# Community Detection in Social Networks (clustering)

Dimitrios Vasdekis
School of Informatics
Aristotle University
Thessaloniki, Greece
vasdekis@csd.auth.gr

Doxakis Chovardas
School of Informatics
Aristotle University
Thessaloniki, Greece
doxakick@csd.auth.gr

Kostantinos Serderidis
School of Informatics
Aristotle University
Thessaloniki, Greece
kserderc@csd.auth.gr

Theodoros Liapikos
School of Informatics
Aristotle University
Thessaloniki, Greece
tliapikos@csd.auth.gr

## ABSTRACT

Revealing the community structure, which is of great importance in understanding the properties of a network, is a problem in network analysis affecting various fields in science and economy. There is a vast amount of literature about this topic approaching the problem from various aspects. In this paper a review of eight published scientific papers regarding community detection in social networks using clustering is sought. Each reviewed paper was chosen for being contemporary and for attempting to address the community detection problem in a novel approach based on previous work in this field. The key aspects of the proposed algorithm in each paper is presented in brief and a critique of the adapted experimental method along with possible limitations is given at the end of each section. At the end of the paper a set of take-away interesting conclusions is drawn.

## KEYWORDS

Community detection; clustering; social networks; structure mining; algorithm's review.

## 1. INTRODUCTION

In social network analysis a network is represented by a graph where the vertices are the entities of the network and the edges are their connections. Most networks display a community structure where the vertices are organized into groups called communities. Community structures are found in many real-world networks used in various fields such as biology, engineering and economy to name a few. Defining communities is crucial in understanding the features and properties of a network. It allows us to understand how the network is organized, study the parts that have a degree of autonomy within the network and how these parts are connected [1]. Inside the community, we can understand the role each vertex has in its community and classify them accordingly. Unfortunately detecting these communities is most of the times a challenging task and in order to tackle it various approaches have been proposed.

Communities in networks are also called clusters. Clusters are usually considered to be groups of nodes in which the connections among members of the group are denser than the connections among members of different groups [8].

Community detection generally speaking, aims to group nodes of a network according to one or more aspects of the relationship between them. Since networks are usually represented as graphs, the goal of community detection is to identify strongly connected subgraphs from the entire graph.

Extensive research work has been made and various algorithms have been proposed to address the problem of community detection. A somehow arbitrary classification of these algorithms could be:

a. Traditional methods that include graph partitioning, hierarchical, partitional and spectral clustering.

b. Modern methods that include divisive, modularity-based, spectral, dynamic, based on statistical inference and alternative methods.

c. methods to find overlapping communities.

d. multiresolution methods.

e. methods for detecting dynamic communities.

Given this huge diversity in various approaches is not easy to analyze, compare and evaluate existing work.

In this paper a study of eight scientific papers where each of them proposes a novel approach in community detection by presenting a new algorithm is attempted. Each paper was chosen for being contemporary, for implementing a new algorithm and for applying a methodology where experimental results and metrics are being taken into account in order for the authors to evaluate their algorithms' performance. It is worth noting that in each of these papers the authors claim that their method, based on experimental results, addresses a specific aspect of the problem of community detection in a more efficient way than the current state-of-the-art algorithms.

The algorithms to be reviewed in this paper are:

a. HICODE, a Hidden Community Detection Algorithm in Social Networks: In this paper the algorithm HICODE is proposed

in order for one to identify smaller communities that cannot be found by conventional community detection algorithms due to interference by stronger communities in the network. The authors, present evidence that the use of HICODE can also increase the quality of stronger communities identified besides finding the hidden ones.

b. ASOCCA, a Community Detection Algorithm Based on Local Similarity of Clustering Coefficient in Social Networks: This paper proposes an algorithm that utilizes a similarity index presented by the authors and based on this, forms an initial community structure for the network. Later on, through a proposed merging strategy the optimal community structure is retrieved and the final network occurs. The authors present evidence that indicate a better performance in most given networks than other state-of-the-art algorithms.

c. EADP, an Extended Adaptive Density Peaks Clustering Algorithm for Overlapping Community Detection in Social Networks

d. DSCAN, a Distributed Algorithm for Large-Scale Graph Clustering

e. ABCD, Weighted Graph Clustering for Community Detection of Large Social Networks

The concept of weighted networks to represent the real-world social networks is analyzed in this paper [2]. The aim is to identify the number of communities by deploying a less complex algorithm. The evaluation of the algorithm is performed by three well-known datasets.

f. Weighted Adjacent Matrix for K-means Clustering

This paper [3] introduces an improved version of the classic K-means clustering algorithm by building two new adjacent matrices to represent the original data points. The performance improvement is demonstrated against three comparison clustering algorithms (k-means, k-means++, and normalized spectral clustering algorithms) using variable metrics.

g. DCK, Density-Canopy-K-means Algorithm using MDS Embedding. A Community Detection and Visualization in Complex Network algorithm.

h. CC-GA. A clustering coefficient based genetic algorithm for detecting communities in social networks.

For each of the above papers, a quick review of the algorithms' workflow, some necessary definitions, the metrics and methods used in evaluation and a brief critique, is given in the second section of this review.

The rest of this paper is organized as follows: In the second section, as mentioned above, is the review of each one of the eight algorithms and finally, a set of conclusions is drawn at the last section.

## 2. PAPER REVIEWS

### 2.1. Weighted graph clustering for community detection of large social Networks

The paper [2] discusses the weighted information concept application in the community detection problem in social networks, in particular large-scale ones.

The means deployed to resolve the problem is **graph clustering**. The social network is classified as a graph, each person-user is a represented by a network node, whereas the relationship between persons is represented by the network edges. Each person (or a community) is assigned a density value, node weight and each pair of persons (or communities) an attractiveness value, the edge weight.

A social network-graph can split into a set of sub-graphs (clusters), identified by an optimization function based on a weight value. If we consider a sparse graph, G (V, E, $W_V$, $S_E$) which consists of the node set V, the edge set E, the weight of the node set $W_V$, and the weight of edge set $S_E$, communities are identified as the Graph G clusters.

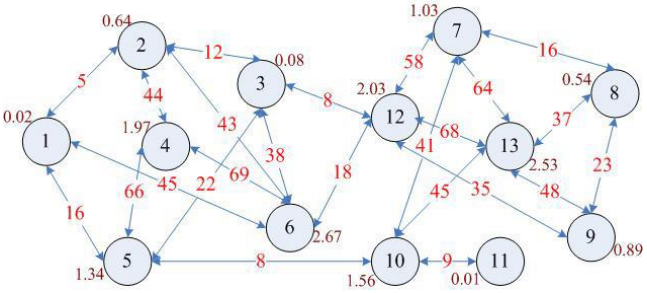


**Figure 1 Weighted Graph example [2]**

### 2.1.2 Methodology

Provided that the No of edges between clusters i and j is **q**, the edgesweight **Se**, e ∈ 1, 2, ..., q, communities i and j have **Qi** and **$Q_j$** nodes respectively, the node weight is **$W_a$**, a set of definitions is constructed [2]:

**Cluster density:** $W_i = \frac{\sum_{a=1}^{Qi} W_a}{Qi}$

**Attractiveness** between clusters i and j:

$$Sij = \frac{\sum_{e=1}^{q} Se}{QixQj}$$

**Inter-interested** clusters: Clusters i and j are inter-interested, if both conditions q ≥ Qi, q ≥ $Q_j$ apply.

**Community**: A cluster i is classified as community if

$$S_{ij} < W_i + W_j \ \forall j$$

An agglomerative algorithm named **A**ttractiveness-**B**ased **C**ommunity **D**etection (**ABCD**) is developed [2]. The rationale is to merge the pair of clusters with the highest value of attractiveness up to the point where the cluster structure remains unchangeable. Starting with each node as a single cluster, namely i., the goal is to identify the cluster with the highest value of attractiveness among all its inter-interested clusters to the starting one, say this is j. The clusters are merged if the following condition applies:

$$Sij \geq Wi + Wj$$

The prerequisites are that inter-interested clusters do exist, the candidate clusters have weight higher than the attractiveness

with other inter-interested clusters, and the adopted principle is that there is no preference among possible inter-interested clusters.

An Attractiveness matrix S is built, where each element $Sij$ represents the attractiveness between the respective clusters i, j. At each iteration, each merging, the density and the attractiveness values change, and the matrix is iteratively updated.

### 2.1.2 Results

The experiments involve:

Sina Weibo, a Chinese micro-blogging website. The ABCD algorithm takes into account info such as the number of interested users (in-degree/user activity), the number of fans (out-degree/user popularity), the number of inter-interested users, as opposed to classic algorithms like CNN, which consider only the interrelations between users. The algorithm has proven to provide better results.

College Football dataset: the network nodes represent teams and edges represent the game between two teams. Teams belong in 12 conferences in US, and the ones in the same conference have a higher degree of inter-connections (more frequent games). The algorithm identified 11 communities (no prior knowledge of the conferences number).

Renren: a Chinese social network service (like Facebook).

The authors state that that in the experimental results, the number of iterations is much smaller than the number of nodes, especially for large-scale networks, and the number of mergers is of several orders of magnitude smaller than the number of nodes.

### 2.1.3 Critique

The paper [2] introduces the application of the weighted concept with no extensive literature review though, as the related work is not detailed to a great extent.

On the other hand, the algorithm description is adequate with clear and simple definitions provided. The proposed algorithm and its pseudo code are presented stepwise and explained.

Regarding the results evaluation, only a single metric is used (number of identified communities), which raises questions on the performance evaluation reliability. The undirected graph is also the only type considered. Moreover, the testing is limited as regards the attributes used, since certain ones are exempted for simplification putting constraints on the actual user related data. Time complexity is demonstrated by the improvement with a reference to the experimental results output. In the end, there is a very short reference to the planned activities to enhance the approach adopted.

## 2 Weighted adjacent matrix for K-means clustering

The k-means clustering is a renowned algorithm for applications in data mining. The goal of k-means clustering is to achieve the minimum sum-squared-error (SSE), i.e. the minimal total intra-cluster variance by a given cluster number k.

Given a set of input data points X = {$x_1$, …, $x_n$} represented as matrix X, the SSE is defined as:

$$SSE = \sum_{j=1}^{k} \sum_{i=1}^{tj} |xi - cj|^2$$

where $c_j$ is the centroid, $t_j$ is the number of data points in the respective j cluster.

The K-means faces certain limitations such as the determination of the cluster number k, the centroids initialisation and the similarity measurements definition (for evaluating the similarity between two data points).

### 2.1.2 Methodology

Two novel clustering methods: The **A**djacent **M**atrix-based **K-Me**ans clustering method (**AMKM**) and the **W**eighted **A**djacent matrix-based k-means clustering method (**WAMKM**) are introduced [3]. These methods are based on building two new representations of the original datasets, i.e., an Adjacent matrix and a Weighted Adjacent matrix. Following that, k-means clustering is applied on the new representations to output the clustering results.

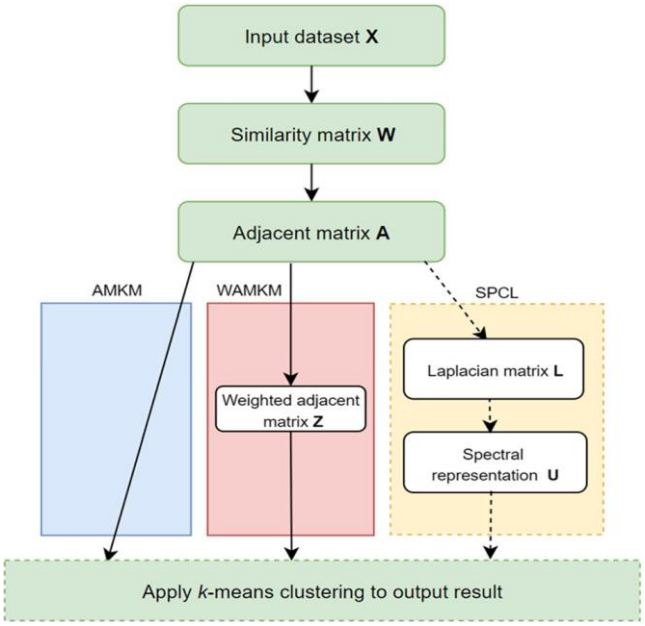


**Figure 2 AMKM, WAMKM and SPCL with k-means [3]**

The datapoints are represented by an undirected graph G, G= (V, E): V = {$v_1$, …, $v_n$} are the vertices, and E = {$e_1$, …, $e_m$} the edges between vertices, and assuming a fully connected graph m = n× (n-1)/2).

The algorithm is based on the construction of the similarity matrix as the data points representation in the undirected graph, called $W = (wi, j)_{i,j=1}^{n}$ where $w_{i,j} \geq 0$ is the similarity between xi and xj under a given a distance metric. The adjacent matrix A is constructed based on W by determining the distance between two

data points via a similarity function, i.e. a kernel function like Polynomial, Gaussian or Sigmoid. The novelty is that the k-means clustering is applied directly on the adjacent matrix instead of the Laplacian eigenvector matrix of the classic approach for simplification and complexity reduction.

The weighted adjacent matrix-based k-means clustering follows the same approach but is to be more representative of a real test case where data points consists of multiple features with different priorities and different influence on the clustering output. The features are assigned a weight as a percentage of each one among all features. A new similarity weighted adjacent matrix is produced by the application of a weighted vector on the original adjacent matrix.

In both methods, the k-means clustering is then applied to the Adjacent matrix produced in the former step.

### 2.1.2 Results

To evaluate the two clustering methods, as regards the reliability and effectiveness of the methods, twelve datasets were selected by the UCI Machine Learning Repository and the data mining centre website, of different categories and wide range varieties of characteristics.

The evaluation was done in terms of three clustering metrics [accuracy (ACC), normalised mutual information (NMI) and purity (PUR)] [3], and against three comparison clustering algorithms (k-means, k-means++, and normalised spectral clustering algorithms). The tests are dependent on the adjustable parameter σ which controls the similarity between two data points and has an impact on the kernel performance. The result is that the proposed algorithms outperform the compared algorithms, k -means, k-means++ and spectral clustering algorithms, in terms of the respective metrics ACC, NMI and PUR. However, the proposed clustering methods are data-driven and their performance can vary on different types of datasets.

### 2.1.3 Critique

The authors demonstrate in a convincing manner their methodology. The literature review is quite extensive, demonstrating the pros and cons of different algorithms to justify the development of the proposed new ones. Different metrics in conjunction with a variety of datasets and algorithms provide comparative values projecting the results.

Nevertheless, there is no concrete data on the time complexity. Furthermore, the results are data-driven. The importance of σ parameter has been highlighted, which is apparently the drawback in the proposed method. The algorithm provides a new approach based on the widely used k-means algorithm, even though there are no concrete experimental results, at this stage, as regards the community detection in real large social networks with numerous communities, since attention is mostly drawn to multiple features. The future work is only briefly addressed.

## 2.3. Hidden Community Detection in Social Networks

In this paper a new paradigm that is important for community detection in real-world networks is introduced. Networks contain a set of strong, dominant communities thus interfering with the detection of weak, natural community structure. When a large proportion of the members of the weak communities are also members of stronger ones, they are extremely hard to be uncovered. The authors refer to weak communities as "hidden community structure".

They propose an approach called HICODE (Hidden Community Detection). It identifies the hidden community structure as well as the dominant one. The main idea is that by weakening the dominant structure one would be able to uncover the hidden structure beneath. Also, by weakening the hidden structure a more accurate identification of the dominant structure could occur.

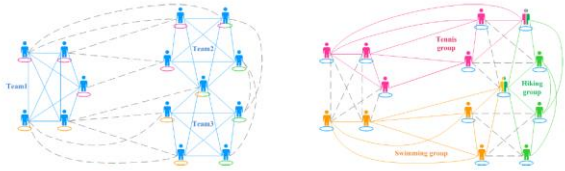


**Figure 3 Illustration of dominant and hidden communities**

The authors in order to address the hidden community issue defines the hiddenness value of a community as the portion of its nodes that also belong to a stronger community [4]. HICODE begins by applying a base community detection algorithm to a network. Then the structure of the detected communities is weakened using methods that are presented later in this paper, making the hidden community structure visible. This step is repeated until no further significant community structure can be detected. Finally, HICODE weakens the structure of the hidden communities, thus obtaining a more accurate representation of the dominant community structure.

### 2.3.1 Metrics

The authors in order to measure the strength of a set of communities that partition the network, adopt the modularity metric. Modularity is the ratio of the number of intra-community edges to the expected number of edges in the same set of communities if the edges had been distributed randomly while preserving degree distribution [6].

Also, in order to compare the similarity of two partitions the Normalized Mutual Information (NMI) is used. This metric is utilized to compare the similarity between HICODE result partitions and ground-truth partitions of networks in the dataset used [7].

### 2.3.2 Definitions

#### 2.3.2.1. Hiddenness Value

Assuming that we have some metric function $F: (G, C_k) \rightarrow \mathbb{R}$ that assigns a quality score to a community $C_k$, let $F_k$, the higher $F_k$ is, the stronger the community it denotes. Therefore, the hiddenness value $H(C_k)$ of community $C_k$, is the fraction of nodes of $C_k$ belonging to various communities with higher $F$ score.

Let $\mathbb{S}_k$ be the set of all strong communities for community $C_k$.

$$\mathbb{S}_k = \{C_i | F_i > F_k, C_i > C\}$$

The hiddenness value of $C_k$ is:

$$H(C_k) = \frac{1}{|C_k|} \cdot \left| \bigcup_{C_i \in \mathbb{S}_k} C_i \cap C_k \right|$$

$H(C_k) \in [0, 1]$. The higher a community' s hiddenness value is, the more difficult is for the community to be uncovered [4]. The goal of HICODE is to locate overlapping communities in a network in a way that such communities can be found.

#### 2.3.2.2. Community Layer

Layer is a set of communities that partitions or covers the nodes of the network [4].

#### 2.3.2.3. Layer's Hiddenness Value

The hiddenness value $H(L_i)$ of a layer $L_i$ is the weighted average hiddenness values of the communities in this layer.

$$H(L_i) = \frac{\sum_{C_k \in L_i} |C_k| \cdot H(C_k)}{\left|\sum_{C_k \in L_i} C_k\right|}$$

the dominant layer is the layer with lowest hiddenness value. It is usually the set of communities found by a standard community detection algorithm that optimizes metric $F$ [4].

### 2.3.3. Algorithm Workflow

HICODE contains two stages: Identification and Refinement.

**Stage 1. Identification:**

This stage determines the initial layers of communities as follows [4]:

(1) Identify a layer using the base community detection algorithm.

(2) Weaken the structure of the detected layer.

(3) Repeat until the appropriate number of layers are found.

A crucial aspect of this stage is to automatically determine the number of layers $n_L$ in a network. This is accomplished by increasing $n_L$ until a stopping condition is met. This process is described below.

**Stage 2. Refinement:**

After the identification stage one has only a rough approximation of the various community structures, because as mentioned before, stronger communities can obscure weaker ones and vice versa. Due to this fact the refinement stage is needed in order to improve the quality of the detected community structures. Refinement is an iterative process. In each iteration we consider each layer and improve the current layer as follows [4]:

(1) Weaken the structure of all other layers to obtain a reduced network.

(2) Apply the base algorithm to the resulting network.

The key difference between weakening the layers in identification and refinement stage is that in the first we weaken only the layers found so far (i.e., the stronger layers) and in the later we weaken both the stronger and weaker layers. This is necessary because the weaker layers can impair detection of the layer currently under consideration, even though they have a smaller impact on the network structure than the stronger layers. Through this process, a more accurate version of the current layer is produced.

### 2.3.4. Weakening Methods

The authors propose three methods to reduce a single layer of community structure: RemoveEdge, ReduceEdge, and ReduceWeight. RemoveEdge works well when the number of layers is small and the overlapping of communities in different layers is small. The best-performing method is ReduceWeight but it has the disadvantage that it requires a base algorithm that supports weighted networks. If there are overlapping communities in a layer in order to avoid duplicate weakening, the communities are first being sorted according to their size and then, larger communities are being weakened first and the overlapping portion is ignored for subsequent communities.

**RemoveEdge** weakens a community by removing all intra-community edges.

**ReduceEdge** approximates each layer as a single stochastic block-model, where other edges are regarded as background noise. This method randomly removes some edges within each community block so that the edge probability in the block matches the background edge probability of this block.

**ReduceWeight**.This method reduces the weight of each community by a factor $q_k$ which is the edge probability of the background block. Like with ReduceEdge, we wish to set the weighted probability within the community equal to the average weighted background block probability $q_k$.

### 2.3.5. Number of Layers ($n_L$) Selection

Selecting the proper number of community layers $n_L$ is a major challenge for HICODE according to the authors and experimental results have proven that if $n_L$ is chosen correctly during the refinement stage, the average modularity of the detected layers increases. Otherwise this trend declines.

The authors based on this observation establish a rule for determining the best $n_L$: if one selects the appropriate number of layers, the output will generally be of higher quality.

At the beginning let $n_L = 2$ and increase until a stopping condition is met. For each candidate number of layers $n_L$, HICODE first calculates the modularity of the weakest layer obtained at the Identification stage. If the value is very low, then there are no more significant layers, so we set $n_L = n_L - 1$ and return; otherwise, let $Q_t$ be the average modularity of all the detected layers at step $t$:

(1) Calculate $Q_0$ for $t = 0$, i.e. after identification, before any refinement is conducted;

(2) Perform $T = 10$ tentative iterations of refinement, and calculate $Q_t$ for each $t \in \{1, \dots, T\}$

(3) Calculate the average improvement ratio of modularity per iteration:

$$R_T = \frac{\sum_{t=1}^{T} Q_t}{T \cdot Q_0}$$

$R_T$ represents how much refinement improves the detected layers. If $n_L$ is too high, then when we remove the structure of the extra layers, we will be removing structure that actually belongs to some earlier layer. This will result in a lower quality partition, such that the refinement stage actually lowers the quality of the detected layers. Thus, we choose $n_L$ corresponding to the peak $R_T$ [4].

### 2.3.6. Critique

The authors use adequate metrics to evaluate the performance of the algorithm. They use modularity which is a popular measure of the quality of a community in network analysis and also NMI and Jaccard-based Precision, Recall, and F-score metrics to measure the similarity between their experimental results and ground-truth communities of the dataset used. As for the dataset used, there are two groups of real-world networks with various sizes. The properties of the dataset's networks are as follows:

| Source | Domain | Dataset | $\|V\|$ | $\|E\|$ |
|---|---|---|---|---|
| Facebook | Social | Caltech | 769 | 16,656 |
| | | Smith | 2,970 | 97,133 |
| | | Rice | 4,087 | 184,828 |
| | | Vassar | 3,068 | 119,161 |
| | | Wellesley | 2,970 | 94,899 |
| | | Bucknell | 3,826 | 158,864 |
| | | Carnegie | 6,637 | 249,967 |
| | | UIllinois | 30,809 | 1,264,428 |
| SNAP | Social | YouTube | 31,150 | 202,130 |
| | Products | Amazon | 13,288 | 41,730 |
| | Collaboration | DBLP | 49,097 | 170,284 |

Table 1 Real-world Datasets

As for scalability the HICODE algorithm is strongly depended on the base algorithm it uses. If the base algorithm is scalable, so will be HICODE. In this direction, the authors use in their evaluation experiments four different base algorithms, divided in two categories: Overlapping Detection Methods and Disjoint Detection Methods which is an adequate sample to draw experimental conclusions.

## 2.4. A Novel Community Detection Algorithm Based on Local Similarity of Clustering Coefficient in Social Networks

In this paper, an adjacent node similarity optimization combination connectivity algorithm (ASOCCA) is proposed for accurate community detection. ASOCCA uses the local similarity measure based on clustering coefficient to identify the nearest neighbors of each node, then obtains various sets of connected components combining different node pairs and finally initial community is formed. In addition, a community merging strategy is implemented to further optimize community structure. ASOCCA focuses on the non-overlapping community detection for undirected and unweighted networks.

In order to achieve what is described above a measure of local similarity between nodes based on clustering coefficient is proposed. It is given as follows:

$$Sim_{ij} = \sum_{t \in \Gamma(i) \cap \Gamma(j)} CC_t$$

where $CC_t$ is the local clustering coefficient of $v_t$, and $v_t$ is the common neighbor of node $v_i$ and $v_j$. This similarity measures cumulant of the degree of aggregation of two nodes' common neighbors. The cumulative value is determined by two aspects: one is the number of common neighbors and the other is the clustering coefficient of each common neighbor. The greater the number of common neighbors, the closer the relationship between the two nodes. It should be made clear that ASOCCA takes into account the similarity only of adjacent nodes and mainly considers the neighbors with the highest similarity among all neighbors of $v_i$. The most similar node of $v_i$ is given by,

$$j = argmax_{j \in \Gamma(i)} Sim_{ij}$$

### 2.4.1. Algorithm Workflow

a. Compute the clustering coefficient of each node and list it.
b. Calculate the local similarity of adjacent nodes.
c. Retrieve the neighbors with the highest similarity for each node.
d. If a node has more than one neighbor with the highest similarity, create all possible pair sets using combinations.
e. Generate the sets of connected components for each combination. Each connected component is a non-overlapping community at this point.
f. Eliminate duplicate connected component groups. For large-scale networks, limit the maximum number of combinations to 100.
g. Once the initial partition is ready, merge some small communities to make the community structure more reasonable and achieve a better modularity. A proper merging threshold $\lambda$

should be set such that small communities with fewer nodes than the threshold $\lambda$ will be merged into other communities.

h. The merging strategy proposed is as follows:

i. Select the node with the largest clustering coefficient in the small community.

ii. Compare the degrees of its neighbors that are not in the small community.

iii. Select the community where the neighboring node with the highest degree is located

iv. Merge the small community into the selected community.

i. Set the appropriate threshold to perform the merging of small communities for each set of connected components. Then modularity is calculated separately on each community structure after merging, and the community structure corresponding to the highest modularity is the final detection result.

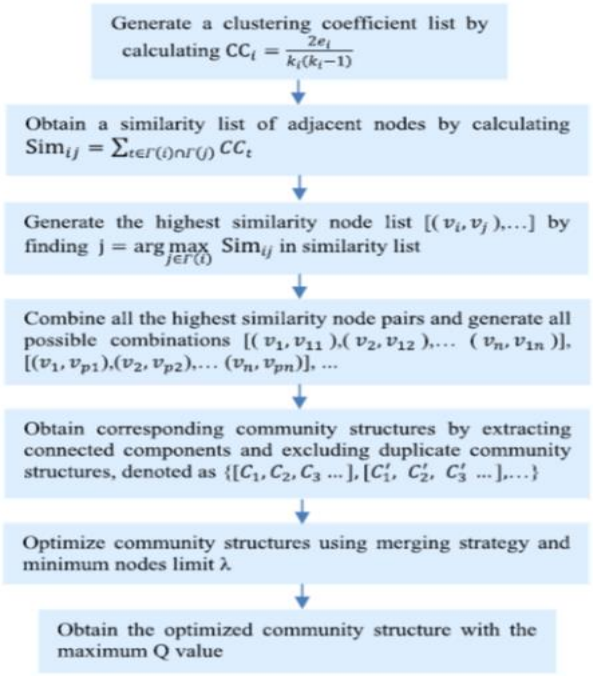


Figure 4 ASOCCA's Workflow Overview

### 2.4.2. Critique

In this paper the authors use modularity and NMI in their experiments which are commonly accepted metrics in network analysis. In order to measure similarity of the nodes, the authors propose a new similarity index based on clustering coefficient instead of any of the commonly used similarity indexes presented in the table below, claiming better experimental results.

| similarity index | Formula |
|---|---|
| Salton index | $\frac{\lvert\Gamma(i)\cap\Gamma(j)\rvert}{\sqrt{k_i k_j}}$ |
| Jaccard index | $\frac{\lvert\Gamma(i)\cap\Gamma(j)\rvert}{\lvert\Gamma(i)\cup\Gamma(j)\rvert}$ |
| Sørensen Index | $\frac{2\lvert\Gamma(i)\cap\Gamma(j)\rvert}{k_i+k_j}$ |
| Resource Allocation (RA) index | $\sum_{t\in\Gamma(i)\cap\Gamma(j)} \frac{1}{k_t}$ |

Table 2 Local Similarity Index

Regarding the datasets the authors use six real-word and two LFR networks with size ranging from tens to hundreds of thousands, involving small, medium and large-scale networks to evaluate their algorithm. One consideration that might occur about the results of ASOCCA is that as stated by the authors, experiments have shown that real-world communities don't always have the highest modularity so since this metric is used to decide the network's final structure the real community structure could be abandoned. Another point that could be made is that in large-scale networks for computational reasons, only the first one-hundred combinations are taken into account. This limitation could lead in failure finding the optimal community structure of a given network. Finally, the merging strategy proposed, is the one that gave the best experimental results. There is no strongly documented reason to follow the proposed approach regarding merging.

## 2.5 EADP, An Extended Adaptive Density Peaks Clustering for Overlapping Community Detection in Social Networks

In this paper is addressed the problem of finding the overlapping communities of a social network. In most networks of all types, communities are not disjoint but overlap each other to some degree. The existing overlapping community detection methods are not taking network weights in consideration and if they do, they are not very accurate in complex weighted networks. Density Peaks Clustering (DPC) method [9] can find communities accurately based on the assumption that cluster centers are surrounded by lower density nodes and that the distance between cluster centers is big. But PDC cannot detect overlapping communities and requires a distance matrix when in most cases we are provided with an adjacency matrix. Lastly in DPC cluster centers are not decided automatically and a human must appoint them manually which is very difficult in large scale networks.

The authors propose a new method of clustering, the Extended Adaptive Density Peaks (EADP) clustering that is an improvement of the DPC method in the following points:

a. It extends DPC in order to be able to detect overlapping communities.

b. It calculates the distance between nodes with a new function that takes into consideration the weights of the edges and thus can be implemented in weighted and unweighted networks.
c. It implements a linear fitting strategy of deciding which nodes will be the centers of the clusters.

### 2.5.1 Methodology

Because the proposed method is based on density peak, the author gives a brief introduction on DPC method. In DPC, the centers of each cluster are the nodes that have higher local density than that of their neighbor nodes and large distance from nodes with higher local density than them. DPC calculates two quantities. The local density ρ and the separation distance δ. The decision graph is drawn with the two quantities as horizontal and vertical axis. The nodes with very particularly large ρand δare becoming the cluster centers and all the remaining nodes are assigned to the cluster that its center has the higher density and the shortest distance from them.

The authors continue with the methodology of EADP. Consider a graph $G = \{V, E_w\}$ where $V$ is a set of vertices and $E_w$ is a set of edges between vertices. For a graph with $n$ nodes, an $n * n$ adjacent matrix $A$ is constructed with $A_{ij}$ is the linking weight between two nodes $v_i$ and $v_j$.

The first step in EADP is the calculation of the distance matrix $D$ from the adjacency matrix $A$. The distance between two nodes is affected by the common nodes of these two. The reason is that even if these two nodes do not connect directly, it is very possible to interact in the future, if they have more common nodes. In order to find the distance between two nodes we first calculate the contribution of the common nodes and the link strength. The contribution that the common nodes do to their link strength is calculated as followed,

$$cc(i,j) = \sum_{p \in V_{ij}} w_{ipj} * \exp\left(-\left(\frac{w_{ipj} - maxw}{r * t + \eta}\right)^2\right)$$

where $V_{ij}$ is the set of the common nodes between nodes $v_i$ and $v_j$, $w_{ipj}$ is the minimum of $A_{ip}$ and $A_{jp}$, $maxw$ is the maximum weight in the whole network, $r$ is the range of weights, $t \in [0,1]$ is a small parameter controlling how significantly a common node $v_p$ influences the link strength between nodes $v_i, v_j$, and $\eta$ is small positive number to avoid zero denominator. Then the link strength,

$$ls(i,j) = \frac{\left(cc(i,j) + A_{ij}\right) * (|V_{ij}| + 1|)}{\min\left(I_i, I_j\right)}$$

where $|V_{ij}|$ is the number of common nodes that the nodes $v_i, v_j$ have links and $I_i, I_j$ is the total weight of outgoing links of nodes $v_i, v_j$. It is noted that the link strength and consequently the distance is affected not only by the contribution of the common nodes but also from their number. Finally, the distance is given by

$$dist(i,j) = \frac{1}{ls(i,j) + \varepsilon}$$

where $\varepsilon$ is a very small non-negative number to avoid zero denominator. For the isolated nodes a very big number $\frac{1}{\varepsilon}$ is assigned automatically.

The method proceeds with the estimation of the local density of the nodes,

$$\rho_i = \sum_{j \in knn_i} \exp\left(-\left(\frac{dist(i,j)}{d_c}\right)^2\right)$$

where $knn_i$ is the set of node's $v_i$ k nearest neighbors and $d_c$ is the cut-off distance. In contrast to DPC, the local density in EADP is calculated by considering only the k-neighbors of the node and not all the nodes of the network. Thus, only the local structure of the network will be considered.

EADP, unlike DCP, decides which nodes will be the cluster centers automatically with a linear fitting method based on the difference of a vector $\gamma_i$. Separation distance is the minimum distance between a node $v_i$ and any other nodes with higher density. Separation distance and local density from the previous step are first normalized and then vector $\gamma_i$ is estimated for every node $v_i$.

$$\gamma_i = \rho_i^* \delta_i^*$$

It is noted that vector $\gamma_i$ is estimated only for those nodes with big enough separation distance and local density because only them have a chance of being cluster centers. The nodes that will become cluster centers are those with anomalously large $\gamma_i$. In order to find the cluster centers, the vectors are sorted in ascending order and the difference vector $d\gamma$ is calculated

$$d\gamma = \{d\gamma_\iota | d\gamma_\iota = \gamma_{\iota+1}^s - \gamma_\iota^s\}$$

Then the index of the biggest element in $d\gamma$ is obtained and it is denoted as $idx$ and its value as $d\gamma_{idx}$. A linear fitting to predict the value $d\gamma_{idx}$is performed

$$\hat{d}\gamma_{idx} = a * idx + b$$

in which $a$ and $b$ are two variables to make the fitting process reach a minimum mean square error and $\hat{d}\gamma_{idx}$ is the predicted value of $d\gamma_{idx}$. Next, the difference between the real value and the predicted value is estimated and it is named as $\Delta\gamma$. If the following condition is true:

$$d\gamma_{idxn} > 2\Delta\gamma$$

this node will be a cluster center and the process continue to the next vertex. If it is not the process is terminated. This process will continue until idx< 3.

Lastly, EADP allocates all the remaining nodes into the clusters in two steps. In the first step, all the remaining nodes are assigned into the same cluster as the node with the higher values in density and minimum distance to it. In the second step, the nodes can become members of an additional cluster. If a node is an envelope node -a node that all the neighbor nodes are in the same cluster- it will not be a member of another cluster. For all the non-envelope nodes the membership $p^c$ is calculated for every cluster that at least one of its k-nearest neighbor nodes belongs.

$$p_i^c = \sum_{j \in knn_i, cl_j = c} ls(i,j) \frac{\sum_{j \in knn_i, cl_j = c} ls(i,j)}{\sum_{j \in knn_i} ls(i,j)}$$

The membership is affected not only by the number of the k-neighbor nodes of this cluster but also by how often cluster label

occur in its own neighbors. Then, the membership of the node is compared with the membership of its original cluster and if it is greater than or equal to a threshold σ, will also become member of this cluster.

$$\frac{p_i^c}{p_i^{cl_i}} \geq \sigma$$

The time complexity of EADP is O(vn² + nlogn+ m + nm) where n is the number of nodes, m the number of clusters and v the average number of common nodes.

### 2.5.2 Experiments

EADP was tested in 12labeled and unlabeled, weighted and unweighted real-world networks and synthetic networks generated form LFR benchmark [10]. It was compared with other algorithms for overlapping community detection which are MOSES [11], SPLA [12], HOCTracer [13], OCDDP [14] are SMFRW [15]. The metric that were used were Ω index [16], F1[17] and Overlapped Normalized Mutual Information (ONMI) [18] for the labeled datasets and the Overlapping version of Modularity ($Q_{ov}$) [19] for the unlabeled. According to the tests on real life networks, the accuracy of EADP was found to the best in the most cases. In structural quality it could get the best $Q_{ov}$. It could also handle weighted networks better than the other algorithms. The experiments on synthetic networks first tested the performance of EADP individually. When the overlapping degree or the number of nodes was growing, the performance of the algorithm was decreasing. Also, when the complexity of the weight distribution was growing, the performance of the algorithm was decreasing, but with slower pace. Compared to the other methods, EADP had better performance in most of the cases.

### 2.5.3 Critique

In this paper, the authors present a new method of finding overlapping communities in social network. This method is based on density peaks clustering (DPC). The paper starts with the related work, it continues with the presentation of the method and concludes with the conducted experiments and their results. In the related work section, the review is extensive and analyses the pros and cons of the relating methods explaining the importance of the proposed method. In the presentation section, the methodology is comprehensive, explaining each part of the algorithm. The conducted experiments are numerous and test the performance of the method individually and compared to 5 related methods of overlapping community detection. From the results it is obvious that the proposed method is generally better in detecting overlapping communities, while in terms of speed is average. The future improvements can be its speed improvement and the ability to handle dynamic networks.

## 2.6 DSCAN, a Distributed Structural Clustering Algorithm for Network

In this paper is addressed the problem of clustering very large graphs. Structural Clustering Algorithm for Networks (SCAN)[20] is an algorithm that detects clusters, hubs and outliers in networks. It is based on graph topology and groups vertices that share the maximum number of neighbors. Also, it computes the similarity between all the edges of the graph in order to perform the clustering. But the similarity computations step is linear to the number of edges, so it is difficult to be applied in large graphs.

The authors propose the Distributed Structural Clustering for Networks (DSCAN) that it is a distributed implementation of SCAN algorithm and is capable of clustering very large graphs.
Methodology

### 2.6.1 Methodology

Initially, the authors explain how SCAN algorithm works and afterwards they propose the distributed version. Consider a graph $G = \{V, E\}$ where $V$ is a set of vertices and $E$ is a set of edges between vertices. If $v, u$ two vertices in $V$ and $(v, u)$ the edge between them, $u$ is said to be a neighbor of $v$.

In the first step of the SCAN algorithm, the structural neighborhood $N(v)$ of each vertex $v$ is computed, which consists of all the neighbor vertices and the vertex itself.

$$N(v) = \{u \in V | (v, u) \in E\} \cup \{u\}$$

Secondly, the structural similarity $\sigma(u, v)$ for each pair of neighboring vertices is calculated, which represents the amount of shared structural neighbors between the two vertices.

$$\sigma(u, v) = \frac{|N(u)| \cup |N(v)|}{\sqrt{|N(u)|.|N(v)|}}$$

If the structural similarity of two vertices is bigger than a threshold $\varepsilon$, it is considered as a strong connection. All the strong connections of a vertex form its ε-neighborhood.

$$N_\varepsilon(u) = \{N(u) | \sigma(u, v) \geq \varepsilon\}$$

If a vertex has more strong connections than a predefined number $\mu$, it is considered as a core vertex.

$$V_c = \{u | N_\varepsilon(u) \geq \mu\}$$

The strong connected neighbors of the core vertex are called borders and weak ones are called noise vertices.

In order to implement the clustering, the algorithm starts from a random core vertex and adds this vertex and all its borders in the cluster. If a border is a core vertex, the algorithm adds its borders to the cluster until adding all the borders of the connected cores. After that, it continues with other cores until checking all of them.

Also, the algorithm identifies two more categories of vertices. A vertex that has no strong connections will not be a part of a cluster. if the neighbors of this vertex belong to at least two different clusters, it will be called a bridge. If not, it will be called an outlier.

The proposed DSCAN method is based on a master/slave architecture and it is implemented on top of BLADYG framework [21], which is a distributed graph processing framework. Master machine's job is to coordinate the slaves are responsible for the execution of a specific computation that it is appointed to them by

the master. The DSCAN consists of two steps: the partitioning step and the clustering step.

In the partitioning step the master machine divides equitably the input graph file into sub-files according to the number of edges and sends the sub files to the workers. Each worker takes a list of vertices and a list of edges from the file and sends the list of vertices in all the other workers, in order to determine the frontier vertices. Frontier vertices are the vertices of a partition that have connections with vertices outside of their partition.

In the clustering step there are two sub-steps, the local clustering and the merging. In the local clustering sub-step, the frontier vertices of each partition are duplicated into the neighbor partitions in order to avoid the loss of information. Then, the clustering is performed in the same way as in the SCAN algorithm. In the merging sub-step, the results from all the workers are combined and the conflicts that arise are solved. There are three types of conflicts, the merging of local clusters, the change of an outlier to bridge and the change of a bridge to outlier.

a. *Merging of clusters*. When a cluster from a partition shares at least a core vertex with another cluster from another partition, then they are merged into one cluster.
b. *Outlier to Bridge*. When two clusters from different partitions share an outlier, this means that it is weakly connected with two different clusters and is a bridge.
c. *Bridge to Outlier*. When there is a bridge between two different clusters from the same partition and in the merging step there is a third cluster from another partition which shares core vertices with the two clusters, then the three are merged and the bridge becomes an outlier.

### 2.6.2 Experiments

DSCAN was tested and compared with basic SCAN, pSCAN [22], anySCAN [23] and ppSCAN [24] in five different real-world large datasets with an increasing number of vertices and edges. In terms of speed, DSCAN is the slowest of all, especially in the smaller graphs, due to additional time costs related to data distribution, synchronization and communication. From the other hand, DSCAN was the only method that succeed in calculating the larger dataset. In terms of scalability the authors showed that the algorithm is horizontally scalable. Also, the execution time slowly rises if there is an increase in the ε value. When ε value increases, the number of outliers becomes larger and thus, the outlier evaluation costs more in terms of time because the check for the outliers requires more communication between the workers. Also, the tests showed that the slower part of the algorithm is the merging part.

### 2.6.3 Critique

In this paper, the authors present an implementation of the existing method SCAN, for processing very large graphs. The paper starts with the related work, it continues with the presentation of the method and concludes with the conducted experiments and their results. In the related work section, the review is extensive and explains the importance of the proposed method. In the presentation section, the methodology was comprehensive, explaining each part of the algorithm. The time complexity of the algorithm was not given. The conducted tests were only time-related, and the compared algorithms were all implementations of SCAN method.

The effectiveness of the method is implied that is the same with SCAN method. The proposed method is useful only in computing very large graphs with millions of nodes and edges. It is not recommended in small and medium size graphs. The future improvements can be its speed, especially in merging phase, and the ability to handle dynamic networks.

## 2.7 Community Detection and Visualization in Complex Network by the Density-Canopy K-means Algorithm and MDS Embedding

In this paper authors try to overcome some common problems found in traditional algorithms for community detection and especially in *K*-means, a well-known unsupervised clustering algorithm. In particular:

a. Most traditional methods for detecting community structure have limitations in dimension reduction or parameter optimization.
b. *K*-means algorithm has moderate accuracy and stability
c. *K*-means algorithm has increased complexity

### 2.7.1 Methodology

First thing one usually do, for community detection in a complex network, is to cluster the nodes found in the network. So, it is critical to define a way to measure the similarity between nodes. Based on this similarity the clustering algorithm can be applied to divide the complex network into smaller parts, and then into communities. Quality of resulting clustering strongly depends on the distance metric selected and the parameters of the clustering algorithm.

Another thing that affects the overall quality of the detected communities structure visualization is the choice of the dimensional reduction algorithm, used to map the nodes of the network to a low-dimensional Euclidean space. There are several dimensional reduction algorithms to choose from, such as principal components analysis (PCA), multidimensional scaling (MDS), Laplacian eigenmaps (LE), T-SNE etc.

The authors choose to work on complex, undirected and unweighted static networks, consisting of n nodes. For the representation of the network they simple use the corresponding adjacency matrix A.

The first step is to choose an appropriate metric to compute the distance and the corresponding similarity between nodes. Among various similarity metrics found in bibliography [25] they

choose to use the Jaccard similarity, as the most suitable for measuring topological closeness. Assuming that $N_i$ is the set of the neighbor nodes of node *i*, and $N_j$ is the set of the neighbor nodes of node *j*, then the Jaccard similarity of the two nodes is defined as:

$$s_{ij} = \frac{|N_i \cap N_j|}{|N_i \cup N_j|} \quad \text{(eq. 2.7.1)}$$

where $|N_x|$ is the size (number of elements) of the neighbor set of node *x*. Since adjacency matrix is a binary matrix, containing the neighbors of each node, we assume that the symbol $L_{mn}$, $m,n \in \{0,1\}$, denotes the total number of bits having a value of *m* in row *i* and *n* in row *j*. So, for instance, $L_{01}$ denotes the total number of bits that are 0 in row *i* and 1 in row *j*. Using the above notation, the equation 2.7.1 is converted to:

$$s_{ij} = \frac{L_{11}}{L_{01} + L_{10} + L_{11}}$$

And, since Distance = 1 – Similarity, the corresponding Jaccard distance values, which compose the distance matrix D, are given by the equation:

$$d_{ij} = 1 - s_{ij} \Rightarrow d_{ij} = \frac{L_{01} + L_{10}}{L_{01} + L_{10} + L_{11}}$$

The *Jaccard* similarity metric takes values ranging from 0 to 1. Two nodes have a *Jaccard* similarity equal to 1 when they share exactly the same set of neighbor nodes. On the other hand, when two nodes have no common neighbor nodes then the *Jaccard* similarity between these nodes is equal to 0. These node pairs, having 0 *Jaccard* similarity, are very common in the case of huge networks and are problematic since MDS methods cannot apply to compute their low dimensional coordinates. To overcome this situation, the authors introduced a new metric to calculate the distance between two nodes, multiplying the *Jaccard* distance value by the *community structure coefficient* γ, and adding a random value ranging from 0 to 1:

$$d_{ij} = \gamma \cdot dist_{ij} + dist\_random_{ij}$$

These distance values create a new modified distance matrix *D*, which is used, by the MDS method, to compute the low dimensional projection of the network nodes. A new matrix *B* is created, using the contents of matrix *D* and the equation below to compute its contents:

$$b_{ij} = -\frac{1}{2}\left( d_{ij}^2 - \frac{\sum_{j=1}^{n} d_{ij}^2 - \sum_{i=1}^{n} d_{ij}^2}{n} + \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}^2}{n^2} \right)$$

The next step is the *eigenvalue* decomposition of matrix *B*. The eigenvalues $\lambda_i, i \in \{1,2,..,n\}$, produced are descending ordered and the two largest eigenvalues, $\lambda_1$ and $\lambda_2$, are chosen to form a diagonal matrix **Λ**, whereas the corresponding *eigenvectors* form the matrix **Ψ**. Finally, the equation below is used to compute the two-dimensional representation of network nodes:

$$X = \Psi\sqrt{\Lambda}$$


**Figure 5. Example of how Canopy algorithm works**

After applying MDS to reduce the dimensionality of network nodes, the Density-Canopy-*K*-means (DCK) algorithm is applied to analyze and divide network into communities. The method consists of three algorithms, applied sequentially according to their order of reference. Starting from the end, the *K*-means is a classic, well established unsupervised clustering algorithm. *K*-means provides excellent clustering results in its implementation, but the main disadvantage is its computational complexity and the necessity to precisely define in advance the number of clusters *K* and the initial seed. Finding the optimal solution to the general *K*-means clustering problem, for observations in *d* dimensions, is an *NP*-hard. If *K* and *d* values are fixed, the problem can be exactly solved in time $O(n^{dK+1})$, where *n* is the number of entities to be clustered.

A very effective solution to the above problem, is to use the Canopy algorithm to provide *k* value and initial seed to K-means algorithm. Canopy is a simple, fast, and efficient clustering algorithm. It makes use of 2 predefined distance values $T_1$ and $T_2$ ($T_1 > T_2$), to assign each node to a specific canopy (cluster), while it allows a node to belong to more than one canopies (clusters) at the same time [Figure 5] [26]. It is obvious that $T_1$ and $T_2$ values directly affect the overlap rate and granularity of the detected canopies. Unfortunately setting the optimal value for variables $T_1$ and $T_2$ is not clear. In order to improve the stability and computational efficiency of the canopy algorithm, the principle of *density clustering* is applied further by the authors. They relate *R* and *T* values (which in turn are related to $T_1$ and $T_2$ values) to the distance matrix *D*, using the following equations:

$$R = d_{\min} + r \cdot (d_{\max} - d_{\min})$$

$$T = d_{\min} + t \cdot (d_{\max} - d_{\min})$$

where $d_{min}$ is the average value of the minimum distance between each node and other nodes, $d_{max}$ is the average value of the maximum distance between each node and other nodes, *r* is the density radius coefficient, and *t* is the distance threshold coefficient. The authors concluded that the best values for *r* and *t* parameters is both 1/6. By combining all of the above approaches and assumptions, the authors managed to reduce the computational complexity of the overall algorithm to $O(n^3)$.

## 2.7.2 Evaluation - Results

*Modularity* was used to measure the quality of detected communities and test the effectiveness of the DCK algorithm. In general, a greater degree of modularity reflects a higher quality of network partitioning (communities detected). The network modularity evaluation function *Q* (eq. 2.7.2) was used to set the optimal value of the hyperparameters γ, *r* and *t*, used by the model:

$$Q = \sum_{u=1}^{K} \left( \frac{h_u}{H} - \left( \frac{d_u}{2H} \right)^2 \right) \quad \text{(eq. 2.7.2)}$$

where $K$ is the number of network communities, $H$ is the total number of network connections, $h_u$ is the total number of connections in the community $u$, and $d_u$ is the sum of the degree of nodes in the community. The modularity value is in the range of [-1,1], where value of 1 represents the best community structure, while a negative value reflects the absence of a community structure. Modularity value of 0 indicates that all network nodes are organized in a single community.

The authors used *artificial* networks, with known community structure, including both connected and mixture of connected-disconnected communities, to compare MDS with other known dimensional reduction – projection methods. MDS delivered the best visualization effects, followed by PCA, whereas LE can only effectively express the clustering of connected networks and T-SNE cannot express communities at all, using the specific distance metric.

Then the authors used *real-world* networks, of known community structure (number of communities included in the network), to measure the performance of DCK algorithm against the classic *K*-means and other known clustering methods, in terms of accuracy of the number of detected communities. In every case DCK algorithm performed better than simple *K*-means algorithm.

So, as a final conclusion, combining density clustering and canopy clustering with the *K*-means clustering algorithm, improves the performance of the overall process on detecting communities on complex networks, providing higher classification accuracy, compared to traditional community detection methods. Additionally, using MDS as a dimensional reduction – projection method, results to better visualization effect.

### 2.7.3 Critique

The combined algorithm presented by the authors has some clear advantages, such as:

a. Simplicity of the network representation, using only the adjacency matrix. No additional prior knowledge about the community structure is needed.
b. Reduced computational complexity of the overall classification process.
c. Enhanced algorithm's classification accuracy and stability.

But inevitably there are some disadvantages too:

a. They are forced to add randomness to the system by introducing a new variable ($\gamma$), which, as the authors themselves report, drastically affects the outcome of the following community detection.
b. Additional experiments needed to set optimum values for hyperparameters introduced to the algorithm
c. Algorithm can become very slow when applied on large networks.
d. Incomplete information about the Canopy method they use, it required a combination of information from other sources.

## 2.8 CC-GA. A clustering coefficient based genetic algorithm for detecting communities in social networks

Community detection, based on graph partitioning, is an NP-hard problem. One class of algorithms used to solve NP-hard problems is termed Genetic Algorithm (GA), a set of algorithms that mimic biological evolution. Gas are used in a wide variety of real-world optimization applications, community detection in networks being one of them.

Over the last 15 years many GA based approaches have been proposed for community detection in complex networks, with many variations on genetic operators used, such as generation of initial population, fitness function used etc. These techniques have some limitations, the major one being the generation of the initial population. Random generation of initial population, usually ends up finding a local optimum and takes much longer to reach a near-global optimum. An algorithm that uses a better technique to generate initial populations, may reach a near-global optimum faster with fewer iterations.

### 2.8.1 Methodology

CC-GA follows the typical steps of a genetic algorithm.

*Initial population*

An initial population (*generation* 0) is generated, consisting of a finite number of chromosomes $P = \{P_1, P_2, .., P_n\}$, in order to encode the network under investigation. The representation of each chromosome is based on locus adjacency representation [27]. Each chromosome is separate graph containing all $N$ nodes of original network and each node is connected to only one of its neighbors in initial network. Each connected component of the resulting mixed graph corresponds to a community present in the original network [Figure 6]. Instead of a random initial population, the authors use the, well-known in social network analysis, *Clustering Coefficient*, which quantifies how close are the neighbors of a specific node $v_i$ to being a clique (complete graph):

$$C_i = 2 \times \frac{L_i}{k_i (k_i - 1)} \quad \text{(for undirected graph)}$$

where $L_1$ is the total number of edges among $v_i$ 's neighbors, excluding the node $v_i$, and $k_i$ is the node $v_i$ 's degree. The algorithm exploits nodes' clustering coefficients during

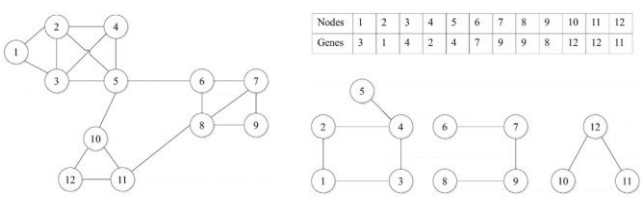


Figure 6 Creation of initial population

generation of initial popularity in a way that if a node has multiple neighbors, the algorithm chooses the neighbor with the *highest* clustering coefficient. In this way algorithm efficiently removes edges that act as local bridges in the original network. The idea is that node pairs connected with edges-bridges tend to have lower clustering coefficient, as compared to other nodes of the network, since their neighbors are not directly connected. So, during the initial stage (generation of initial population), the algorithm will not select these nodes to be connected to each other. As a final result, the algorithm splits the original network to simpler connected components, by removing preferably the edge-bridges. This is very important, since local bridges are the natural cut points separating various communities in original network.

*Fitness function*

Fitness function helps in evaluating the performance of a particular population and in selecting the *best performing* chromosomes. Modularity, a well-known measure of the quality of community structure in a network (eq. 2.7.2), is used to evaluate the community structure in each different generation of chromosomes. Each generation has its own distinct community structure.

*Crossover operator*

Crossover operator combines, in a random basis, the gene structure of two chromosomes to generate the gene of the next generation's chromosomes [Figure 7]. Only chromosomes that show *improved* performance compared to their ancestors are considered for the next generation. Crossover operator contributes to avoid local optima and to improve the community structure of a network.

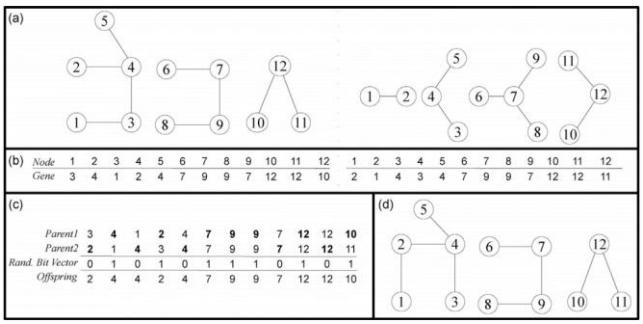


**Figure 7 Example of uniform crossover operator**

*Mutation operator*

Authors introduced a novel mutation operator, that ensures that smaller communities are merged into larger ones, and increases the community structure quality (modularity value) at the same time. To achieve this, algorithm selects random chromosomes from different communities and joins them. If this particular edge doesn't exist in the original network, then the algorithm chooses a different random pair of chromosomes.

*Population update*

The algorithm replaces the least fit chromosomes from the previous generation, with the new ones produced by crossover and mutation operators, to produce the population for the next generation. Then the procedure is repeated once again, until no farther improvement in the quality of detected communities' structure, as judged by the modularity value.

*Hyperparameters*

Based on above mentioned steps it is obvious that a series of hyperparameters must be carefully set before algorithm's execution. These hyperparameters include: population size, crossover and mutation rates, maximum number of iterations without any improvement (used as a termination criterion) and the percentage of chromosomes from the current population to be involved in the next iteration of genetic operators.

## 2.8.2 Evaluation - Results

The authors compared the performance of CC-GA algorithm against 3 other GA algorithms and 6 community detection algorithms, using 11 different real-world networks from various categories (social, biological, collaborative, online-social). The final results indicate that CC-GA outperforms existing GA-based approaches, returning a better community structure. In addition, CC-GA outperforms the community detection algorithms in most of the networks used, while remains only slightly behind on the rest of them. Another important point is that CC-GA is capable of detecting the community structure very quickly, since modularity approaches its maximum values within a few iterations.

## 2.8.3 Critique

The positive points of this paper are:

a. CC-GA algorithm identifies communities from dense networks, as well as finds cohesive groups of nodes in sparse networks.
b. The authors take advantage of known social network analysis measures, such as Clustering Coefficient, to improve performance of a GA based algorithm for the first time.
c. CC-GA does not require any a priori knowledge of the size, number or structure of the communities.
d. Simplicity of the network representation, using only the adjacency matrix.
e. Extensive evaluation of the algorithm against many state-of-the-art algorithms, using many real or artificial networks of different sizes.
f. Clustering Coefficient provides high quality initial population and, as a result, algorithm converges within a few iterations.
g. Authors plan to expand algorithm usage to overlapping communities' detection.

On the other hand, the negative points of this paper are:

a. An additional linear time decoding step is required to identify communities within the network.
b. Calculating Clustering Coefficient process has polynomial time complexity $O(n^k)$, where k<2.376.
c. Authors don't report overall algorithm time complexity.

d. GA algorithms' performance (finding an optimum solution, quality of detected communities) is highly dependent on initial population selection, which is mainly a random process.
e. A relatively large number of hyperparameters are required to be set as algorithm input.
f. Algorithm evaluation didn't include really large networks (>23K).
g. Algorithm evaluation against other GAs algorithms included only small networks.

## 3. CONCLUSIONS

This paper provides a combined overview of eight contemporary clustering algorithms in the field of community detection in social networks.

The majority of the algorithms are classified as structural, whereas HICODE adopts a more generic approach applicable to any basic conventional community detection algorithm.

Regarding the selected algorithms features, we investigate both unweighted and weighted scenarios. Furthermore, the mixture of the investigated algorithms is applied to datasets of different origin, i.e. real-world (ABCD, HICODE, AMKM) and synthetic ones (ASOCCA, EADP). The datasets deployed are also variable in size covering a wide range from small to large scale networks (|V| up to 335,000 in ASOCCA, |E| up to 4,800,000 in ABCD).

Depending on the algorithm different metrics are used to assess the performance like the Accuracy (ABCD), Modularity (HICODE, EADP), NMI (AMKM, HICODE, ASOCCA), speed (DSCAN) among others. In certain algorithms, the time complexity, an important performance index, is addressed like in ABCD, ASOCCA, EADP with claimed significant improvements. In certain cases, the algorithms are constrained though by limitations such as the introduction of controlling parameters (σ in AMKM-WAMKM), affecting the performance or user-defined thresholds (ε in DSCAN). On the other hand, the set of selected algorithms provides for special features like detecting hidden/overlapping communities and incorporating the dataset features gravitation in the clustering process.

To sum up, this paper addresses a wide range of methodological approaches to accommodate the community detection requirements. In this respect, this study-review allows for a future more thorough investigation of the applicability of the selected algorithms.

## ACKNOWLEDGMENTS

Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here. Insert paragraph text here.

## REFERENCES

[1] Michelle Girvan and Mark E. Newman. 2002 Community structure in social and biological networks. Proceedings of the National Academy of Sciences 99, 12 (2002), 7821–7826.

[2] Ruifang Liu, Shan Feng, Ruisheng Shi, Wenbin Guo. Weighted graph clustering for community detection of large social networks, 2nd International Conference on Information Technology and Quantitative Management, ITQM 2014, Procedia Computer Science 31 (2014) 85 – 94

[3] Jukai Zhou, Tong Liu, Jingting Zhu, 2019. Weighted adjacent matrix for K-means clustering. # Springer Science + Business Media, LLC, part of Springer Nature 2019, https://doi.org/10.1007/s11042-019-08009-x

[4] Kun He, Yingru Li, SuchetaSoundarajan, John E. Hopcroft. Hidden Community Detection in Social Networks. Conference of Canada 2017

[5] Xiaohui Pan, Guiqiong Hum Bing Wang, Tao Zhang, A Novel Community Detection Algorithm Based on Clustering Coefficient in Social Networks IEEEAccess August 30 2019

[6] Mark EJ Newman and Michelle Girvan.2004. Finding and evaluating communitystructure in networks.Physical review E69, 2 (2004), 026113.

[7] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparingcommunity structure identification.Journal of Statistical Mechanics: Theory and Experiment 9 (2005), P09008.

[8] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi, ``Defining and identifying communities in networks,'' Proc. Nat. Acad. Sci. USA, vol. 101, no. 9, pp. 2658-2663, 2004.

[9] A. Rodriguez, A. Laio, Clustering by fast search and find of density peaks, Science 344 (6191) (2014) 1492–1496.

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

[11] A. McDaid, N. Hurley, Detecting highly overlapping communities with model-based overlapping seed expansion, in: Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, IEEE, 2010, pp. 112–119.

[12] J. Xie, B. K. Szymanski, Towards linear time overlapping communitydetection in social networks, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2012, pp. 25–36.

[13] S.Y.Bhat, M.Abulaish, Hoctracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks, IEEE Transactions on Knowledge and Data engineering 27 (4) (2015) 1019–1013

[14] X. Bai, P. Yang, X. Shi, An overlapping community detection algorithm based on density peaks, Neurocomputing 226 (2017) 7–15.

[15] W. Li, J. Xie, M. Xin, J. Mo, An overlapping network community partition algorithm based on semi-supervised matrix factorization and random walk, Expert Systems with Applications 91 (2018) 277–285.

[16] A. Lancichinetti, S. Fortunato, J. Kert´esz, Detecting the overlapping and hierarchical community structure in complex networks, New Journal of Physics 11 (3) (2009) 033015.

[17] L. M. Collins, C. W. Dent, Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions, Multivariate Behavioral Research 23 (2) (1988) 231–242.

[18] J. Yang, J. Leskovec, Overlapping community detection at scale: a nonnegative matrix factorization approach, in: Proceedings of the

sixth CM international conference onWeb search and data mining, ACM, 2013, pp.587–596.

[19] V. Nicosia, G. Mangioni, V. Carchiolo, M. Malgeri, Extending the definition of modularity to directed graphs with overlapping communities, Journal of Statistical Mechanics: Theory and Experiment 2009 (03) (2009)P03024.

[20] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger. Scan: a structural clustering algorithm for networks. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 824–833. ACM, 2007.

[21] S. Aridhi, A. Montresor, and Y. Velegrakis. Bladyg: A graph processing framework for large dynamic graphs. Big Data Research, 9:9–17, 2017.

[22] L. Chang, W. Li, L. Qin, W. Zhang, and S. Yang. pscan: Fast and exact structural graph clustering. IEEE Transactions on Knowledge and Data Engineering, 29(2):387–401, 2017.

[23] W. Zhao, G. Chen, and X. Xu. Anyscan: An efficient anytime framework with active learning for large-scale network clustering. In 2017 IEEE International Conference on Data Mining (ICDM), pages 665–674. IEEE, 2017.

[24] Y. Che, S. Sun, and Q. Luo. Parallelizing pruning-based graph structural clustering. In Proceedings of the 47th International Conference on Parallel Processing, ICPP 2018, Eugene, OR, USA, August 13-16, 2018,pages 77:1–77:10, 2018.

[25] Lu, L., & Zhou, T., Link prediction in complex networks: A survey, Physica A, 390, p. 1150–1170, 2011.

[26] Zhang G., Zhang C., Zhang H. Improved K-means Algorithm Based on Density Canopy, Knowledge-Based Systems, 2018

[27] Y. Park, M. Song, A genetic algorithm for clustering problems, Proceedings of the Third Annual Conference on Genetic Programming, p. 568–575, 1998.

Experiment
Conference Name:ACM Woodstock conference
Conference Short Name:WOODSTOCK'18
Conference Location:El Paso, Texas USA
ISBN:978-1-4503-0000-0/18/06
Year:2018
Date:June

DOI:10.1145/1234567890
RRH: F. Surname et al.
Price:$15.00