chosen initial population may get struck into local minima and does not produce a near-optimum result. Usually, In GSA the initialization is random and hence take more iterations and computations to reach close to the optimum solution. If proper input is chosen, then GSA explores the solution space properly and provides an optimum result. The k-means result provides a good initial population to GSA. Hence, with better exploration near the result of k-means, GSA can provide a high-quality solution in fewer iterations. GSA has been utilized in many engineering applications like combinatorial optimization problems, clustering, and classification problem, training neural networks (Siddiquea and Adelib 2016).

4 Proposed community detection algorithm

This section presents the details of the algorithm adopted in this paper, which utilizes network embedding technique and gravitational search algorithm (GSA) to identify the communities in the network. These network embedding techniques aims to learn low-dimensional representations of vertices in networks, intending to capture and preserve the network structure. Using network embedding techniques, we consider nodes as points in d- dimensional vector space, such that two nodes that are connected by an edge are closer in feature space and non-connected nodes are away. The network embedding techniques aim to learn low-dimensional representations of vertices in networks and capturing the network structure. Using network embedding techniques, we consider nodes as points in d- dimensional vector space, such that two nodes that are connected by an edge are closer in feature space, and non-connected nodes are away. We intend to exploit such low-dimensional vector representation of nodes and the structural features of the network to perform community detection. The outline of the proposed algorithm is presented in Algorithm 2.

Algorithm 2 Proposed community detection algorithm.

```
Input: $G = (V, E)$: Graph, $k$ : Number of communities , $true\_labels$
Output: $NMI$
 1: $d \leftarrow$ select an appropriate dimension for embedding
 2: $method\_name \leftarrow$ chose a method for embedding
 3: $emb \leftarrow$learn_embedding($G, d, method\_name$)
 4: $var \leftarrow$ Choose appropriate var. for low rank approx.
 5: $emb\_low \leftarrow$ low_rank_approximation ($emb, var$)
 6: $predicted \leftarrow$ $k$-means ($emb\_low, k$)
 7: $gsa\_result \leftarrow$ GSA (predicted, S, $r_i$, $r_j$, iter,G, $\alpha$)
 8: $NMI \leftarrow$ calc_NMI (gsa_result, truth_labels)
 9: $Modularity \leftarrow$ calc_modularity (gsa_result, truth_labels)
10: $Precision \leftarrow$ calc_precision (gsa_result, truth_labels)
11: $Recall \leftarrow$ calc_recall (gsa_result, truth_labels)
12: return $NMI, Modularity, Precision, Recall$
```

The detailed descriptions of the proposed algorithm are as follows:

- Step 1: We first choose the appropriate vector space size, d, to embed the nodes of the graph into low-level vector representations. If we choose too small d, then embedding may not be able to capture the complete information from the graph. The more the dimension, the better it can achieve graph properties, but high computation time it requires. Here, d is a hyper-parameter, higher the value d captures the more graph properties. In the case of a simple graph with a uniform distribution of edges, even smaller value of d can properly capture the graph properties. If the graph is more complex, then we should select the d by considering the trade-off between graph properties to capture and computational complexity.
- Step 2: There are three classes of embedding methods, namely graph factorization based, random walk based, and deep learning-based. We consider one latest technique from each category to compare the results of community detection. We use the HOPE method from the graph factorization category, node2vec method from the random walk category, and SDNE method from the deep learning category.
- Step 3: In this step, we generate the vector embedding from the input graph (G), dimension (d), and the chosen method for embedding either HOPE, node2vec, or SDNE. The outcome of this step is the nodes of the input graph, each represented in a d dimensional vector space that captures the features of the network. Here, *emb* is a

two-dimensional matrix containing all nodes along with their d dimensional low-level vector representations.

– Step 4: To avoid the results from being affected by noise or outliers, we remove noise by low-rank approximation of the learned embedding. For low-rank approximation, we need to choose the amount of variance to preserve, which is also a hyperparameter. Therefore, this step involves the selection of variance for low-rank approximation. The variance should be optimum to remove unwanted noise.
– Step 5: In this step, we learn low-rank approximation using SVD (Singular Value Decomposition) method on embedded matrix emb as obtained in step 3 with chosen variance in the previous step. The diagonal matrix of SVD represents variance along different dimensions. Depending on the amount of variance to preserve, we keep some diagonal values and make others as zero. The result obtained in this step is stored in the new matrix, emb_low.
– Step 6: In this step, we apply k-means clustering algorithm to the result of low-rank approximation obtained in the previous step along with the number of clusters as input. We employ k-means technique as it is simple and scales well to large data items. Also, k-means finds a near-optimal solution in a feasible amount of time. We use k-means++ initialization for better results. The result of the k-means is a $k \times d$ dimension vector representing coordinates of k cluster centers.
– Step 7: The performance of k-means depends on the initial position of centroids and may confine in local optima. The gravitational search algorithm (GSA) is a famous population optimization-based technique based on the law of gravity. GSA is used to obtain a near-optimal solution and hence is a potential way to improve the performance of k-means. In this step, we perform a GSA search with an initial population of S particles, and the number of iterations (iter). It also takes parameters initial gravitational constant (G), the decay rate of the gravitational constant (α), and two random coefficients r_i and r_j for force and velocity equation in GSA respectively.

  Note that community detection is an NP-hard problem and an optimal solution can not be obtained in polynomial time unless $P = NP$. As it is unlikely that $P = NP$, one needs to find a near optimal solution in polynomial time. This is what GSA tries to do. But it is expected that if the results of k-means are input to GSA, high chances are there that GSA will be able to find a near optimal solution. Since, GSA is based on both exploration and exploitation, even if k-means result is struck in local optima, GSA will explore other solutions too and may lead to a better solution, which is more likely to be a global optimum or near global opimum. The execution and efficiency of the GSA algorithm mainly depend on S (number of initial solution particles) and iter (number of iterations). The value of S should be kept reasonable with respect to the problem size. A smaller value may not provide a good exploration whereas a large value will make the algorithm computationally expensive and may increase noise in search space. The outcome of this step is a two-dimensional vector having coordinates of k clusters flattened into a $k \times d$ dimension space. The outcome of this step is a two-dimensional vector having coordinates of k clusters flattened into a $k \times d$ dimension space and hence the result of the community detection of the input graph. We made the following changes to the original gravitational search algorithm, as mentioned in Algorithm 2, to adapt in our proposed approach and leading to optimize results in community detection:-

  (a) Initial Population: The initial population is not random, but it has the result of k-means clustering as one of the inputs. The rest of the particles are generated randomly.

(b) Fitness Function: The calc_fitness function, as in line number 3 of Algorithm 1 is defined as the squared sum of euclidean distances between the cluster centers in solution particle and the nodes belonging to that cluster. When cluster centers are close to all the points belonging to the cluster, the fitness value may be small. The optimization is a minimization problem to minimize this fitness value.

(c) Terminating Condition:- Since we are improving the results of clustering through GSA, stating whether we achieved minimum fitness is difficult. Hence, we run the GSA algorithm for a certain fixed number of iterations, and the solution with the best fitness value is returned as a result.

– Step 8 − 11: For effective evaluation of the obtained results, we analyze different performance criteria. Normalized Mutual Information (NMI) (McDaid et al. 2013) is an entropy-based evaluation criteria. To calculate NMI, we need to first calculate the community entropy with ground truth $H(Y)$ and predicted labels $H(C)$. Then we calculate the conditional probabilities, and then NMI is calculated using a series of equations from (1) to (6). To calculate the modularity, which is a measure of division among communities, is calculated using (7). Precision and recall are a measure of relevance of the result obtained and are calculated using (8) and (9), respectively.

We present the complete simulation of the proposed algorithm on Zachary Karate Club dataset having two ground-truth communities using embedding in the vector space of a minimal dimension (d)= 2, as depicted in Fig. 1. Figure 1a shows the karate dataset as input with 34 nodes and 78 edges. The input network is then converted to 2D coordinates using the node2vec embedding technique. In Fig. 1b each point represent a node in the

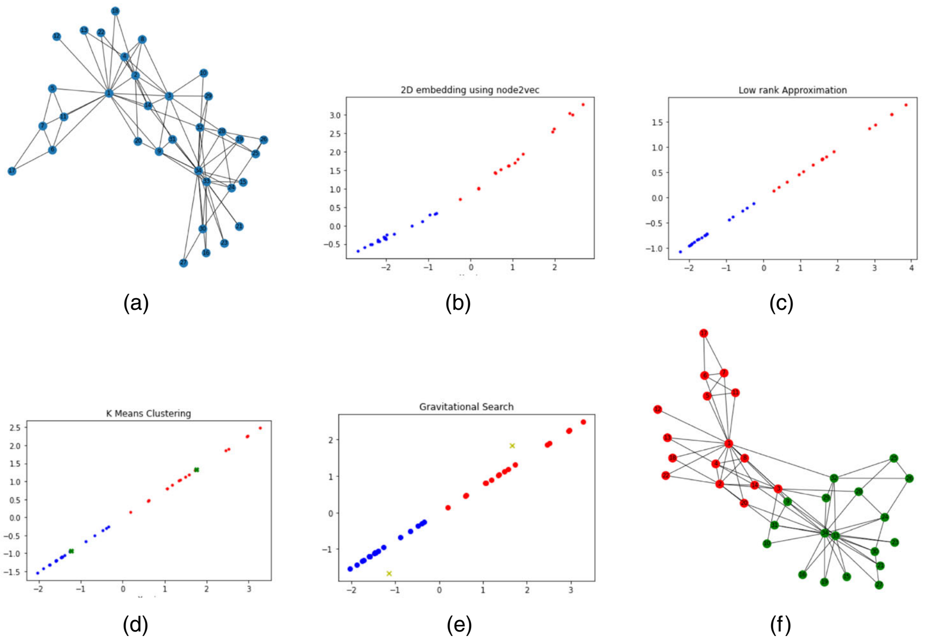


Fig. 1 Simulation of proposed Algorithm on Zachary Karate Club network with $2-d$ node2vec embedding-**a** input Zachary Karate Club dataset, **b** 2-D embedding using node2vec, **c** Low Rank Approximation, **d** Cluster Centers With k-Means, **e** Cluster Centers after GSA, and **f** Finally, obtained communities as output

network, and the color of each coordinate point shows the community to which it belongs according to the ground truth value. Figure 1c shows the new coordinates of the points after performing low-rank approximation and removing noise. Notice that the points are almost in a straight line making it easier to define boundaries of the clusters. Figure 1d depicts two cluster centers obtained by applying k-means. The result of k-means is optimized through the GSA, and new and improved cluster centers are obtained through GSA as presented in Fig. 1e. The cluster centers seem to be far apart from coordinates points but are better at classifying each coordinate into a different community. The final network divided into two communities, with red and green colors, each color representing different communities, as shown in Fig. 1f.

5 Datasets and evaluation criteria

For a better evaluation of the proposed algorithm, we choose different real-life and synthetic graph datasets with varying sizes, degrees of intermixing, density, and the number of clusters. We adopt eight datasets, out of which five are real-world datasets, and rest three are synthetic. Karate Club (Zachary 1977), American Football (Girvan and Newman 2002), Polblogs (Adamic and Glance 2005), Polbooks (Krebs 2020), Word (Newman 2006b) are the real-world networks datasets and three synthetic datasets created using LFR benchmark (Lancichinetti et al. 2008) with 128 nodes and varying mixing parameter (μ). For each chosen dataset, the ground-truth community value is known for each vertex. The brief description of each dataset is as follows:

5.1 Real-life datasets

(i) Zachary Karate Club (Zachary 1977): A social network of 34 members of a karate club at a US university in the 1970s. The nodes represent the members of the club and edges represent the friendship relationship between them. The network with 34 nodes has 78 edges and average degree 4.88. It is one of the most popular dataset used in community detection with two ground truth communities.

(ii) American Football (Girvan and Newman 2002): It is a network of football games between Division IA colleges during the Fall season, 2000. The network has 115 nodes and 613 edges with an average degree of 10.66, and 12 predefined communities.

(iii) Political Blogs (Adamic and Glance 2005): A network of hyperlinks between different weblogs on US politics, recorded in 2005 by Adamic and Glance. The political blogs network with 1490 nodes, 16718 edges, and an average degree of 25 with two ground-truth communities.

(iv) Political Books (Krebs 2020): A network of books on US politics published around the time of the 2004 election for president. Nodes represent books, and edges between books represent frequent co-purchasing of books by the same buyers. This network has 105 nodes and 441 edges with an average degree of 8. This is a relatively medium-sized dataset with three ground-truth communities.

(v) Word (Newman 2006b): It is a network of common words (nouns and adjectives) in the novel David Copperfield by Charles Dickens. This network has 112 nodes, and 425 edges with an average degree of 7. This is a relatively medium-sized dataset with loosely defined communities and 2 ground-truth communities.

5.2 Synthetic datasets

The datasets in this section are created using the LFR benchmark for an undirected graph (Lancichinetti et al. 2008). The create three datasets having all characteristics similar, but the level of intermixing among communities is varied. The level of intermixing is defined by the mixing parameter, which is the average fraction of neighboring nodes of a node that do not belong to any community that the benchmark node belongs to. At the extremes, when the parameter is set to 0, all links are within community links, and for mixing parameter=1, all links are between nodes belonging to different communities. We use three different variant of LFR datasets.

(i) LFR-1: It is a synthetic dataset created using LFR benchmark for undirected graphs. The graph has 128 nodes and an average degree of 16. The other LFR parameters are community size as 32, mixing parameter (μ) as 0.1, and exponent for weight distribution as 1. The graph has four ground-truth communities and is a graph with dense communities and less inter-mixing.

(ii) LFR-2:- It is a synthetic dataset created using the same LFR benchmark. All parameters are same as of LFR-1 except the mixing parameter 0.3. The graph has 128 nodes and average degree 16. The graph has 4 ground truth communities and is a graph with dense communities and small amount of inter-mixing.

(iii) LFR-3:- It is also a synthetic dataset created using the same LFR benchmark. All parameters are same as of LFR-1 except the mixing parameter which is set to 0.5 to get a graph with more intermixing among communities. The graph has 128 nodes and average degree 16 with four ground truth communities.

5.3 Evaluation criteria

(i) Normalized Mutual Information: The normalized mutual information (NMI) (Estévez et al. 2009) of two random variables is a normalized measure of the mutual dependence relation between the two variables. More specifically, it quantifies the ”amount of information” obtained about one random variable by observing the other random variable. The concept of mutual information is intricately linked to that of entropy of a random variable, a fundamental notion in information theory that quantifies the expected ”amount of information” held in a random variable. The three steps involved in the calculation are:

Calculating Entropy of Community Labels $H(Y)$ represents the entropy of communities using ground truth, whereas $H(C)$ represent the entropy using the predicted labels. Here n represents the total number of distinct communities in the ground truth, and m represents the number of distinct communities in the ground truth. Similarly, G_i represents probability of community i using ground truth whereas P_i represents probability calculated using predicted labels.

$$H(Y) = \sum_{i=1}^{n} G_i log(G_i) \tag{1}$$

, where G_i is the number of nodes belonging to community i using ground truth labels up on total number of nodes in the network.

$$H(C) = \sum_{i=1}^{m} P_i log(P_i) \tag{2}$$

, where P_i is the number of nodes belonging to community i using predicted labels up on total number of nodes in the network.

Calculating Conditional Entropy ($H(Y|C)$):- This is dependent on the outcome of the algorithm. It is calculated for each community C, and its sum is used further. Here n represents the total number of clusters in which the nodes are divided.

$$H(Y|C) = \sum_{i=1}^{n} P(i) \sum_{j=1}^{m} f(i, j) \tag{3}$$

$$f(i, j) = P(Y = j|C = i)log(P(Y = j|C = i)) \tag{4}$$

Calculating NMI:- Before calculating NMI, we calculate MI (Mutual Information). Then NMI is calculated by normalizing MI using (6).

$$MI = H(Y) - H(Y|C) \tag{5}$$

$$NMI = \frac{2 * MI}{H(Y) + H(C)} \tag{6}$$

(ii) Modularity: Modularity (Q) is designed to measure the strength of the division of a network into modules or communities. Modularity is often used for evaluating the qualities of the detecting community structure in networks. The value of modularity is computed using (7).

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{7}$$

where, m represents the number of edges in the network, A is the Adjacency matrix of the network, k_i is the degree of i and $\delta(c_i, c_j)$ is 1 if i and j are in the same community and 0 otherwise. The term $k_i k_j$ represents the probability that a random edge would go between i and j.

(iii) Precision and recall: precision, also called positive predictive value, is the fraction of relevant instances among the retrieved instances, while recall, also known as sensitivity, is the fraction of the total amount of relevant instances that were actually retrieved. Both precision and recall are, therefore, based on an understanding and measure of relevance. Precision shows the agreement between the known labels and predicted labels, whereas recall shows the effectiveness of the algorithm in predicting labels.

$$Precision = \frac{\sum_{i=1}^{n} tp_i}{\sum_{i=1}^{n}(tp_i + fp_i)} \tag{8}$$

$$Recall = \frac{\sum_{i=1}^{n} tp_i}{\sum_{i=1}^{n}(tp_i + fn_i)} \tag{9}$$

, where tp_i, fp_i, fn_i represents true positives, false positives and false negative for community c_i and n represents the total number of communities.

6 Experimental results and analysis

To properly evaluate the proposed algorithm, we chose a variety of networks varying in size, complexity, and number of communities. We perform the investigations on each of the selected real-life and synthetic datasets, as mentioned in the Sections 5.1, and 5.2. To perform the low-rank approximation using SVD, the variance is set to 50 to extract important information and leave out the less useful information. The initial population for GSA is set to 20, as we got the optimal performance in some datasets on this particular value of the initial population. The performance of the GSA depends on the initial population for different dimensions, to maintain uniformity in evaluation for all datasets, we chose the same the initial population. The rate of decay of gravitational constant determines the rate at which the force will decay, a higher value may lead to a local optimum, whereas a smaller value may not provide result in a reasonable number of iterations. To maintain a balance, the value of the decay rate of the gravitational constant is set to 20. We fix the number of iterations for the gravitational search algorithm as 100 as it is a balance value to obtain good results in a reasonable time. We obtain the predicted labels for each node with each of the three embedding methods, namely HOPE, node2vec, and SDNE. We perform the embedding till 128 dimensions of the low-dimensional vector representation. The results are evaluated based on the four performance measures, namely NMI, modularity, precision, and recall, as listed in Section 5.3. Figure 2 depicts the pictorial view of results of community detection on various datasets. From Fig. 2a to f obtained results are compiled using node2vec embedding technique except in Fig. 2d for Word dataset result is drawn using SDNE.

6.1 Normalized mutual information

Normalized Mutual Information (NMI) is a normalization of the mutual information (MI) score to scale the results between 0 and 1, where 0 corresponds to no mutual information, and 1 corresponds to perfect correlation. We observe a general trend is that NMI value increases with the increase in the dimension of embedding. Node2vec gives the best result except in the case of sparse graphs like Word, where SDNE performs better. Also, HOPE performs better than SDNE with datasets like American Football having more number of communities (ref. Fig. 3b), but with a lesser number of communities like Word, SDNE dominates, which is evident from Fig. 3e. Both are approximately the same sized datasets, but American Football has more communities than Word. The gravitational search help in finding better results in lower dimensions, so we see more flat sections in the plot and better values even at lower dimensions. GSA plays a significant role in large networks as it leads to better-separated communities with even with low dimension embedding. Reducing dimensions in embedding saves computational time as it is the most time-consuming subprocess in community detection. The brief description of the NMI results obtained on each dataset is discussed below:-

Zachary Karate Club (Fig. 3a):- Node2vec is performing best, with NMI value 1 for dimension greater than 32. Hope is better than SDNE for lower dimensions but saturates sooner, and for higher dimension value for SDNE and HOPE are almost equal.

American Football (Fig. 3b):- For smaller dimensions, the NMI values are less for all three embedding methods. However, it increases as we increase the dimension and then remains constant for dimension greater than 32. Node2vec gives the best result, followed by HOPE and then SDNE. In this case, HOPE performs better then SDNE.

Political Blogs (Fig. 3c):- It is a relatively large dataset with a lesser number of communities. Here, node2vec gives a throughout better result when compared with other methods. SDNE performs well as compared to HOPE with lower dimensions. For higher dimensions, NMI values with HOPE improves but slightly less than that of SDNE, and NMI values are constant for dimension greater than 16 for all datasets. For larger datasets, the algorithm tends to give better results even at low dimensions as GSA improves the clustering.

Political Books (Fig. 3d):- Here also, node2vec gives a better result even for smaller dimensions of embeddings, but for higher dimensions, SDNE gives NMI values close to that of node2vec. The result remains unchanged for a dimension greater than 32.

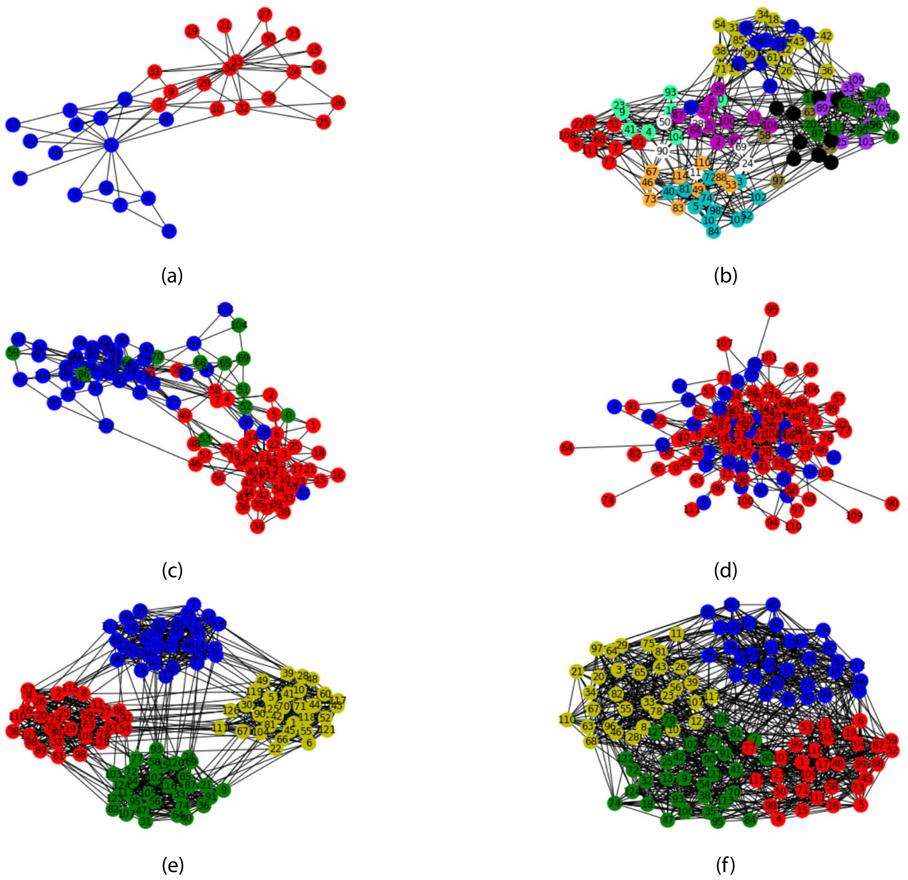


Fig. 2 Community detection results on various networks **a** Karate club with two visible communities, **b** American Football with twelve visible communities, **c** political Books with three communities, **d** Word dataset with two communities, **e** LFR-1, a synthetically generated dataset, and the graph is well separated into four communities, **f** community detection on LFR-2 dataset the division among the four communities is visible

Word (Fig. 3e):- In the case of the Word dataset, SDNE performs better for all dimensions. Node2vec, which produces the best results in other graphs, fails to give significant results for this dataset due to not so well defines communities.

LFR-1 (Fig. 3f):- In a graph with less intermixing among communities, HOPE and node2vec show the same trend. The NMI values of both node2vec and HOPE achieves optimum with an embedding dimension of 16 due to GSA. SDNE gives a poor result with fewer dimensions but improves as we increase dimension.

LFR-2 (Fig. 3g):- The trends captured in LFR-1 are almost the same in LFR-2 for node2vec and sdne, both giving maximum value for embedding dimension greater than 16. Here, the performance of SDNE does not improve well with a higher dimension.

LFR-3 (Fig. 3h):- Though the trends remain the same in LFR-3, maximum obtained NMI values decrease significantly due to more intermixing among communities. Node2vec performs better if the intermixing is more. After the embedding dimension of 32, performance remains the same for all the methods of embedding.

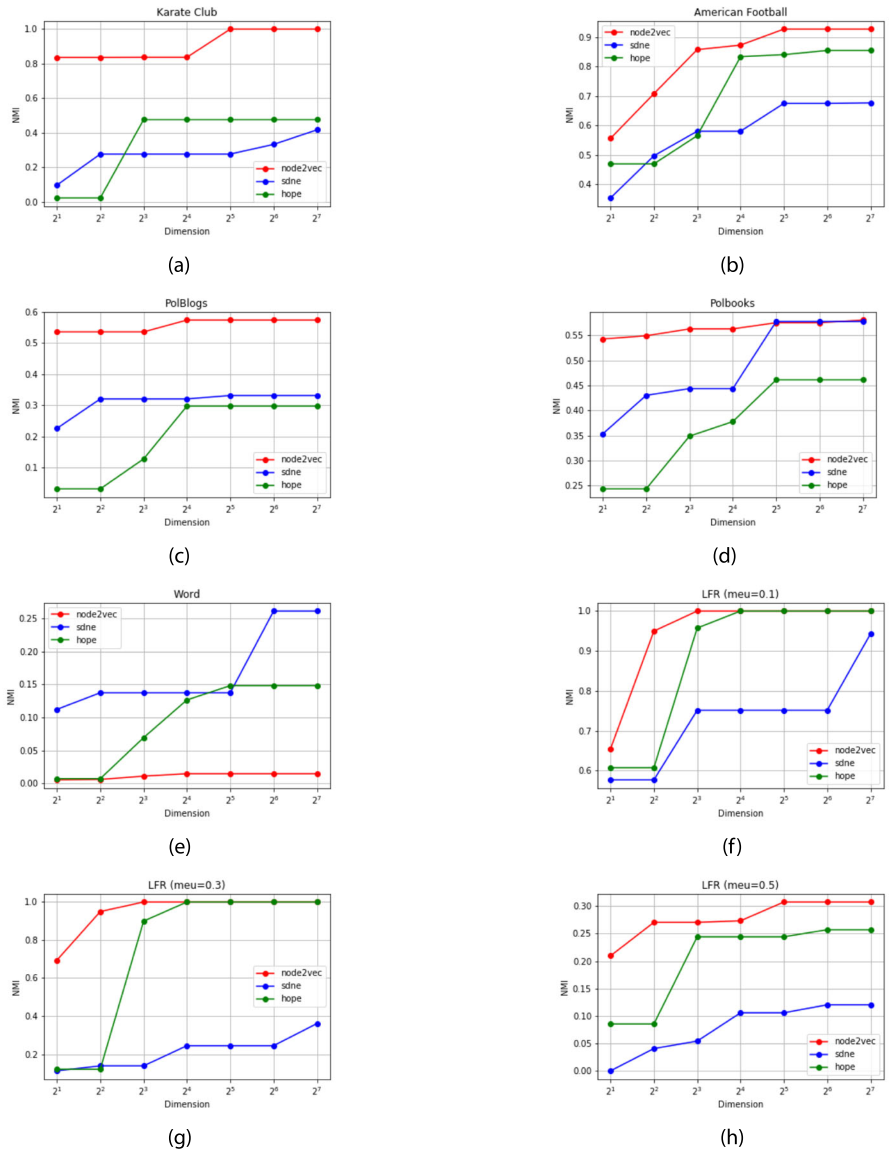


Fig. 3 NMI results on different datasets **a** Karate club, **b** American Football, **c** Polblogs , **d** political Books, **e** Word dataset **f** LFR-1, **g** LFR-2, and **h** LFR-3

We compare the performance of our proposed algorithm namely network embedding using HOPE (NE-HOPE), network embedding using SDNE (NE-SDNE) and network embedding using node2vec (NE-N2V) with some well-known approaches of community detection like Girvan-Newman (Edge-Betweenness) method (Girvan and Newman 2002), fast greedy (Newman 2006a), label propagation (Cordasco and Gargano 2010), and infomap (Rosvall and Bergstrom 2008). Table 1 lists the value of normalized mutual information (NMI) obtained on various datasets by the well-known approaches of community detection

Table 1 NMI results of different methods on various used datasets where FG, LP, GN, NE-HOPE, NE-SDNE, NE-N2V refers to fast greedy, label propagation, Girvan-Newman, network embedding using HOPE, network embedding using SDNE, and network embedding using node2vec methods respectively

Dataset	Infomap	FG	LP	GN	NE-HOPE	NE-SDNE	NE-N2V
Karate	0.8371	0.6924	0.1733	0.8364	0.4764	0.3333	**1**
Football	0.8829	0.6977	0.1193	0.3592	0.8543	0.6745	**0.9268**
Word	0.0332	0.0052	0.0836	0.0030	0.1480	**0.2610**	0.0147
Polbooks	0.5368	0.5308	0.1419	0.4051	0.46128	**0.57764**	0.5752
Polblogs	0.3363	0.37824	0.0019	0.1161	0.2972	0.3311	**0.5732**
LFR-1	**1**	**1**	0.1072	0.5771	**1**	0.7515	**1**
LFR-2	**1**	0.9497	0.0960	0.5771	**1**	0.2440	**1**
LFR-3	0.0064	**0.4233**	0.1096	0.0152	0.2571	0.1203	0.30764

along with the proposed approach with three variants by taking the size of embedding (d) as 64. From the results, it is evident that our network embedding based algorithms perform well.

6.2 Modularity

Modularity is a popular measure to obtain the strength of the division of a network into modules. We observe an almost similar trend in modularity, like that of NMI for all the datasets. Node2vec is the best technique that gives the most modular division of graph among communities. Employing GSA shows better results even at lower dimensions.

Karate Club (Fig. 4a):-In this dataset, node2vec gives a more modular community division than other embedding techniques, which is similar to NMI-dimension results, as in Fig. 3a. The performance of HOPE remains after node2vec, and SDNE performs least. There is also no improvement for dimensions greater than 8 in each method.

American Football (Fig. 4b):- In terms of modularity, node2vec is the best among three embeddings. Here, SDNE gives a better result than HOPE for higher dimensions, which is the opposite of what we observe from the NMI-dimension graph, as in Fig. 3b. Here, the performance of SDNE and HOPE are similar for the embedding dimension 32 onwards.

Political Books (Fig. 4c):- The trend in modularity value is almost the same as compared to NMI, but node2vec tends to give a more modular distribution than SDNE, although the NMI values were almost the same with node2vec and SDNE for dimension greater than 32.

Political Blogs (Fig. 4d):- This relatively large dataset shows a near constant plot with node2vec. SDNE does not perform good for low dimesions but the modularity value is near to node2vec values when dimension is kept higher.

Word (Fig. 4e) Word dataset is a relatively sparse graph. The modularity trend in this dataset changes entirely in comparison to NMI. In NMI results, SDNE performed best. However, here in modularity, node2vec produces better than both SDNE and HOPE. For HOPE, there is no change seen in modularity even we increase the dimension of embedding. Also, SDNE performs inadequately in this case.

LFR-1 (Fig. 4f):- In a graph with less inter-mixing among communities, modularity, and NMI, both metrics tend to show the same trend with node2vec giving better results for lower dimensions. SDNE and HOPE also perform better for larger dimensions.

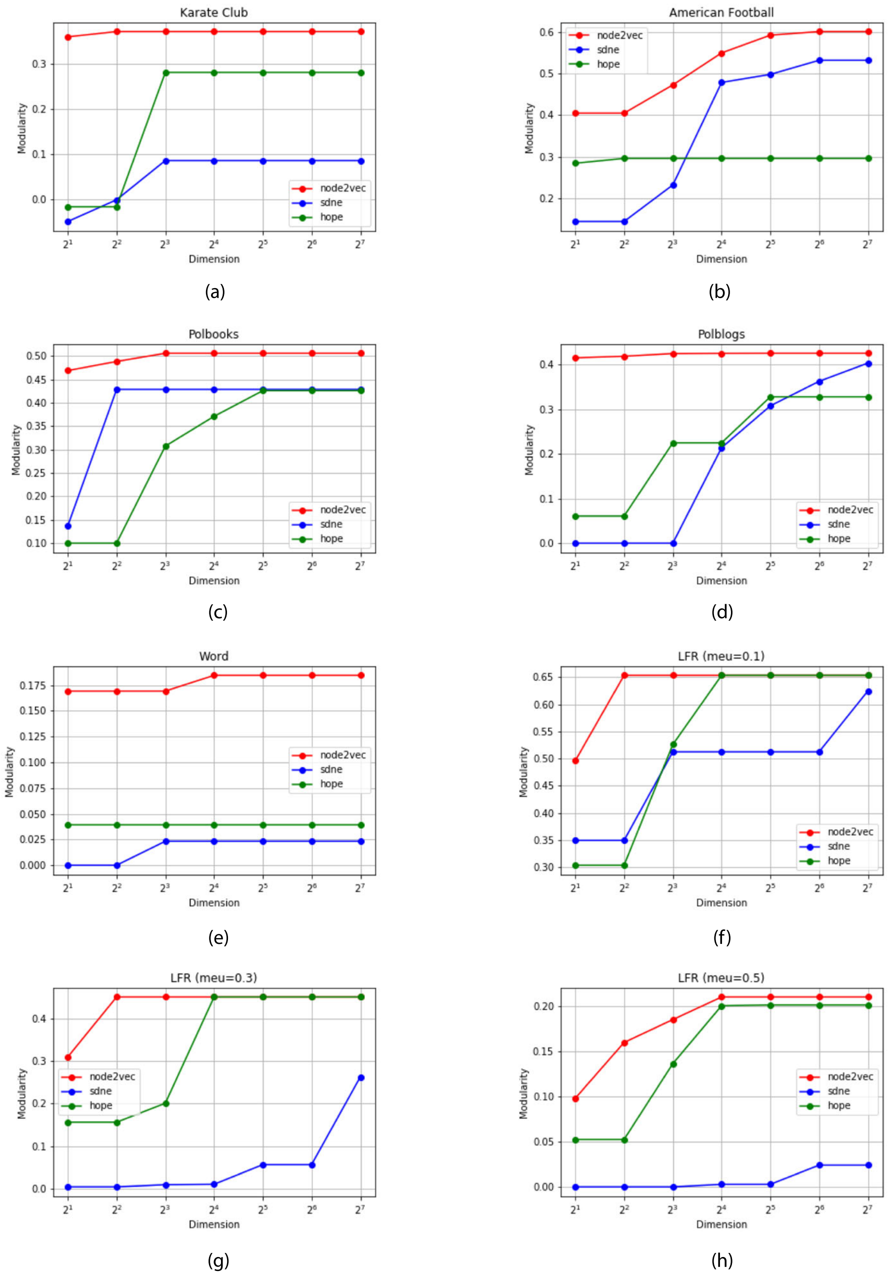


Fig. 4 Modularity results on different datasets **a** Karate club, **b** American Football, **c** political Books, **d** Polblogs **e** Word dataset, **f** LFR-1, **g** LFR-2, and **h** LFR-3

LFR-2 (Fig. 4g):- For a lower dimension of embedding, node2vec gives better results. However, for dimension 32 onwards, the performance of node2vec and HOPE are similar. The performance with SDNE is not good in terms of modularity when intermixing is increased.

Table 2 Modularity results of different methods on various used datasets where FG, LP, GN, NE-HOPE, NE-SDNE, NE-N2V refers to fast greedy, label propagation, Girvan-Newman, network embedding using HOPE, network embedding using SDNE, and network embedding using node2vec methods respectively

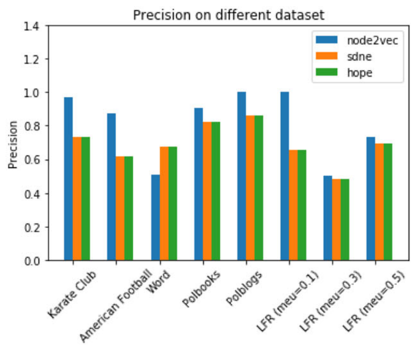
Dataset	Infomap	Greedy	LP	Girwan	NE-HOPE	NE-SDNE	NE-N2V
Karate	**0.4020**	0.3806	0.3547	0.3599	0.2813	0.0864	0.3714
Football	0.6008	0.5497	0.5509	0.4003	0.2957	0.5319	**0.6010**
Word	0.0084	**0.2946**	0.0074	0.0092	0.0395	0.0233	0.1845
Polbooks	0.5267	0.5019	0.4811	0.4428	0.4258	0.4284	**0.5357**
Polblogs	0.4248	**0.4270**	0.4262	0.0012	0.3275	0.3621	0.4249
LFR-1	**0.6533**	**0.6533**	**0.6533**	0.3281	**0.6533**	0.5123	**0.6533**
LFR-2	**0.4521**	0.4361	0.2645	0.2246	**0.4521**	0.0560	**0.4521**
LFR-3	0.0019	0.2088	0.0011	0.0001	0.2016	0.0241	**0.2104**

LFR-3 (Fig. 4h):- As communities in this graph dataset tend to get more inter-mixed, node2vec performs slightly better than HOPE. The results with the SDNE technique deteriorates as intermixing is increased.

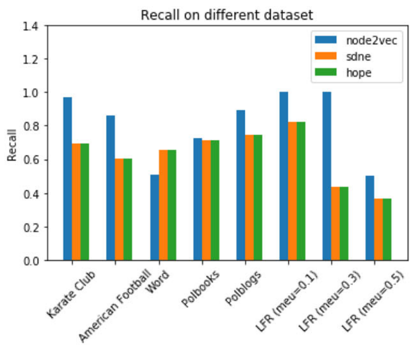
We examine our network embedding based proposed algorithm based on modularity value with some well-known approaches of community detection like the Girvan-Newman (Edge-Betweenness) method (Girvan and Newman 2002), fast greedy (Newman 2006a), label propagation (Cordasco and Gargano 2010), and infomap (Rosvall and Bergstrom 2008). Table 2 presents the modularity value received on different datasets by various community detection methods considered and the proposed approach using network embedding by taking the size of embedding (d) as 64. From the results, it is clear that network embedding using node2vec (NE-N2V) algorithm outperforms others in most of the cases.

6.3 Precision

Precision is the fraction of relevant instances among the retrieved instances. Figure 5a depicts the precision obtained on the various dataset with different embedding techniques. Precision value is best in the node2vec embedding technique for all the datasets except in case of sparse graphs like Word. For Karate club, LFR-1, and LFR-2 datasets, the value of



(a) Precision



(b) Recall

Fig. 5 Precesion and recall value of different methods in each dataset

precision obtained is almost one using the node2vec embedding technique. However, the precision value of SDNE and HOPE is almost the same on all datasets. This implies that although SDNE and HOPE predict the labels with the same accuracy, but they vary in terms of structural division and having different modularity and NMI on different datasets.

6.4 Recall

Figure 5b presents the recall value obtained on various datasets. The recall trends are the same as precision Section 6.3, where node2vec performs better on all the datasets except Word. Here, node2vec tends to perform even better in terms of recall on all the datasets except Word. For Karate club, LFR-1, and LFR-2 datasets, the value of precision achieved is almost one using the node2vec embedding technique. This indicates that node2vec is a more sensitive embedding technique and is good at predicting relevant instances.

7 Conclusion

Community detection is one of the most studied problems in the field of network science. In this paper, we utilized the idea of network embedding and gravitational search algorithm to uncover community structure in complex networks. We obtained the low-dimensional vector representation of nodes by three popular methods of embedding techniques i.e, HOPE from matrix factorization category, node2vec from random walk, and SDNE from deep learning category. Then we apply low-rank approximation followed by k-means clustering. Then gravitational search algorithm (GSA) is used to improve the results of centroids of clusters obtained through k-means leading to better results even at low dimensions. As the output solution of the k-means clustering is one of the member of initial population of GSA, our algorithm does not trap in local optima, which usually happens with the evolutionary-based algorithm and clustering techniques like k-means. We performed comprehensive investigations on five real-life, and three synthetic network datasets using various performance measures, and the results obtained are satisfactory enough to suggest that the proposed algorithm presents a general framework for detecting communities in a variety of networks using network embedding techniques. Comparing benchmarks of various traditional embedding techniques, we achieved the best results with node2vec embedding for all the graphs except networks where communities have a high level of intermixing, SDNE performs better in such a scenario. This is due to how these embedding techniques operate. Node2vec is a random walk based embedding technique and is better at capturing path based relationships between the nodes, so it is proved to be more a modular embedding technique. The node2vec performs better if community boundaries are simpler. SDNE, on the other hand, is good at capturing complex relationships between nodes, which is evident from results on dataset like Word. In precision and recall criteria, SDNE and HOPE perform almost similar but node2vec performs better than others for all the datasets except on the Word dataset. Our study is a noble effort to employ popular network embedding methods for the community detection problem, and try to span the gap between machine learning techniques, evolutionary approaches, and network analysis operation. The extension of our research work can be to find a better fitness function to get better results using GSA. Different statistical distributions can also be used to generate the initial population and study their effect on the results.

References

Adamic, L.A., & Glance, N. (2005). The political blogosphere and the 2004 US Election. In *Proceedings of the WWW-2005 workshop on the weblogging ecosystem*.

Aggarwal, C.C. (2011). An introduction to social network data analytics. In *Social network data analytics* (pp. 1–15). Boston: Springer.

Aggarwal, C., & Subbian, K. (2014). Evolutionary network analysis a survey. *ACM Computing Surveys (CSUR)*, *1;47*(1), 1–36.

Arasteh, M., & Alizadeh, S. (2019). A fast divisive community detection algorithm based on edge degree betweenness centrality. *Applied Intelligence*, *49*(2), 689–702.

Bamakan, S.M., Nurgaliev, I., Qu, Q. (2109). Opinion leader detection: a methodological review. *Expert Systems with Applications*, *115*, 200–22.

Battiston, F., Iacovacci, J., Nicosia, V., Bianconi, G., Latora, V. (2016). Emergence of multiplex communities in collaboration networks. *PLoS ONE*, *11*(1), e0147451.

Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, *2008*(10), P10008.

Chouchani, N., & Abed, M. (2020). Online social network analysis: detection of communities of interest. *Journal of Intelligent Information Systems*, *54*(1), 5–21.

Clauset, A., Newman, M.E., Moore, C. (2004). Finding community structure in very large networks. *Physical Review E*, *70*(6), 06611.

Cordasco, G., & Gargano, L. (2010). Community detection via semi-synchronous label propagation algorithms. In *2010 IEEE international workshop on: business applications of social network analysis (BASNA)* (pp. 1–8): IEEE.

Cui, P., Wang, X., Pei, J., Zhu, W. (2018). A survey on network embedding. *IEEE Transactions on Knowledge and Data Engineering*, *31*(5), 833–52.

Danon, L., Arenas, A., Díaz-Guilera, A. (2008). Impact of community structure on information transfer. *Physical Review E*, *3;77*(3), 036103.

Ding, Z., Zhang, X., Sun, D., Luo, B. (2018). Low-rank subspace learning based network community detection. *Knowledge-Based Systems*, *155*, 71–82.

Ding, X., Zhang, J., Yang, J. (2018). A robust two-stage algorithm for local community detection. *Knowledge-Based Systems*, *152*, 188–199.

Estévez, P.A., Tesmer, M., Perez, C.A., Zurada, J.M. (2009). Normalized mutual information feature selection. *IEEE Transactions on Neural Networks*, *20*(2), 189–201.

Eustace, J., Wang, X., Li, J. (2014). Approximating web communities using subspace decomposition. *Knowledge-Based Systems*, *70*, 118–127.

Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, *486*(3), 75–174.

Fortunato, S., & Hric, D. (2016). Community detection in networks: a user guide. *Physics Reports*, *659*, 1–44.

Freeman, L. (2004). The development of social network analysis. *A Study in the Sociology of Science*, *1*, 687.

Girvan, M., & Newman, M.E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, *11;99*(12), 7821–6.

Goyal, P., & Ferrara, E. (2018). Graph embedding techniques, applications, and performance: a survey. *Knowledge-Based Systems*, *151*, 78–94.

Grover, A., & Leskovec, J. (2016). node2vec: scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 855–864): ACM.

Guerrero, M., Montoya, F.G., Baños, R., Alcayde, A., Gil, C. (2017). Adaptive community detection in complex networks using genetic algorithms. *Neurocomputing*, *266*, 101–113.

Gui, Q., Deng, R., Xue, P., Cheng, X. (2018). A community discovery algorithm based on boundary nodes and label propagation. *Pattern Recognition Letters*, *109*, 103–9.

Guo, K., He, L., Chen, Y., Guo, W., Zheng, J. (2020). A local community detection algorithm based on internal force between nodes. *Applied Intelligence*, *50*(2), 328–40.

Honghao, C., Zuren, F., Zhigang, R. (2013). Community detection using ant colony optimization. In *2013 IEEE congress on evolutionary computation* (pp. 3072–3078): IEEE.

Huang, H., Shen, H., Meng, Z., Chang, H., He, H. (2019). Community-based influence maximization for viral marketing. *Applied Intelligence*, *49*(6), 2137–50.

Jaradat, A.S., & Hamad, S.B. (2018). Community structure detection using firefly algorithm. *International Journal of Applied Metaheuristic Computing (IJAMC)*, *9*(4), 52–70.

Krebs, V. (2020). unpublished, http://www.orgnet.com/.

Kumar, S., & Panda, B.S. (2020). Identifying influential nodes in social networks: neighborhood coreness based voting approach. *Physica A: Statistical Mechanics and its Applications*, 124215.

Lancichinetti, A., Fortunato, S., Radicchi, F. (2008). Benchmark graphs for testing community detection algorithms. *Physical review E*, *78*(4), 046110.

Ma, L., Huang, H., He, Q., Chiew, K., Liu, Z. (2014). Toward seed-insensitive solutions to local community detection. *Journal of Intelligent Information Systems*, *43*(1), 183–203.

Mahmood, A., & Small, M. (2015). Subspace based network community detection using sparse linear coding. *IEEE Transactions on Knowledge and Data Engineering*, *28*(3), 801–812.

Mahmoud, H., Masulli, F., Rovetta, S., Russo, G. (2013). Community detection in protein-protein interaction networks using spectral and graph approaches. In *International meeting on computational intelligence methods for bioinformatics and biostatistics* (pp. 62–75). Cham: Springer.

McDaid, A.F., Greene, D., Hurley, N. (2013). Normalized mutual information to evaluate overlapping community finding algorithms arXiv:1110.2515v2.

Messaoudi, I., & Kamel, N. (2019). A multi-objective bat algorithm for community detection on dynamic social networks. *Applied Intelligence*, *49*(6), 2119–36.

Mitchell, M. (2006). Complex systems: network thinking. *Artificial Intelligence*, *170*(18), 1194–1212.

Newman, M.E. (2001). The structure of scientific collaboration networks. *Proceedings of the National Academy of Science*, *98*(2), 404–409.

Newman, M.E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, *103*(23), 8577–8582.

Newman, M.E. (2006). Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, *74*(3), 036104.

Newman, M.E. (2012). Communities, modules and large-scale structure in networks. *Nature Physics*, *8*(1), 25.

Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W. (2016). Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1105–111.4): ACM.

Palla, G., Derényi, I., Farkas, I., Vicsek, T. (2005). Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, *435*(7043), 814–818.

Pattanayak, H.S., Sangal, A.L., Verma, H.K. (2019). Community detection in social networks based on fire propagation. *Swarm and evolutionary computation*, *44*, 31–48.

Ramezani, M., Khodadadi, A., Rabiee, H.R. (2018). Community detection using diffusion information. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, *23;12*(2), 1–22.

Rashedi, E., Nezamabadi-pour, H., Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Elsevier Information Sciences*, *179*, 2232–2248.

Rosvall, M., & Bergstrom, C.T. (2008). Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, *105*(4), 1118–1123.

Sahebi, S., & Cohen, W.W. (2011). Community-based recommendations: a solution to the cold start problem. In *Workshop on recommender systems and the social web, RSWEB* (p. 60).

Saleh, M., Esa, Y., Mohamed (2019). A applications of complex network analysis in electric power systems. *Energies*, *11*(6), 1381.

Siddiquea, N., & Adelib, H. (2016). Applications of gravitational search algorithm in engineering.

Wang, D., Cui, P., Zhu, W. (2016). Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1225–1234): ACM.

Yadav, R.K., & Rai, A.K. (2020). Incorporating communities' structures in predictions of missing links. Journal of Intelligent Information Systems. https://doi.org/10.1007/s10844-020-00603-y.

Zachary, W.W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*, 452–473.

Zhou, Z., & Amini, A. (2019). Analysis of spectral clustering algorithms for community detection: the general bipartite setting. *Journal of Machine Learning Research*, *20*(47), 1–47.