Noname manuscript No.
(will be inserted by the editor)

Community Detection in Complex Networks using Stacked Autoencoders and Crow Search Algorithm

Sanjay Kumar · Abhishek Mallik · Sandeep Singh Sengar

Received: date / Accepted: date

Abstract The presence of community structures in complex networks reveals meaningful insights about such networks and their constituent entities. Finding groups of related nodes based on mutual interests, common features, objectives, or interactions in a network is known as community detection. In this paper, we propose a novel Stacked Autoencoder-based deep learning approach augmented by the Crow Search Algorithm (CSA) based k-means clustering algorithm to uncover community structure in complex networks. As per our approach, firstly, we generate a modularity matrix for the input graph. The modularity matrix is then passed through a series of stacked autoencoders to reduce the dimensionality of the matrix while preserving the topology of the network and improving the computational time of the proposed algorithm. The obtained matrix is then provided as an input to a modified k-means clustering algorithm augmented with the crow search optimization to detect the communities. We use Crow Search Algorithm based optimization to generate the initial centroids for the k-means algorithm instead of generating them randomly. We perform extensive experimental analysis on several real-world and synthetic datasets and evaluate various performance metrics. We compare

Sanjay Kumar
Department of Computer Science and Engineering, Delhi Technological University, New Delhi-110042, India
E-mail: sanjay.kumar@dtu.ac.in

Abhishek Mallik
Department of Computer Science and Engineering, Delhi Technological University, New Delhi-110042, India
E-mail: abhishekmallik265@gmail.com

Sandeep Singh Sengar
Department of Computer Science, Cardiff Metropolitian University, United Kingdom- CF5 2YB
E-mail: Email: sssengar@cadiffmet.ac.uk

the results obtained by our algorithm with several traditional and contemporary community detection algorithms. The obtained results reveal that our proposed method achieves commendable results.

Keywords Community detection · Complex networks · Crow search algorithm (CSA) · Stacked autoencoders · k-means clustering · Social Networks

1 Introduction

In recent years, the study of complex networks has drawn much attention from researchers of different disciplines. Various real-life networks like online social networks, biological networks, communication networks, collaboration networks are examples of complex networks consisting of a large number of nodes or entities and intricate relationships between the nodes [1]. Complex network analysis or network science offers a plethora of research problems like node classification [2], community detection [3], link prediction [4], influence maximization [5], information diffusion [6], and many more. Community detection is one of the widely studied topics in the fields of network science, which aims at grouping nodes of a network into clusters or modules such that each cluster has a dense internal connection and is sparsely connected with other clusters [7, 8]. The presence of community structure is ubiquitous and quite common in real-life networks [9]. Community detection has been extensively applied to various fields, such as biology, sociology, and computer science. For instance, in online social networks such as Facebook, Twitter, or Instagram, users with similar hobbies or a common cause can form a community. In protein-protein interaction networks, communities correspond to functional groups where proteins having similar functions are anticipated to be involved in the same processes [10]. In social networks, discovering communities can help in targeted advertisements, suggesting similar users, controlling rumors, crowdfunding campaigns, etc. [11]. E-commerce companies can target prospective communities to promote their products to exploit the mutual trust between the users of a community to achieve optimal advertisement of their products.

Traditional algorithms for community detection like hierarchical, partitional, and spectral clustering perform poorly and give results with low accuracy, and are cost-intensive [12]. To overcome this challenge lower-dimensional representation of the network can be used, which would improve the computation cost while maintaining the topological details of the network. Low dimensional representation of a network is the task of representing a network with fewer dimensions while retaining all important features [13]. However, various low-dimensional representation of the networks suffers from a loss of crucial information about the network topology.

In this paper, we propose a novel Community Detection algorithm, named CD-SACS, using Stacked Autoencoders and Crow Search algorithm based k-means clustering technique. Firstly, we generate a modularity matrix for the

network from the adjacency matrix of the network under consideration. The generated modularity matrix has high dimensions which make it infeasible to use in a reasonable computation time. Moreover, this modularity matrix contains several irrelavant and highly correlated features which are redundant and useless to solve the problem. To improve upon the modularity matrix, we pass it through a network of stacked autoencoders which reduces the dimensionality of the original modularity matrix and keeps the most relevant and least correlated features while preserving the network topology. This reduces the loss of original data in the lower dimensional matrix. The lower dimensional matrix is then passed to our novel crow search based k-means clustering algorithm. The clustering algorithms like k-means use a random initialisation of the centroids or initial solutions which often leads the algorithm to converge to a local optimum. To avoid such scenario, we use the Crow Search Algorithm (CSA) to generate the initial centroids for the k-means clustering algorithm. This helps our algorithm to explore a relatively bigger search space without converging to a local optimum in a reasonable time. Our crow search optimisation based k-means clustering algorithm helps us in optimally detecting communities in the network. Finally, we perform the experimental results of the proposed algorithm against several traditional and contemporary community detection algorithms on ten real-life and three synthetic networks of varying size, dimensions, and topologies. The obtained results echo that the proposed algorithm produces commendable results. For example, the NMI value obtained by the proposed method for Karate, Football, Citeseer, and LFR 0.1 is 1, 0.94, 0.879, and 1, respectively. Similarly, the precision value reported by the proposed algorithm for Karate, Football, Citeseer, and LFR 0.1 is 0.868, 0.868, 0.908, 0.94, and 0.878, respectively.

The main contributions of the proposed work can be summarised as follows.

- We propose a novel stacked autoencoder-based deep learning approach augmented by the Crow Search Algorithm (CSA) based k-means clustering algorithm to uncover community structure in complex networks.
- We utilize the modularity matrix for feature generation and then use a network of stacked autoencoders for dimensionality reduction and feature selection.
- We introduced a modified Crow-Search based k-means clustering algorithm to avoid local optimum and explore a larger search space to achieve a better result close to global optimum.
- The performed intensive experimentation and the results obtained establish the efficacy of the proposed algorithm.

The rest of the paper is organized as follows. Section 2 shows related works for community detection. Section 3 describes various preliminary related to our work like autoencoders, modularity matrix, k-means clustering, and crow search algorithm. Section 4 presents the proposed community detection algorithm in detail. The networks datasets on which we apply the proposed algorithm and the evaluation metrics that we use to calculate the performance of the proposed algorithm are mentioned in Section 5. Section 6 presents the

experimental results performed based on numerous performance measures and real-life and synthetic datasets. Finally, in Section 7 we conclude our work.

2 Related Work

Revealing community structure is a significant measure for an insight into complex systems, which go beyond the local structure of the members in the network. Uncovering community structure finds numerous applications in various disciplines like biology, sociology, and computer science. Over the years, numerous community detection algorithms have been proposed [14]. Modularity based methods become quite popular for community detection [15]. Since modularity optimization is an NP-hard problem, various techniques exist to solve for a close approximation. Some of them are greedy techniques, simulated annealing, extremal optimization, spectral optimization and some other techniques. Greedy Technique was the first one invented for modularity maximization by Newmann [16] where vertices that are grouped together are joined to increase modularity. Simulated annealing uses probabilities to explore the space of possible states to maximize the given function . Extremal optimization uses heuristic search procedure to improve computation time while keeping the accuracy comparable to simulated annealing [17]. In Spectral Optimization, we optimize the modularity using the eigenvalues and eigenvectors of the modularity matrix. Some algorithms like Infomap [18], SC(Spectral Clustering) [19], AP (Affinity Propagation) [20], AC (Agglomerative Clusttering) [21] and LP (Label Propagation) [22] have also worked well on detecting communities. Infomap is a flow based method that detects communities on the basis of the map equation. In Spectral Clustering (SC), communities are detected on the basis of edges. It uses a similarity matrix for clustering. Affinity Propagation (AP), creates clusters by sending signal between data points until they converge. LP is a semi supervised machine learning algorithm which starts from a small subset points having labels. These labels are propagated to the unlabeled points to label them. Membership in a community is determined by the majority label the points have in its neighbourhood upto a degree of one. Agglomerative Clusttering (AC) is a bottom up hierarchical clustering method in which each data point initially is considered a cluster. After each iteration, similar clusters merge until k clusters are produced.

Bhih et al. [23] proposed a topological and content based community detection algorithm. In their work, they proposed a new hybrid similarity matrix representing a weighted contribution of attribute information, information shared by the neighbors of a node, and the connectivity information among the nodes. Once this similarity matrix is generated then it can be used with any state-of-the-art community detection algorithms. Jalali et al. [24] used a dynamic local and overlapping community detection based approach to propose a dynamic collaborative filtering based social recommender system. To overcome the issues of scalability, sparsity, and cold start issues of the collaborative filtering approach they used users' temporal rating data, temporal friend-

ship relationships amongst the users and a local community detection based method. Mohammadi et al. [25] proposed an accelerated implementation of the classical Louvian method called adaptive CUDA Louvain method (ACLM) algorithm. Their implementation benefits from the performative excellence of the graphic processing unit (GPU). ACLM implementation minimizes parallelization overhead and accelerates the modularity parameters calculation by using shared memory in GPU along with optimal threads in the GPU block. ACLM allocates thread to the blocks of GPU on the basis of the number of the required streaming multiprocessors and warps on the GPU. Jaradat et al. [26] introduced community detection using the firefly algorithm by exploiting the effect of the attractiveness function, which is unique to the firefly behavior and maximizing the modularity measures. Kumar et al. [27] proposed a community detection method using the idea of network embedding and gravitational search algorithm (GSA) and produced better results. Messaoudi et al. [28] presented a multi-objective Bat Algorithm, which practices the Mean Shift algorithm to produce the initial population to manage randomness in initial population choice, and produced good results for community detection in dynamic social networks. Pattanayak et al. [29] proposed a fire propagation-based community detection algorithm by relating the seed nodes in the community detection as the fire started forming a source. They defined the search space by a two-radius neighborhood graph. Guo et al. [30] suggested a local community detection algorithm based on the internal force between nodes. They obtained the seed nodes using local degree central nodes and the Jaccard coefficient. The node with maximum degree among seeds is pre-extended by the fitness function. Bandyopadhyay and Peter [31] proposed an unsupervised constrained community detection algorithm using a self-expressive graph neural network. They incorporated the principle of self-expressiveness framework with a self-supervised graph neural network for unsupervised community detection.

Zhang et al. [32] presented a Structural Deep Nonnegative Matrix Factorization model, named SDNMF. They proposed a multi layer Negative-Matrix Factorisation model comprising of an encoder and decoder module similar to a deep autoencoder. They exploit the first order and second order similarities to capture the local and global structural details of the network. Xu et al. [33] presented a stacked autoencoder and ensemble framework based community detection model (SAECF). They obtain efficient low-dimensional feature representation of the network using transfer learning and a stacked autoencoder. They use clustering algorithms for obtaining consistent matrix which is later improved using non-negative matrix factorisation. Wang et al. [34] presented a proximity group formation game based model (PFGM). It works on the notion that the higher second-order pairwise proximity is due to higher number of shared communities. They illustrate the evolution of community structures using a two-step non-cooperative game model. A local community detection (LCD) model was presented by Shang et al. [35]. Their proposed algorithm relies on higher-order structure and edge information. They start by using the connectivity of the first node to the local community based on a seed de-

gree. Then they propose a new modularity function to generate an extended and more tightly connected local community. The central part of the local community is more clearly defined using a community central node. The edge information amongst the nodes is used to determine the membership strength of a node to a particular community.

3 Preliminaries

3.1 Community Detection

In this section, wwe describe the problem of community detection. Finding groups of related nodes which are densely connected in a network is known as community detection. The presence of community structures in complex networks like online social networks, biological networks, collaboration networks is ubiquitous, revealing meaningful insights about such networks and their constituent entities. Community detection is one of the prominent research topics in network science and contains numerous real-life applications. Let there be a graph $G = (V, E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of nodes and $E = \{e_{ij}\}_{i,j=1}^{n}$ represents the vertices in the network. The problem of community detection refers to partitioning the entire graph G into a set of k communities $C = \{C_1, C_2, ..., C_k\}$. Each node in the graph belongs to one of the communities which represent the local topological details of the community. Fig. 1 illustrates the community detection problem statement.

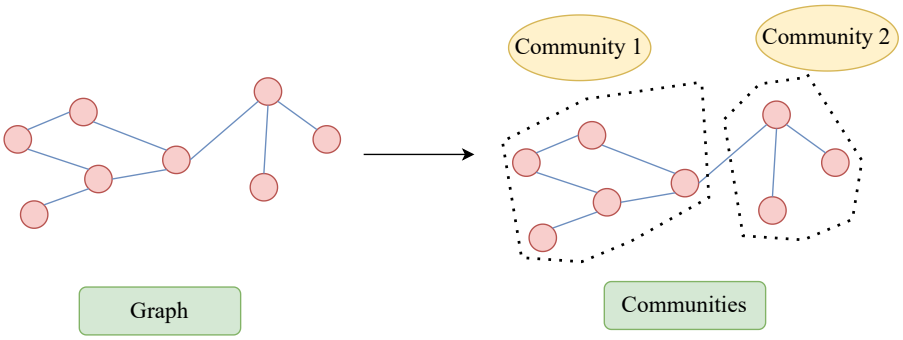

Fig. 1: Illustration of the Community detection problem statement.

3.2 Autoencoder

Our proposed model relies on autoencoders to learn features from the graph. Autoencoders are unsupervised artificial neural networks that learn how to efficiently encode(compress) data and how to decode(reconstruct) original data

from the compressed data[36]. It does so by applying backpropagation while setting the target values to be equal to the input matrix **X**. i.e. it sets **Y**=**X**, where **Y** is the output matrix of the encoder. It tries to learn a matrix **h** on input **X** defined in Eq. 1.

$$h = f(W_e^T x_i + b_e) \tag{1}$$

where **h** is a low dimensional embedding of **X** and **f(.)** is an element-wise nonlinear mapping, such as the **sigmoid** function or **tanh** function and W,b are weights and biases for the neural networks. This process of transforming a high dimensional data into a low dimensional embedding is encoding. Similarly, a decoder transforms a low dimensional embedding to reconstructed data according to the given Eq.2.

$$x_o = f(W_d^T h + b_d) \tag{2}$$

Here x_o is the reconstruction matrix and $f(.)$, W and b are similar to that used in the encoding layer.

3.3 Modularity Matrix

The adjacency matrix A gives us limited information about the relationship between nodes. Hence, we convert it to another form of matrix known as modularity matrix. Formally, we describe the similarity relationship between nodes by a modularity matrix $B = [b_{pq}] \epsilon R^{NXN}$, which can be written as shown below in Eq. 3:

$$b_{ij} = a_{ij} - \frac{k_i k_j}{2m} \tag{3}$$

where $\frac{k_i k_j}{2m}$ is the expected number of edges between nodes i and j if the edges are placed randomly, k_i is the degree of vertex i, and $m = \frac{1}{2}\sum_i k_i$ is the total number of edges in the network[37]. The modularity maximization model is taken as a reference to use the matrix B to maximize to improve the quality of partition in the network.

3.4 k-means clustering

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centres or cluster centroid), serving as a prototype of the cluster [38]. The way the k-means algorithm works is as follows.

1. Specify the number of clusters k.
2. Initialize centroids by first shuffling the network and then randomly selecting k data points for the centroids without replacement.

3. Keep iterating until there is no change to the centroids. i.e assignment of data points to clusters isn't changing.

The objective function is described below in Eq. 4:

$$J = \sum_{i=1}^{m} \sum_{k=1}^{K} w_{ik} ||x_i - \mu_k||^2 \tag{4}$$

In the proposed work, k-means algorithm is optimized using crow search method explained in Section 3.5.

3.5 Crow Search Algorithm

The accuracy of the k-means clustering algorithm is dependent on the random initialization of centroids. The final result of k-means depends on the initial position of centroids and may confine in local optima. To counter this, we apply a recent swarm intelligence technique known as the crow search algorithm (CSA) which is a metaheuristic optimizer based on the behavior of crows [39]. CSA is a population based technique that works on the idea of crows hiding excess food and retrieving it when required. It is considered that there are n crows, where each crow denotes an initialization position in a d-dimensional search space. Each crow stores the memory of its best hiding place as $m^{i,k}$ and moves in the environment, aiming to find a better hiding position search space. Assume that a crow j visits its hiding position for an iteration k. Two cases can occur for a crow i while following crow j:

- Case 1 − Crow j is unaware of crow i while visiting its hiding location, so crow i approaches the hiding position of crow j.
- Case 2 − Crow j is aware of crow i following it and it decides to visit another random location.

The new position of crow i is given by Eq. 5.

$$x^{i,k+1} = \begin{cases} x^{i,k} + r_i \times fl^{i,k} \times (m^{j,k} - x^{i,k}) & \text{if } r_j \geq AP^{i,k} \\ a\ random\ position & \text{otherwise} \end{cases} \tag{5}$$

where,
$x^{i,k}$ is the position of crow i at k-th iteration which is given by the d-dimensional vector $[x_1^{i,k},\ x_2^{i,k},\ x_3^{i,k},\ .....\ x_d^{i,k}]$
r_i = a random number with value between 0 and 1
$fl^{i,k}$ = flight length of crow
$m^{j,k}$ = hiding spot of crow
AP = awareness probability, denotes awareness of crow w.r.t its surroundings

In this algorithm, AP is used to control the search space. If AP is decreased, CSA will search in local region whereas on increasing the AP, CSA will be less likely to search in local regions and much more likely to explore and search

space on global scale. The details of the algorithms are presented in Algorithm 1.

Algorithm 1 Crow Search Algorithm

Input: S : Flock Size,
d : number of dimensions,
F : Set of all possible solutions and values,
$itermax$: Max number of iterations,
fl : flight length,
AP : Awareness probability
Output: Optimal positions

$crowPositions \leftarrow InitializeRandomPosition(d, F, S)$
$Memory \leftarrow InitializeMemory(d, F, S)$
$Fitness \leftarrow CalculateFitness(CrowPositions)$
$score \leftarrow NMIscore(ActualLabels, PredictedClusterLabels)$
while $k < itermax$
  for $i \leftarrow$ 1 *to* S
    $r_i \leftarrow GenerateRandom()$
    $RandomCrow \leftarrow ChooseRandomCrow(S)$
    $NewPosn \leftarrow UpdatePosn(CrowPositions, i, Memory, RandomCrow, AP, r_i)$
  endfor
  if $NewPosn \in F$
    $CrowPostions \leftarrow NewPosn$
  endif
  $NewFitness \leftarrow CalculateFitness(CrowPostions)$
  if $NewFitness > Fitness$
    $Memory \leftarrow CrowPostions$
  endif
  $k \leftarrow k + 1$
endwhile
return crow position with best fitness out of all CrowPostions

4 Methodology

In this section, we present our proposed stacked autoencoder-based deep learning approach augmented by the Crow Search Algorithm (CSA) based k-means clustering algorithm for community detection, and in short, we name our algorithm as CD-SACS. The proposed algorithm can be used to uncover community structure based on the interactions and relationships between the nodes in the network. We start by first generating a modularity matrix from the adjacency matrix, which is described in Section 4.1. Then we reduce the dimensionality of the modularity matrix to extract the most relevant and the least correlated features using a network of stacked autoencoders, which is presented in Section 4.2. The newly generated feature matrix is then fed to our Crow-Search optimization based k-means clustering algorithm to detect the community as described in Section 4.3. The detailed description of the proposed algorithm, CD-SACS, is explained in Section 4.4.

4.1 Modularity matrix generation

To perform any network analysis task like community detection, we first need to represent it mathematically. One possible way to do so is the Adjacency matrix, which is a basic representation of the existence or non-existence of edges amongst the nodes of the network. The Adjacency matrix represents only the relationships amongst individual nodes and not between the communities to which they belong. To better understand the relationships amongst the nodes of the network as part of sub-modules within the network, we use the Modularity matrix as discussed in Section 3.3. It helps as a measure to ascertain the strength of the partition of a network into modules or communities. It also helps to understand the density of the connection within the modules and amongst the nodes in various modules. This is the reason we generate and use the Modularity matrix.

4.2 Dimensionality reduction using Stacked Autoencoders

The Modularity matrix obtained in the previous step has very high dimensionality and may contain several irrelevant and highly correlated information, which is redundant and increases the computational cost of the algorithm. We perform dimensionality reduction on the modularity matrix to extract the essential features using a network of stacked autoencoders, which helps to reduce dimensionality while preserving the topological network information. This is done by reducing the modularity matrix to capture the network topology, further encoded by a series of stacked autoencoders to represent the modularity matrix in a lower-dimensional feature space. Followed by passing the modularity matrix through a feed-forward neural network. Finally, it generates the feature vectors for the nodes in the network such that the nodes having similar topology are closely placed in the lower dimensional feature space. We use a network of multiple autoencoders (single layer autoencoders) stacked together in series with the encoded output of previous autoencoders as an input to successive autoencoders. Adam optimizer is used for updating autoencoder weights. The encoding and decoding functions are as mentioned in Eq. 1 and 2. In our proposed model, we have taken $f(.)$ as hyperbolic tangent or $tanh(.)$, which is given by Eq. 6.

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6}$$

Autoencoder aim to learn a low dimensional non-linear representation that can be reconstructed into modularity matrix B with minimized loss under parameters $\{W_e, b_e, W_d, b_d\}$. We utilize the Mean Squared Loss as our reconstruction error function given by Eq. 7:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2 \tag{7}$$

where f_i is the value returned by the model and y_i is the actual value. After training the autoencoders, the obtained parameters are used in the encoding equation given above to generate the new representation for the graph G with reduced dimensions. Same is denoted by line 2-5 of Algorithm 2.

4.3 Crow Search optimisation based k-means clustering algorithm

The final feature matrix obtained in the last step is then passed to a clustering algorithm to cluster the nodes into separate clusters or to detect communities in the network. For detecting communities, we propose a Crow-Search optimisation based k-means clustering algorithm. Crow Search Algorithm is a relatively new metaheuristic algorithm that simulates the intelligent behavior of crows and finds an optimal solution to various optimization problems. It requires less number of parameters, and it is easier to implement the crow search algorithm [40]. The fewer parameters and easy implementation make it suitable for tasks like community detection. Also, to the best of our knowledge, we are the first to apply Crow Search Algorithm in combination with Stacked Autoencoders to uncover community structure in complex networks. The classical k-means clustering algorithm heavily depends upon its random initialization of centroids, and hence it has a tendency to converge at a local optimum. To improve upon this drawback of the k-means clustering algorithm, we use the Crow-Search based swarm intelligence algorithm. The Crow-Search optimization generates a solution set of multiple centroid initializations, and then it works on arriving at the best solution based on the fitness function. This helps our algorithm to explore a much larger search space and obtain results much closer to the global optimum.

The results obtained for the Crow-Search optimization depend on the flock size S, dimension of the search space d, flight length fl, iterations $iter_{max}$ and awareness probability AP[39]. The flock size is decided as per the problem. For a smaller flock size, the algorithm would converge faster to a local optimum, while for a larger flock size, the task of obtaining the optimal solution becomes computationally expensive. The dimension d of search space for the crows is the number of clusters to be detected in the network under consideration. The values of AP and fl are chosen so that the crows search for a globally optimum solution rather than limiting in a local search space. For the proposed Crow-Search optimized k-means algorithm, the For optimizing k-means, the initial population details, fitness function, and termination conditions are described as follows.

- Initial Population: The Crow-Search optimization starts with initializing the position and the search space of the crows. An initial population or a flock of crows of size S is randomly generated having a d dimensional search space. Here, each crow represents a set of possible centroids, and d is the number of centroids that is given by the actual number of clusters to be detected in the network under consideration.

– Fitness function: The goal of Crow-Search optimization is to maximize the value of the fitness function. For our study, we choose the NMI score, which is defined as the percentage of nodes correctly classified for each cluster, as the fitness function. It is given by Eq. 8. Every possible solution tries to maximize the value of the NMI score, and the solution with the highest NMI score becomes the fittest solution.
– Termination condition: The Crow-Search optimization is run until a termination condition is met. For our study, we have defined the maximum number of iterations as $iter_{max}$ as the termination condition. This implies that our algorithm will run $iter_{max}$ number of times after which the algorithm would be terminated.

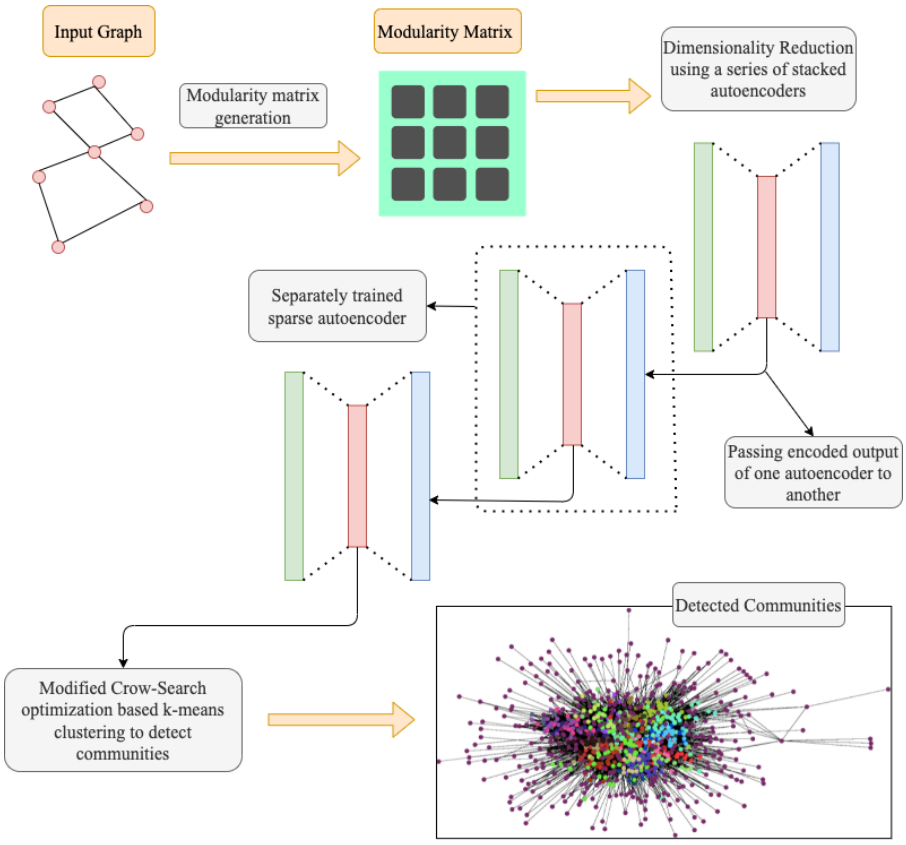

Fig. 2: The algorithmic flow chart of our proposed CD-SACS method.

4.4 Community Detection Using Stacked Autoencoder & Crow-Search optimzation based k-means clustering (CD-SACS)

This section presents a consolidated view of our proposed community detection algorithm using Stacked Autoencoder and Crow-Search optimization based k-means clustering. Firstly, we represent the network as an adjacency matrix which is then converted into a modularity matrix as shown in Section 4.1. The generated Modularity matrix is then passed through a series of stacked autoencoders to obtain a lower-dimensional representation of the modularity matrix while preserving the network topology and relationships. This is described in Section 4.2. The stacked-autoencoders extract the most relevant and the least correlated features from the modularity matrix while reducing its dimensionality to make it more computationally efficient. These extracted features are then passed through a Crow-Search based k-means clustering algorithm to detect the communities in the network as shown in Section 4.3. This allocates the nodes in the networks to their predicted communities. Various performance metrics are evaluated on the community results obtained by our algorithm. Fig. 2 shows the entire algorithmic flow of our proposed CD-SACS algorithm. The algorithmic steps for our proposed algorithm are shown in Algorithm 2.

Algorithm 2 Proposed Community Detection Using Stacked Autoencoder & Crow-Search optimzation based k-means clustering (CD-SACS) algorithm

Input: G :Graph, A : Adjacency Matrix , α : Learning rate,
P, Q, R :Number of hidden layers in each autoencoder m_1, m_2, m_3
$actualLabels$:Actual Clustering Labels of nodes
$N_{clusters}$:Total number of clusters
Output: NMI, $Modularity$ score, $Precision$, $Recall$ and clustering result ($predictedLabels$)
1: $ModularityMatrix \leftarrow$ create_modularity_matrix(AdjacencyMatrix)
2: $m_1, m_2, m_3 \leftarrow$ create_autoencoder(P,Q,R)
3: $encoded\_output1 \leftarrow$ train_autoencoder(m_1,ModularityMatrix,α)
4: $encoded\_output2 \leftarrow$ train_autoencoder(m_2,encoded_output1,α)
5: $encoded\_output3 \leftarrow$ train_autoencoder(m_3,encoded_output2,α)
6: $centroidPositions \leftarrow crowSearchAlgorithm(N, N_{clusters}, S, fl, AP, itermax)$
7: $predictedLabels \leftarrow$ Kmeans(encoded_output3,centroidPositions)
8: $NMI \leftarrow$ NMIscore($actualLabels$,$predictedLabels$)
9: $Modularity \leftarrow$ ModularityScore($actualLabels$,G)
10: $Precision, Recall \leftarrow$ PrecisionAndRecall($actualLabels$,$predictedLabels$)
11: **return** $predictedLabels$, NMI, $Modularity$, $Precision$, and $Recall$

The various steps of our proposed algorithm are explained as follows.

– Step 1 : This step covers line number 1 of the proposed Algorithm. We convert the input adjacency matrix to a modularity matrix according to Eq. 9. The modularity matrix represents the similarity relationship between nodes of the network. The modularity matrix has the same dimensions as

the adjacency matrix, i.e., $N*N$ where N is the number of nodes in the network.

- Step 2 : This step covers line number 2, in which we create the stacked autoencoders m_1, m_2, m_3 with learning rate α and P, Q and R as their hidden layers as explained in Section 4.2. The layer settings of autoencoders are written as $N - s1 - s2 - s3$ where $m1$, $m2$ and $m3$ have their layers as $N - s1 - N$, $s1 - s2 - s1$, and $s2 - s3 - s2$ respectively. Here, N is the number of nodes in the network, $s1$, $s2$ and $s3$ are powers of two such that $s1$ is the greatest power of two less than N, while $s2 = \frac{s1}{2}$ and $s3 = \frac{s1}{4}$. Layer settings for real world and synthetic networks are shown in Section 5.1 and 5.2 respectively.
- Step 3 : This step covers line number 3,4,5, where we train stacked autoencoders $m1$, $m2$ and $m3$ with modularity matrix as an input to the first autoencoder $m1$, while for the subsequent autoencoders $m2$ and $m3$, encoded output of previous autoencoders $m1$ and $m2$ acts as their input respectively. These steps are performed so as to minimize the reconstruction loss compared to the original network. Stacked autoencoders are used to reduce dimensionality along with preserving the relationships. We observe that if the autoencoder is trained for too many epochs, the output matrix does not yield optimal results for community detection. Hence, we limit the number of epochs for each network to obtain the optimum result. Hence, based on the analysis made, for small networks, only two stacked autoencoders are used, while for relatively larger networks, we use three stacked autoencoders.
- Step 4 : In line number 6, we apply the crow search algorithm to obtain optimal centroids for applying k-means algorithm. We give the number of clusters as dimensions, S, which denotes flock size as 30. We also set the decision variables and constraints like flight length fl, Awareness probability AP, and the maximum number of iterations $iter_{max}$ for the Crow-Search optimization.
- Step 5 : In line number 7, the encoded output of $m3$ and centroids are fed to a modified k-means algorithm for obtaining clustering result $predictedLabels$.
- Step 6 : The $predictedLabels$ are then used to calculate the NMI score, Modularity score, Precision, and Recall, as shown in lines 8-10.
- Step 7 : Finally in line number 11, we return NMI score, Modularity score, Precision, Recall and the predicted labels.

4.5 Time Complexity Analysis

In this section, we discuss the time complexity analysis for our proposed CD-SACS algorithms. We start by generating a modularity matrix which can be obtained in $O(n^2)$ from the adjacency matrix. Here, n is the number of nodes in the network. Then we perform feature reduction using a stacked autoencoder which is a network of successive layers of neurons. Let l be the neurons in the layers of the stacked autoencoder. Training a neuron layer involves two

phases, namely, forward propagation and back propagation having time complexities of $O(l^4)$ and $O(l^5)$ (https://lunalux.io/series/introduction-to-neural-networks/computational-complexity-of-neural-networks). Hence, the total time complexity of training the stacked autoencoder will be $O(E \times C \times (l^4 + l^5)) \approx O(E \times C \times l^5)$. Here, E is the number of epochs and C is the number of layers in the stacked autoencoder. Then we go onto generating the cluster centres using the modified crow search algorithm. Let the flock size be S, the dimensionality of the search space be d and the total number of iterations be I. Since the fitness function is evaluated and the representations of the crows are updated for the entire flock on every iteration, hence, the time complexity of the modified crow search algorithm is $O(I \times S \times d)$. Finally, we use the K-means to classify the nodes into their respective communities with a complexity of $O(n^2)$. Hence, the final time complexity for our proposed CD-SACS algorithm is $O(n^2 + E \times C \times l^5 + I \times S \times d + n^2)$. Here, $E, C, andl$ are constant, so the time complexity becomes $O(n^2 + I \times S \times d)$=$O(n^2)$ where S, I, and d can be considered as constant.

5 Datasets and Evaluation metrics

We perform the experimental simulations on various real-life datasets of varying size, number of edges and number of communities. We also apply the proposed algorithm on LFR synthetic network for three different values of mixing parameters. The experimental results were evaluated on several benchmarked performance metrics. The description of the datasets and performance metrics used by us is given as follows.

5.1 Real world networks

The description of the various real-life datasets is given below.

1. Zachary's Karate Club [41]: It is a network of 34 members of a karate club in US university in the 1970s. It has two ground-truth communities.
2. Dolphin Social network [42]: It is an undirected network of frequent association among 62 Dolphins in New Zealand. This network contains four ground-truth communities.
3. Polbooks network [43]: It is based on a book about US politics sold on Amazon. Edges between books represent frequent co-purchasing of books by the same buyer. This network has 105 nodes and three communities.
4. Word [44]: Word is a network containing the nouns and adjectives used in Charles Dicken's novel David Copperfield. The dataset has 112 nodes and 425 edges. There are two ground-truth communities present in this network.
5. American college football[45]: It is a network of American college football having 115 nodes and is divided into twelve communities.

6. Email network [46]: This network is obtained from a large European institution that has 1005 nodes and 25571 edges. Each edge (u, v) represents that person u has sent an email to person v, and there are 42 ground-truth communities present in the network.
7. Polblogs network [47]: It is a network of various blogs surrounding US politics. It was published by Adamic and Glance in 2005. It has 1490 nodes, 16718 edges, and has two ground-truth communities.
8. Cora network [48]: It is a scientific publication network consisting of 2708 scientific publications and 5429 citation links. There are 7 communities in the network.
9. Citeseer network [49]: It is also a scientific publications network which consists of 3312 scientific publications and 4732 citation links with six ground-truth communities in the network.
10. Facebook network [50]: It is a Facebook social network consisting of 4039 nodes and 81800 edges. There are twelve communities present in this network.

Tab. 1 gives the stastical details about various real-world datasets used by us including number of nodes, number of edges, and their brief descriptions.

Table 1: Statistical details about the real-world network

Data set	**# of Nodes**	**# of Edges**	**Description**
Karate[41]	34	78	Zachary's karate club
Dolphin[42]	62	159	Dolphin Social Network
Polbooks[43]	105	441	Books about US politics
Word[44]	112	425	Adjectives and nouns in novel
Football[45]	115	613	American college football—2000
Email[46]	1005	25571	Email data from a large European research institution
Polblogs[47]	1490	16718	Blogs about US politics
Cora[48]	2702	5429	Scientific publication network
Citeseer[49]	3312	4732	Scientific publication network
Facebook[50]	4039	81800	Facebook social network

5.2 Synthetic networks

Lancichinetti–Fortunato–Radicchi (LFR) algorithm [51] is commonly used for generating benchmark networks. They have a priori known communities and are used to compare different community detection methods. It requires some parameters to generate the required type of graph. $LFR(N, k, m, \gamma, \phi, \mu)$ requires N which denotes the number of nodes, k denotes the average degree of nodes, m denotes the maximum degree of nodes, γ is the exponent for the degree distribution, ϕ is the component for the community size distribution, μ is the mixing parameter which is used to control the ratio between the degree of intra-communities of a node and its total degree.

By setting different parameters, we can generate various graphs with ground truth values on which the proposed algorithm to detect communities can be tested. We generate three different graphs keeping the number of nodes and the number of edges constant in each case but varying k, m, γ, ϕ, μ using these parameters, we generate three synthetic networks, namely, LFR 0.1, LFR 0.3 and LFR 0.5 networks. In all these networks, N=128, the total number of edges =1024 and μ in the range of 0.1 to 0.3. The total number of communities existing in all three networks is four. The details of the synthetic networks used by us are given below.

1. LFR 0.1: This network has $\mu = 0.1$ due to which it's clusters are dense and are easily separable with less intermixing between each other. This serves as a simple benchmark test for community detection.
2. LFR 0.3: This network has $\mu = 0.3$ due to which its clusters are not so dense and are not as easily separable as in LFR 0.1. Communities have a small amount of intermixing between them which serves as a better testing benchmark for testing of community detection algorithms.
3. LFR 0.5: This network has $\mu = 0.5$ due to which its clusters are much less dense than in LFR 0.3 and are not separable by the naked eye. Communities have a significant amount of intermixing between them due to which community detection is very difficult for this dataset.

Tab. 2 gives the stastical details about various real-world datasets used by us including number of nodes, number of edges, number ofground-truth communities, and their descriptions.

Table 2: Statistical details about the used synthetic networks

Data set	**# of Nodes**	**# of Edges**	**Description**
LFR 0.1	128	1024	LFR network with $\mu = 0.1$
LFR 0.3	128	1024	LFR network with $\mu = 0.3$
LFR 0.5	128	1024	LFR network with $\mu = 0.5$

5.3 Evaluation Metrics

We have evaluated all the algorithms on various evaluation metrics to benchmark our proposed algorithm and compare it with other traditional and contemporary community detection algorithms. The descriptions of these evaluation metrics are as follows.

5.3.1 Normalized Mutual Information (NMI)

The normalized mutual information (NMI) was introduced by Estevez et al.)[52]. The NMI of two random variables represents the co-dependency amongst the two variables. Formally, it is a measure of the amount of information that can be revealed by observing the other variable. The fundamental notion of NMI is linked to that of entropy of a random variable, i.e., a basic concept in information theory that is a measure of the expected "amount of information" held in a random variable. It is an evaluation measure that helps us measure how clusters correlate with each other. It is defined as given in Eq. 8.

$$NMI(Y, C) = \frac{2 \times I(Y; C)}{H(Y) + H(C)} \tag{8}$$

where Y, C, $H(.)$, and $I(Y;C)$ denotes class labels, cluster labels, entropy, and Mutual Information between Y and C, respectively.

5.3.2 Modularity score

Modularity score (Q) represents the difference between the actual number of connections and the expected number of connections in random conditions. It is used to assess the quality of the network cluster structure. This score is computed using Eq. 9

$$Q = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) \tag{9}$$

where m denotes number of edges in the network, A represents adjacency matrix, k_i denotes degree of node i, δ denotes Kronecker delta and value of $\delta(c_i, c_j)$ is 1 if node i and j are in the same community otherwise 0.

5.3.3 Precision and Recall

Precision is used as a measure to estimate the positive predictive value of an algorithm. Formally, it can be described as the number of positive instances with respect to the total number of instances. On the other hand, recall is an estimate of the sensitivity of an algorithm. It measures the total number of relevant cases with respect to the instances that were positive. In simpler terms, precision shows the agreement of the algorithm between labels and predicted labels, while recall gives an idea of the algorithm's effectiveness in

predicting labels. Mathematically, precision and recall can be expressed as below.

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

Here, TP, FP, and, FN represents the True Positives, False Positives, and False Negatives, respectively for the communities detected.

6 Result Analysis

In this section, we present the experimental analysis of our proposed approach, CD-SACS. We have conducted experiments on various real-life as well synthetic datasets as described in Sections 5.1 and 5.2. The results have been evaluated on popularly used evaluation metrics as described in Section 5.3. We compared the performance of our purposed algorithm with some popular traditional community detection algorithms like KM (K-means), IF (Infomap) [18], SC (Spectral Clustering) [19], AP(Affinity Propagation) [20], AC (Agglomerative Clustering) [21] and LP(Label Propagation) [22]. The performance of our proposed algorithm is also compared with several contemporary community detection algorithms like: SAECF [33], PFGM [34], and LCD [35]. Fig. 3 pictorially shows the community detection results obtained by our algorithm on various real-world and synthetic datasets with varying dimensions and community structures. We utilized python programming with several libraries like Sklearn, NumPy, Pandas, Networkx, etc., and some publicly available GitHub repositories to perform simulations and obtain results. We performed simulations on a personal computer with an intel i7 11th generation processor, 16 GB RAM, and RTX 3070 graphics card. The various parts of the analysis of the results are organized in the subsections like parameter settings, comparison of results based on normalized mutual information (NMI) score, modularity score, precision, and recall score.

6.1 Parameter Settings

To obtain optimal results, we use proper parameter settings in our algorithm. We set different layer configurations for the stacked autoencoder based on the network. The network has a minimum of 2 and at max three stacked autoencoders. The dimensions of autoencoders are decided in such a way that the encoded result is of the dimension $NX2^t$, where 2^t is less than N(Size of input data). The layer settings are shown in Tab. 3. Note that layer setting for a network is written as $N-512-256-128$, which means that N is the number of nodes in the network followed by three autoencoders of configuration $N - 512 - N$, $N - 256 - N$, and $N - 128 - N$. All autoencoders were trained

separately, and the encoded result of each autoencoder was used as the input for successive ones. The learning rate for each autoencoder was 0.001 trained up to 1000 epochs.

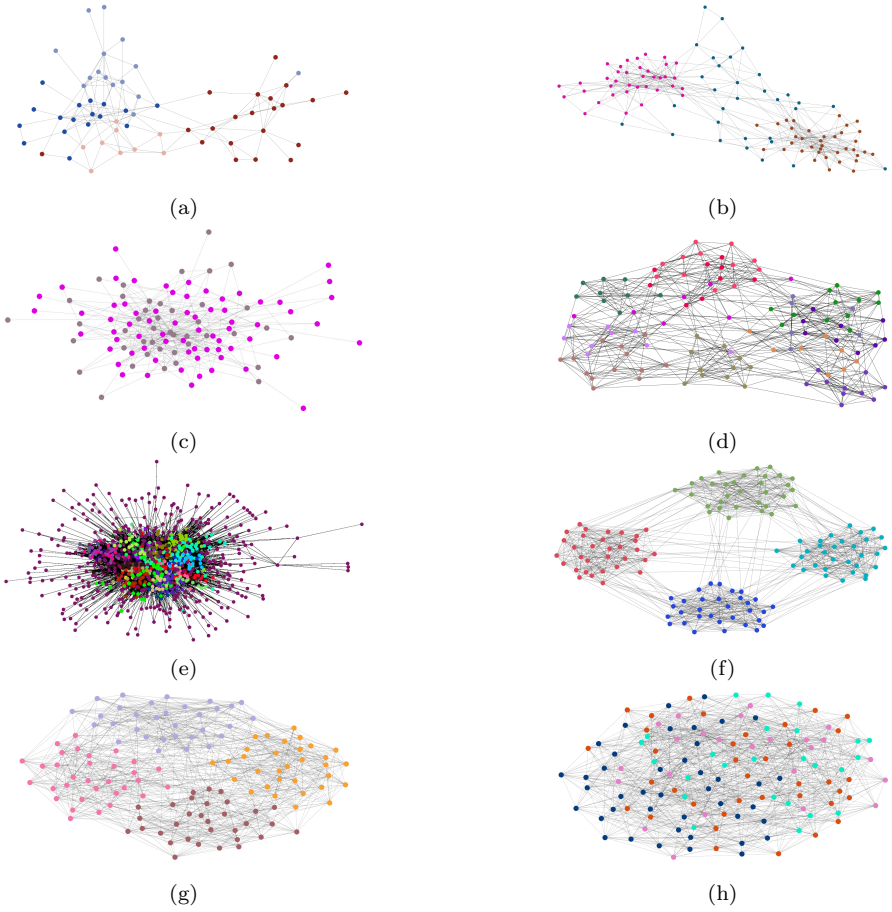

Fig. 3: Community detection results on various real-world and synthetic networks (a) Dolphin Social network with four visible communities, (b) Polbooks network with three visible communities, (c) Word network with two communities, (d) American college football with twelve communities, (e) Email network with 42 communities (f) LFR 0.1 synthetic network with four visible communities, (g) LFR 0.3 synthetic network with four visible communities, and (h)LFR 0.5 synthetic network with four visible communities.

Table 3: Layer Settings for the stacked autoencoders

Networks	**# of Nodes**	**Layer Settings**
Karate	34	N-32-16
Dolphin	62	N-32-16
Polbooks	105	N-64-32
Word	112	N-64-32
Football	115	N-64-32
Email	1005	N-512-256-128
Polblogs	1490	N-1024-512-256
Cora	2702	N-2048-1024-512
Citeseer	3312	N-2048-1024-512
Facebook	4039	N-2048-1024-512
LFR 0.1	128	N-64-32-16
LFR 0.3	128	N-64-32-16
LFR 0.5	128	N-64-32

The final output of the stacked autoencoders was then fed to the Crow Search optimization based k-means clustering algorithm. For the Crow Search optimization, we set the maximum number of iterations $iter_{max}$ as 100 as it is a balanced value to obtain optimal results in a reasonable amount of time. The initial population plays an important role in the performance of any swarm intelligence algorithm. We select the initial flock size, S, to be 30, as we achieved optimal results on several networks based on this value. The flight length is chosen to be 0.9, i.e., closer to 1 so that the particles have larger flight lengths and travel larger steps to exploit the feature space globally. We also tune the hyperparameters manually depending on the experimental results so that we adopt the best hyperparameters.

6.2 Normalized Mutual Information (NMI) Score

This section presents the results obtained for the Normalized Mutual Information (NMI) Score for our algorithms as evaluated on all the networks mentioned in sections 5.1 and 5.2. The NMI score results were also calculated for various other community detection algorithms and the results were compared with our proposed algorithm. Tab. 4 shows the results obtained for various algorithms as evaluated on various networks for the NMI score. Brief description of the obtained NMI scores is given below: Karate[41]: This network has 2 communities having highly sparse edges between them. Spectral Clustering and CD-SACS have the best NMI scores of 1. Infomap, AP and LP also perform well on this network but have a low score due to less number of nodes(32) in the network. Dolphin[42]: CD-SACS performs the best with a score 0.86 closely followed by Infomap with a score of 0.83. SC, AP, LP and AC have relatively low scores. Polbooks[43]: This network have 3 communities with relatively dense connection between them which makes prediction of clusters a tough tasks and leads to relatively lower NMI scores for all of the algorithms. For Polbooks network also our proposed CD-SACS algorithm performs the

best with a NMI score of 0.556. This is closely followed by Spectral Clustering, Infomap, and Agglomerative Clustering, with NMI score of 0.538, 0.536, and 0.515, respectively. Word[44]: We see that for the Word network, k-means is the worst performer, while our proposed CD-SACS is the best performer. Agglomerative clustering and Infomap also perform well on the word dataset. Football [45]: This network has 12 communities which are easily divisible due to sparse connection between them. All of the algorithms perform well and proposed algorithm CD-SACS performs the best with an NMI score of 0.94 followed by Agglomerative Clustering with a score of 0.92. Email [46]: The network 42 communities with dense connections between the clusters which makes prediction of communities challenging task. This results in lower NMI scores with CD-SACS performing the best with a score of 0.56 followed by Infomap. All other algorithms have similar results except Agglomerative Clustering which has a score of 0.05. LFR 0.1: This network is too simple for testing and is used to check whether proposed algorithm is working fine. All the community detection algorithms including proposed algorithm peforms optimally. Some algorithms like AP and AC fail to do so. LFR 0.3: Our model is able to detect all communities but so does other algorithms like k-means, Infomap and Spectral Clustering. In this network, communities are not as sparsely connected as in LFR 0.1 but still many community detection algorithms are able to detect communities perfectly including proposed algorithm LFR 0.5: This network is challenging for community detection since it is hard even after seeing the graph with naked eye to detect the communities. All algorithms struggle on this network including proposed algorithm. Best result is obtained by CD-SACS of 0.288 closely followed by AP with an NMI score of 0.218.

Table 4: NMI Score for all the algorithms evaluated on all the networks chosen by us.

Network	**KM**	**IF**	**SC**	**AP**	**LP**	**AC**	**SAECF**	**PFGM**	**LCD**	**CD-SACS**
Karate	0.004	0.699	1.0	0.535	0.448	0.085	0.489	0.746	0.596	1.0
Dolphins	0.747	0.834	0.376	0.437	0.634	0.495	0.494	0.705	0.605	0.865
Polbooks	0.454	0.536	0.538	0.321	0.436	0.515	0.393	0.345	0.383	0.556
Word	0.614	0.705	0.697	0.653	0.673	0.723	0.522	0.764	0.644	0.764
Football	0.892	0.882	0.902	0.851	0.682	0.924	0.678	0.721	0.644	0.941
Email	0.478	0.539	0.439	0.459	0.368	0.587	0.391	0.315	0.389	0.561
Polblogs	0.518	0.558	0.538	0.535	0.575	0.593	0.372	0.382	0.345	0.648
Cora	0.623	0.632	0.661	0.753	0.758	0.773	0.562	0.583	0.754	0.827
Citeseer	0.737	0.758	0.763	0.774	0.849	0.827	0.609	0.694	0.814	0.879
Facebook	0.683	0.728	0.763	0.772	0.822	0.789	0.653	0.706	0.801	0.843
LFR 0.1	1.0	1.0	1.0	0.681	1.0	0.667	0.689	0.686	0.715	1.0
LFR 0.3	1.0	1.0	1.0	0.641	0.577	0.667	0.718	0.761	0.745	1.0
LFR 0.5	0.022	0.083	0.119	0.218	0.064	0.052	0.767	0.904	0.832	0.288

The above results show that our proposed CD-SACS algorithm is the best performer in terms of the NMI score obtained on all the networks. This can be attributed to the careful dimensionality reduction by our algorithm using

the stacked autoencoder while maintaining all the topological details of the network. The centroid generation for the k-means algorithm using the Crow Search optimization also helps in detecting the community structures optimally. In addition to this, the proper parameter tuning done by us also helps in achieving the superior performance in terms of NMI score obtained by our algorithm.

6.3 Modularity Score

This section presents the results obtained for Modularity score by various algorithms over all the datasets mentioned in Sections 5.1 and 5.2. The scores were also compared with several contemporary community detection algorithms to obtain a comparative study. Brief description for results of modularity for the networks are as follows.
Karate[41]: CD-SACS has the highest modularity score for this network followed by SC, Infomap and LP. AC, AP and k-means have very low modularity scores.
Dolphin[42]: In prediction of community detection for this network, proposed algorithm performs best with a modularity score of 0.529 and closely followed to Infomap. This is followed by LP, k-means and AC. SC and AP have very low modularity scores for the Dolphin network.
Polbooks[43]: This network has 3 communities and is relatively diffcult for community detection but proposed algorithm performs better than others with a score of 0.527. CD-SACS has the highest modularity score for this network followed by AC, SC, LP, Infomap and k-means. AP has very low modularity score.
Word[44]: For the Word network, our proposed CD-SACS algorithm is the best performer with a modularity score of 0.573. This is closely followed by AC and LP algorithms.
Football [45]: This network has 12 communities to predict which proposed algorithm does very well. This is shown by modularity score of 0.634 for proposed algorithm which is higher than other community detection algorithms. CD-SACS has the highest modularity score for this network followed by AC, Infomap, LP, k-means, SC and AP. Apart from AP all algorithms have similiar modularity scores.
Email[46]: This network has 24 communities which makes it difficult to perform community detection but proposed algorithm out performs other community detection algorithm with a modularity score of 0.468. CD-SACS has the highest modularity score for this network followed by Infomap and k-means. Remaining algorithms used for comparision have very low scores.
Polblogs[47]: For the Polblogs network we see that our proposed algorithm is the best performer with a modularity score of 0.489. This is closely followed by AC, SC and k-means algorithms.
Cora[48]: All the algorithms generate competitive results for the Modularity scores on the Cora network. But even for the Cora network, we see that our

proposed CD-SACS algorithm is the best performer closely followed by AC and AP algorithms.
Citeseer [49]: The Modularity results obtained for the Citeseer network show that k-means clustering algorithm is the worst performer. Our proposed CD-SACS algorithm is the best performer and all the other algorithms follow thus.
Facebook [50]: For the facebook network also our proposed CD-SACS algorithm is best performer outperforming the other community algorithms by a considerable margin. k-means clustering algorithm tends to be the worst performer while the LP algorithm is the second best performer.
LFR 0.1: Same modularity score is obtained by the best performing algorithms. All other algorithms also perform well on this network. This is because community detection on this network is easy due to which many community detection algorithms are able to detect the communities optimally including our proposed CD-SACS algorithm.
LFR 0.3: The best modularity score is obtained by our CD-SACS algorithm. This shows that despite being much more difficult for community detection than LFR 0.1, most best algorithms are able to perform community detection including proposed algorithm.
LFR 0.5: Modularity scores obtained for this network is very low. This shows that it is difficult to perform community detection on networks where communities are not sparsely connected and are close to each other. Even still our proposed community detection algorithm CD-SACS performs best in detecting the communities optimally.

Table 5: Modularity Score for all the algorithms evaluated on all the networks chosen by us.

Network	**KM**	**IF**	**SC**	**AP**	**LP**	**AC**	**SAECF**	**PFGM**	**LCD**	**CD-SACS**
Karate	0.188	0.332	0.371	0.017	0.325	0.402	0.345	0.346	0.307	0.434
Dolphins	0.491	0.514	0.213	0.086	0.498	0.412	0.484	0.406	0.599	0.529
Polbooks	0.438	0.451	0.454	0.146	0.481	0.489	0.595	0.542	0.575	0.527
Word	0.436	0.457	0.469	0.485	0.493	0.542	0.492	0.565	0.438	0.573
Football	0.579	0.587	0.551	0.396	0.583	0.596	0.578	0.522	0.446	0.634
Email	0.234	0.417	0.368	0.326	0.386	0.423	0.382	0.417	0.582	0.468
Polblogs	0.465	0.461	0.465	0.453	0.463	0.474	0.377	0.373	0.345	0.489
Cora	0.513	0.528	0.517	0.527	0.518	0.553	0.567	0.591	0.557	0.594
Citeseer	0.497	0.483	0.545	0.512	0.523	0.537	0.511	0.407	0.413	0.576
Facebook	0.495	0.502	0.548	0.546	0.586	0.566	0.502	0.506	0.401	0.614
LFR 0.1	0.653	0.552	0.657	0.628	0.653	0.443	0.495	0.483	0.514	0.674
LFR 0.3	0.432	0.452	0.446	0.468	0.226	0.315	0.418	0.464	0.546	0.512
LFR 0.5	0.145	0.223	0.231	0.187	0.268	0.292	0.254	0.204	0.233	0.334

The above results shows that our proposed CD-SACS algorithm performs more optimally in terms of modularity score as compared to other community detection algorithms on all the chosen networks. Moreover, our CD-SACS algorithm also gives stable results across real-world as well as synthetic networks.

The optimised performance can be attributed to the efficient use Crow-Search swarm-intelligence algorithm paired with the stacked autoencoders to cluster the nodes in the networks in a much better way as compared to other contemporary algorithms. The efficacy of our proposed CD-SACS algorithm can be observed over both synthetic as well as real-life networks of varied dimensions and topology.

Table 6: Precision values for all the algorithms evaluated on all the networks chosen by us

Network	**KM**	**IF**	**SC**	**AP**	**LP**	**AC**	**SAECF**	**PFGM**	**LCD**	**CD-SACS**
Karate	0.594	0.851	0.403	0.247	0.782	0.828	0.466	0.699	0.683	0.868
Dolphins	0.649	0.813	0.511	0.403	0.725	0.782	0.466	0.707	0.628	0.888
Polbooks	0.662	0.884	0.514	0.089	0.721	0.725	0.569	0.681	0.518	0.816
Word	0.596	0.879	0.389	0.556	0.755	0.721	0.475	0.773	0.722	0.894
Football	0.773	0.866	0.837	0.247	0.81	0.755	0.659	0.722	0.716	0.868
Email	0.828	0.863	0.849	0.118	0.739	0.81	0.618	0.795	0.855	0.776
Polblogs	0.87	0.81	0.365	0.16	0.844	0.739	0.721	0.756	0.825	0.904
Cora	0.826	0.828	0.303	0.771	0.886	0.844	0.641	0.704	0.722	0.885
Citeseer	0.727	0.863	0.323	0.908	0.779	0.886	0.621	0.807	0.785	0.908
Facebook	0.745	0.836	0.405	0.688	0.742	0.741	0.847	0.658	0.804	0.883
LFR 0.1	0.722	0.769	0.444	0.461	0.686	0.674	0.742	0.685	0.739	0.868
LFR 0.3	0.767	0.957	0.368	0.739	0.862	0.768	0.831	0.704	0.739	0.905
LFR 0.5	0.878	0.88	0.849	0.869	0.822	0.808	0.668	0.904	0.827	0.913

Table 7: Recall values for all the algorithms evaluated on all the networks chosen by us

Network	**KM**	**IF**	**SC**	**AP**	**LP**	**AC**	**SAECF**	**PFGM**	**LCD**	**CD-SACS**
Karate	0.558	0.675	0.685	0.823	0.677	0.851	0.596	0.856	0.338	0.833
Dolphins	0.628	0.757	0.747	0.685	0.818	0.709	0.543	0.744	0.515	0.752
Polbooks	0.624	0.77	0.852	0.537	0.795	0.769	0.811	0.587	0.645	0.78
Word	0.648	0.806	0.805	0.77	0.704	0.771	0.914	0.729	0.496	0.848
Football	0.757	0.843	0.671	0.823	0.672	0.723	0.833	0.699	0.467	0.786
Email	0.872	0.877	0.719	0.85	0.796	0.717	0.839	0.869	0.461	0.88
Polblogs	0.911	0.881	0.683	0.833	0.808	0.778	0.897	0.491	0.842	0.672
Cora	0.563	0.751	0.606	0.829	0.573	0.826	0.274	0.338	0.861	0.828
Citeseer	0.638	0.789	0.677	0.868	0.556	0.551	0.523	0.595	0.854	0.83
Facebook	0.614	0.809	0.716	0.666	0.608	0.693	0.381	0.857	0.804	0.81
LFR 0.1	0.772	0.857	0.831	0.815	0.611	0.71	0.576	0.742	0.694	0.837
LFR 0.3	0.927	0.883	0.698	0.796	0.742	0.763	0.547	0.781	0.739	0.867
LFR 0.5	0.945	0.899	0.719	0.796	0.787	0.804	0.903	0.901	0.827	0.805

6.4 Precision and Recall

This section presents the result analysis for the precision and recall performance metrics. Tab. 6 shows the precision values for all the algorithms evaluated on all the networks chosen by us, while Tab. 7 shows the recall values for all the algorithms evaluated on all the networks chosen by us. From Tab. 6, we see that our proposed CD-SACS algorithm performs best in terms of precision except for Polbooks, Email, Cora, and the LFR 0.3 synthetic network. From Tab. 7, we can see that our proposed CD-SACS algorithm gives competitive results in terms of recall values being the best performer for Email, Cora, and Facebook networks. The relatively lower performance of our algorithm in terms of recall can be understood due to the inverse proportionality relation between precision and recall.

7 Conclusion

Community detection is one of the most researched topics in complex network analysis. In this work, we proposed a novel community detection algorithm, named CD-SACS, using stacked autoencoders and a Crow Search Algorithm based k-means clustering algorithm. We start by obtaining a modularity matrix for the input graph, which is then passed through a network of stacked autoencoders to reduce the dimensionality of the modularity matrix while preserving crucial topological details and the least correlated and most relevant features. This feature matrix is then passed through a Crow-Search optimization based k-means clustering algorithm, which groups the nodes into their clusters and divides the network into various communities. We exploited Crow-Search optimization to generate the initial centroids for the k-means algorithm to avoid early convergence into a local optimum and to allow the algorithm to explore a much larger global search space to obtain a solution much closer to the global optimum. We performed experimental analysis on several real-life and synthetic benchmark networks based on various evaluation metrics. The results of our proposed approach were also compared with some of the traditional and contemporary community detection algorithms. The obtained results exhibit the efficacy of our proposed algorithm over others community detection algorithms. One major limitation of the proposed work is that, like other metaheuristics, the modified crow search algorithm may also suffer from the problem of early convergence and falling into local optimum. Also, the proposed work can not be used for weighted and multi-layer complex networks. In the near future, we can extend our work in weighted and multi-layer complex networks.

Conflict of interest

The authors declare that they have no conflict of interest.

Data availability

All data generated or analysed during this study are included in this article.

References

1. Boccaletti S, Latora V, Moreno Y, Chavez M, Hwang DU (2006) Complex networks: Structure and dynamics. Physics Reports 424(4):175–308, DOI https://doi.org/10.1016/j.physrep.2005.10.009, URL `https://www.sciencedirect.com/science/article/pii/S037015730500462X`
2. Li B, Pi D (2019) Learning deep neural networks for node classification. Expert Systems with Applications 137:324–334
3. Sattar NS, Arifuzzaman S (2022) Scalable distributed louvain algorithm for community detection in large graphs. The Journal of Supercomputing 78(7):10275–10309
4. Kumar S, Mallik A, Panda B (2022) Link prediction in complex networks using node centrality and light gradient boosting machine. World Wide Web pp 1–27
5. Kumar S, Gupta A, Khatri I (2022) Csr: A community based spreaders ranking algorithm for influence maximization in social networks. World Wide Web pp 1–20
6. Kumar S, Saini M, Goel M, Aggarwal N (2020) Modeling information diffusion in online social networks using sei epidemic model. Procedia Computer Science 171:672–678
7. Plantié M, Crampes M (2013) Survey on social community detection. In: Social media retrieval, Springer, pp 65–85
8. Kumar S, Hanot R (2020) Community detection algorithms in complex networks: A survey. In: International Symposium on Signal Processing and Intelligent Recognition Systems, Springer, pp 202–215
9. Girvan M, Newman ME (2002) Community structure in social and biological networks. Proceedings of the national academy of sciences 99(12):7821–7826
10. Tripathi B, Parthasarathy S, Sinha H, Raman K, Ravindran B (2019) Adapting community detection algorithms for disease module identification in heterogeneous biological networks. Frontiers in genetics 10:164
11. Tang L, Liu H (2010) Community detection and mining in social media. Synthesis lectures on data mining and knowledge discovery 2(1):1–137
12. Yang Z, Algesheimer R, Tessone CJ (2016) A comparative analysis of community detection algorithms on artificial networks. Scientific reports 6(1):1–18
13. Intrator N, Edelman S (1996) Making a low-dimensional representation suitable for diverse tasks. In: Learning to learn, Springer, pp 135–157
14. Fortunato S (2009) Community detection in graphs/santo fortunato. Physics Reports

15. Newman MEJ (2006) Modularity and community structure in networks. Proceedings of the National Academy of Sciences 103(23):8577–8582, DOI 10.1073/pnas.0601602103, URL https://www.pnas.org/content/103/23/8577, https://www.pnas.org/content/103/23/8577.full.pdf
16. Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99(12):7821–7826, DOI 10.1073/pnas.122653799, URL https://www.pnas.org/content/99/12/7821, https://www.pnas.org/content/99/12/7821.full.pdf
17. Duch J, Arenas A (2005) Community detection in complex networks using extremal optimization. Physical review E 72(2):027104
18. Burk CF, Horton FW (1988) Infomap: a complete guide to discovering corporate information resources
19. Von Luxburg U (2007) A tutorial on spectral clustering. Statistics and computing 17(4):395–416
20. Wang K, Zhang J, Li D, Zhang X, Guo T (2008) Adaptive affinity propagation clustering. arXiv preprint arXiv:08051096
21. Murtagh F, Legendre P (2014) Ward's hierarchical agglomerative clustering method: which algorithms implement ward's criterion? Journal of classification 31(3):274–295
22. Gregory S (2010) Finding overlapping communities in networks by label propagation. New Journal of Physics 12(10):103018, DOI 10.1088/1367-2630/12/10/103018, URL https://doi.org/10.1088/1367-2630/12/10/103018
23. Bhih A, Johnson P, Randles M (2020) An optimisation tool for robust community detection algorithms using content and topology information. The Journal of Supercomputing 76(1):226–254
24. Jalali S, Hosseini M (2021) Social collaborative filtering using local dynamic overlapping community detection. The Journal of Supercomputing pp 1–21
25. Mohammadi M, Fazlali M, Hosseinzadeh M (2021) Accelerating louvain community detection algorithm on graphic processing unit. The Journal of Supercomputing 77(6):6056–6077
26. Jaradat AS, et al. (2018) Community structure detection using firefly algorithm. International Journal of Applied Metaheuristic Computing (IJAMC) 9(4):52–70
27. Kumar S, Panda B, Aggarwal D (2020) Community detection in complex networks using network embedding and gravitational search algorithm. Journal of Intelligent Information Systems pp 1–22
28. Messaoudi I, Kamel N (2019) A multi-objective bat algorithm for community detection on dynamic social networks. Applied Intelligence 49(6):2119–2136
29. Pattanayak HS, Sangal AL, Verma HK (2019) Community detection in social networks based on fire propagation. Swarm and evolutionary computation 44:31–48

30. 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):328–340
31. Bandyopadhyay S, Peter V (2021) Unsupervised constrained community detection via self-expressive graph neural network. In: Uncertainty in Artificial Intelligence, PMLR, pp 1078–1088
32. Zhang M, Zhou Z (2020) Structural deep nonnegative matrix factorization for community detection. Applied soft computing 97:106846
33. Xu R, Che Y, Wang X, Hu J, Xie Y (2020) Stacked autoencoder-based community detection method via an ensemble clustering framework. Information sciences 526:151–165
34. Wang Y, Cao J, Bu Z, Jiang J, Chen H (2021) Proximity-based group formation game model for community detection in social network. Knowledge-Based Systems 214:106670
35. Shang R, Zhang W, Zhang J, Feng J, Jiao L (2022) Local community detection based on higher-order structure and edge information. Physica A: Statistical Mechanics and its Applications 587:126513
36. Vincent P, Larochelle H, Lajoie I, Bengio Y, Manzagol PA, Bottou L (2010) Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. Journal of machine learning research 11(12)
37. Beecks C, Uysal MS, Seidl T (2010) Similarity matrix compression for efficient signature quadratic form distance computation. In: Proceedings of the Third International Conference on SImilarity Search and APplications, Association for Computing Machinery, New York, NY, USA, SISAP '10, p 109–114, DOI 10.1145/1862344.1862361, URL `https://doi.org/10.1145/1862344.1862361`
38. Arthur D, Vassilvitskii S (2007) k-means++: the advantages of careful seeding, p 1027–1035. In: SODA'07: proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA
39. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Computers & Structures 169:1–12
40. Hussien AG, Amin M, Wang M, Liang G, Alsanad A, Gumaei A, Chen H (2020) Crow search algorithm: theory, recent advances, and applications. IEEE Access 8:173548–173565
41. Zachary WW (1977) An information flow model for conflict and fission in small groups. Journal of anthropological research 33(4):452–473
42. Lusseau D (2003) The emergent properties of a dolphin social network. Proceedings of the Royal Society of London Series B: Biological Sciences 270(suppl_2):S186–S188
43. Ying X, Pan K, Wu X, Guo L (2009) Comparisons of randomization and k-degree anonymization schemes for privacy preserving social network publishing. In: Proceedings of the 3rd Workshop on Social Network Mining and Analysis, pp 1–10

44. Newman ME (2006) Finding community structure in networks using the eigenvectors of matrices. Physical review E 74(3):036104
45. Evans T (2012) American college football network files
46. Leskovec J, Krevl A (2014) SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data
47. Adamic LA, Glance N (2005) The political blogosphere and the 2004 us election: divided they blog. In: Proceedings of the 3rd international workshop on Link discovery, pp 36–43
48. Getoor L (2005) Link-based classification. In: Advanced methods for knowledge discovery from complex data, Springer, pp 189–207
49. Sen P, Namata G, Bilgic M, Getoor L, Galligher B, Eliassi-Rad T (2008) Collective classification in network data. AI magazine 29(3):93–93
50. Yang T, Jin R, Chi Y, Zhu S (2009) Combining link and content for community detection: a discriminative approach. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 927–936
51. Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78:046110, DOI 10.1103/PhysRevE.78.046110, URL https://link.aps.org/doi/10.1103/PhysRevE.78.046110
52. Estévez PA, Tesmer M, Perez CA, Zurada JM (2009) Normalized mutual information feature selection. IEEE Transactions on neural networks 20(2):189–201