Received June 10, 2021, accepted June 26, 2021, date of publication July 7, 2021, date of current version July 14, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3095335

A Review on Community Detection in Large Complex Networks from Conventional to Deep Learning Methods: A Call for the Use of Parallel Meta-Heuristic Algorithms

MOHAMMED NASSER AL-ANDOLI[1], SHING CHIANG TAN[1], WOOI PING CHEAH[2], AND SIN YIN TAN[1]

[1]Faculty of Information Science and Technology, Multimedia University, Malacca 75450, Malaysia
[2]Faculty of Science and Engineering, School of Computer Science, University of Nottingham Ningbo China, Ningbo 315100, China

Corresponding author: Shing Chiang Tan (sctan@mmu.edu.my)

This work was supported by the Fundamental Research Grant Scheme (FRGS) through the Ministry of Higher Education Malaysia under Project FRGS/1/2018/ICT02/MMU/02/1.

ABSTRACT Complex networks (CNs) have gained much attention in recent years due to their importance and popularity. The rapid growth in the size of CNs leads to more difficulties in the analysis of CNs tasks. Community Detection (CD) is an important multidisciplinary research area where many machine/deep learning-based methods have been applied to map CNs into a low-dimensional representation for extracting information similarity among members of CNs. Currently, Deep Learning (DL) is one of the promising methods to extract knowledge and learn information from high dimensional space and represent it in low dimensional space. However, designing an accurate and efficient DL-based CD method especially when dealing with large CNs is always an on-going research endeavor to pursue. Meta-Heuristic (MH) algorithms have shown their potentials in improving DL models in terms of solution quality and computational cost. In addition, parallel computing is a feasible solution for building efficient DL models. The algorithmic principle of MH is parallel in nature; however, its computation framework in DL training that is reported in the literature is not really implemented in a parallel computing setup. In this paper, we present a systematic review of CD in CNs from conventional machine learning to DL methods and point out the gap of applying DL-based CD methods in large CNs. In addition, the relevant studies on DL with parallel and MH approaches are reviewed and their implications on DL models are highlighted to prospect effective solutions to overcome the challenges of DL-based CD methods. We also point out research challenges in the field of CD and suggest possible future research directions.

INDEX TERMS Community detection, deep learning, complex networks, meta-heuristic algorithms, parallel computing.

I. INTRODUCTION

The study of Complex Networks (hereinafter referred to as CNs) has gained much attention in recent years due to their importance and popularity. With the development in science and technology, a variety of research in the field of CNs have attracted a considerable amount of attention from the research community. Recently, unstructured data has increased rapidly due to the fast development of Internet technology. A large part of the data is in the form of CNs. Large complex systems can also be presented as a CN, such as social networks [1], biochemical networks [2], protein-protein interaction networks [3], computer networks [4], citation networks [5], etc. CNs consist of interconnected nodes. Due to proliferation of social networks such as Facebook, Twitter, LinkedIn, etc., there are billions of users. The study of user interactions on these networks is of a great importance to many parties, including academia, industry, governments, etc [6]. Hence,

The associate editor coordinating the review of this manuscript and approving it for publication was Feiqi Deng.

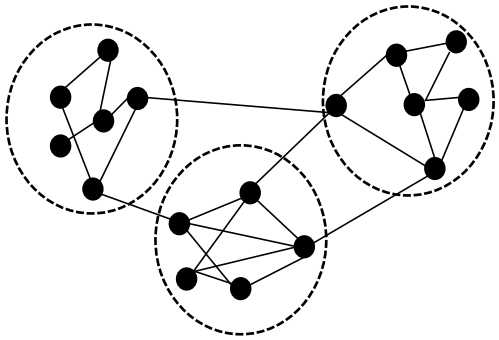

FIGURE 1. Community detection visualizations with three groups of nodes.

it is worthwhile to understand the relevant literature and study research insights from the CNs of which their complexities are ever increasing.

Community detection (hereafter referred to as CD) is one of the most important research fronts in the field of CNs. It is also a key multi-disciplinary field of research and is useful for understanding the structure of networks. [7]. The idea of CD in CNs is depicted in Fig. 1. The interest of CD is valuable in several applications. Usually, a CN is divided into several groups (or communities) so that the number of relationships among nodes in a group is more than the number of relationships the nodes between groups. CD is one of the fundamental tasks in CNs. After the relationship of nodes within a group and among groups have been identified through CD, one can analyze and mine the potentially useful information from different network communities in a CN that are knowledge representations in many applications such as social networks, medical science, machine learning, criminology, and biology [1], [6]. In addition, communities in a CN can exchange and suggest information due to similar desires among the members, and this feature is useful for tasks in a variety of applications which require recommendations, segmentation, vertex labelling, link inference, and influence analysis. CD is also helpful to detect subspecies groups or individuals [8], [9].

A great deal of effort has been devoted to develop new CD methods to extract meaningful features from a CN and represent them in a low-dimensional space (sometimes called learning embedding-based CD); then, clustering algorithms such as k-means are applied to find communities in the low dimensional space [8], [9]. Many conventional machine-learning (ML) algorithms were developed to solve the problem of CD [10], [11]. However, due to the increase in network size and complexity, the traditional ML methods such as spectral mapping and non-negative matrix factorization methods are no longer viable. Moreover, these methods can easily provide local optimal solutions and have high complexity [6], [12]. An alternative way is to introduce deep learning-based methods to cope with the CD problems and challenges. This trend has brought deep learning an attractive solution and research area for addressing the CD problem [13]. In addition, meta-heuristic algorithms were adapted to deal with relevant issues of CD [14].

Deep learning (hereinafter referred to as DL) has become one of the most active research areas in artificial intelligence and machine learning. It has achieved impressive results in many areas, such as speech recognition, image analysis and text comprehension, for both supervised and unsupervised learning strategies [15]. Unfortunately, due to the unique properties of graphs, applying DL-based methods to the CN problems has never been an easy task [16]. The low-dimensional representation capability of DL-methods in solving the CD problem was investigated in recent studies; these DL methods are, for example, stack and sparse autoencoders [8], [17], graph neural networks [18], [19], and other variants of DL [6], [20]. Despite the presence of DL-based CD methods, there is still a lack of finding communities effectively and efficiently, especially when dealing with large CNs. Unfortunately, in reality, the size of the networks increases considerably. In addition, most of the existing DL-based CD methods are trained with gradient decent strategy and backpropagation algorithm, which have three limitations: (1) training process is slow, especially with big data; (2) sensitive to parameter initiation; and (3) the solutions from these methods are likely to fall in local minima [21], [22].

Meta-heuristic (MH) is a concept of a set of algorithms including evolutionary algorithms such as naturally inspired algorithms such as Particle Swarm Optimization [23], and Genetic Algorithm [24]. MH algorithms have been successful used for optimizing machine learning models. They are efficient methods to solve complex problems and could find optimal solutions within an acceptable duration of time. Nowadays, MH algorithms are the state of the art for various optimization problems, especially for problems that are very complex and have high dimensionality [25]; the CD problem is no exception. MH algorithms were adapted and employed to address the problem of CD [14]. Even though many effort have been made to develop effective MH algorithms for the CD problem, they encounter the problem of low accuracy and deficiency in dealing with large CNs [14]. On the other hand, MH algorithms have been applied to optimize DL models, where the gradient descent optimization is replaced by iterative MH algorithms to optimize the parameters and model structure [26]. However, the number of paper publications related to the application of MH in large-scale DL is still few [21], [22].

The emergence of big data is observed recently. DL is playing an important role to deal with big data and to harvest valuable knowledge from complex systems [15], [27]. On the other hand, the complexity of DL models increases significantly due to massive computational work and memory requirements. The development of High-Performance Computing (HPC) devices is helpful to support the development of large-scale DL models. The utilization of parallel computing helps to improve the efficiency for the training of large-scale DL models on big data by dividing a task into subtasks and solving them simultaneously. Parallel computing improves the efficiency of large-scale DL models by utilizing Graphics

Processing Unit (GPU) devices [28], [29], and clusters of CPUs [30], [31].

A. EXISTING REVIEWS

The field of community detection is developing rapidly and plays an important role in solving various complex problems in real life. There are various review articles about the applications and methods of community detection, most of them are developed for conventional CD methods [1], [32]. Recently, a few studies review recent techniques, such as DL and MH. The study by Liu *et al.* [13] presented a review on CD with DL. It contains a discussion of deep graph embedding, deep neural networks, and graph neural networks. The survey by Jin *et al.* [9] contained a discussion of probabilistic graphical model and DL methods. In addition, Abduljabbar *et al.* [14] presented a generic overview Nature-inspired optimization algorithms and their role in solving CD problems. The heuristic and MH based CD algorithms were reviewed by Bara'a *et al.* in [33]. Furthermore, the authors described hybrid MH and hyper heuristic algorithms for CD. Despite the availability of some studies in the literature that provide the researcher with valuable information about CD using DL and MH techniques, there is still a lack of literature that gives a macroscopic view of effective and efficient DL- and MH-based CD in large CNs.

One of our main goals in this paper is to show the DL-based CD need of MH and parallel approaches to bridge the gap exist so far in a unified perspective. Our survey differs from the published ones in four aspects. First, we present a recent trend in the development of methods for CD, i.e., from conventional to DL, while the others focus mainly on individual techniques, e.g., conventional [1], [32], DL [9], [13], or MH [14], [33]. Second, the paper focuses on a survey of DL and MH- based methods for CD in large CNs that are defined as a big data problem. The scope of survey is different from the existing papers [9], [13] where the solutions are not applied for handling big data. Third, the survey summarizes recent research work involving the integration of DL and MH. Fourth, this paper refers to the most successful studies of using large DL models with MH and parallel techniques in other domains to encourage researchers to carefully reconsider the design of effective and efficient DL-based solutions for CD in large CNs.

B. CONTRIBUTIONS

This paper presents a systematic review of CD in CNs from conventional machine learning to DL methods, and point out the gap of applying DL-based CD methods in large CNs. In addition, the relevant studies on DL with parallel and MH approaches in various domains are reviewed and their implications on DL models are highlighted to prospect effective solutions to overcome the challenges of DL-based CD methods. We summarize the contribution of this paper as follows:

- We provide a comprehensive review of learning-based CD from conventional machine learning to deep learning-based methods, considering existing methods in MH and parallel approaches to deal with large CNs. To our knowledge, this is the first effort dedicated to summarize current research work on the application of the mentioned methods for CD. We also point out challenges and open research questions related to DL-based methods for CD.
- We present an overview of recent remarkable studies that were developed to improve the effectiveness and efficiency of large-scale DL models in various fields, especially those using MH and parallel approaches to deal with big data. In addition, the existing challenges and open issues in such methods are discussed. The aim of this review is to motivate and inspire the research community to adapt DL to use MH and parallel approaches to overcome the existing problems of DL-based methods in CD, especially applied in large CNs.
- Finally, a summary of review, future directions, and research gaps in the field of CD are presented at before the end of the paper, which shed light on future endeavours in developing effective and efficient computing solutions for large-scale CNs.

C. TERMINOLOGY

A set of acronyms are used throughout the presentation. For a quick access, Table 1 lists all acronyms used in this article.

D. PAPER ORGANIZATION

The paper is organized as follows. Section II presents a detailed methodology to conduct this study. The preliminary concepts of CD in CNs with DL are described in Section III. The technical overview of the research on earlier conventional CD, as well as parallel conventional CD methods are presented in Section IV. Section V overviews the research progress on DL-based CD methods. Section VI overviews the research on DL-based parallel computing on regular and irregular domains. An overview of MH algorithms, CD-based MH algorithms, and DL-based MH optimization methods are presented in Section VII. The open benchmark datasets that have been used in the area of CD are presented in Section VIII. Section IX summarizes the review and point out research challenges involving the use of the mentioned methods in Section VII for CD. Section X suggests future research directions. Finally, the paper is concluded in Section XI.

II. REVIEW METHODOLOGY

A survey of the literature was conducted to identify publications describing CD algorithms from conventional to DL algorithms as well as DL with parallel and MH approaches. For this purpose, keywords and phrases were used to search for articles in well-known repositories such as Google Scholar, IEEE Xplore, Science Direct, Web of Science, Scopus, and arXiv. The overall procedure

TABLE 1. List of acronyms.

Acronym	Meaning
CN	Complex network
CD	Community detection
DL	Deep learning
MH	Meta-heuristic
ML	Machin learning
HPC	High-performance computing
GPU	Graphics processing unit
SBM	Stochastic block model
NMF	Non-negative matrix factorization
KL	Kullback-Leibler
ANN	Artificial neural network
CNN	Convolutional neural network
GNN	Graph neural network
GCN	Graph convolutional network
SGD	Stochastic gradient descent
DNN	Deep Neural Network
RNN	Recurrent neural network
DBN	Deep Belief Network
CAE	Convolutional autoencoder
GA	Genetic algorithm
PSO	Particle swarm optimization
ACO	Ant colony optimization
ABC	Artificial bee colony
FA	Firefly algorithm
CS	Cuckoo Search
BA	Bat algorithm
SA	Simulated Annealing
SFSO	Sigmoid fish swarm optimization
WWO	Water wave optimization
WOA	Whale optimization algorithm
GWO	Grey wolf optimization
GD	Gradient descent
BP	Backpropagation

was as follows: first, the identified keywords were employed along with their synonyms to find relevant articles. The keywords were set with the following searching criteria: ((**"community detection∗" OR "community discovery∗" OR "deep learning∗ community detection" OR "metaheuristic∗ community detection" OR "finding communities∗") AND ("complex networks" OR "large complex networks" OR "social networks") AND/OR ("parallel∗ deep learning∗" OR "metaheuristic ∗ deep learning"**)). Second, the title and abstract of the articles were read to remove unrelated articles. Next, the related papers were selected for full-text reading according to the following criteria:

- Publications after 2015 were given a higher priority. If they did not exist, those articles published after 2010 were reviewed. Some remarkable studies before 2010, which were few, were also selected.
- To overcome the language difficulty, only the articles written in the English language were considered.
- The literature review was based on papers that applied DL and ML for community detection, and that introduced supporting techniques to improve the efficiency and scalability of large-scale DL models (such as MH and parallel techniques).

As a result, a total number of 184 paper publications were identified to form the basis for this review. Approximately two-thirds of the publications (63.2%) were journal articles. The rest were presented at conferences (34.1%), and book chapters (2.7%).

The selected papers were studied in details and the key information were extracted, e.g., the approach of CD and algorithms, the applications and datasets used and their attributes, the improvement techniques applied in terms of efficiency and effectiveness, and merits and demerits of individual CD methods.

III. PRELIMINARY CONCEPT

A complex network $CN(n, m)$ can be expressed as a graph $G(V, E)$. Here V represents a set of nodes n, $V(G) = \{v_1, v_2, \ldots, v_n\}$ and E represents a set of relations m between nodes. In the abstract form of the relations, they are considered as undirected, unweighted and unsigned relations. The adjacency matrix $A = [a_{ij}] \in R^{n\times n}$ is the most widely used representation of these relations, where the pair nodes i and j are represented as a_{ij} and defined as $a_{ij} = w_{ij}, if\,(a_{ij} = 1)$ means the nodes are connected, $a_{ij} = 0$ the nodes are disconnected. The number of relationships between the node v_i and other nodes is called node-degree that can be calculated as:

$$d(v_i) = \sum_{i}^{n} a_{ij} \tag{1}$$

The node with a high degree refers to the importance of the node in the CN. Another representation is a modularity matrix $\mathbf{B} = [b_{ij}] \in R^{n\times n}$. The element b_{ij} can be expressed as $\mathbf{B} = a_{ij} - (k_i k_j/2m)$, where k_i and k_j indicate the degree of node i, j and m is the total number of all edges in the *G/CN*. Each node v_i has a vector with a length size equals n, which represents the node i relationships with all nodes in the *G/CN*, $v_{i,j}, j = \{1, 2, \ldots, n\}$.

The goal of CD algorithms is to partition a given CN into a set of k communities $C = \{c_1, c_2, \ldots, c_k\}$, so that the relationships inside each community, e.g., c_1, is denser than the relationships between the communities, e.g., $c_1 \leftrightarrow c_2$. This is typically calculated by a Q-Modularity function that should be maximized. The Q-Modularity function is denoted as:

$$Q = \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{d_i d_j}{2m})\delta_{ij}, \tag{2}$$

Here, m, d_i and A_{ij} are the number of edges between the nodes in the sub-graph, the degree of the node i, and the adjacency matrix, respectively. In case of $\delta_{ij} = 1$, i and j are in the same community, otherwise $\delta_{ij} = 0$.

The nodes with a nearly similar vector representation are assigned to the same community. This can be performed as follows. First, some nodes are selected as community centers correspond to k communities $v_c = \{v_{c1}, v_{c2}, \ldots, v_{ck}\}$; usually, by referring to Eq. (1), these nodes have high degree of relationships. Next, the Euclidean distance between each

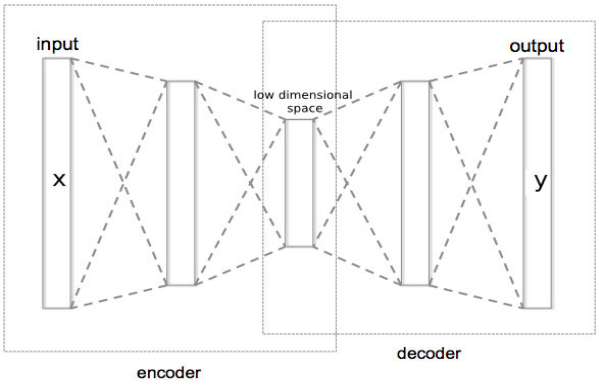

FIGURE 2. Structure of an autoencoder with 3 fully connected hidden layers.

node v_i and all community centers v_c is computed according to the following equation:

$$dis(v_i, v_{cj}) = \left\| v_i - v_{cj} \right\|_2^2, \quad i \in n,\ j \in k \tag{3}$$

Nodes v_i (where $i > 1$ and $i < n$) with a similar representation obtain similar distances from the same community center v_{cj}. As a result, they are grouped in the same community.

The main problem in this process is how to find an effective low-dimensional representation of the node so that the similarity between the nodes is promoted and the original network representation is preserved.

DL can effectively extract low-dimensional representation and find information similarity among nodes of CNs. For example, the deep autoencoder model reconstructs the output data so that it is similar to the input data [34]. Figure 2 shows the general structure of the autoencoder. It has two phases: (1) The first phase is called encoding, where the hidden layer h converts the data x and its features n into a latent representation with feature spaces d, where d is smaller than n. The mapping process is conducted with the $f1$ function, $h = f1(W_1x+b_1)$. (2) The second phase is called decoding, which attempts to reconstruct the original data from the output of the hidden layer using the $f2$ function $h = f2(W_2x + b_2)$. The *Sigmoid*, *Tanh*, and *Relu* activation functions are normally used with the decoding and encoding functions since they are nonlinear mapping functions. The parameters of autoencoder model are referred to as $\theta = [W_1, b_1, W_2, b_2]$ and trained by minimizing the loss function:

$$J_\theta = \frac{1}{n}\sum_{i=0}^{n}(x^i - y^i) \tag{4}$$

with respect to all samples n (i.e., nodes), where x and y denote the original input data and the reconstruction data, respectively.

The training steps of autoencoder include a feedforward process and a backward process. The time complexity of the autoencoder model is calculated by:

$$T(J_\theta + \partial J_\theta) = \mathrm{O}(2nho + 2nho) = \mathrm{O}(nho) \tag{5}$$

where n, h, and o refer to the number of nodes (i.e., samples of training), the number of neurons in the hidden and input/output layers. The deep autoencoder requires l layers and t iterations, and this changes the time complexity to O($tlnho$). In CN representation, a node i is represented by a vector of size n. Therefore, the time complexity is O(tln^2h).

The time complexity of assigning nodes to the nearest community is O(nkd), where n, d, and k denote the number of nodes, the size of the dimensional representation, and the number of communities, respectively.

IV. CONVENTIONAL COMMUNITY DETECTION METHODS

A great effort has been devoted for CD using conventional ML methods. This section presents remarkable and recent literature studies related to use of conventional ML-based methods for performing CD.

A. EARLIER CONVENTIONAL COMMUNITY DETECTION METHODS

In the last two decades, several conventional ML methods have been developed to address CD in CNs. Spectral mapping and clustering algorithms were introduced to find communities in networks. Ng *et al.* [35] presented a spectral clustering algorithm to detect communities in a CN. The main idea of the algorithm was to represent the data of a CN in a small-dimensional space. Then, a clustering algorithm for the new small dimensional space was used to discover communities. White and Smyth [11] devised a spectral clustering algorithm for CD based on the reformulation of eigenvectors of matrices. The study mapped the node data of the network into Euclidean space and then grouped them into clusters. The spectral algorithm was improved by Niu *et al.* [36]. The authors used the page rank method to find the core nodes, and then the cores were employed to initiate the cluster centers in the spectral clustering algorithm. Zhang *et al.* [37] proposed an extension of the spectral clustering algorithm to support the detection of overlapping communities, which included spectral mapping and Fuzzy c-means clustering [38]. Newman and Girvan [39] proposed a hierarchical clustering algorithm to solve the CD issue. It was based on a measurement called ''betweenness centrality'' [40]. It started by calculating the betweenness of all edges. Then, the edges with high betweenness were removed, and the betweenness for remaining edges was recalculated. Brandes et al [41] proposed a method for grouping the nodes of a CN into communities based on the modularity index. The method was aimed to find the best clustering with maximum modularity. A Random-Walk algorithm was built based on a random walk process, which was a process of navigating node at random [42]. Rosvall and Bergstrom [43] proposed an algorithm for CD based on flow of information between nodes, called an Infomap algorithm. They used maps to represent the flow of information between nodes. A group of nodes between which information flows were grouped into a single cluster. A random walk was used as a proxy for information flow.

Stochastic Block Model (SBM) is a generative model [44] to solve the CD problem. The model uses a node membership probability function to find hidden communities in the network. Karrer and Newman [45] improved the performance of the SBM model by proposing a new model based on the corrected degree. They overcame the limitations of the original SBM that used the basic node degrees (i.e., adjacent degrees) in graph clustering, which were not uniformly distributed. The study introduced parameter degrees for all nodes that could scale with edge probabilities and expected a new appropriate degree. Non-Negative Matrix Factorization (NMF) is another Conventional ML method developed for CD [46]. It mapped the topology information of a CN to a latent low-dimensional space. The new space was soft membership vectors that assign each node to a particular cluster. Shi *et al.* [47] developed a new NMF-based pairwise constrained method to improve the performance of CD. Wang *et al.* [48] also adapted the NMF algorithm to detect overlapping and non-overlapping communities in CNs.

B. PARALLEL CONVENTIONAL COMMUNITY DETECTION METHODS

Recently, many researchers have dedicated their efforts in developing parallel conventional ML methods to find communities in CNs. They use HPC that is available in either CPU or GPU devices to design parallel computation models. The research works are reviewed in the following sections.

1) CPU BASED PARALLEL COMPUTATION

Prat-Pérez et al [49] proposed a parallel algorithm based on optimizing weighted community clustering for CD in large graphs. The algorithm first used the clustering coefficient as an evidence to find initial communities, then refined these communities by moving nodes between them. The calculation of the clustering coefficient of vertices and the refinement functions were performed in parallel. He *et al.* [50] proposed a parallel CD algorithm based on the distance dynamics model [51], which made the interaction scope of each node is only affected by its neighbors. The algorithm divided the large network into sub-networks by a divide-and-conquer strategy. Fazlali *et al.* [52] proposed a parallel method to tackle Louvain CD algorithm. The method introduced a thread level parallelism for the calculation of adding qualified neighbor nodes to the community in a parallel way. Parallel processing with a threaded binary trees data structure method was proposed for CD in [53]. The method was performed over weighted networks in irregular topologies. The aforesaid CD methods were evaluated in multi-processors platforms.

Distributed parallel CD algorithm was proposed by Moon *et al.* in [54]. The authors developed two parallel versions of Newman and Girvan's algorithm [39] to handle CD in large CNs. MapReduce and the vertex-centric models were adopted for performing parallelization operations of the Newman and Girvan's algorithm. In addition, the algorithms were implemented together with GraphChi and Hadoop. Similarly, a parallel method based on a partitioning algorithm was proposed to solve the CD problem [55]. It used a subtree-split strategy to divide the network into sub-networks. The performance of above algorithms was evaluated on a cluster of CPUs platform and applied to different large networks. The experimental results showed that they could give a good performance in terms of time computation.

2) GPU BASED PARALLEL COMPUTATION

Some researchers have also suggested that the use of GPU devices could improve the efficiency of CD methods. Soman and Narang [56] presented a new parallel algorithm for CD with GPU device. It adopted a weighted-label propagation algorithm. Li [57] proposed a parallel version of Newman algorithm based on parallel computing for CD. The algorithm utilized the architecture of GPU architecture. In the relevant issue, Al-Ayyoub *et al.* [58] proposed a parallel algorithm for CD in social networks. The algorithm used dynamic parallelism based on GPU device. It provided three parallel implementations of Zhang *et al.*'s algorithm [37]: parallel CPU, hybrid CPU-GPU, and dynamic parallelism-based GPU. Mohammadi *et al.* [59] adapted the Louvain algorithm for CD in large CNs to perform in parallel with GPU devices. They used thread-level parallelism and shared memory with threads in the GPU block. They also proposed parallel implementations with hybrid CPU-GPU. In [60], Souravlas *et al.* proposed a parallel CD algorithm with hybrid CPU-GPU devices. The algorithm transformed the network nodes into a set of threaded binary trees. It first made the CPU to take samples of CN communities and represented them in the form of threaded-binary-trees, then the GPU read the large load data and sent it into a path matrix, finally the matrix was sent back to the CPU to analyze it and find communities. The results of the above algorithms showed that they achieved a good speed-up gain compared to non-parallel CD algorithms.

In short, according to the review in Section IV (A and B), we observed that great efforts have been devoted to solve the CD issue in CNs by applying conventional ML methods with both sequential and parallel implementations. However, these conventional methods could achieve local optimal solutions, which lead to a decrease in the quality of CD solutions. In addition, these approaches are highly dependent on the properties of the data. For example, spectral mapping methods adopt eigenvectors for CD, but they do not perform well on sparse networks. These methods also struggle in the face of today's complex data and increasing scaling of CNs and dimensionality of data [13]. Therefore, researchers were motivated by the success of DL in various fields and shifted their attention to developing powerful techniques to obtain effective and efficient performance for CD with feasible computational speed. [6], [9].

V. DEEP LEARNING MODELS-BASED COMMUNITY DETECTION

DL has become one of the most proactive research areas in artificial intelligence and machine learning. It has achieved remarkable success in many domains and applications, such

as speech recognition, image analysis and text comprehension [15], communications and networking [61], [62], etc., and CD is no exception. In this section, we review studies that address CD using DL models. The existing DL models in the field of CD are classified as follows.

A. STACKED AUTOENCODER-BASED COMMUNITY DETECTION

Tian *et al.* [17] proposed a CD method using the DL model, which is one of the earliest studies in this field. They designed a stacked autoencoder model for learning a nonlinear embedding of the graph. The model was developed by exploiting the similarity between the spectral clustering algorithm and the autoencoder model. The model took similarity, D was diagonal matrix and n nodes. The method then created a latent representation z with low-dimensional space $d, z \in R^{n\times d}$ The model was trained with the loss function that is calculated with Eq. (4). with respect to all samples m (i.e., nodes. It also consisted of five layers and adapted Kullback-Leibler (KL) divergence for sparsity besides loss reconstruct function. The new loss function was then introduced, as follows.

$$J_\theta = \frac{1}{m}\sum_{i=0}^{m}(x^i - y^i)^2 + \beta KL(P||\hat{P}), \tag{6}$$

where x and y denote the original input data and the reconstruction data, respectively, β controls the weights of sparsity, and p is a small constant value, e.g., 0.01 and $\hat{P} = \frac{1}{n}\sum_{j=1}^{n} h_j$, which is the average of activation of units in the hidden layer (h), and $KL(P||\hat{P})$ denotes the KL-divergence and is formulated as follows:

$$(P||\hat{P}) = \sum_{i=1}^{|\hat{P}|} p\log\frac{P}{\hat{P}_j} + (1-p)\log\frac{1-p}{1-\hat{P}_j} \tag{7}$$

Backpropagation algorithm and greedy layer-wise training process were used for training the model. Afterward, k-mean clustering algorithm was applied on the extracted latent representation for dividing the graph into groups (or communities). The results demonstrated that the model outperforms spectral clustering algorithm.

In the same direction of research work, Yang et al. [8] developed a stack autoencoder model for CD [63]. They adopted the modularity matrix $B = [b_{ij}] \in R^{n\times n}$ as an input of the model. Then, the stack autoencoder model was trained to create a useful hidden representation of the graph in low dimensional space. The model was extended to a semi-supervised by incorporating pairwise constraints. Similarly, a stacked and sparse autoencoder was developed by Wang *et al.* in [12] for CD. The proposed model encapsulated stack autoencoder as embedding stage and k-means as a clustering stage. It received a normalized adjacent matrix A as input. Fei *et al.* [64] proposed a method for CD using a deep sparse autoencoder model. In this method, a similarity matrix S was first constructed. Then, a sparse autoencoder was performed to find an effective and low-dimensional feature space of CNs. Continuing in the same direction, a CD method based on a denoising autoencoder was developed by Geng *et al.* in [65]. A probability transfer matrix T of a CN was first computed. Then, the transfer matrix was nonlinearly mapped to a new space by the denoising autoencoder model. Moreover, autoencoder and Convolutional Neural Network (CNN) models were combined for CD in [66]. The method extracted the spatial localization features by autoencoder and CNN models. The above methods used k-mean clustering algorithm to group nodes into communities. A set of experiments were conducted to evaluate the performance of the aforesaid methods with small-sized real CNs, and the results showed that the methods were promising for finding communities in CNs.

The autoencoder based on the integration of the network topology and content was suggested to solve CD problem. Cao *et al.* [67] proposed autoencoder that can utilize network topology and network content for such problem. Modularity B and Laplacian L matrices were used as input to the model. Graph regularization was also added to the proposed autoencoder to achieve robust integration of network content and topology, even though there was a mismatch in between them. In [68], the authors proposed an autoencoder model for CD based on the combination of topology and node contents. Both Markove M matrix and modularity B matrix were used as input data to the autoencoder. In the same direction of research work, Cao *et al.* [69] proposed a deep autoencoder model for CD by integrating network structures and node contents. The autoencoder obtained low dimensional space of the network representation. Modularity B and Markove M matrices were also used as input data to the proposed model. The experimental results showed that the methods gave a more robust performance than some popular CD methods.

Some researchers have also suggested other variants of the autoencoder for discovering communities in CNs. Xie *et al.* [70] proposed a transitive autoencoder model for CD. They calculated a new similarity matrix S based on eigenvectors and eigenvalues, and used them as input to the transitive autoencoder. The model was trained to obtain a low dimensional representation of a CN, then the communities were discovered with the k-mean clustering algorithm. More recently, Xu *et al.* [71] proposed a new method for CD using DL techniques. The method used four similarity matrices of CNs based on modularity, diagonal, transition operations, called *B, D, T, M*. All these matrices were fed to the model as source and target. Then, a stack autoencoder and transfer learning were combined to find an efficient low-dimensional feature space. Finally, several clustering algorithms were integrated to effectively find communities of CNs. The algorithms were evaluated on medium to large CNs and outperformed traditional methods in this area. Table 2 lists research publications relevant to autoencoders -based CD methods, as well as their merits and demerits and related properties.

TABLE 2. Summary of DL-based CD studies (Autoencoders).

Ref.	Methods	Input data	Net. size	Efficiency	Remarks ("+" refers to the merits and "-" to the demerits)
[17]	Sparse Autoencoder	$D^{1}S$	5K	Low	+The structure of the models was mentioned, and they outperformed traditional spectral clustering. - The cost of the models was not discussed, and it was evaluated with small datasets.
[8]	Stack autoencoder based modularity	B	2.7K		
[12]	Deep transitive autoencoder clustering	A	0.11K	Low	+ It performed more effectively than the spectral community detection algorithm. + The model cost was calculated and its structure was presented. - Performance analysis and comparison of the model is poor and has been evaluated on small datasets.
[64]	Sparse autoencoder	S	8K	Low	+ They outperform traditional ML methods in the field. - They require high memory, extensive trainable parameters. - No improvement in efficiency, and the evaluation was performed with small datasets.
[65]	Denoising autoencoder	T	1.2K		
[66]	Autoencoder-based CNN	A	0.1 K		
[67]	Autoencoder with the integration of network structure and contents	B, L	0.26K	Low	+ They can robustly combine network topology and node content, resulting in robust performance - Time cost not measured and discussed. - Autoencoder structure was not mentioned.
[68]		M, B	19K		
[69]		M, B	39K		
[70]	Transitive autoencoder	S	371K	Moderate	+ They outperform traditional and existing methods. + They can extract local and global network structure effectively. - Time complexity was not measured and discussed, and extensive trainable parameters were needed. - Many data representations are used that require high complexity in space and time.
[71]	Stacked autoencoder	B, D, T, M	1M		

B. GRAPH NEURAL NETWORK-BASED COMMUNITY DETECTION

Graph Neural Networks (GNNs) are a technical merging of graph mining and DL. The rapid development of GNNs in recent times is proof of their ability to model and capture the complex relationships between members of CNs. Chen *et al.* [19] developed a GNN model for supervised CD to perform as a node-wise classification. The model used an adjacency matrix A to exploit the information of edge adjacency by including the non-backtracking operator of the graph. Graph Convolutional Network (GCN) is a popular type of GNNs models for integrating graphs and supervised DL, which is developed following the success of CNN in other domains [72]. Jin *et al.* [18] proposed a method based on autoencoder and GCN for CD. The method incorporated network topologies and network contents. It consisted of encoder model, CD model, which is based on multinomial logistic regression, and structure reconstruction model. Similarly, Wang *et al.* [73] proposed a method based on marginalized autoencoder and GCN models for CD. GCN was mainly used for graph classification, the authors adapted GCN to a purely unsupervised clustering task by applying a stack autoencoder model. The approach integrated the content and the structure of the network and supported clustering tasks.

Recently, the GNN model was adapted to data-driven spectral analysis as well as CD in [74]. The method used a SBM and generic inference algorithms employing supervised learning for the data-driven process. In [75], a new GNN encoding method was proposed for the problem of CD in CNs. The method used a multi-objective evolutionary algorithm to solve this problem. In GNN encoding step, the edge in an attribute network was associated with a continuous variable, then non-linear transformation was used to transfer a continuous value into a discrete-values, which indicated communities of a CN. Sun *et al.* [76] also proposed a CD method based on GCN-autoencoder for clustering nodes. The method extracted the network embedding through the GCN autoencoder model. It also used the community structure to maximize the modularity index and grouped the network members with a clustering algorithm. Similarly, Park *et al.* [77] proposed a symmetric GCN autoencoder for learning the representation of a CN in unsupervised manner. The extracted new representation was utilized for node clustering.

The performance of the aforementioned methods [69]–[72] was evaluated on small-sized CNs, and they achieved good results in terms of accuracy and effectiveness. However, these methods encounter efficiency issues and are applicable to deal with small CNs only. Table 3 lists research publications relevant to GNN and GCN -based CD methods, as well as their advantages and related properties. properties.

C. OTHER TYPES OF DEEP LEARNING-BASED COMMUNITY DETECTION

Other variants of DL models were developed to solve the problem of CD in CNs. For example, Xie *et al.* [6] proposed a deep sparse filtering method for this task. In the method, a new network similarity representation $S\emptyset$ was proposed to denote direct and non-direct relationships between nodes.

TABLE 3. Summary of DL-based CD studies (GNNs and GCNs).

Ref.	Methods	Net. size	Input data	Efficiency	Remarks ("+" refers to the merits and "-" to the demerits))
[18, 19]	GNN and joint GCN Embedding	19K	*A, X*	Low	+ They have a good performance. + Effective performance comparison, analysis and interpretation of results were performed. - Time costs were neither computed nor discussed. - No improvement in efficiency.
[73]	Marginalized autoencoder based GCN	5.5K	*A, X*	Low	+ It achieved good performance compared to other methods in the field. of CD, and it integrated new variant of autoencoder to a GCN. + The time taken was calculated and discussed. - No improvement in efficiency, and only small datasets are included in performance evaluation.
[74]	GNN	58K	*A*	Moderate	+ It harvests good performance and can work with medium-sized CNs. - Time costs have not been introduced and discussed. - No improvement in efficiency.
[75]	GNN-based evolutionary	1.4K	*A, X*	Low	+ It uses multi-objective function to improve the method effectiveness, and outperforms traditional ML methods in the field. - It doesn't suggest improvement in efficiency (time, space complexity), and performs the evaluation with small-sized CNs.
[76, 77]	GCN-based autoencoder	3K	*A, X*	Low	+ They have the superiority performance. - They require large memory and extensive trainable parameters. - No improvement in efficiency, evaluation only with small-sized CNs.

Sparse filtering model was adapted to extract meaningful features of network and represent them in low-dimensional space. The k-mean clustering algorithm was used to divide the network into communities. The authors also incorporated new constrained similarities of pairwise nodes to the loss function of sparse filtering. In the same direction of research, Ye *et al.* [78] integrated the concept of non-negative matrix factorization into the layer structure of the autoencoder model. The encoder and decoder layers were incorporated in non-negative matrix factorization to form a unified lose function. The method used adjacency A and non-negative factor matrices U as input data. A convolution neural network CNN model was adapted for CD by Sperli [20]. The CNN model used adjacent matrix A as an input data representation. The convolution and max pooling layers mapped the similarity graph into the latent representation in a low dimensional space. A full connected neural network was performed on the new representation to assign nodes to communities. The method was trained with a supervised learning strategy to do a classification task. The mentioned methods were tested on a set of small-sized CNs, and the results showed that the they are promising to solve CD compared to conventional method.

In the relevant problem, Zhou *et al.* [79] proposed an approach that used an autoencoder to address CD based on local and global structure of graphs. In this approach, the graph was first divided into subgraphs and sent to the autoencoder model to extract graph features and represent them in a low-dimensional space. Finally, it executed Student t-distribution to refine initial K subgraphs clustering. Recently, in [80], the authors developed a new learning method for CD in CNs. They divided a given CN into several parts and chunks in order to bring them into the autoencoder model with low trainable parameters. In addition, the proposed method adopted a reduction and sharing of trainable parameters to improve the efficiency of deep autoencoder model for such a task. The method used $S\emptyset$ similarity representation as input data representation. Also, a parallel design was designed in the method. The above methods were evaluated with medium to CNs, and the results showed that they outperformed existing methods in the field. Table 4 lists research publications relevant to other variants of DL models for CD, including those dealing with large CNs, as well as their advantages and related properties are presented.

In a nutshell, according to the recent studies, we witnessed acceptable attention has been devoted to CD in CNs using DL models, especially with an unsupervised approach via applying autoencoder models. However, most of the existing methods have a challenge in terms of effectiveness, efficiency, and scalability. In terms of the efficiency and scalability, three issues are still open for improving better solutions: most of current methods confront low efficiency, where they do not operate in a setup with parallel CPU-GPU computing; the methods treat an entire CN as a single object; and they are usually evaluated on small CNs, except for a few recent methods. Therefore, this paper reviews the recent and relevant studies on DL with parallel approaches in Section VI to show how parallel approaches play an important role in developing efficient large-scale models. The aim of Section VI to highlight valuable solutions to overcome the challenges in developing efficient and large-scale DL-based CD methods. As for effectiveness limitations, all existing DL-based-CD methods use the gradient descent strategy with

TABLE 4. Summary of DL-based CD studies (other variants of models, including those dealing with large CNs).

Ref.	Methods	Net. size	Input data	Efficiency	Remarks ("+" refers to the merits and "-" to the demerits)
[78]	Nonnegative matrix factorization with autoencoder	19K	*A, U*	Low	+ It has effective performance compared to traditional methods, and the analysis was done effectively. - It has a high complexity and challenges to deal with large CNs.
[20]	CNN-based CD	50K	*A*	Moderate	+ It greatly reduced memory consumption. - It worked as a classification and has poor performance. analysis and comparison.
[6]	Deep sparse filtering	3.3K	*SØ*	Low	+ An effective representation of the CN was proposed. + The performance comparison was effectively analyzed. - The time cost was neither calculated nor discussed. - It was inefficient and was evaluated on small datasets.
[80]	Autoencoder-based CD with network partitioning	22K	*SØ*	Moderate	+ The method reduced the required trainable parameters. +It improved the efficiency using parallel and partitioning methods. - It was evaluated with non-large networks.
[81]	Degeneracy for graph embedding using autoencoder	1M	*G*	Moderate	+ It handled large graphs with sampling and propagation approaches. - It is a very tradeoff strategy conflict between effectiveness and efficiency. - It requires matrix-multiplications in large graphics, and leads computationally inefficient.
[79]	Multiple Autoencoders-based embedding and clustering	1M	*T, G*	Moderate	+ It extracted local and global network structure and scales for large graphs. - Time complexity was not measured and discussed. - Autoencoder structure was not mentioned. - Poor performance analysis and comparison.

the backpropagation algorithm for the training step, which leads to a fall into local optima (i.e., low effectiveness and poor CD quality) and slow convergence. Hence, an overview of combining MH algorithms with DL models is presented in Section VII to prove the positive impact of these algorithms in developing global and effective solutions for DL models and to motivate the research community to integrate MH algorithms into the DL and CD domain.

VI. PARALLEL DEEP LEARNING

In recent years, it is witnessed the emergence of big data, which has increased significantly, due to the rapid increase in most of the real-world data. As a result, the complexity of DNNs increases as the computational intensity and memory requirements of DL models increase significantly. The evolution of HPC devices helps to efficiently design large-scale DL models by utilizing parallel computing. Parallel computing plays an important role in designing efficient DL models.

In this section, we list the remarkable and recent studies related to DL with parallel computing in regular domain (e.g., images and speech) and irregular domains (i.e., graphs).

A. PARALLEL DEEP LEARNING ON REGULAR DOMAINS

In regular domains, the underlying data representation often has a low-dimensional space, a regular and clear grid structure, which is friendly to hardware accelerators such as the utilization of GPU [28]. Dean *et al.* [30] proposed a powerful framework, called Distbelif. In the framework, large-scale clusters of machines were used to distribute and parallel the training of Deep Neural Networks (DNNs). Two parallel architectures have been proposed: the first was implemented on a single machine using multithreading and the other was on multi-machines. In addition, three parallelism schemas were performed with this framework: model parallelism, data parallelism and pipelining parallelism. Two optimization methods were adopted in Distbelif; the first was named Downpour Stochastic gradient descent (SGD), which used multiple replicas of a single model (i.e., a dataset was divided into several partitions and a copy of the model was executed on each partition), while the second optimization called Sandblaster, which used distributed parameter storage and manipulation. Similarly, Ma *et al.* [82] proposed an autoencoder model for detecting outliers in large data. They divided the training data randomly into several parts and learned a representation by training a replicator autoencoder model using parallel-SGD for each part. Then the outputs of each replicator autoencoder were aggregated to detect outliers. The performance evaluation results showed that the above frameworks accelerated the training of models with large datasets.

Other studies also used GPUs to develop an efficient and scalable DL models. Chen and Huo [83] proposed a distributed Recurrent Neural Networks (RNNs) and DNNs for speech recognition applications based on a cluster of GPU devices. They leveraged data parallelism and intra-block parallel. Makkie *et al.* [84] presented a scalable and fast

distributed deep Convolutional Autoencoder (CAE) for functional magnetic resonance imaging (MRI) with big data analysis. CAE consisted of convolutional layers and max pooling operations. Data parallelism was leveraged to improve the efficiency of the proposed approach by distributing a copy of the entire model to all executer nodes, and each executer addresses sub-mini-batches via asynchronous SGD. In addition, Apache Spark and Tensorflow were utilized. Harlap *et al.* [85] suggested hybrid models in parallelism. They presented a parallel framework for DNNs, named PipeDream. It used data parallelism, model parallelism, and pipeline parallelism based on GPUs. A group of layers was divided into several chunks, and each chunk was assigned to a single GPU. Forward and backward tasks were processed in the chunk. PipeDream produced data parallelism through processing different mini-batches simultaneously; while, model parallelism and pipelining were conducted via overlapping forward and backward tasks. Experimental results demonstrated that mentioned methods were very efficient and obtained high speedup values.

More recently, Sriram *et al.* [86] proposed a DL model for multi-coil MRI reconstruction, called GrappaNet. The method could generate high quality reconstructions even at high speed-up factors. This was achieved by integrating parallel imaging methods into the DL model. In [87], Park *et al.* proposed a distributed DL training for CNN model. In the method, two parallelism schemas were utilized: data parallelism and model parallelism. In the same direction, Kim *et al.* [88] proposed a distributed DL method based on heterogenous systems. The schema was built by using multiple heterogenous GPUs that worked together. It also used data parallelism schema that worked via asynchronous large mini-batch training mechanism. Four types of GPUs were utilized for the evaluation process. The methods reaped a good speedup gains compared to baseline methods.

B. PARALLEL DEEP LEARNING ON IRREGULAR DOMAINS

As for irregular domain, the underlying data representation often has a high-dimensional space, an irregular and unclear grid structure, which makes it difficult to perform parallel computations [28]. Recently, large-scale DL models were developed in several studies from irregular domains with a supervised learning strategy (i.e., classification). In [89], Zeng *et al.* proposed a scalable GCNs model based on the combination of sampling and parallelization approaches. In the method, the graph was divided into sub-graphs and submitted each subgraph independently to the GCN model to perform in parallel. The sampling method utilized the frontier algorithm for the sampling approach [90]. In the training step was performed in parallel for each subgraph. In the same way, Chiang *et al.* [91] extended the variants of GCNs and designed a new variant of GCN that divided of the graphs into subgraphs. The partition task was performed as a preprocessed step by a clustering algorithm. Each a sub-graph was addressed either as a single batch or multi sub-graphs in a single batch. The experimental results showed that the methods achieved a good improvement in terms of memory and computational efficiency.

Ma *et al.* [28] proposed a new framework to support parallel neural networks computations to graphs. The authors investigated the capability of data partitioning, parallelism, and scheduling of the graph in neural networks in order to move beyond low dimensional regular grids. It also minimized the communication between GPU and host (CPU) and maximized the overlap computation. In the same direction, Liu *et al.* [92] presented a framework for GNN based on the GPU device. The framework aimed to improve the efficiency of GNN training by using parallel graph processing. In [93], Zeng *et al.* proposed a new parallel framework for GNN models. The proposed framework used a parallel sampling graph approach during the training process. Then, the main computation steps were parallelized on a shared memory system. Data partitioning was utilized in the framework to reduce memory traffic and improve cash utilization. Experimental results demonstrated that the methods achieve linear speedup values.

As for unsupervised learning, relevant works along this direction are still few. However, some authors have suggested deep parallel computing to develop large scale- DL models. For example, Bhatia and Rani [31] proposed a method for discovering overlapping clustering based on autoencoder model with distributed parallel computing. Hadoop platform involved 8 machines was used. An iterative-bulk synchronous parallel-based graph processing framework Giraph was utilized. The authors extended their work in [94], and applied the same idea, just added degree metrics to compute the importance of nodes during cluttering process. The authors tested their methods on large networks, and the results showed that the methods achieved an acceptable efficiency.

In a brief, based on the review in this section, we have found that parallel computing is seen as one of the important solutions for addressing large-scale DL models. In a regular domain, it is observed that a great effort has been devoted to developing large-scale DL models, but in irregular domains, e.g., graph, less attention is paid to build effective and efficient large DL models. However, there has been few works related to DL-based graph applications, especially in relation to the supervised learning techniques, e.g. classification using GCNs. However, GCN does not fit into the clustering task because the embedding driven from GCN is not oriented for CD [18]. In unsupervised DL-based graph applications, there are very few studies that use parallel approaches, for example, [31], [94] proposed parallel DL overlapping clustering. However, there are several limitations in these methods, such as: massive synchronization between nodes, since bulk synchronous parallel was used; the methods required extensive trainable parameters; they also did not use GPU parallelism, model parallelism and also data parallelism. Moreover, these DL models were optimized based on random walk and PageRank techniques, which focused on the local structure of the network and were considered conventional ML techniques. So, DL-based graph applications

TABLE 5. Summarization of recent and major studies in DL with parallel computing of regular and irregular domains.

Ref.	Methods and Enhancement Techniques	Resources	dataset	Domains	Applications	Efficiency
[30]	- Distributed parallel DNNs. Thread-level parallelism, distribute-level parallelism. - Data, model and pipeline parallelism.	6000 CPU-Machines	16 M	Regular (Image)	Classification	Very high
[82]	-Parallel Autoencoder for outlier detection - Data parallelism with Hadoop systems.	32 CPU-Machines	32 M		Outlier detection	High
[83]	- Distributed DNNs and RNNs. - Data parallelism.	64 GPU-Machines	1.8 K hour		Speech recognition	High
[84]	- Distributed CNNs with autoencoder. - Data and pipeline parallelism with Apache Spark.	16 GPU-Machines	9M		Analyzing task-based fMRI	High
[85]	- Distributed parallel DNNs. - Data, model and pipeline parallelism.	16 GPU-Machines	1.3 M		Visual recognition	High
[86]	- Parallel imaging with DNNs. - Data parallelism model.	8 GPU-Machines	1.59 K		MRI reconstruction	Moderate
[87]	- A distributed CNN - Data parallelism and model.	32 GPU-Machines	1.28 M		Classification	High
[88]	- Distributed DL-based on heterogeneous multi-GPUs. - It leverages data parallelism.	4 GPU-Machines	0.203 K		-	Moderate
[89]	- Parallel GCN based-sampling - Data parallelism.	40 cores-Xeon servers	1.4 M	Irregular (Graph)	Classification	High
[91]	- GCN based dividing graph - Data parallelism.	20 cores-Xeon servers	2.4 M		Classification	High
[28]	- GNN based-data partitioning - Data parallelism	28 cores-Xeon servers + 8 GPUs	8 M		-	High
[92]	- GNN training based on CPU-GPU. - Data parallelism.	10 cores-Xeon servers + Tesla GPU	19.7 K		Classification	Moderate
[93]	- Parallel framework for training of GNNs. - Data parallelism, data sampling, partitioning.	40-core Xeon	1.59 M		Classification	Moderate
[31],[94]	-Autoencoder based parallel and distributed computing. - Data parallelism.	8 CPU-Machines	4 M		Overlapping Clustering	High

(e.g., DL-based CD in CNs) needs further investigation to design efficient and scalable DL models. Table 5 provides summarization of recent and important studies in the area of DL and parallel computing of regular and irregular domains.

VII. META-HEURISTIC-BASED OPTIMIZATION METHODS

MH is a concept of a set of algorithms including evolutionary algorithms such as Genetic Algorithm (GA) [24], and natural inspired algorithms such as Particle Swarm Optimization (PSO) [23]. MH algorithms have been successfully applied for optimizing machine learning models and are known as efficient methods to solve complex problems and find optimal solutions in a suitable lapse of time. They are now state of the art methods for solving various optimization problems, especially for those that are very complex and high-dimensional spaces, and the detection of communities using DL in CNs is no exception [25]. In this section, first the popular MH algorithms are presented and then conventional studies-based MH algorithms that have developed for CD (i.e., without deep learning models), and finally the methods that have used for training deep neural network models.

A. COMMON META-HEURISTIC ALGORITHMS

GA was developed in the early 1970s [95]. It is the most popular and most commonly-used evolutionary method. GA uses of a binary data representation. It can also support other types of data representations. GA comprises four key parts: chromosomes (i.e., means solutions), crossover, selection, and mutation, and fitness procedures. All these procedures work to get optimal or near optimal solutions. In the crossover process, the offspring are inherited and formed from two parents. In the mutation phase, one or more chromosomes

are randomly selected and altered to increase the diversity of the population [96]. The algorithm was applied to optimize solutions to problems in various domains and applications, e.g., CD in CNs [97], classification in images [98], and DL applications [99].

PSO algorithm is a population-based optimization algorithm [23], [100]. It uses a stochastic optimization technique and emulates the behavior of bird swarms to find the solution to the optimization problem. Many particles are randomly created over the search space to form a swarm. Each particle is generated in such a way that it represents a solution candidate for an optimization problem. The particle moves in the search space with a specific velocity. The particles can go back to the best prior position according to their memory, which maintains the previous best position. PSO has been successfully used to provide solutions to optimization problems in various fields, e.g., CD in CNs [101], [102], images [103], wireless sensor networks [104], etc.

Ant Colony Optimization (ACO) is a population-based MH that can be used to find nearly optimal solutions for difficult optimization problems. It was developed in 1992 by Dorigo [105] based on the behavior of ants. ACO was inspired by the behavior of ants in their search for food to find the optimal path in the search space to solve an optimization problem. Each ant first explores the search space near its nest to search for food at random. When the ant returns to the nest, the path is traced by chemical pheromones. This helps other ants to find the shortest path to the food source through chemical pheromones left on the ground. ACO has been utilized to a variety of optimization problems in various domains and applications, e.g., CN applications [106], DL applications [107], internet-of-things applications [108], etc.

Differential Evolution (DE) algorithm was introduced in 1996 by Storn and Price [109]. It is a method that optimizes a problem by trying iteratively to improve a candidate solution with respect to a given evaluation function. The optimization process of the algorithm starts with the random initiation of a solution candidate. Then new individuals are created by crossing and mutation in the evolutionary process for each generation. The target individual and the mutated individual are recombined to create the trial individual containing a useful solution from the prior generation. Some DL and image applications have been successfully addressed using the DE algorithm [110], [111].

The Artificial Bee Colony (ABC) is a swarm based on MH algorithm introduced in 2005 [112]. The algorithm was developed inspired by the natural performance of the bee colony. The bee colony method can be effectively used for the design of intelligent system models as it includes several features such as foraging, communication, task selection, group decisions, etc. The model has three main components: employed and un-employed forager bees, and food sources. Employed and un-employed forager bees look for plentiful food sources near their hive. The model also defines two guiding behaviors required for self-organization and group intelligence: the recruitment of feed gatherers for rich sources of food, leading to positive feed-back, and the abandonment of poor sources by foragers, leading to negative feed-back. The ABC algorithm was used for optimization in various fields, such as image and DL applications [115], [116].

The Firefly Algorithm (FA) is a new MH algorithm based on the flashing patterns and behavior of fireflies. It was developed by Yang in 2007 [117]. The brightness of FA is the objective function. The attractiveness in FA depends on the brightness, so the less bright firefly will move towards the brighter one. If it is not brighter, the movement will be random. FA has been used for CN applications [118], image and DL applications [119].

Cuckoo Search (CS) is a new MH optimization algorithm introduced in 2009 [113]. The algorithm uses the breeding parasitism of some cuckoo species together with Levy flights that run randomly to solve optimization problems. CS has been used for optimization in various domains and applications, e.g., CD in social networks [120], classification and prediction in images [121], [154]. Bat Algorithm (BA) is a recent MH algorithm for global optimization. It was developed by Yang in 2010 [114]. The algorithm is inspired by the echolocation behavior of micro-bats, which involves the use of varying pulse rates of emission and loudness. Echolocation is used to determine distance, and all bats know the difference between prey and background barriers. BA was applied to solve optimization problems in various fields, e.g., CD in CNs [122] and DL applications [123].

Readers can refer to a comprehensive study that is presented in [124] about the state of art of MH algorithms. The important advantages and disadvantages of these MH algorithms are summarized in Table 6.

B. COMMUNITY DETECTION BASED ON META-HEURISTIC ALGORITHMS

In today's world, conventional ML studies-based MH algorithms have received much attention to address the CD problem, and the literature has expanded with diverse applications. Important and current methods of CD using MH algorithms are presented in this section.

In [97], Said *et al.* proposed an algorithm for CD-based GA in CNs. The crossover and mutation operations were iterated until the stopping conditions are met, and selected the best solutions. The algorithm used a clustering coefficient-based GA to generate the initial population and mutation method, and these improved its efficiency and accuracy. It used the modularity function as a fitness function. Likewise, Zaire and Meybodi [125] suggested a method to find communities in CNs based on the integration of GA and object migrating automata. The method sought to maximize modularity function. In [126], Rostami *et al.* developed a new CD based on GA for feature selection. The method consisted of three steps: the features were computed and selected, then these features were clustered; finally select the best clustering by GA.

The PSO algorithm is also applied to solve the community discovery problem. In [127], Cai *et al.* developed a PSO

TABLE 6. A simple comparison of popular MH algorithms.

Algorithm	Merits	Demerits
GA [95]	The complexity of the algorithm can be reduced by parallel computing.	The exact solution is not always guaranteed.
PSO [23, 100]	It is simply implemented and has a few parameters to set. Can perform parallel computations, and converge quickly. Have higher probability and efficiency in finding the global Optima. Can be efficient for solving problems that have difficulty. to find accurate mathematical models.	It is a challenge to define initial parameters. May converge prematurely and be stuck in a local minimum, particularly in the case of complex problems.
ABC [112]	It adapts to the greedy heuristic search, solves the problem quickly and is able to find global convergence	The initial solution restricts the search space.
FA	It flexible to address continuous problems, classifications, and clustering, and combinatorial optimization.	Slow convergence speed and high possibility of being trapped in local optimum.
DE [109]	It has limited hyper parameters. It also used for multi-objective optimization.	It has fallen on early convergence and its parameters are fixed.
ACO [105]	It is valuable algorithm for optimizing parameters (i.e., weight).	Unstable time convergence.
CS [113]	It has flexible parameters that lead to an efficient and effective optimization.	the algorithm is inefficient, it has a slow convergence.
Bat [114]	It is a newer and more promising algorithm due to its good results compared to other algorithms.	It is inefficient in terms of storage space.

algorithm to find communities in social networks. The PSO particles were adapted to a discrete scenario. The update rules used by the method depend on the greedy strategy and the network topology. It found communities in large social networks. A discrete PSO-based multi-objective optimization algorithm was developed by Gong *et al.* [101] to discover communities in CNs. The method attempted to minimize the two objective functions: Kernel k-means and Ratio cut functions. It dealt with signed and unsigned networks. In [128], Liu *et al.* presented a multi-objective PSO algorithm for extracting network embedding and finding communities. The method mapped the nodes in a low-dimensional space. Thus, this led to an increase in search efficiency as the search space was reduced. Another variant of PSO algorithm was developed for CD in [129], called Sigmoid Fish Swarm Optimization (SFSO). The algorithm used the sigmoid function for different fish movements in a swarm. It outperformed the traditional PSO algorithm in this task.

The Firefly algorithm (FA) has been adapted for CD in CNs. Amiri *et al.* [118] developed a CD method based on FA optimization. The method used multi-objective optimizations so that a set of solutions with Pareto optimum can be obtained. The parameters were tuned based on self-adaptive probabilistic mutation and a chaotic mechanism. Similarly, Del Ser *et al.* [130] proposed a CD method based on FA algorithm. It emulated the behavioral patterns of fireflies that attract each other while flying, which was used for numerical optimization. In [131], Jaradat and Hamad also used an FA optimization algorithm to solve the CD problem. They also proposed a parallel approach to perform this task. The above FA-based CD methods have been evaluated in real and synthetic CNs, and their results were promising for this task.

BA is another MH algorithm that has been used to develop and improve CD methods. In [132], Hassan *et al.* proposed a new CD method based on discrete BA optimization. The method used the modularity function as a fitness function to maximize. It did not require prior information about the number of communities, which can be inferred based on the locus-based adjacency coding scheme. Likewise, Song *et al.* [133] solved the CD problem with a discrete BA optimization. The method found the global optimal solution, and it automatically determined the number of communities. Doush *et al.* [122] also adapted multi-objective BA optimization to address the CD problem. The concept of Pareto dominance was used to select the optimal solution in multi-objective optimization. The results of the above methods showed that BA optimization is able to find the communities in CNs, and showed a better performance of the CD.

ACO algorithm has also attracted the attention of researchers to address the CD problem. In [134], Chen *et al.* proposed an algorithm based on ACO to solve this problem. The algorithm used artificial ants traveling on a logical digraph to generate CD solutions. Each ant chose a path according to the heuristic information about each path. The degree of association was used as heuristic information. Similarly, Guo *et al.* [135] adapted ACO to find communities in CNs. The algorithm included initialization, and three various of searching; employed bee, onlooker, and scout bee. The CD solutions were first initialized and then tuned and addressed by the three search phases. In the same direction,

the multi-objective ACO algorithm has been adapted to discover communities in CNs in [106]. Two objective functions were used: community score and community fitness, which measured the density of groups and minimized the external relations, respectively. During the process of the algorithm, a Pareto was considered to store non-dominated solutions. The results showed that these above algorithms successfully detect community structures and competitive CD performance was achieved.

CS algorithm has also been successfully used to solve the CD challenge. Zhou *et al.* [136] proposed a discrete multi-objective CS algorithm for CD. Two objective functions were minimized, namely negative ratio association and ratio cut. It found high-quality communities without prior information. On the same topic, Babers and Hassanien [120] proposed a CS algorithm for CD in social networks. The method used the modularity function as an objective function. The locus-based adjacency scheme was used to represent individual solutions in the network and community structure. The results of the mentioned algorithms show that the CS algorithm is promising to solve the CD problem.

In a nutshell, remarkable efforts have been made to develop conventional ML-based MH algorithms for CD. These algorithms have global search capabilities and good local learning, and can deal with a wide range of CD problems. Moreover, they can automatically determine the number of communities and they can be implemented in parallel and efficiently. However, they have shortcomings of achieving low accuracy and efficiency, especially when dealing with high dimensional and complex data, such as large CNs [14]. Therefore, the development of effective and efficient algorithms is of greater interest and worth considering, especially when dealing with large and complex data. Therefore, in the next subsections, the studies on the integration of MH algorithms and DL models in different domains are reviewed to offer interested researchers the opportunity to bridge the existing gap in the CD field.

C. META-HEURISTIC ALGORITHMS INCORPORATING WITH DEEP LEARNING

Recently, several studies have proposed MH algorithms for optimizing parameters and hyperparameters of DL models to create effective and efficient models. The success of the MH algorithms in optimization tasks has prompted the research community to solve large and difficult problems. In the field of DNNs, the Gradient Descent (GD) is replaced by iterative MH algorithms for tuning parameters, e.g., weights. The main purpose of using the methods is to overcome limitations of gradient descent methods (i.e., Back-propagation (BP)). Three shortcomings exist in GD-based BP algorithm: training process is slow, especial with big data, sensitive to parameter initiation and has fallen to local minima, which leads to a drop in performance [21], [22]. In this subsection, we outline the recent and important studies that used MH algorithms for training and optimization of DL models.

1) DEEP LEARNING BASED ON META-HEURISTIC ALGORITHMS

An earlier study adapting GA for training Artificial Neural Networks (ANNs) was proposed by Leung *et al.* in [137]. The authors developed an improved version of GA to train neural networks. All functions of GA were redefined, including crossover, mutation operations and fitness function. The authors showed that the parameters of the neural network can be efficiently tuned using the improved GA algorithm. In the same research direction, recently, GA was adapted to optimize the structure and hyperparameters of deep CNNs in [138]. The proposed method was evaluated for the amyloid brain images dataset for disease diagnosis. Pan *et al.* [98] also adapted GA to optimize deep CNNs for classification in multi-unmanned aerial vehicles. GA obtained the scenario states and path segments to train CNNs. Then, the CNN reproduced the path planning resulting from GA's experience. Similarly, David and Greental [99] incorporated GA in a deep autoencoder. The GA was used to optimize parameters (i.e., weights) of autoencoder. The proposed method overcame the problem of tied weights of encoder and decoder models to enhance the accuracy of classifications. The experimental results indicated that the GA algorithm was promising to optimize DL models.

PSO algorithm was adapted for optimizing DL models. Gudise and Venayagamoorthy [139] developed a PSO-based population algorithm to train ANNs [140]. The fitness value of each particle is the value of the cost function, which was evaluated by the current position and corresponded to the weight matrix. The authors presented a comparative study of training neural network based on PSO and BP algorithms. Recently, Rajagopal *et al.* [103] proposed a deep CNNs for scene classification in unmanned aerial vehicles. The authors developed the optimization of the model depending on a multi-objective PSO algorithm. The method allowed the vehicle to acquire videos, then the pre-processing step was applied to the videos. Then the training step was performed with the CNN-based multi-objective PSO model. Finally, the images were classified. In [141], Band *et al.* proposed a new DNN approach for modelling gully erosion susceptibility. The approach used PSO optimization in the training step. The experimental results showed that PSO was effective and efficient for training DL models.

Mavrovouniotis and Yang [107] incorporated ACO algorithm in feedforward neural network models. They proposed two optimization algorithms: firstly, the model was trained with a stand-alone ACO optimization algorithm, secondly GD-based BP and ACO were combined for an optimization process to train the model. Both optimization algorithms were evaluated on several datasets for pattern classification. The result showed that the ACO algorithm could achieve efficient performance especially when integrating ACO and GD. Recently, Zhang *et al.* [142] proposed DNN-based ACO optimization to forecast the cost of mining projects. BA was also adapted to optimize ANNs by Jaddi *et al.* [123]. It optimized the structure of and the parameters of ANNs models.

The algorithm improved the effectiveness and reduce the complexity of such models. The authors also proposed two modifications of the bat algorithm (namely MBatDNN and MeanBatDNN) to improve optimization process. The results demonstrated that the proposed methods outperformed traditional optimization algorithms.

Other MH algorithms were also adapted to train ANNs. For example, Karaboga *et al.* [115] suggested to train ANNs based on ABC algorithm. It was the first study that trained ABC based on neural networks. Recently, In [116], ABC was combined with DL model to detect the pattern of COVID-19 patients. In this method, the deep CNN model was used to extract the features of the X-ray images, then the ABC algorithm refined the features to select the best ones. CS algorithm [143] was also adapted to optimize ANNs structure and parameters to improve the accuracy and the efficiency of the model. In [144], Cristin *et al.* proposed a DL method for plant disease determination and prediction. The method used a Deep Belief Network (DBN) model for this task, and the model was optimized by integrating Reiter optimization algorithm and CS algorithm. Another algorithm was proposed by Zhou *et al.* [145] to optimize DNNs, called water wave optimization (WWO). The method was used to solve the high-dimensional optimization problem. Similarly, FA algorithm was proposed by Strumberger *et al.* [119] to select optimal CNNs stricture. The hyperparameters were tuned by FA, including number of layers, kernels, kernel size. In the same research direction, Rere *et al.* [146] used Simulated Annealing (SA), which was a single solution-based algorithm [147] for optimizing CNNs models. The experimental results showed that the above-mentioned methods achieved good performance and were more effective than other optimization techniques in regular domains (e.g., Images-classification). Table 7 lists research publications relevant of DL-based standalone MH optimization algorithms.

2) DEEP LEARNING BASED ON HYBRID META-HEURISTICS AND GRADIENT DESCENT ALGORITHMS

Several studies have suggested an optimization process by integrating MH algorithms with GD-based algorithm. Bakhshi *et al.* [121] proposed GA to explore a suitable CNN architecture tune hyperparameters such as learning rates, number of layers. In the method, the hyperparameters and parameters (i.e., weights) were optimized by GA and BP algorithms, respectively. Lander and Shang [110] proposed a new evolutionary framework (called EvoAE) to optimize autoencoders. EvoAE evolved a population of autoencoders and searched for structures and features at the same time. It learned different features from large datasets and reduced training time. The EvoAE generated new autoencoder models from crossover and mutation operations with chromosomes. It also supported training with BP algorithm.

In the same research direction, in [111], Sui *et al.* incorporated PSO into marginalized stacked denoising autoencoder for tuning parameters. PSO could hold the best configuration of the marginalized autoencoder. Similarly, Silhan *et al.* [148] suggested an approach to tune the hyperparameters of stack autoencoder model by using evolutionary algorithms. The results showed that the methods could find the best hyperparameters, but the training time of the model increased approximately threefold. Similarly, Zhang *et al.* [149], integrated between PSO and BP to work together for neural network training. This integration aimed to take advantage of both algorithms, with PSO performing a global search at initializing phase of the weights. The BP algorithm then performed a local search around the global optimum. The experimental results showed that the combination of the algorithms was better in terms of efficiency and effectiveness than the application of standalone algorithm, PSO, or BP.

Yi *et al.* [22] proposed an incorporation method between BP and CS algorithms for the optimization of regression neural network model. CS assisted BP in weigh initialization. In addition, FA [150] was adapted to optimize the parameters (i.e., weights) and thresholds between the input layer and the hidden layer of neural networks. Similarly, Rojas-Delgado *et al.* [151] also proposed an approach that included three MH algorithms: PSO, FA and CS algorithms. They have added a continuation method to the above-mentioned algorithms, where the problem was solved by moving it progressively from the simple to the actual problem. The approach was evaluated using public benchmark datasets and the experimental results showed that continuation method of the three studied MH algorithms could reduce the execution time in by 5–30% as compared to standard MHs.

More recently, there are new hybrid optimization of gradient descent and MH algorithms. Yadav [152] developed a hybrid optimization of ANNs models for medical diagnostic applications. The method integrated PSO and GA algorithms to BP algorithm with Adam optimizer. The PSO and GA algorithms overcame the shortcoming of the BP algorithm. In [153], the PSO algorithm was also integrated to the BP algorithm to perform hybrid optimization for wind power prediction. A good comparison was made with BP and GA-PB algorithms. The PSO-BP outperformed the others. In [154], the PSO algorithm was combined with the GD-BP algorithm to train a radial basis function (RBF) neural network. The purpose of this combination was to take advantage of the two in the meantime. Similarly, Mohapatra *et al.* [155] integrated PSO to BP-Adam optimizer to train ANNs for mathematical equivalence of error gradients. Seifi and Soroush [156] proposed an ANN-based MH algorithm for climate prediction. The method used three MH algorithms: GA, Whale Optimization Algorithm (WOA) and Grey Wolf Optimization (GWO) algorithms with ANN models for the estimation process. Tran-Ngoc *et al.* [157] also developed ANNs for damage detection based on a hybrid MH optimization algorithm. The method combined GA and CS algorithms to quickly find the best solution and avoid local solutions. In addition, the method used a vectorization technique for the objective function to reduce the computational cost. The

TABLE 7. Summarization of studies in deep learning-based standalone MH optimization algorithms.

Ref.	Methods	Applications	Remarks ("+" refers to the merits and "-" to the demerits)
[137]	ANNs using GA algorithm	Benchmark-dataset-prediction	+ It tunes the parameters and structure of neural networks. + It is more effective than the BP algorithm. - Poor performance analysis and comparison. - Neural network structure used was simple.
[138] [98, 99]	Deep CNNs, autoencoder-based-GA algorithm	Images-classification	+ They achieve superior performance on a classification task. - Time cost was not measured clearly. - Poor performance comparison.
[139]	ANNs using PSO	Synthetic dataset	+It is more efficient than the BP. + Training recurrent neural networks has been performed. - Neural network is messed, and poor performance comparison was done.
[103]	Deep CNN-based multi-objective PSO algorithm	Images-classification	+ The method is effective, and it uses multi-objective optimization. + It requires less computation time. - Neural network is messed, and poor performance comparison was done.
[141]	DNN-based PSO algorithm	Images-prediction	+ It shows good performance. - Time cost was not measured clearly, and DNN structure is not clear.
[107]	ANNs using ACO and BP algorithms	Benchmark-dataset-Classification	+ Hybrid ACO-BP shows superior performance. + It can deal with large dimensional problems. - ACO Standalone does not achieve a good result. - The structure of the neural network is simple.
[142]	DNN-based ACO algorithm	Benchmark-dataset - Forecasting	+ DNN-based ACO harvests superior performance. + Good performance comparison and analysis. - Time cost was not measured clearly, and DNN structure is simple.
[123]	ANNs using Bat algorithm	time series datasets	+ It can successfully predict the future values. + It is also less complex than some existing optimization algorithms. - It handles low-dimensional data, and cannot extend to large data.
[115]	ANNs with ABC algorithm	Benchmark-dataset for Classification	+ It is an early study using the ABC for training neural networks. + It outperforms the BP and GA algorithms. - It is approached with a very simple neural network.
[116]	Deep CNNs with ABC algorithm	Images-pattern recognition	+ It achieves good recognition accuracy. - Neural network is messed, poor performance comparison. - Time cost was not measured clearly.
[143]	ANNs with CS algorithm	Video- emotion classification	+ It provides accurate and fast classification. - Poor performance analysis and comparison. - Time cost was not measured clearly.
[144]	DBN-based Rider-CSA algorithm	Images-classification	+ It outperforms other existing methods in the same field. - Neural network is messed, Poor performance comparison. - Time cost was not measured clearly.
[145]	DL with WWO algorithm	Benchmark-dataset-classification	+ It Performed better than BP. + It can solve high-dimensional optimization problems. - It has high consumption time compared to BP.
[119]	CNNs-based FA algorithm	Images-classification	+ It achieves promising performance in images classification. - It tunes only hyperparameters. - Time cost was not measured.
[146]	CNNs-based SA algorithm	Images-classification	+ It optimizes CNNs effectively. - Poor performance analysis.

TABLE 8. Summarization of studies in DL-based hybrid gradient decent and MH optimization algorithms.

Ref.	Methods	Applications	Remarks ("+" refers to the merits and "-" to the demerits)
[121]	CNN with GA and BP	Image- classification	+ It can explore CNN structure effectively. - It requires high computation time. - Poor performance comparison and analysis.
[110]	Autoencoders based hybrid evolutionary-BP (i.e., EvoAE)	Image-dataset for classification	+ Training time of autoencoder is reduced. + A population of autoencoders can be executed simultaneously in parallel. - Poor performance comparison, and model structure was not mentioned. - CS is only used for initialization.
[111]	Marginalized denoising autoencoder using PSO	Images- feature extraction and classification	+ It achieves an improvement in accuracy compared to other autoencoders. + It can reduce the training time. - Poor performance comparison, and time costs were not introduced.
[148]	Stacked Autoencoder with evolution algorithm	Images- feature extraction and classification	+ It effectively finds suitable sets of hyperparameter values. + It improves the performance of the model. - The model loses 3-time efficiency.
[149]	Optimization of neural networks with hybrid PSO-BP	Benchmark data for classification problem	+ It gets higher training accuracy than the PSO and the BP. + Time cost was computed, and it performed better than the BP. - Neural networks structure was not mentioned. - Performance comparison to other MH algorithms is low.
[22]	ANNs-based CS-BP algorithms	Benchmark datasets- regression	+ It trains fast, can obtain the global optimal solution. - Neural networks structure was not mentioned.
[150]	ANNs-based FA_BP algorithms	time series- forecasting	+ It has superior performances for forecasting task. + Effective performance analysis and compassion. - Time and space consumption were neither calculated nor discussed.
[151]	ANNs-based PSO, FA, CS and continuation algorithms	Benchmark datasets - prediction	+ A continuation approach improved the model efficiency significantly. + Time consumption was calculated and discussed. - Poor performance analysis and comparison.
[152]	ANN-based on PSO-GA-BP algorithms	Images- medical diagnosis	+ The hybrid optimization outperformed the traditional one. + It uses multi-heuristic algorithm for optimization. - Poor performance comparison, and time cost was not measured.
[153]	ANN-based on PSO-BP algorithms	wind power- forecasting	+ The hybrid optimization outperformed the traditional one. + Good performance comparison, and time cost was not measured.
[154, 155]	ANNs-based PSO-BP algorithms	-	+ The method is superior to traditional ANNs. - It has probability to fall into the local minimum. - Poor performance comparison, and time cost was not measured.
[156]	ANNs-based GA, GWO WOA algorithms	Climates- estimation	+ They achieve a good level of accuracy, and have a good performance analysis. - Time cost was not measured.
[157]	ANNs-based GA-CS	Damage-detection	+ The method is superior to traditional ANNs. + It is effective and efficient, and time cost was measured. - It deals with simple ANNs, and no deep ANNs.

above hybrid optimization algorithms were compared to traditional GD-BP optimizers, the theory and algorithmic performance were supported by promising results. Table 8 lists research publications relevant of DL-based hybrid GD and MH optimization algorithms.

3) LARGE SCALE DEEP LEARNING-BASED META-HEURISTIC ALGORITHMS

A few MH algorithms and DL techniques are used for the analysis of big data. Hegde and Mundada [158] presented a method for the analysis of big data based on PSO and DL. The meaningful features were extracted by PSO algorithm, and DL model was applied for the classification task. The aim was to produce high classification accuracy. Cao *et al.* [159] extended the combination of PSO and BP for building a distributed parallel computing of them and improving the performance and efficiency. The parallel model was designed on MapReduce on Hadoop platform. The proposed model was evaluated on images classification. The result showed that the proposed method received high classification accuracy and efficiency. Recently, Saranya and Nagarajan [160] proposed a large-scale MH optimization for DNNs training. The model was used to deal with large data for predicting agricultural yields using the Hadoop framework in parallel. The model was evaluated on large data from plant images.

Coelho et al [161] proposed DL-based MH algorithm for time series forecasting with GPU device. With this method, a multithreaded GPU-based strategy was developed

TABLE 9. Summarization of studies in DL-based MH optimization methods with big data.

Ref.	Methods	Applications	Remarks ("+" refers to the merits and "-" to the demerits)
[158]	DL- based PSO for the analysis of Big Data	Feature extraction from big data	+ It can examine large amounts of data to extract valuable information effectively. - The method lacks an efficient technique that supports the handling of big data (i.e., parallel computing). - Time and space were not discussed.
[159]	Parallel PSO-BP with Hadoop for training ANNs	Scene-image-dataset for Classification	+ It has the ability to process and analysis the big data. + It is scalable and can extend to several machines to run in parallel. - The size of the evaluated dataset and the dimensional feature space is small. - PSO algorithm cannot find the global optimal solution quickly and simply.
[161]	ANNs-based MH using GPU	Electricity dataset for time series forecasting	+ It is scalable and support massive parallelism based-GPU. + It used the GPU global memory. - Information about the MH algorithm used is not provided. - It depends only on level-thread parallelism (i.e., only one machine).
[160]	ANNs-based MH using Hadoop	Images- agricultural yield prediction	+ It optimizes neural networks effectively and efficiently. + It processes large data using parallel computing. - It lacks information about efficiency (i.e., time and space complexity).

to improve the efficiency of the DL-based MH model. The experimental results showed that the proposed method was scalable and efficient and achieved with GPU high speed-up as compared to a sequential implementation with CPU. Table 9 lists recent studies about the application of DL-based MH optimization methods in big data research.

In a short, it is clearly shown that the incorporation of MH algorithms with DL models is attracting increasing attention in the literature. We observe that MH algorithms play an important role in improving the performance and efficiency of DL models. Some of them work as stand-alone optimization algorithms, others are combined with gradient descent optimization (i.e., BP algorithm). Most of the existing algorithms could perform with low accuracy and efficiency, especially when dealing with large and complex data. Therefore, the development of effective and efficient DL-based metaheuristic algorithms is of a greater interest for dealing with large and complex data. To our best knowledge, no study was suggested to apply DL and MH algorithms for performing community detection.

VIII. OPEN DATASETS FOR COMMNITY DETECTION

Open datasets are available to encourage and facilitate research in CD. These datasets can be categorized into two groups: synthetic datasets and real datasets.

A. SYNTHETIC DATASETS

Synthetic dataset is artificially produced rather than collected from real events. In the field of CD, there are two widely used synthetic networks with known community structure, i.e., the Girvan Newman (GN) [162] and Lancichinetti–Fortunato–Radicchi (LFR) [163], [164] synthetic networks. The GN is a small synthetic network consisting of 128 nodes divided into four communities where 32 nodes belong to each community. Each node in the GN network has an average of 16 relations, including relations with nodes in the same community and in different communities. The LFR network is more complicated than the GN network. It has important properties like real world networks, such as scaling of network size and heterogeneity in the distributions of node connections and community sizes. These distributions can be tuned according to a power law with different exponents. The LFR is a popular benchmark synthetic dataset in community detection than others because of these good properties.

B. REAL-WORLD DATASETS

Real-world datasets are more effective and accurate benchmarks for evaluating the performance of algorithms for CD. In this subsection, we introduce real-world datasets from 27 remarkable and common open real networks, which are classified into four categories: social networks, citation networks, collaboration networks, and others. These datasets are listed in Table 10. All ten datasets (numbered from 1 to 10 in Table 10) consist of numerical records of individuals and their relationships in a social network. The eight datasets of Citation networks (numbered from 11 to 18 in Table 10) consist of records for papers or patents and their relationships, such as citations. The four datasets from collaboration networks (numbered from 19 to 23) comprise relationship records about scientists and their collaborations. Another four datasets (numbered from 24 to 27) are the records from other types of community network. The size of these datasets ranges from small to large in each category in terms of the number of nodes and edges.

IX. SUMMARY AND CHALLENGES

The review first presented the important conventional methods proposed to solve the problem of CD in CNs, i.e., without using DL techniques in Section IV. It is shown that great efforts have devoted to the CD issue on CNs by applying early

TABLE 10. Information of real-world datasets.

Categories	Index	Datasets	#Nodes	#Edges
Social Networks	1	Karate [165]	34	78
	2	Dolphins [166]	62	159
	3	School6 [167]	68	220
	4	school7 [167]	68	220
	5	Football [162]	115	613
	6	Facebook [168]	4039	81800
	7	Wikipedia [169]	7100	107000
	8	Artists [170]	50515	819306
	9	ComputerBib [171]	317080	1049866
	10	Youtube [172]	1134890	2987624
Citations Networks	11	Small-hep[1]	397	812
	12	Pol-blogs [173]	1490	16,718
	13	Cora [174]	2708	5429
	14	Citeser [174]	3312	4732
	15	Large-hep1	11752	134956
	16	Pubmed [175]	19729	44338
	17	arXiv	576000	6640000
	18	US patents [176]	3700000	16500000
Collaboration Networks	19	Net-science [177]	379	914
	20	Computer-science [178]	22000	96800
	21	Engineering [178]	14900	49300
	22	Chemistry [178]	35400	157400
	23	Medicine [178]	63300	810300
Other Networks	24	Pol-books [63]	105	441
	25	Amazon [179]	410236	3356824
	26	Web-Google [180]	875713	5105039
	27	RoadNet-PA[2]	1088092	1541898

conventional methods as presented in Subsection IV (A). However, most of these methods face challenges in dealing with large network efficiently. Then, efficient methods that adopted parallel computing for handling such task were introduced in Subsection IV (B). However, these conventional methods could not perform CD effectively and efficiently on CNs where the datasets have features in high dimensionality and diversity. In addition, they have high probability to get into local optimal solutions and struggle in the face of today's complex data and increasing scaling of CNs and dimensionality of data [13]. Hence, researchers were motivated by the success of DL applications in several fields and recently moved their attention to investigating effective and efficient solutions using DL models, especially with the use of autoencoder models for CD.

Section V presented the literature proposed for DL-based solutions for CD. After review, we found that an acceptable effort was devoted to develop unsupervised autoencoder models, GNNs, and other variants of DL models for CD. All these works were summarized in Tables 2, 3, and 4, respectively. However, most of the existing methods show shortcomings in terms of low efficiency and scalability, and in addition, they were evaluated on small-scale networks with at most a few thousand nodes. A few studies were proposed for addressing the mentioned problem in the field such as [31], [94]. However, these methods still showed other drawbacks, e.g., they require extensive trainable parameters, use massively synchronization between nodes through applying -bulk synchronous parallel. In addition, DL-based CD methods were not deployed in GPU parallelism and model parallelism. The other method was proposed to tackle such problem by a sampling approach, which was a speed–accuracy trade-off strategy [81]. Moreover, all existing DL-based CD methods depend on GD with BP optimization algorithm. Thus, there are three inherent limitations: the training process is slow, especially when processing large datasets; the method is sensitive to parameter initialization; and the solutions provided by the method could lead to local minima, which reduce the quality of CD.

Section VI presented a simple review of different DL models that use parallel computing methods in both regular, and irregular fields, which were listed in Table 5. According to the review of this Section, it is clearly that the parallel computing is considered a better solution for DL on a large-scale data. In regular areas (i.e., clear and grid structure, low-dimensional space) we have seen great efforts to develop large-scale models for DL based on parallel computing. As in irregular areas (i.e., unclear structure, high-dimensional space such as graphs or CNs), an acceptable effort has been made to use parallel computing to improve the efficiency of DL-based graph applications, especially in supervised learning methods, such as classification with GCNs. On the other hand, only very limited studies were proposed to improve unsupervised DL methods (i.e. stack autoencoders for CN clustering) such as [31], [94], however, there is a gap in deploying these unsupervised DL methods in a parallel computing framework. For example, no GPU parallelization or efficient partitioning technique was used, and parallel computing also suffers from high synchronization overheads.

Section VII introduced the common MH algorithms. Their advantages and disadvantages are summarized in Table 6. Then the section presented the utilization of MH algorithms to solve the problem of CD. According to the review in Subsection VII (B), it is clear that the MH algorithms have global search capabilities and good local learning, and can deal with a wide range of CD problems. Moreover, they can automatically determine the number of communities and they can be implemented in parallel and efficiently. However, they show low capabilities of achieving high accuracy and efficiency, especially when dealing with large and CNs.

Section VII also presented the integration of MH algorithms into DL models for the optimization process. The optimization of DL models could be based on stand-alone MH algorithms, or on a hybrid of gradient descent and MH

[1]https://www.cs.cornell.edu/projects/kddcup/datasets.html
[2]http://snap.stanford.edu/data/roadNet-PA.html

algorithms, which were summarized in Table 7 and 8 respectively. In addition, some methods have been introduced to integrate parallel computing and MH algorithms with DL models to process big data more efficiently (see Table 9). The MH algorithms have demonstrated their ability in improving the performance of DL models in various applications. However, most of the existing methods focus on improving the effectiveness of DL models and ignoring their efficiency in terms of space and time complexities, especially when the models are applied to handle large datasets. Nonetheless, some studies have been developed to improve the efficiency of DL models. MH algorithms have been blended into parallel computations to improve a DL model's efficiency, but the number of such research work across different application domains is still very few. Likewise, in the CD field, the use of MH algorithms for optimizing DL models in CN applications (either oriented on clustering or classification) is still very rare, and DL-based CD is no exception.

X. FUTURE DIRECTIONS AND PERSPECTIVES

Despite the fact that DL-based CD has shown superior performances, there are several problems that are still open and have not been fully solved. In this section, we briefly discuss these future prospects that can help overcome these problems.

One of the main challenges of CD is to efficiently identify communities in large-scale CNs. Many existing DL-based CD methods are inefficient when dealing with large CNs due to prohibitive demands on memory and computation (see Sections V and VI). However, nowadays, large networks exist in many real-world CNs, e.g., Facebook and Twitter networks. Therefore, novel DL-based CD methods shall be developed to efficiently and effectively utilize the rich information of large CNs. One possible direction is to develop DL-based CD methods that can achieve HPC with three features: efficient sub-network training, data and model parallelisms, the use of both CPU and GPU devices. In addition, a wise consideration is to develop methods from a parallelism approach to reduce the synchronization and communication operations and increase the asynchronization while focusing on improving the performance of CD. Memory consumption is an important issue that has not yet been fully addressed. CNs are usually available in a high-dimensional feature space, which leads to the generation of massive number of trainable parameters for refinement as well as requiring a large amount of memory for storage and computation. Thus, it is worthwhile to utilize techniques such as data partitioning and sharing the trainable parameters to reduce the trainable parameters [80].

MHs are promising algorithms used for optimizing the parameters of DL models. However, most of the existing algorithms show shortcomings in terms of low accuracy and efficiency when dealing with data that are collected from large and complex problems. Several possible research directions are suggested to explore for research in the future. In the research direction of improving effectiveness of a machine learning or DL model, a long-term goal is to develop powerful MH algorithms that can be saved from the trap of local optimum and premature convergence. This can be achieved by integrating different MH algorithms together or integrating MH with a GD-BP algorithm. Moreover, considering the continuation approach' ability to solve the optimization problem [151], [181], its ability in optimizing DL models shall be further investigated. Objective functions play an important role in guiding a search process for high quality solutions. Many efforts have been made to formulate a single objective solution, but multi-objective functions are still rarely used. Thus, it is worthwhile to further investigate the development of DL based on MH algorithms with multiple objective functions [98].

MH algorithms are known as a time-consuming process, especially for big data. Thus, we find that most of the DL with MH algorithms are evaluated with small sized problems. Since time is a very crucial factor, this is an important issue since the real world is full of big data, especially in CN applications. In the research direction of improving efficiency, parallel computing with CPUs and GPUs will be an essential solution to overcome this problem. In addition, scalability is another important issue as real-world data is growing rapidly and the dimensions of the data are very huge. Therefore, efficient and scalable MH algorithms are long-term preferable solutions used for training large-scale DL models.

For CNs and their applications, the use of DL models with MH algorithms is rather new and there are no studies that propose to solve CNs problems, e.g., CD, with those methods. Therefore, several possible research directions can be explored in the future, e.g., optimization of model parameters, automatic definition of model structure, reduction of dimensional space to reduce computational cost. In addition, it is worthwhile to investigate the possibility of integrating MH algorithms into DL-based CD models. Furthermore, the MH algorithms could be integrated into GD-BP algorithms to improve the effectiveness and efficiency of such models. Finally, since CNs in real words are often large, it is worth to explore the integration of parallel computing, including CPU, GPUs, or hybrids thereof, to improve the efficiency and scalability of DL-based CD models with the MH algorithms.

Finally, there are general issues related to DL-based CD models in CNs that need further investigation. Most of the existing models deal with complete, static, or homogeneous networks. However, in the real world, many networks are incomplete, dynamic, or heterogeneous. Incomplete networks [182] are those that lack of information about topology or features. Dynamic networks [183] incorporate topology and attributes that shall be updated over time. Heterogeneous networks [184] contain members belonging to different types, e.g., images and texts. These complex structures in networks cannot be solved with the algorithms proposed for complete, static and homogeneous networks. As a result, several issues arise. For example, for incomplete networks, the entire network structure cannot be learned and discovered; for dynamic networks, the network structure shall be re-learned and

re-explored, which is computationally costly; the network members of different types require complex algorithms to deal with their features and topology. Unfortunately, it is rarely possible to find accurate and complete networks without noise in the real world. Therefore, it is worthwhile to investigate the performance of DL and MH algorithms in community detection with noisy networks, which are characterized as incomplete, dynamic, and/or heterogeneous networks.

XI. CONCLUSION

In this paper, we present a literature review of CD in CNs from conventional ML to DL methods and point out the gap of applying DL-based CD methods in large CNs. In addition, the relevant studies on DL with parallel and MH approaches are reviewed and their implications on DL models are highlighted. One of our main goals is to present and organize most of the work done so far on DL-based CD methods, as well as on DL with MH and parallel computing approaches, in a unified perspective. This is intended to encourage the research community to bridge the gap between DL-based CD and DL with MH and parallel computing approaches. Specifically, we have reviewed the recent literature on CD methods, including those using early classical ML and MH approaches, as well as more recently DL approaches, and presented their merits and demerits. Despite considerable research efforts have been made to propose solutions for this topic, there are still lack of effective and efficient methods that can fully meet performance requirements, particularly when large CNs are involved. Therefore, this paper presents an overview on the success of integrating DL models with MH algorithms and parallel computing in different domains, explains their pros and cons. How DL with MH and parallel computing methods could contribute to improve the effectiveness and efficiency of DL models in the CD field has also been discussed. Finally, we summarize the work and point out some research directions for bridging the gaps between DL-based CD methods with MH and parallel computing approaches to deal with large CNs effectively and efficiently.

REFERENCES

[1] M. Khatoon and W. A. Banu, "A survey on community detection methods in social networks," *Int. J. Educ. Manage. Eng.*, vol. 5, no. 1, pp. 8–18, May 2015, doi: 10.5815/IJEME.2015.01.02.

[2] K. Yugi, H. Kubota, A. Hatano, and S. Kuroda, "Trans-omics: How to reconstruct biochemical networks across multiple 'omic' layers," *Trends Biotechnol.*, vol. 34, no. 4, pp. 276–290, Apr. 2016, doi: 10.1016/j.tibtech.2015.12.013.

[3] N. Malod-Dognin, K. Ban, and N. Pržulj, "Unified alignment of protein-protein interaction networks," *Sci. Rep.*, vol. 7, no. 1, pp. 1–11, Apr. 2017, doi: 10.1038/s41598-017-01085-9.

[4] Z. Xiao, N. Kathiresshan, and Y. Xiao, "A survey of accountability in computer networks and distributed systems," *Secur. Commun. Netw.*, vol. 9, no. 4, pp. 290–315, Mar. 2016, doi: 10.1002/sec.574.

[5] T. Velden, S. Yan, and C. Lagoze, "Mapping the cognitive structure of astrophysics by infomap clustering of the citation network and topic affinity analysis," *Scientometrics*, vol. 111, no. 2, pp. 1033–1051, Feb. 2017, doi: 10.1007/s11192-017-2299-9.

[6] Y. Xie, M. Gong, S. Wang, and B. Yu, "Community discovery in networks with deep sparse filtering," *Pattern Recognit.*, vol. 81, pp. 50–59, Sep. 2018, doi: 10.1016/j.patcog.2018.03.026.

[7] M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig, "Community detection in networks: A multidisciplinary review," *J. Netw. Comput. Appl.*, vol. 108, pp. 87–111, Apr. 2018, doi: 10.1016/j.jnca.2018.02.011.

[8] L. Yang, X. Cao, D. He, C. Wang, X. Wang, and W. Zhang, "Modularity based community detection with deep learning," in *Proc. IJCAI*, vol. 16, Jul. 2016, pp. 2252–2258, doi: 10.5555/3060832.3060936.

[9] D. Jin, Z. Yu, P. Jiao, S. Pan, P. S. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep representation," 2021, *arXiv:2101.01669*. [Online]. Available: http://arxiv.org/abs/2101.01669

[10] X. Zhang and M. E. J. Newman, "Multiway spectral community detection in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 92, no. 5, Nov. 2015, Art. no. 052808, doi: 10.1103/PhysRevE.92.052808.

[11] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proc. SIAM Int. Conf. Data Mining*, Apr. 2005, pp. 274–285, doi: 10.1137/1.9781611972757.25.

[12] F. Wang, B. Zhang, and S. Chai, "Deep auto-encoded clustering algorithm for community detection in complex networks," *Chin. J. Electron.*, vol. 28, no. 3, pp. 489–496, May 2019, doi: 10.1049/cje.2019.03.019.

[13] F. Liu, S. Xue, J. Wu, C. Zhou, W. Hu, C. Paris, S. Nepal, J. Yang, and P. S. Yu, "Deep learning for community detection: Progress, challenges and opportunities," 2020, *arXiv:2005.08225*. [Online]. Available: http://arxiv.org/abs/2005.08225

[14] D. A. Abduljabbar, S. Z. M. Hashim, and R. Sallehuddin, "Nature-inspired optimization algorithms for community detection in complex networks: A review and future trends," *Telecommun. Syst.*, vol. 74, no. 2, pp. 225–252, Jan. 2020, doi: 10.1007/s11235-019-00636-x.

[15] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018, doi: 10.1016/j.inffus.2017.10.006.

[16] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, early access, Mar. 17, 2020, doi: 10.1109/TKDE.2020.2981333.

[17] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proc. 28th AAAI Conf. Artif. Intell.*, Jun. 2014, pp. 1–7. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/8916

[18] D. Jin, B. Li, P. Jiao, D. He, and H. Shan, "Community detection via joint graph convolutional network embedding in attribute network," in *Proc. Int. Conf. Artif. Neural Netw.* Cham, Switzerland: Springer, Sep. 2019, pp. 594–606, doi: 10.1007/978-3-030-30493-5_55.

[19] Z. Chen, X. Li, and J. Bruna, "Supervised community detection with line graph neural networks," May 2017, *arXiv:1705.08415*. [Online]. Available: http://arxiv.org/abs/1705.08415

[20] G. Sperlí, "A deep learning based community detection approach," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput.*, Apr. 2019, pp. 1107–1110, doi: 10.1145/3297280.3297574.

[21] S. Fong, S. Deb, and X.-S. Yang, "How meta-heuristic algorithms contribute to deep learning in the hype of big data analytics," in *Progress in Intelligent Computing Techniques: Theory, Practice, and Applications*. Singapore: Springer, Jul. 2018, pp. 2–3, doi: 10.1007/978-981-10-3373-5_1.

[22] J.-H. Yi, W.-H. Xu, and Y.-T. Chen, "Novel back propagation optimization by cuckoo search algorithm," *Sci. World J.*, vol. 2014, pp. 1–8, Jan. 2014, doi: 10.1155/2014/878262.

[23] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intell.*, vol. 1, no. 1, pp. 33–57, Oct. 2007, doi: 10.1007/s11721-007-0002-0.

[24] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, no. 2, pp. 95–99, Oct. 1988, doi: 10.1023/A:1022602019183.

[25] Z. Tian and S. Fong, "Survey of meta-heuristic algorithms for deep learning training," in *Optimization Algorithms: Methods and Applications*. U.K.: InTech, Sep. 2016, doi: 10.5772/63785.

[26] V. K. Ojha, A. Abraham, and V. Snášel, "Metaheuristic design of feedforward neural networks: A review of two decades of research," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 97–116, Apr. 2017, doi: 10.1016/j.engappai.2017.01.013.

[27] L. Deng, D. Yu, and J. Platt, "Scalable stacking and learning for building deep architectures," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2012, pp. 2133–2136, doi: 10.1109/ICASSP.2012.6288333.

[28] L. Ma, Z. Yang, Y. Miao, J. Xue, M. Wu, L. Zhou, and Y. Dai, "NeuGraph: Parallel deep neural network computation on large graphs," in *Proc. USENIX Annu. Tech. Conf. (USENIX)*, 2019, pp. 443–458. [Online]. Available: https://www.usenix.org/conference/atc19/presentation/ma

[29] R. Raina, A. Madhavan, and A. Y. Ng, "Large-scale deep unsupervised learning using graphics processors," in *Proc. 26th Annu. Int. Conf. Mach. Learn. (ICML)*, Jun. 2009, pp. 873–880, doi: 10.1145/1553374.1553486.

[30] J. Dean, G. S. Corrado, R. Monga, K. Chen, M. Devin, Q. V. Le, M. Z. Mao, M. Ranzato, A. Senior, P. Tucker, K. Yang, and A. Y. Ng, "Large scale distributed deep networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1223–1231. [Online]. Available: https://research.google/pubs/pub40565/

[31] V. Bhatia and R. Rani, "DFuzzy: A deep learning-based fuzzy clustering model for large graphs," *Knowl. Inf. Syst.*, vol. 57, no. 1, pp. 159–181, Oct. 2018, doi: 10.1007/s10115-018-1156-3.

[32] S. Papadopoulos, Y. Kompatsiaris, A. Vakali, and P. Spyridonos, "Community detection in social media: Performance and application considerations," *Data Mining Knowl. Discovery*, vol. 24, no. 3, pp. 515–554, May 2012, doi: 10.1007/s10618-011-0224-z.

[33] B. A. Attea, A. D. Abbood, A. A. Hasan, C. Pizzuti, M. Al-Ani, S. Özdemir, and R. D. Al-Dabbagh, "A review of heuristics and metaheuristics for community detection in complex networks: Current usage, emerging development and future directions," *Swarm Evol. Comput.*, vol. 63, Jun. 2021, Art. no. 100885, doi: 10.1016/j.swevo.2021.100885.

[34] W. H. L. Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli, "Autoencoders," in *Machine Learning*. Amsterdam, The Netherlands: Elsevier, 2020, pp. 193–208, doi: 10.1016/B978-0-12-815739-8.00011-0.

[35] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, Sep. 2002, pp. 849–856.

[36] S. Niu, D. Wang, S. Feng, and G. Yu, "An improved spectral clustering algorithm for community discovery," in *Proc. 9th Int. Conf. Hybrid Intell. Syst.*, vol. 3, Aug. 2009, pp. 262–267, doi: 10.1109/HIS.2009.268.

[37] S. Zhang, R.-S. Wang, and X.-S. Zhang, "Identification of overlapping community structure in complex networks using fuzzy c-means clustering," *Phys. A, Stat. Mech. Appl.*, vol. 374, no. 1, pp. 483–490, Jan. 2007, doi: 10.1016/j.physa.2006.07.023.

[38] J. C. Bezdek, *Pattern Recognition With Fuzzy Objective Function Algorithms*. USA: Springer, Jul. 1981.

[39] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 69, no. 2, Feb. 2004, Art. no. 026113, doi: 10.1103/PhysRevE.69.026113.

[40] L. C. Freeman, "A set of measures of centrality based on betweenness," *Sociometry*, vol. 40, pp. 35–41, Mar. 1977, doi: 10.2307/3033543.

[41] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner, "On modularity clustering," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 2, pp. 172–188, Feb. 2008, doi: 10.1109/TKDE.2007.190689.

[42] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Proc. Int. Symp. Comput. Inf. Sci.* Berlin, Germany: Springer, 2005, pp. 284–293, doi: 10.1007/11569596_31.

[43] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proc. Nat. Acad. Sci. USA*, vol. 105, no. 4, pp. 1118–1123, Jan. 2008.

[44] P. W. Holland, K. B. Laskey, and S. Leinhardt, "Stochastic blockmodels: First steps," *Social Netw.*, vol. 5, no. 2, pp. 109–137, Jun. 1983, doi: 10.1016/0378-8733(83)90021-7.

[45] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 83, no. 1, Jan. 2011, Art. no. 016107, doi: 10.1103/PhysRevE.83.016107.

[46] R.-S. Wang, S. Zhang, Y. Wang, X.-S. Zhang, and L. Chen, "Clustering complex networks and biological networks by nonnegative matrix factorization with various similarity measures," *Neurocomputing*, vol. 72, nos. 1–3, pp. 134–141, Dec. 2008, doi: 10.1016/j.neucom.2007.12.043.

[47] X. Shi, H. Lu, Y. He, and S. He, "Community detection in social network with pairwisely constrained symmetric non-negative matrix factorization," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2015, pp. 541–546, doi: 10.1145/2808797.2809383.

[48] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining Knowl. Discovery*, vol. 22, no. 3, pp. 493–521, May 2011, doi: 10.1007/s10618-010-0181-y.

[49] A. Prat-Pérez, D. Dominguez-Sal, and J.-L. Larriba-Pey, "High quality, scalable and parallel community detection for large real graphs," in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, Apr. 2014, pp. 225–236, doi: 10.1145/2566486.2568010.

[50] T. He, L. Cai, T. Meng, L. Chen, Z. Deng, and Z. Cao, "Parallel community detection based on distance dynamics for large-scale network," *IEEE Access*, vol. 6, pp. 42775–42789, Jul. 2018, doi: 10.1109/ACCESS.2018.2859788.

[51] J. Shao, Z. Han, Q. Yang, and T. Zhou, "Community detection based on distance dynamics," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2015, pp. 1075–1084, doi: 10.1145/2783258.2783301.

[52] M. Fazlali, E. Moradi, and H. T. Malazi, "Adaptive parallel Louvain community detection on a multicore platform," *Microprocessors Microsyst.*, vol. 54, pp. 26–34, Oct. 2017, doi: 10.1016/j.micpro.2017.08.002.

[53] S. Souravlas, A. Sifaleras, and S. Katsavounis, "A parallel algorithm for community detection in social networks, based on path analysis and threaded binary trees," *IEEE Access*, vol. 7, pp. 20499–20519, Feb. 2019, doi: 10.1109/ACCESS.2019.2897783.

[54] S. Moon, J.-G. Lee, M. Kang, M. Choy, and J.-W. Lee, "Parallel community detection on large graphs with MapReduce and GraphChi," *Data Knowl. Eng.*, vol. 104, pp. 17–31, Jul. 2016, doi: 10.1016/j.datak.2015.05.001.

[55] T. Lopes, V. Ströele, M. Dantas, R. Braga, and J.-F. Meháut, "A parallel graph partitioning approach to enhance community detection in social networks," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2020, pp. 1–6, doi: 10.1109/ISCC50000.2020.9219602.

[56] J. Soman and A. Narang, "Fast community detection algorithm with GPUs and multicore architectures," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, Sep. 2011, pp. 568–579, doi: 10.1109/IPDPS.2011.61.

[57] P. Li, "Community structure discovery algorithm on GPU with CUDA," in *Proc. 3rd IEEE Int. Conf. Broadband Netw. Multimedia Technol. (IC-BNMT)*, Oct. 2010, pp. 1136–1139, doi: 10.1109/ICBNMT.2010.5705267.

[58] M. Al-Ayyoub, M. Al-Andoli, Y. Jararweh, M. Smadi, and B. Gupta, "Improving fuzzy c-mean-based community detection in social networks using dynamic parallelism," *Comput. Electr. Eng.*, vol. 74, pp. 533–546, Mar. 2019, doi: 10.1016/j.compeleceng.2018.01.003.

[59] M. Mohammadi, M. Fazlali, and M. Hosseinzadeh, "Accelerating Louvain community detection algorithm on graphic processing unit," *J. Supercomput.*, vol. 22, pp. 1–22, Nov. 2020, doi: 10.1007/s11227-020-03510-9.

[60] S. Souravlas, A. Sifaleras, and S. Katsavounis, "Hybrid CPU-GPU community detection in weighted networks," *IEEE Access*, vol. 8, pp. 57527–57551, Mar. 2020, doi: 10.1109/ACCESS.2020.2982227.

[61] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cognit. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017, doi: 10.1109/TCCN.2017.2758370.

[62] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: Experimental evaluation, lessons learned, and challenges," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 2, pp. 445–458, Jun. 2019, doi: 10.1109/TNSM.2019.2899085.

[63] M. E. J. Newman, "Modularity and community structure in networks," *Proc. Nat. Acad. Sci. USA*, vol. 103, no. 23, pp. 8577–8582, Jun. 2006, doi: 10.1073/pnas.0601602103.

[64] R. Fei, J. Sha, Q. Xu, B. Hu, K. Wang, and S. Li, "A new deep sparse autoencoder for community detection in complex networks," *EURASIP J. Wireless Commun. Netw.*, vol. 2020, no. 1, pp. 1–25, Dec. 2020, doi: 10.1186/s13638-020-01706-4.

[65] X. Geng, H. Lu, and J. Sun, "Network structural transformation-based community detection with autoencoder," *Symmetry*, vol. 12, no. 6, p. 944, Jun. 2020, doi: 10.3390/sym12060944.

[66] L. Wu, Q. Zhang, C.-H. Chen, K. Guo, and D. Wang, "Deep learning techniques for community detection in social networks," *IEEE Access*, vol. 8, pp. 96016–96026, 2020, doi: 10.1109/ACCESS.2020.2996001.

[67] J. Cao, D. Jin, and J. Dang, "Autoencoder based community detection with adaptive integration of network topology and node contents," in *Proc. Int. Conf. Knowl. Sci., Eng. Manage.* Cham, Switzerland: Springer, Aug. 2018, pp. 184–196, doi: 10.1007/978-3-319-99247-1_16.

[68] D. Jin, M. Ge, Z. Li, W. Lu, D. He, and F. Fogelman-Soulie, "Using deep learning for community discovery in social networks," in *Proc. IEEE 29th Int. Conf. Tools With Artif. Intell. (ICTAI)*, Nov. 2017, pp. 160–167, doi: 10.1109/ICTAI.2017.00035.

[69] J. Cao, D. Jin, L. Yang, and J. Dang, "Incorporating network structure with node contents for community detection on large networks using deep learning," *Neurocomputing*, vol. 297, pp. 71–81, Jul. 2018, doi: 10.1016/j.neucom.2018.01.065.

[70] Y. Xie, X. Wang, D. Jiang, and R. Xu, "High-performance community detection in social networks using a deep transitive autoencoder," *Inf. Sci.*, vol. 493, pp. 75–90, Aug. 2019, doi: 10.1016/j.ins.2019.04.018.

[71] R. Xu, Y. Che, X. Wang, J. Hu, and Y. Xie, "Stacked autoencoder-based community detection method via an ensemble clustering framework," *Inf. Sci.*, vol. 526, pp. 151–165, Jul. 2020, doi: 10.1016/j.ins.2020.03.090.

[72] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: http://arxiv.org/abs/1609.02907

[73] C. Wang, S. Pan, G. Long, X. Zhu, and J. Jiang, "MGAE: Marginalized graph autoencoder for graph clustering," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 889–898, doi: 10.1145/3132847.3132967.

[74] Z. Chen, X. Li, and J. Bruna, "Supervised community detection with line graph neural networks," May 2017, vol. 1050, p. 27, *arXiv:1705.08415*. [Online]. Available: http://arxiv.org/abs/1705.08415

[75] J. Sun, W. Zheng, Q. Zhang, and Z. Xu, "Graph neural network encoding for community detection in attribute networks," *IEEE Trans. Cybern.*, early access, Feb. 10, 2021, doi: 10.1109/TCYB.2021.3051021.

[76] H. Sun, F. He, J. Huang, Y. Sun, Y. Li, C. Wang, L. He, Z. Sun, and X. Jia, "Network embedding for community detection in attributed networks," *ACM Trans. Knowl. Discovery From Data*, vol. 14, no. 3, pp. 1–25, May 2020, doi: 10.1145/3385415.

[77] J. Park, M. Lee, H. J. Chang, K. Lee, and J. Y. Choi, "Symmetric graph convolutional autoencoder for unsupervised graph representation learning," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6519–6528, doi: 10.1109/ICCV.2019.00662.

[78] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 1393–1402, doi: 10.1145/3269206.3271697.

[79] Y. Zhou, A. Amimeur, C. Jiang, D. Dou, R. Jin, and P. Wang, "Density-aware local siamese autoencoder network embedding with autoencoder graph clustering," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 1162–1167, doi: 10.1109/BigData.2018.8621992.

[80] M. Al-Andoli, W. P. Cheah, and S. C. Tan, "Deep learning-based community detection in complex networks with network partitioning and reduction of trainable parameters," *J. Ambient Intell. Humanized Comput.*, vol. 12, pp. 1–19, Jul. 2020, doi: 10.1007/s12652-020-02389-x.

[81] G. Salha, R. Hennequin, V. A. Tran, and M. Vazirgiannis, "A degeneracy framework for scalable graph autoencoders," 2019, *arXiv:1902.08813*. [Online]. Available: http://arxiv.org/abs/1902.08813

[82] Y. Ma, P. Zhang, Y. Cao, and L. Guo, "Parallel auto-encoder for efficient outlier detection," in *Proc. IEEE Int. Conf. Big Data*, Oct. 2013, pp. 15–17, doi: 10.1109/BigData.2013.6691791.

[83] K. Chen and Q. Huo, "Scalable training of deep learning machines by incremental block training with intra-block parallel optimization and blockwise model-update filtering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2016, pp. 5880–5884, doi: 10.1109/ICASSP.2016.7472805.

[84] M. Makkie, H. Huang, Y. Zhao, A. V. Vasilakos, and T. Liu, "Fast and scalable distributed deep convolutional autoencoder for fMRI big data analytics," *Neurocomputing*, vol. 325, pp. 20–30, Jan. 2019, doi: 10.1016/j.neucom.2018.09.066.

[85] A. Harlap, D. Narayanan, A. Phanishayee, V. Seshadri, N. Devanur, G. Ganger, and P. Gibbons, "PipeDream: Fast and efficient pipeline parallel DNN training," Jun. 2018, *arXiv:1806.03377*. [Online]. Available: http://arxiv.org/abs/1806.03377

[86] A. Sriram, J. Zbontar, T. Murrell, C. L. Zitnick, A. Defazio, and D. K. Sodickson, "GrappaNet: Combining parallel imaging with deep learning for multi-coil MRI reconstruction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14315–14322, doi: 10.1109/CVPR42600.2020.01432.

[87] J. H. Park, S. Kim, J. Lee, M. Jeon, and S. H. Noh, "Accelerated training for CNN distributed deep learning through automatic resource-aware layer placement," Jan. 2019, *arXiv:1901.05803*. [Online]. Available: http://arxiv.org/abs/1901.05803

[88] Y. Kim, H. Choi, J. Lee, J.-S. Kim, H. Jei, and H. Roh, "Towards an optimized distributed deep learning framework for a heterogeneous multi-GPU cluster," *Cluster Comput.*, vol. 23, no. 3, pp. 2287–2300, Jul. 2020, doi: 10.1007/s10586-020-03144-9.

[89] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Accurate, efficient and scalable graph embedding," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2019, pp. 462–471, doi: 10.1109/IPDPS.2019.00056.

[90] B. Ribeiro and D. Towsley, "Estimating and sampling graphs with multidimensional random walks," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*, Nov. 2010, pp. 390–403, doi: 10.1145/1879141.1879192.

[91] W.-L. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C.-J. Hsieh, "Cluster-GCN: An efficient algorithm for training deep and large graph convolutional networks," May 2019, *arXiv:1905.07953*. [Online]. Available: http://arxiv.org/abs/1905.07953

[92] H. Liu, S. Lu, X. Chen, and B. He, "G3: When graph neural networks meet parallel graph processing systems on GPUs," *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 2813–2816, Aug. 2020, doi: 10.14778/3415478.3415482.

[93] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, and V. Prasanna, "Accurate, efficient and scalable training of graph neural networks," *J. Parallel Distrib. Comput.*, vol. 147, pp. 166–183, Jan. 2021, doi: 10.1016/j.jpdc.2020.08.011.

[94] V. Bhatia and R. Rani, "A distributed overlapping community detection model for large graphs using autoencoder," *Future Gener. Comput. Syst.*, vol. 94, pp. 16–26, May 2019, doi: 10.1016/j.future.2018.10.045.

[95] F. Hayes-Roth, "Review of 'adaptation in natural and artificial systems by John H. Holland'', The U. of Michigan Press, 1975," *ACM SIGART Bull.*, vol. 18, no. 53, p. 15, 1975, doi: 10.1137/1018105.

[96] A. Surendran and P. Samuel, "Evolution or revolution: The critical need in genetic algorithm based testing," *Artif. Intell. Rev.*, vol. 48, no. 3, pp. 349–395, Oct. 2017, doi: 10.1007/s10462-016-9504-8.

[97] A. Said, R. A. Abbasi, O. Maqbool, A. Daud, and N. R. Aljohani, "CC-GA: A clustering coefficient based genetic algorithm for detecting communities in social networks," *Appl. Soft Comput.*, vol. 63, pp. 59–70, Feb. 2018, doi: 10.1016/j.asoc.2017.11.014.

[98] Y. Pan, Y. Yang, and W. Li, "A deep learning trained by genetic algorithm to improve the efficiency of path planning for data collection with multi-UAV," *IEEE Access*, vol. 9, pp. 7994–8005, 2021, doi: 10.1109/ACCESS.2021.3049892.

[99] O. E. David and I. Greental, "Genetic algorithms for evolving deep neural networks," in *Proc. Companion Publication Annu. Conf. Genet. Evol. Comput.*, Jul. 2014, pp. 1451–1452, doi: 10.1145/2598394.2602287.

[100] S. Sharma and H. M. Pandey, "Genetic algorithm, particle swarm optimization and harmony search: A quick comparison," in *Proc. 6th Int. Conf.-Cloud Syst. Big Data Eng. (Confluence)*, Jan. 2016, pp. 40–44, doi: 10.1109/CONFLUENCE.2016.7508044.

[101] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 18, no. 1, pp. 82–97, Feb. 2014, doi: 10.1109/TEVC.2013.2260862.

[102] M. Al-Andoli, W. P. Cheah, and S. C. Tan, "Deep autoencoder-based community detection in complex networks with particle swarm optimization and continuation algorithms," *J. Intell. Fuzzy Syst.*, vol. 40, pp. 1–17, Mar. 2021, doi: 10.3233/JIFS-201342.

[103] A. Rajagopal, G. P. Joshi, A. Ramachandran, R. T. Subhalakshmi, M. Khari, S. Jha, K. Shankar, and J. You, "A deep learning model based on multi-objective particle swarm optimization for scene classification in unmanned aerial vehicles," *IEEE Access*, vol. 8, pp. 135383–135393, 2020, doi: 10.1109/ACCESS.2020.3011502.

[104] X. Cheng, D. Ciuonzo, and P. S. Rossi, "Multibit decentralized detection through fusing smart and dumb sensors based on rao test," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 2, pp. 1391–1405, Apr. 2020, doi: 10.1109/TAES.2019.2936777.

[105] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politecnico di Mila, Milan, Italy, 1992. [Online]. Available: https://ci.nii.ac.jp/naid/10027800670/

[106] N. S. Sani, M. Manthouri, and F. Farivar, "A multi-objective ant colony optimization algorithm for community detection in complex networks," *J. Ambient Intell. Humanized Comput.*, vol. 11, no. 1, pp. 5–21, Jan. 2020, doi: 10.1007/s12652-018-1159-7.

[107] M. Mavrovouniotis and S. Yang, "Training neural networks with ant colony optimization algorithms for pattern classification," *Soft Comput.*, vol. 19, no. 6, pp. 1511–1522, Jun. 2015, doi: 10.1007/s00500-014-1334-5.

[108] M. K. Hussein and M. H. Mousa, "Efficient task offloading for IoT-based applications in fog computing using ant colony optimization," *IEEE Access*, vol. 8, pp. 37191–37201, 2020, doi: 10.1109/ACCESS.2020.2975741.

[109] R. Storn and K. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Global Optim.*, vol. 11, no. 4, pp. 341–359, Dec. 1997, doi: 10.1023/A:1008202821328.

[110] S. Lander and Y. Shang, "EvoAE—A new evolutionary method for training autoencoders for deep learning networks," in *Proc. IEEE 39th Annu. Comput. Softw. Appl. Conf.*, vol. 2, Jul. 2015, pp. 790–795, doi: 10.1109/COMPSAC.2015.63.

[111] C. Sui, M. Bennamoun, and R. Togneri, "Deep feature learning for dummies: A simple auto-encoder training method using particle swarm optimisation," *Pattern Recognit. Lett.*, vol. 94, pp. 75–80, Jul. 2017, doi: 10.1016/j.patrec.2017.03.021.

[112] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Eng. Fac., Erciyes Univ., Kayseri, Turkey, Tech. Rep. tr06, 2005.

[113] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214, doi: 10.1109/NABIC.2009.5393690.

[114] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization*. Berlin, Germany: Springer, 2010, pp. 65–74, doi: 10.1007/978-3-642-12538-6_6.

[115] D. Karaboga, B. Akay, and C. Ozturk, "Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks," in *Proc. Int. Conf. Modeling Decisions Artif. Intell.* Berlin, Germany: Springer, 2007, pp. 318–329, doi: 10.1007/978-3-540-73729-2_30.

[116] A. M. Sahan, A. S. Al-Itbi, and J. S. Hameed, "COVID-19 detection based on deep learning and artificial bee colony," *Periodicals Eng. Natural Sci.*, vol. 9, no. 1, pp. 29–36, 2021, doi: 10.21533/pen.v9i1.1774.

[117] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *Stochastic Algorithms: Foundations and Applications* (Lecture Notes in Computer Science), vol. 5792, O. Watanabe and T. Zeugmann, Eds. Berlin, Germany: Springer, doi: 10.1007/978-3-642-04944-6_14.

[118] B. Amiri, L. Hossain, J. W. Crawford, and R. T. Wigand, "Community detection in complex networks: Multi–objective enhanced firefly algorithm," *Knowl.-Based Syst.*, vol. 46, pp. 1–11, Jul. 2013, doi: 10.1016/j.knosys.2013.01.004.

[119] I. Strumberger, E. Tuba, N. Bacanin, M. Zivkovic, M. Beko, and M. Tuba, "Designing convolutional neural network architecture by the firefly algorithm," in *Proc. Int. Young Eng. Forum (YEF-ECE)*, May 2019, pp. 59–65, doi: 10.1109/YEF-ECE.2019.8740818.

[120] R. Babers and A. E. Hassanien, "A nature-inspired metaheuristic cuckoo search algorithm for community detection in social networks," *Int. J. Service Sci., Manage., Eng., Technol.*, vol. 8, no. 1, pp. 50–62, Jan. 2017, doi: 10.4018/IJSSMET.2017010104.

[121] A. Bakhshi, N. Noman, Z. Chen, M. Zamani, and S. Chalup, "Fast automatic optimisation of CNN architectures for image classification using genetic algorithm," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 1283–1290, doi: 10.1109/CEC.2019.8790197.

[122] I. A. Doush, W. A. B. Alrashdan, M. A. Al-Betar, and M. A. Awadallah, "Community detection in complex networks using multi-objective bat algorithm," *Int. J. Math. Model. Numer. Optim.*, vol. 10, no. 2, pp. 123–140, Mar. 2020, doi: 10.1504/IJMMNO.2020.106529.

[123] N. S. Jaddi, S. Abdullah, and A. R. Hamdan, "Optimization of neural network model using modified bat-inspired algorithm," *Appl. Soft Comput.*, vol. 37, pp. 71–86, Dec. 2015, doi: 10.1016/j.asoc.2015.08.002.

[124] A. M. Hemeida, S. A. Hassan, A.-A.-A. Mohamed, S. Alkhalaf, M. M. Mahmoud, T. Senjyu, and A. B. El-Din, "Nature-inspired algorithms for feed-forward neural network classifiers: A survey of one decade of research," *Ain Shams Eng. J.*, vol. 11, no. 3, pp. 659–675, Sep. 2020, doi: 10.1016/j.asej.2020.01.007.

[125] B. Zarei and M. R. Meybodi, "Detecting community structure in complex networks using genetic algorithm based on object migrating automata," *Comput. Intell.*, vol. 36, no. 2, pp. 824–860, May 2020, doi: 10.1111/coin.12273.

[126] M. Rostami, K. Berahmand, and S. Forouzandeh, "A novel community detection based genetic algorithm for feature selection," *J. Big Data*, vol. 8, no. 1, pp. 1–27, Jan. 2021, doi: 10.1186/s40537-020-00398-3.

[127] Q. Cai, M. Gong, L. Ma, S. Ruan, F. Yuan, and L. Jiao, "Greedy discrete particle swarm optimization for large-scale social network clustering," *Inf. Sci.*, vol. 316, pp. 503–516, Sep. 2015, doi: 10.1016/j.ins.2014.09.041.

[128] X. Liu, Y. Du, M. Jiang, and X. Zeng, "Multiobjective particle swarm optimization based on network embedding for complex network community detection," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 2, pp. 437–449, Apr. 2020, doi: 10.1109/TCSS.2020.2964027.

[129] Y. Ahmad, M. Ullah, R. Khan, B. Shafi, A. Khan, M. Zareei, A. Aldosary, and E. M. Mohamed, "SiFSO: Fish swarm optimization-based technique for efficient community detection in complex networks," *Complexity*, vol. 2020, pp. 1–9, Dec. 2020, doi: 10.1155/2020/6695032.

[130] J. Del Ser, J. L. Lobo, E. Villar-Rodriguez, M. N. Bilbao, and C. Perfecto, "Community detection in graphs based on surprise maximization using firefly heuristics," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 2233–2239, doi: 10.1109/CEC.2016.7744064.

[131] A. S. Jaradat and S. B. Hamad, "Community structure detection using firefly algorithm," *Int. J. Appl. Metaheuristic Comput.*, vol. 9, no. 4, pp. 52–70, Oct. 2018, doi: 10.4018/IJAMC.2018100103.

[132] E. A. Hassan, A. I. Hafez, A. E. Hassanien, and A. A. Fahmy, "A discrete bat algorithm for the community detection problem," in *Proc. Int. Conf. Hybrid Artif. Intell. Syst.* Cham, Switzerland: Springer, May 2015, pp. 188–199, doi: 10.1007/978-3-319-19644-2_16.

[133] A. Song, M. Li, X. Ding, W. Cao, and K. Pu, "Community detection using discrete bat algorithm," *IAENG Int. J. Comput. Sci.*, vol. 43, no. 1, pp. 37–43, 2016. [Online]. Available: http://www.iaeng.org/IJCS/issues_v43

[134] B. Chen, L. Chen, and Y. Chen, "Detecting community structure in networks based on ant colony optimization," in *Proc. Int. Conf. Inf. Knowl. Eng. (IKE)*, 2012, pp. 1–7.

[135] Y. Guo, X. Li, Y. Tang, and J. Li, "Heuristic artificial bee colony algorithm for uncovering community in complex networks," *Math. Problems Eng.*, vol. 2017, pp. 1–12, Oct. 2017, doi: 10.1155/2017/4143638.

[136] X. Zhou, Y. Liu, B. Li, and H. Li, "A multiobjective discrete cuckoo search algorithm for community detection in dynamic networks," *Soft Comput.*, vol. 21, no. 22, pp. 6641–6652, Nov. 2017, doi: 10.1007/s00500-016-2213-z.

[137] F. H. F. Leung, H. K. Lam, S. H. Ling, and P. K. S. Tam, "Tuning of the structure and parameters of a neural network using an improved genetic algorithm," *IEEE Trans. Neural Netw.*, vol. 14, no. 1, pp. 79–88, Jan. 2003, doi: 10.1109/TNN.2002.804317.

[138] S. Lee, J. Kim, H. Kang, D.-Y. Kang, and J. Park, "Genetic algorithm based deep learning neural network structure and hyperparameter optimization," *Appl. Sci.*, vol. 11, no. 2, p. 744, Jan. 2021, doi: 10.3390/app11020744.

[139] V. G. Gudise and G. K. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," in *Proc. IEEE Swarm Intell. Symp. (SIS)*, Jun. 2003, pp. 110–117, doi: 10.1109/SIS.2003.1202255.

[140] I. Boussaïd, J. Lepagnot, and P. Siarry, "A survey on optimization metaheuristics," *Inf. Sci.*, vol. 237, pp. 82–117, Jul. 2013, doi: 10.1016/j.ins.2013.02.041.

[141] S. S. Band, S. Janizadeh, S. C. Pal, A. Saha, R. Chakrabortty, M. Shokri, and A. Mosavi, "Novel ensemble approach of deep learning neural network (DLNN) model and particle swarm optimization (PSO) algorithm for prediction of gully erosion susceptibility," *Sensors*, vol. 20, no. 19, p. 5609, Sep. 2020, doi: 10.3390/s20195609.

[142] H. Zhang, H. Nguyen, X.-N. Bui, T. Nguyen-Thoi, T.-T. Bui, N. Nguyen, D.-A. Vu, V. Mahesh, and H. Moayedi, "Developing a novel artificial intelligence model to estimate the capital cost of mining projects using deep neural network-based ant colony optimization algorithm," *Resour. Policy*, vol. 66, Jun. 2020, Art. no. 101604, doi: 10.1016/j.resourpol.2020.101604.

[143] M. Sreeshakthy and J. Preethi, "Classification of human emotion from DEAP EEG signal using hybrid improved neural networks with cuckoo search," *Broad Res. Artif. Intell. Neurosci.*, vol. 6, nos. 3–4, pp. 60–73, 2016. [Online]. Available: https://brain.edusoft.ro/index.php/brain/article/view/519

[144] R. Cristin, B. S. Kumar, C. Priya, and K. Karthick, "Deep neural network based rider-cuckoo search algorithm for plant disease detection," *Artif. Intell. Rev.*, vol. 53, pp. 1–26, Feb. 2020, doi: 10.1007/s10462-020-09813-w.

[145] X.-H. Zhou, M.-X. Zhang, Z.-G. Xu, C.-Y. Cai, Y.-J. Huang, and Y.-J. Zheng, "Shallow and deep neural network training by water wave optimization," *Swarm Evol. Comput.*, vol. 50, Nov. 2019, Art. no. 100561, doi: 10.1016/j.swevo.2019.100561.

[146] L. M. R. Rere, M. I. Fanany, and A. M. Arymurthy, "Simulated annealing algorithm for deep learning," *Procedia Comput. Sci.*, vol. 72, pp. 137–144, Jan. 2015, doi: 10.1016/j.procs.2015.12.114.

[147] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, May 1983, doi: 10.1126/science.220.4598.671.

[148] T. Silhan, S. Oehmcke, and O. Kramer, "Evolution of stacked autoencoders," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jun. 2019, pp. 823–830, doi: 10.1109/CEC.2019.8790182.

[149] J.-R. Zhang, J. Zhang, T.-M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization–back-propagation algorithm for feedforward neural network training," *Appl. Math. Comput.*, vol. 185, no. 2, pp. 1026–1037, Feb. 2007, doi: 10.1016/j.amc.2006.07.025.

[150] D. Wang, H. Luo, O. Grunder, Y. Lin, and H. Guo, "Multi-step ahead electricity price forecasting using a hybrid model based on two-layer decomposition technique and BP neural network optimized by firefly algorithm," *Appl. Energy*, vol. 190, pp. 390–407, Mar. 2017, doi: 10.1016/j.apenergy.2016.12.134.

[151] J. Rojas-Delgado, R. Trujillo-Rasúa, and R. Bello, "A continuation approach for training artificial neural networks with meta-heuristics," *Pattern Recognit. Lett.*, vol. 125, pp. 373–380, Jul. 2019, doi: 10.1016/j.patrec.2019.05.017.

[152] R. K. Yadav and Anubhav, "PSO-GA based hybrid with adam optimization for ANN training with application in medical diagnosis," *Cognit. Syst. Res.*, vol. 64, pp. 191–199, Dec. 2020, doi: 10.1016/j.cogsys.2020.08.011.

[153] G. Bo, H. Hejuan, H. Hui, and R. Yan, "Hybrid PSO-BP neural network approach for wind power forecasting," *Int. Energy J.*, vol. 17, no. 4, pp. 211–222, 2017. [Online]. Available: http://www.rericjournal.ait.ac.th/index.php/reric/article/view/1690

[154] M. Xu, H. Chen, and L. Duan, "A combined training algorithm for RBF neural network based on particle swarm optimization and gradient descent," in *Proc. IEEE 9th Data Driven Control Learn. Syst. Conf. (DDCLS)*, Nov. 2020, pp. 702–706, doi: 10.1109/DDCLS49620.2020.9275049.

[155] R. Mohapatra, S. Saha, C. A. C. Coello, A. Bhattacharya, S. S. Dhavala, and S. Saha, "AdaSwarm: Augmenting gradient-based optimizers in deep learning with swarm intelligence," May 2020, *arXiv:2006.09875*. [Online]. Available: http://arxiv.org/abs/2006.09875

[156] A. Seifi and F. Soroush, "Pan evaporation estimation and derivation of explicit optimized equations by novel hybrid meta-heuristic ANN based methods in different climates of Iran," *Comput. Electron. Agricult.*, vol. 173, Jun. 2020, Art. no. 105418, doi: 10.1016/j.compag.2020.105418.

[157] H. Tran-Ngoc, S. Khatir, H. Ho-Khac, G. De Roeck, T. Bui-Tien, and M. A. Wahab, "Efficient artificial neural networks based on a hybrid metaheuristic optimization algorithm for damage detection in laminated composite structures," *Composite Struct.*, vol. 262, Apr. 2021, Art. no. 113339, doi: 10.1016/j.compstruct.2020.113339.

[158] S. Hegde and M. R. Mundada, "A hybrid approach of deep learning with cognitive particle swarm optimization for the big data analytics," in *Proc. 9th Int. Conf. Comput., Commun. Netw. Technol. (ICCCNT)*, Jul. 2018, pp. 1–5, doi: 10.1109/ICCCNT.2018.8494200.

[159] J. Cao, H. Cui, H. Shi, and L. Jiao, "Big data: A parallel particle swarm optimization-back-propagation neural network algorithm based on MapReduce," *PLoS ONE*, vol. 11, no. 6, Jun. 2016, Art. no. e0157551, doi: 10.1371/journal.pone.0157551.

[160] C. Saranya and N. Nagarajan, "Efficient agricultural yield prediction using metaheuristic optimized artificial neural network using Hadoop framework," *Soft Comput.*, vol. 24, pp. 1–11, Jan. 2020, doi: 10.1007/s00500-020-04707-z.

[161] I. M. Coelho, V. N. Coelho, E. J. D. S. Luz, L. S. Ochi, F. G. Guimarães, and E. Rios, "A GPU deep learning metaheuristic based model for time series forecasting," *Appl. Energy*, vol. 201, pp. 412–418, Sep. 2017, doi: 10.1016/j.apenergy.2017.01.003.

[162] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proc. Nat. Acad. Sci. USA*, vol. 99, no. 12, pp. 7821–7826, Jun. 2002, doi: 10.1073/pnas.122653799.

[163] A. Lancichinetti and S. Fortunato, "Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 80, no. 1, Jul. 2009, Art. no. 016118.

[164] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 78, no. 4, Oct. 2008, Art. no. 046110.

[165] W. W. Zachary, "An information flow model for conflict and fission in small groups," *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977, doi: 10.1086/jar.33.4.3629752.

[166] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behav. Ecol. Sociobiol.*, vol. 54, no. 4, pp. 396–405, Sep. 2003, doi: 10.1007/s00265-003-0651-y.

[167] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 1–35, Aug. 2013, doi: 10.1145/2501654.2501657.

[168] J. Leskovec and J. J. Mcauley, "Learning to discover social circles in ego networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 539–547.

[169] J. Leskovec, D. Huttenlocher, and J. Kleinberg, "Signed networks in social media," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2010, pp. 1361–1370, doi: 10.1145/1753326.1753532.

[170] B. Rozemberczki, R. Davies, R. Sarkar, and C. Sutton, "GEMSEC: Graph embedding with self clustering," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, Aug. 2019, pp. 65–72, doi: 10.1145/3341161.3342890.

[171] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 181–213, Jan. 2015, doi: 10.1007/s10115-013-0693-z.

[172] H. Van Lierde, T. W. S. Chow, and G. Chen, "Scalable spectral clustering for overlapping community detection in large-scale networks," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 4, pp. 754–767, Apr. 2020, doi: 10.1109/TKDE.2019.2892096.

[173] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. Election: Divided they blog," in *Proc. 3rd Int. Workshop Link Discovery (LinkKDD)*, 2005, pp. 36–43, doi: 10.1145/1134271.1134277.

[174] T. Yang, R. Jin, Y. Chi, and S. Zhu, "Combining link and content for community detection: A discriminative approach," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 927–936, doi: 10.1145/1557019.1557120.

[175] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI Mag.*, vol. 29, no. 3, p. 93, Sep. 2008, doi: 10.1609/aimag.v29i3.2157.

[176] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graphs over time: Densification laws, shrinking diameters and possible explanations," in *Proc. 11th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2005, pp. 177–187, doi: 10.1145/1081870.1081893.

[177] M. E. J. Newman, "Finding community structure in networks using the eigenvectors of matrices," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 74, no. 3, Sep. 2006, Art. no. 036104, doi: 10.1103/PhysRevE.74.036104.

[178] O. Shchur and S. Günnemann, "Overlapping community detection with graph neural networks," 2019, *arXiv:1909.12201*. [Online]. Available: http://arxiv.org/abs/1909.12201

[179] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," *ACM Trans. Web*, vol. 1, no. 1, p. 5, May 2007, doi: 10.1145/1232722.1232727.

[180] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney, "Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters," *Internet Math.*, vol. 6, no. 1, pp. 29–123, Jan. 2009, doi: 10.1080/15427951.2009.10129177.

[181] R. Liu, A. Agrawal, W.-K. Liao, A. Choudhary, and Z. Chen, "Pruned search: A machine learning based meta-heuristic approach for constrained continuous optimization," in *Proc. 8th Int. Conf. Contemp. Comput. (IC)*, Aug. 2015, pp. 13–18, doi: 10.1109/IC3.2015.7346645.
[182] W. Lin, X. Kong, P. S. Yu, Q. Wu, Y. Jia, and C. Li, "Community detection in incomplete information networks," in *Proc. 21st Int. Conf. World Wide Web (WWW)*, Apr. 2012, pp. 341–350, doi: 10.1145/2187836.2187883.
[183] D. J. DiTursi, G. Ghosh, and P. Bogdanov, "Local community detection in dynamic networks," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Nov. 2017, pp. 847–852, doi: 10.1109/ICDM.2017.103.
[184] X. Li, Y. Wu, M. Ester, B. Kao, X. Wang, and Y. Zheng, "Semi-supervised clustering in attributed heterogeneous information networks," in *Proc. 26th Int. Conf. World Wide Web*, Apr. 2017, pp. 1621–1629, doi: 10.1145/3038912.3052576.

MOHAMMED NASSER AL-ANDOLI received the B.Sc. degree in computer information system from Mutah University, Jordan, in 2011, and the M.Sc. degree in computer science from the Jordan University of Science and Technology, Jordan, in 2016. He is currently a Graduate Research Assistance (GRA) with Multimedia University (MMU). His main research interests include complex networks analysis, machine learning, high-performance computing, deep learning, and parallel computing.

SHING CHIANG TAN received the B.Tech. (Hons.) and M.Sc. (Eng.) degrees from Universiti Sains Malaysia, Malaysia, in 1999 and 2002, respectively, and the Ph.D. degree from Multimedia University, Malacca, Malaysia, in 2008. He is currently an Associate Professor with the Faculty of Information Science and Technology, Multimedia University. His current research interests include computational intelligence (artificial neural networks, evolutionary algorithms, fuzzy logic, and decision trees), deep learning and their applications, data classification, condition monitoring, fault detection and diagnosis, stroke rehabilitation, and biomedical disease classification and optimization. He was a recipient of the Matsumae International Foundation Fellowship, Japan, in 2010.

WOOI PING CHEAH received the B.S. degree in physics and computer science from Campbell University, Buies Creek, NC, USA, in 1986, the M.Sc. degree in computer science from Universiti Sains Malaysia, in 1993, and the Ph.D. degree in computer science from Chonnam National University, South Korea, in 2009. He is currently a Visiting Assistant Professor with the School of Computer Science, University of Nottingham Ningbo China. He is also working on explainable AI through the integration of deep learning and Bayesian reasoning. His current research interests include artificial intelligence, knowledge engineering, and soft computing.

SIN YIN TAN received the B.Sc. degree (Hons.) in statistics and the master's degree in applied statistics from Universiti Putra Malaysia, Malaysia, in 2006 and 2008, respectively. She is currently a Lecturer with the Faculty of Information Science and Technology, Multimedia University. Her current research interests include deep learning, Bayesian reasoning, and applied statistics.