Dušan Džamić

ADVANCES IN VARIABLE NEIGHBORHOOD SEARCH FOR GRAPH CLUSTERING

Abstract. This paper summarizes the key results of applying Variable Neighborhood Search (VNS) to graph clustering. The paper provides an overview of various adaptations of VNS developed for optimizing different quality measures, including modularity, edge ratio, and E-quality, across different network types. Special attention is given to generic VNS method and its flexibility in solving clustering problems with various quality functions. Additionally, the paper presents experimental findings that demonstrate the robustness and flexibility of VNS in achieving high-quality cluster partitions on artificial and real-world networks. These results demonstrate the important role of VNS in graph clustering methodologies and its potential for future applications.

Mathematics Subject Classification (2020): NN-02, 90C27; 90C59, 05C82

Keywords: graph clustering, community detection, variable neighborhood search

Faculty of Organizational Sciences, University of Belgrade, Serbia
dusan.dzamic@fon.bg.ac.rs
ORCID: 0000-0002-9088-879X

https://doi.org/10.64191/zr2510310107

Contents

1. Graph clustering 240
1.1. Strong and weak cluster 242
1.2. Clustering without a priori definitions 243
1.3. Clustering as a combinatorial optimization problem 244
1.4. Hybrid approaches 245
2. Variable neighborhood search for graph clustering 245
2.1. Modularity 246
2.2. Edge ratio 248
2.3. Sum-of-squares 251
2.4. E-quality function 253
3. Conclusion 256
References 257

1. Graph clustering

Clustering represents the process of grouping data such that the degree of similarity between elements is maximized if they belong to the same group and minimized if they belong to different groups. The groups formed through clustering are called *clusters*, and their number and characteristics are not known before the procedure begins. When the data are represented as a graph, this is referred to as *graph clustering*. In the context where the graph describes the structure of a complex system, the term *community detection* is commonly used. A cluster (community) in a graph represents a set of elements that are densely connected among themselves and sparsely connected with the rest of the graph. In scientific literature, the terms "network" and "graph" are often used interchangeably, with "network" typically referring to real-world systems (e.g., social or metabolic networks) and "graph" denoting their mathematical representations.

The need for clustering can arise in various fields, from social network analysis and identification of communities within them to the analysis of metabolic network and determination of protein functions. Beyond from direct application, clustering can represent just one step in the processing and preparation of data for other research techniques. For example, when analyzing large amounts of data, the data set can be reduced by selecting one or more representatives from each cluster due

to limitations in computational resources. Similarly, clustering can be used to identify outliers, which are often removed from data sets used for training in machine learning techniques.

For graph clustering, researchers aim to develop general techniques that can be used and adapted depending on the context and the available information about the graph. In developing methods, various approaches are used based on optimization, spectral graph theory, statistical inference, and dynamic processes. Developing methods for solving clustering problems and their application are of significant importance for understanding the dynamics and evolution of complex systems [11, 23, 27, 33, 40, 43, 53]. Moreover, they can enable better visualization and provide necessary information about individual nodes and their roles in the network. For example, certain nodes in a cluster may connect the cluster with the rest of the network, while other nodes may have a role in controlling and stabilizing the cluster.

One of the most commonly used networks for illustrating clustering is the network known as *Zachary's karate club* (Figure 1). The network was first described in the work [54], and it resulted from Zachary's monitoring of interactions (represented as edges) among members (nodes) of a university karate club during the period from 1970 to 1972. At that time, 78 interactions outside the karate club were observed among 34 members. A conflict between the coach (node 1) and the president (node 34) resulted in the club members dividing into two groups. One group of members formed a new club with the coach, while other members came together with the club president and hired a new coach. Based on the collected data, Zachary correctly predicted the decisions (indicated by node colors in the

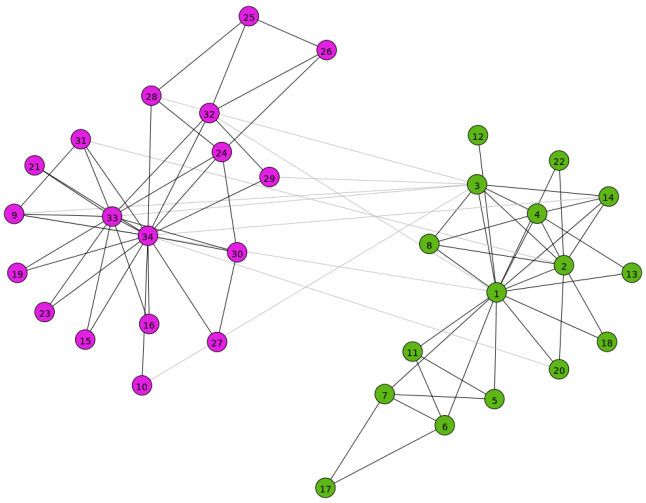


FIGURE 1. Zachary's karate club network, with nodes colored according to the predicted split between the coach's and president's groups after a social conflict.

figure) of all members, except for member 9, who joined the group gathered around the coach instead of the group around the president.

The Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), which represents a consortium of prestigious academic institutions (Rutgers University, Princeton University, Columbia University) and research laboratories (Microsoft, IBM, AT&T, NEC), included the problem of graph clustering in 2012 on the list of the most significant challenges in computer science [1], emphasizing the need to develop and implement efficient algorithms (implementation challenge). Since then, the number of scientific papers dedicated to this problem has increased annually and now is over 10,000 publications.

Let $G = (V, E)$ be a graph with n nodes and m edges. Let I be a finite set of indices, and let $\mathcal{P} = \{C_i : C_i \subset V,\ i \in I\}$ be a *partition* of the graph G, i.e., a collection of subsets of V such that:

1) $\forall i \in I, \quad C_i \neq \emptyset$;
2) $\forall i, j \in I,\ i \neq j, \quad C_i \cap C_j = \emptyset$;
3) $\bigcup_{i \in I} C_i = V$.

The result of clustering on the graph is a partition $\mathcal{P}$ whose elements C_i induce subgraphs G_{C_i} for all $i \in I$, which represent clusters. Often, we refer to the subsets C_i as clusters themselves, as they uniquely determine G_{C_i}. The number of clusters in the graph G is not known in advance; in other words, the partition is not uniquely determined with respect to the cardinality of the set I. Additionally, determining the partition is not uniquely defined with respect to the concept of what makes a subgraph induced by a set of nodes $C \in \mathcal{P}$ a cluster. This is not a consequence of an insufficiently considered problem definition but rather the diversity of contexts in which clustering can be performed and the goals that clustering aims to achieve. The example in Figure 2 demonstrates that in a single graph, there can exist multiple different clusterings.

1.1. Strong and weak cluster. When developing methods for graph clustering, some researchers formally define cluster. Radicchi et al., in the paper [20], attempted to define what makes an induced subgraph G_C a cluster in both the strong and weak sense. According to their definition, an induced subgraph G_C represents a cluster in a *strong sense* if each node has more edges to nodes within the cluster than to nodes outside the cluster, that is:

$$k_v^+(C) > k_v^-(C), \quad \forall v \in C.$$

Here, $k_v^+(C)$ denotes the number of edges from node v to other nodes within cluster C, and $k_v^-(C)$ denotes the number of edges from node v to nodes outside of C.

An induced subgraph G_C represents a cluster in a *weak sense* if the total number of edges between nodes within C is greater than the total number of edges connecting nodes in C to the rest of the graph, i.e.:

$$\sum_{v \in C} k_v^+(C) > \sum_{v \in C} k_v^-(C).$$

[1] http://dimacs.rutgers.edu/programs/challenge/

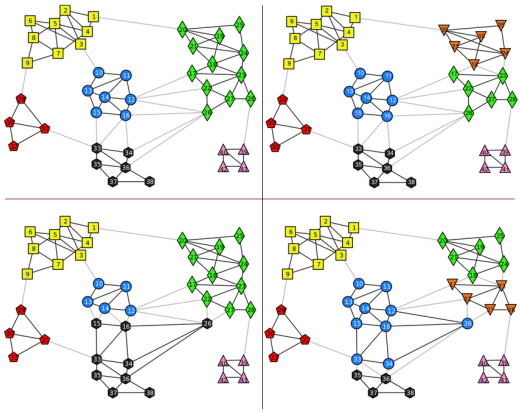


FIGURE 2. Different solutions for graph clustering

1.2. Clustering without a priori definitions. On the other hand, many researchers [29,30,52,56] consider that clusters should represent the final result of an algorithm without a precise a priori definition. The Label Propagation Algorithm (LPA), presented in [52], is the most commonly used representative of this class of algorithms. The idea of the algorithm is very simple and consists of an iterative process through which nodes are grouped into clusters. Initially, all nodes have unique labels representing clusters. In each iteration, each node update its label to the most frequent label among its neighbors. The formal rule for updating the label of node v_i is:

$$l^*_{v_i} = \underset{l}{\operatorname{argmax}} \left(\sum_{j=1}^{n} a_{ij}\, \delta(l_{v_j}, l) \right),$$

where l_{v_i} is the current label, $l^*_{v_i}$ is the new label of node v_i, and a_{ij} is the element of the adjacency matrix indicating the presence of an edge between nodes v_i and v_j. The Kronecker delta function δ is defined to be 1 if its arguments are identical, and 0 otherwise. If there are multiple labels with the maximum frequency, one is selected at random. The process repeats until each node has the label that is the most common among its neighbors. In the end, connected groups of nodes with the same label represent clusters.

The time complexity of the LPA algorithm is approximately linear. Initializing all nodes with unique labels requires $O(n)$ time. An iteration of label propagation for a single node v consists of two steps: in the first step, neighbors are grouped according to their labels, which requires $O(k_v)$ time, where k_v is the degree of node v; in the second step, the label of the group with the largest number of nodes is selected (in the worst case, this requires $O(k_v)$ time). Since label propagation is performed for all nodes, we find that the complexity of one iteration is $O(m)$. The exact number of iterations required to meet the stopping condition cannot be precisely determined. However, experimental results from [52] show that the number of iterations is very small (around 5) and does not depend on the number of nodes in the graph.

It is important to note that sometimes the result may include two or more groups of nodes with the same label that are connected through nodes with different labels. This happens when two or more neighbors receive a label from one node, spread it to different groups of nodes, and then change their own label. For this reason, after obtaining the results, it is necessary to apply the Breadth-First Search (BFS) algorithm to each induced subgraph of nodes that share the same label to obtain connected components, i.e., the final clusters. The time complexity of this step is $O(n + m)$.

1.3. Clustering as a combinatorial optimization problem. An approach that has gained the most attention from researchers focuses on defining a measure to evaluate partition quality and developing methods to identify the partition with the highest quality value. With this approach, the clustering problem is formulated as a combinatorial optimization problem, and various mathematical optimization methods can be used to solve it.

Formally, a quality measure (quality function) of a partition is a function $f : \mathcal{P} \longrightarrow \mathbb{R}$ that assigns a finite real number to each partition. After defining the function f, the clustering problem on a graph G can be formally written as follows:

$$\max_{\mathcal{P} \in \mathcal{P}_G} f(\mathcal{P}),$$

where $\mathcal{P}_G$ is the set of all partitions of graph G.

Among the defined quality measures, *modularity* is the most commonly considered in the literature, defined in 2004 by Mark Newman and Michelle Girvan [34]. Maximizing modularity in large-scale networks often fails to detect clusters with a small number of nodes, even when they are clearly defined. In the literature, this limitation is referred to as the *resolution limit problem*, first identified by Fortunato and Barthelemy in their study [50].

After Fortunato and Barthelemy identified the resolution limit problem, several ideas emerged to refine modularity or define alternative quality measures. Arenas et al. [1] proposed adding self-loops to each node with a weight $r > 0$ and optimizing modularity for varying r values on the resulting weighted graph. This approach allows scaling the total network weight without altering its topology, enabling the detection of smaller clusters for higher r values and larger clusters for lower r values. Similarly, Pons and Latapy [41] introduced a parameter $r \in (0, 1)$ to regulate the difference between actual and expected edge fractions in the modularity definition, producing a *multi-scale modularity*. While these multi-resolution modularity methods enable the identification of clusters of different sizes, they do not fully overcome the resolution limit. Adjusting the resolution parameter to avoid merging smaller clusters and splitting larger ones simultaneously remains challenging, as noted by Lancichinetti and Fortunato [5]. Alternatively, Li et al. [58] proposed *density modularity*, a new quality measure that excludes resolution parameters. In this measure, each cluster is characterized by the average modularity degree, defined as the difference between the internal and external average degrees of the cluster. Their experimental results showed that density modularity avoids the undesired effects associated with original modularity. Five years later, Chen et al. [32] introduced

a fundamentally different measure, also called *density modularity*, that penalizes inter-cluster edges while weighting penalties and terms based on the cluster sizes. This approach uses local cluster properties, such as density, to address the limitations of modularity more effectively. In 2016, Miyauchi and Kawase [6] introduced a new measure called *Z-modularity*. Their approach quantifies the statistical rarity of clusters in a partition using a normalized deviation (Z-score) from a reference model. This measure assigns a higher contribution to statistically rare clusters, addressing some issues with modularity by incorporating statistical significance.

In addition to these, many other functions for measuring partition quality have been proposed in the literature. More details can be found in [28, 47] and review papers [48, 49].

1.4. Hybrid approaches. Combining methods with different strategies for addressing a problem is often done to take advantage of each approach. An example of such an algorithm is the *Modularity-specialized Label Propagation Algorithm* (LPAm) proposed in [35]. Barber and Clark, the authors of the LPAm method, improved the label update rule to choose the label that contributes the most to maximizing the modularity function, instead of simply selecting the most frequent label among neighbors::

$$l^*_{v_i} = \operatorname*{argmax}_l \left(\sum_{j=1}^{n} \left(a_{ij} - \frac{k_{v_i}\, k_{v_j}}{m} \right) \delta(l_{v_j}, l) \right).$$

This hybridization retained the efficiency of the LPA algorithm while improving the quality of resulting partitions in the context of modularity. The weakness of the LPAm algorithm is that it tends to favor partitions into clusters with similar total node degrees, often terminating after reaching a poor local maximum of the modularity function.

Liu and Murata proposed an *advanced modularity-specialized label propagation algorithm* (LPAm+) in [55], introducing a procedure to overcome local maxima far from the global maximum. The LPAm+algorithm starts by applying the LPAm algorithm to the given network. After obtaining a solution, a greedy algorithm is applied to merge the resulting clusters to overcome the local maximum. First, for every pair of clusters (C_i, C_j), where $i, j \in I$, the change in the modularity function value caused by merging these clusters is calculated. Then, the merges that result in the greatest increase in the modularity function are performed. If any cluster merges are successful, the LPAm algorithm is applied again, and the entire merging process is repeated. The algorithm stops when no pair of clusters can be merged to increase modularity.

2. Variable neighborhood search for graph clustering

The *Variable Neighborhood Search* (VNS) method, a single-solution metaheuristic, was first introduced by Mladenović in 1995 at the "Optimization Days" symposium in Montreal [36]. Two years later, in the paper [38], Mladenović and Hansen provided a detailed description of the VNS method and applied it to the

Traveling Salesman Problem. VNS method has been widely applied across various fields, demonstrating its flexibility and effectiveness in solving both graph-based [4, 10, 14, 18, 26, 46] and non-graph-based [2, 3, 13, 21, 37, 39] optimization problems. In the following subsections, VNS methods for graph clustering with various quality measures, are described.

2.1. Modularity. The modularity [34] of a partition $\mathcal{P}$ is defined as

$$MQ(\mathcal{P}) = \sum_{C \in \mathcal{P}} \left(\frac{m_C}{m} - \left(\frac{k_C^*}{2m} \right)^2 \right),$$

where m_C is the number of edges between nodes within the set C, and $k_C^* = \sum_{v \in C} k_v$ is the sum of the degrees of the nodes in set C. The term $\frac{m_C}{m}$ represents the ratio between the number of edges within cluster C and the total number of edges in the graph, while $\left(\frac{k_C^*}{2m} \right)^2$ is the expected value of the same ratio in a network with the same degree sequence but with edges placed randomly. The values of modularity for different partitions of a network are presented in Figure 3.

In the context of modularity, an induced subgraph of a set of nodes C is considered a cluster if the total number of edges within it is greater than the expected number. The expected number of edges is calculated using a null model, which in the original definition represents a graph where edges are randomly distributed among the nodes while retaining each node's degree. Bollobás considered such random graphs determined by node degrees as early as 1980 [8]. Other null models are rarely used, such as the Erdős–Rényi model, the Chung–Lu model [19], or the corrected block model [57].

The first version of the *Variable Neighborhood Decomposition Search* (VNDS) method for modularity maximization was proposed by Aloise et al. in [12]. This initial approach introduced a strictly ascent framework focused on iteratively

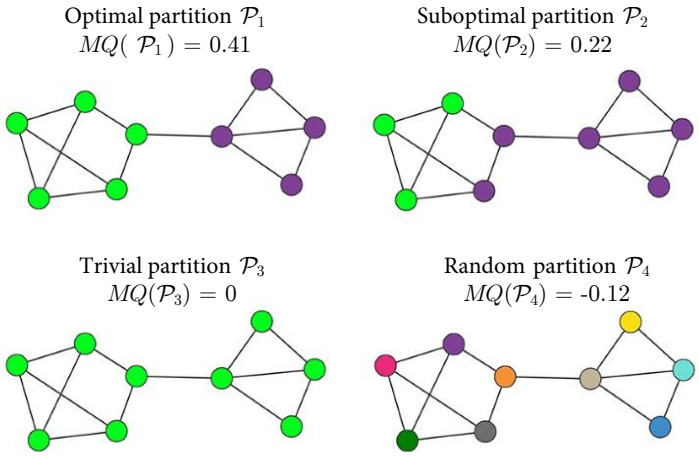


FIGURE 3. Modularity values for different partitions of a graph [7]

improving solutions within a decomposition-based search strategy. Later, this method was enhanced and extended into the *Ascent-Descent Variable Neighborhood Decomposition Search* (ADVNDS) by Džamić et al. [14]. ADVNDS introduced significant improvements, including new neighborhood, advanced problem decomposition techniques and an ascent-descent acceptance criterion, enabling the method to effectively navigate local optima and handle large-scale complex networks.

The solution structure represents a network partition as an integer array where each element corresponds to a node and its assigned cluster. To explore the solution space, four types of neighborhoods are defined: (1) splitting clusters into singleton nodes, (2) balanced bipartitions of clusters, (3) reassigning nodes to adjacent clusters, and (4) merging clusters. These transformations are carefully designed to enable efficient updates to modularity calculations and to overcome local optima by exploring diverse and wide solution spaces.

A key feature of the ADVNDS algorithm is **problem decomposition**, which focuses the search on smaller, structurally meaningful subgraphs. Instead of selecting nodes randomly, which can disrupt important network structures, the algorithm selects s clusters from the current partition to construct an induced subgraph. The

Algorithm 1: ADVNDS for graph clustering by modularity maximization

Input: Graph G, Parameters (a_0, r, s_{max}, t_{max})
Output: Best found solution x^*

$x \leftarrow$ InitialSolution()
$x \leftarrow$ LocalSearch(x, G, LPAm+)
$x^* \leftarrow x$
$s \leftarrow 1$; $a \leftarrow a_0$
while $t < t_{max}$ **do**
  $S \leftarrow$ Decomposition(x, s)
  $k \leftarrow$ RandomChoice({1, 2, 3, 4})
  $x' \leftarrow$ Shaking(x, k, S)
  $x' \leftarrow$ LocalSearch(x', S, LPAm+)
  $d \leftarrow MQ(x') - MQ(x)$
  if $d \geqslant 0$ *or* $e^{d/a} >$ RandomChoice([0, 1]) **then**
    $x \leftarrow$ LocalSearch(x', G, LPAm)
    if $MQ(x) > MQ(x^*)$ **then**
      $x^* \leftarrow x$
    $s \leftarrow 1$; $a \leftarrow a_0$
  else
    $s \leftarrow s + 1$; $a \leftarrow a\, r$
    if $s >$ min(NumberOfClusters(x), s_{max}) **then**
      $s \leftarrow 1$
  $t \leftarrow$ ElapsedCPUTime()

process begins by randomly choosing one cluster and then selecting $s-1$ of its neighboring clusters. All nodes from the selected clusters are used to form an induced subgraph, which defines the subproblem instance. This targeted decomposition increases the likelihood that local improvements will contribute to enhancing the overall solution.

One of the most important components of ADVNDS is its mechanism for overcoming local maxima. The shaking phase effectively diversifies the search through the use of non-nested neighborhoods and a randomized selection process. The ascent-descent criterion enhances this process by allowing moves to worse solutions when the search stagnates, thus navigating plateaus characterized by closely spaced local maxima. This approach utilizes insights from modularity landscapes [9], where high-density regions often contain the global optimum, allowing the method to focus on relevant areas while avoiding exhaustive and computationally expensive searches. The pseudocode of the ADVNDS method is presented in Algorithm 1.

The condition in step 11 is important for ADVNDS method, as it determines whether a local search will be applied in the entire solution space. The standard condition $d \geqslant 0$ is extended by the AD criterion, which is implemented to accept worse solutions with a probability of $e^{d/a}$. The AD condition is considered only if $d < 0$. In such cases, the probability of accepting a worse solution increases as d approaches zero, which indicates that the quality of the worse solution is close to the current solution's quality. Additionally, this probability rises with the increase of the parameter a, which reflects the number of unsuccessful iterations.

The experimental results demonstrate the efficiency of ADVNDS algorithm for modularity maximization. ADVNDS was tested on 22 instances from the *10th DIMACS Implementation Challenge*, categorized into two sets: smaller instances (C1) and larger instances (C2). For all instances in C1 where optimal solutions are known, ADVNDS achieved these optimal values. For 9 of the 13 harder instances in C2, where optimal solutions are not known, ADVNDS improved upon the best-known results, representing a 70% improvement rate.

In comparison with six newly proposed methods: *Label Propagation with weighted Coherent Neighborhood Propinquity* (LPA-CNP) [24], *Genetic Algorithm for Fuzzy Community Detection* (GAFCD) [31], COMBO [51], Fuzzy Clustering (FC) [44], Multiobjective Memetic Community Detection (MMCD) [42], and Membership-Degree Propagation (MDP) [25] — the results demonstrate that the ADVNDS heuristic achieves strong performance for modularity maximization. The ADVNDS heuristic matched or surpassed the best modularity values achieved by these methods across all instances. Figure 4 illustrates the clustering of the Karate club network obtained using the VNS method with modularity as the quality function.

2.2. Edge ratio. While modularity remains one of the most widely used quality measures, alternative criteria such as the edge ratio offer different perspectives on cluster cohesion and separation. In their paper [45], Cafieri et al. introduced the *edge ratio* as a measure for evaluating the quality of a cluster in a graph. This measure, denoted as $r(C)$, is defined as the ratio of the total number of internal edges within a cluster to the total number of cut edges that connect the cluster to

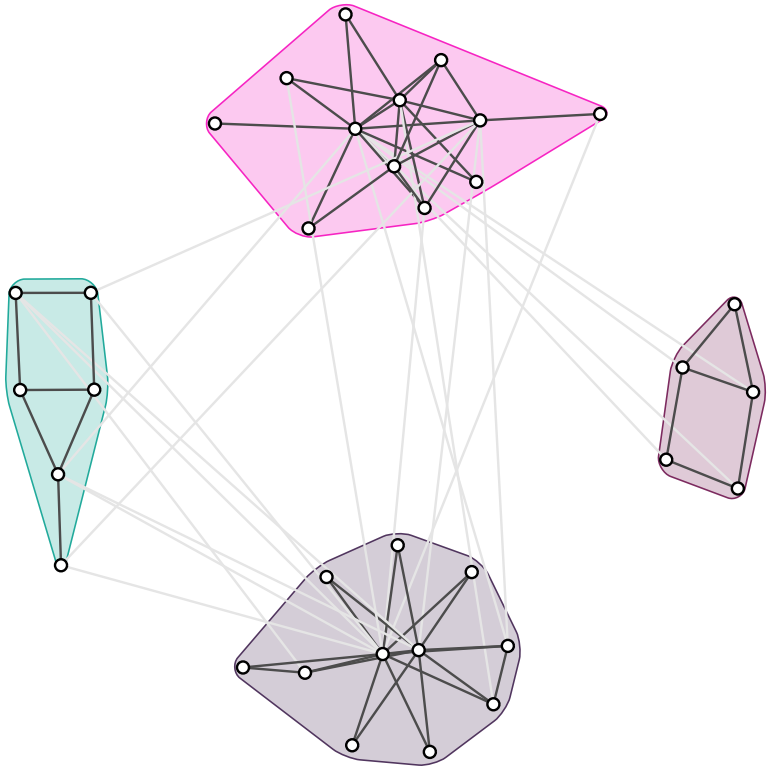


FIGURE 4. Clustering of the Karate club network using VNS with modularity. Nodes represent club members, and colors indicate detected communities.

the rest of the graph. Formally, it is expressed as:

$$r(C) = \frac{\sum_{v \in C} k_v^+(C)}{\sum_{v \in C} k_v^-(C)},$$

where $k_v^+(C)$ represents the number of edges from node v to other nodes within cluster C, and $k_v^-(C)$ denotes the number of edges from node v to nodes outside of C. The edge ratio provides a way to quantify the balance between the internal cohesion of a cluster and its separation from the rest of the network, making it a useful tool for assessing the quality of partitions.

In the same paper, the authors proposed a divisive clustering algorithm that iteratively splits a graph into smaller clusters using the edge ratio. The splitting process involves solving an optimization problem to maximize the smaller edge ratio of the two resulting clusters, C_1 and C_2, resulting from the division, i.e.:

$$\max_{C_1, C_2 \subseteq V} \min(r(C_1), r(C_2)),$$

subject to the constraints $C_1 \cup C_2 = C$, $C_1 \cap C_2 = \emptyset$, and $C_1, C_2 \neq \emptyset$. The optimization problem is solved through a sequence of linear programs with binary variables, integrated in a dichotomous search procedure. This ensures that each division is performed in an optimal way, resulting in a clear and locally optimal criterion for clustering decisions.

Cafieri et al. [46] proposed a Variable Neighborhood Search (VNS) algorithm for bipartitioning in hierarchical divisive clustering, using the edge-ratio criterion as the objective. The pseudocode of the hierarchical divisive clustering is presented in Algorithm 2.

Algorithm 2: Hierarchical divisive algorithm

Input: Graph G
Output: Partition $\mathcal{P}$
1 $\ell \leftarrow 1$, $C_\ell \leftarrow \{v_1, v_2, \ldots, v_n\}$, $\mathcal{P} \leftarrow C_\ell$
2 **while** $\ell \leqslant |\mathcal{P}|$ and $\exists C_\ell \in \mathcal{P}$ not visited **do**
3 Select $C_\ell \in \mathcal{P}$ (not visited)
4 C_1, $C_2 \leftarrow \mathrm{VNS}(G_{C_\ell})$
5 **if** $f \geqslant$ *threshold* **then**
6 $P \leftarrow (\mathcal{P} \cup \{C_1\} \cup \{C_2\}) \setminus \{C_\ell\}$
7 $C_\ell \leftarrow C_1$; $C_{\ell+1} \leftarrow C_2$
8 $\ell \leftarrow \ell + 1$

The VNS algorithm performed in line 4 is structured around several key components. First, it uses a binary vector x to represent the membership of nodes: $x_i = 1$ if node v_i belongs to C_1, and $x_i = 0$ otherwise. Neighborhoods are defined by allocation moves, which involve reassigning a node from one subgroup to another. For example, if a node v_i is in C_1, moving it to C_2 changes $x_i = 1$ to $x_i = 0$. These moves adjust the number of internal edges $k_i^+(C_1)$, $k_i^+(C_2)$ and cut edges $k_i^-(C_1)$, $k_i^-(C_2)$, which are recalculated dynamically after each move to evaluate its impact on the objective function.

To avoid being trapped in local optima, the VNS incorporates a shaking step, which introduces random perturbations to the current solution. Shaking modifies the binary vector x by flipping k bits (reassigning k nodes), creating a new solution at a specified Hamming distance from the current one. The value of k is incrementally increased up to a maximum ($k_{\max}$), ensuring systematic exploration of wider neighborhoods.

After shaking, the algorithm applies local search to refine the perturbed solution. Starting from an initial solution, local search explores the neighborhood by iteratively reassigning nodes to the subgroup that improves the edge-ratio. This continues until no further improvement is possible, resulting in a local optimum being reached. The local search process is computationally efficient, with a complexity of $O(n)$ per iteration, as it evaluates at most n moves (one for each node).

The final step involves updating the current solution based on improvements identified during local search. The binary vector x, as well as the counts of internal

edges and cut edges for each subgroup, are updated to reflect the new optimal partition. The process repeats until a termination condition is met, typically defined by a maximum number of iterations ($\text{iter}_{\max}$). The pseudocode of basic VNS for edge-ratio clustering is presented in Algorithm 3.

Algorithm 3: Basic VNS for edge-ratio clustering bipartition

Input: Induced subgraph $G_c = (C, E_c)$
Output: A bipartition of C

```
Initialization
while iter ⩽ iter_max do
    k ← 1
    while k ⩽ k_max do
        Apply Shaking
        Perform Local Search
        k ← k + 1 (next neighborhood)
        if f improved then
            Update current solution
```

Proposed VNS-based heuristic for edge-ratio clustering was tested on 11 instances, commonly used in graph clustering research. The results demonstrate that the VNS approach achieved high-quality solutions comparable to those obtained by exact methods [45] while significantly reducing computational time. For smaller networks, such as the karate club dataset, the exact and heuristic methods generated identical partitions, validating the accuracy of VNS. For larger networks, VNS provided efficient and effective partitions where exact methods were computationally expensive.

The computational results showed that VNS reduces execution time compared to exact bipartitioning, with time complexity empirically observed to grow approximately as $\sqrt{n}$. This allowed the heuristic to scale to larger networks without compromising solution quality. Furthermore, the flexibility of the VNS algorithm in handling different threshold values for indivisibility allowed fine-tuning of the number of detected clusters. This made it a robust alternative to modularity-based approaches, effectively addressing their resolution limitations. Figure 5 presents the results of clustering the Karate club network using the VNS method with the edge ratio criterion.

2.3. Sum-of-squares. In addition to topological measures like edge ratio, clustering can also be approached from a geometric standpoint. *Minimum Sum-of-Squares Clustering* (MSSC) is a classical clustering method that aims to partition a set of data points into k clusters such that the total within-cluster variance (sum of squared distances between data points and their cluster centroids) is minimized. This method is widely used in data analysis and machine learning for its ability to generate compact and well-separated clusters.

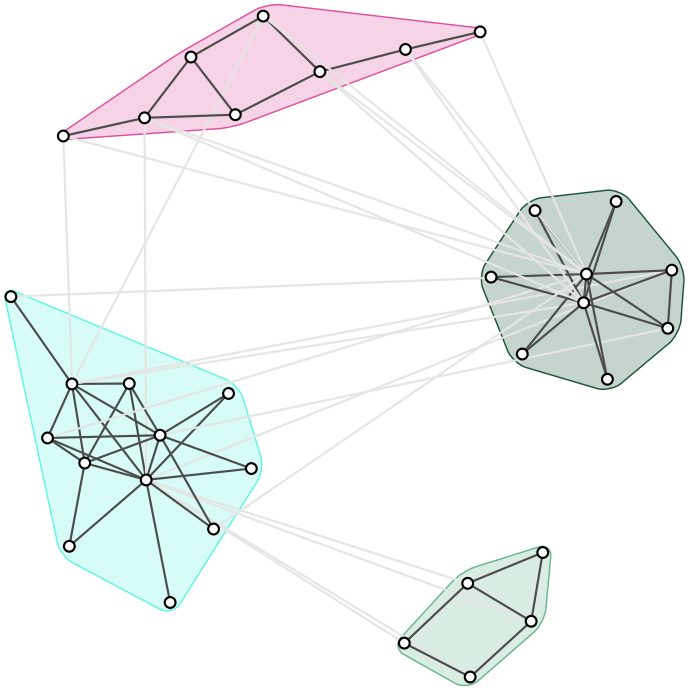


FIGURE 5. Karate club graph clustered using VNS with edge ratio criterion

In [17, 18] authors introduce two graph clustering models, V-MSSC (*Vertex-based Minimum Sum-of-Squares Clustering*) and E-MSSC (*Edge-based Minimum Sum-of-Squares Clustering*), which extend the classical Minimum Sum-of-Squares Clustering (MSSC) to network structures. In V-MSSC the goal is to minimize the sum of squared shortest-path distances between each node and its closest prototype:

$$\min \sum_{u \in V} \sum_{v \in V} d^2(u, v)\, \delta(l_u, v),$$

where l_u is assigned prototype for node u and δ is the Kronecker delta function.

V-MSSC restricts prototypes to network nodes, transforming the problem into a discrete p-median problem. In contrast, E-MSSC allows prototypes to be placed anywhere along the network edges, enabling more flexible cluster configurations that V-MSSC cannot achieve. The formulation is:

$$\min \sum_{v \in V} d^2\left(v, \{x_1, x_2, \ldots, x_p\}\right),$$

where $x_1, x_2, \ldots, x_p \in D(G)$ are prototypes selected from the set $D(G)$, which includes all points along the edges of the graph, and $d(v, \{x_1, x_2, \ldots, x_p\})$ represents the shortest distance from node v to the nearest prototype ($\min_{j=1,\ldots,p} d(v, x_j)$).

In same paper authors explore structural properties of these models, highlighting the potential for E-MSSC to identify more detailed clusters. To solve E-MSSC efficiently, the authors propose a Variable Neighborhood Search (VNS) heuristic. This heuristic includes components for generating initial solutions, systematic neighborhood exploration through shaking, and a modified k-means algorithm for networks. Several variations of the VNS heuristic have been developed, differing in their strategies for initialization and refinement.

Experimental results demonstrate the advantages of E-MSSC over V-MSSC. E-MSSC often produced more accurate and diverse partitions, especially for larger networks, with approximately 52% of tested instances resulting in different cluster configurations. The study also emphasizes the importance of high-quality initial solutions, as advanced initialization significantly improves the outcomes of the VNS heuristic. Figure 6 shows the Karate club network clustered using the VNS method with the sum-of-squares criterion.

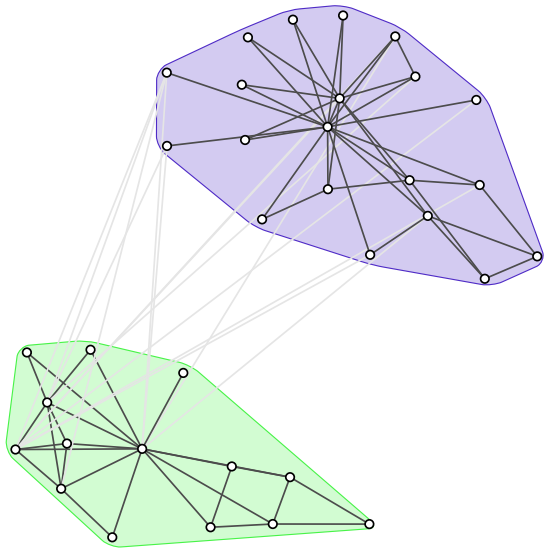


FIGURE 6. Karate club graph clustered using VNS with MSSC

2.4. E-quality function. While sum-of-squares clustering focuses on minimizing within-cluster variance, another important quality function, the *Exponential quality* (E-quality), introduces a different perspective by considering both internal structure and external connectivity [15].

The internal structure of a cluster is well described by its density. For example, a complete subgraph, representing an ideal cluster, has a density of one, which is the maximum value. However, a measure of a cluster's internal quality should also be sensitive to the structure of the network. Thus, a cluster with density d_C in a sparse network should be assigned a higher internal quality value than a cluster

with the same density d_C in a dense network. For this reason, a good measure of internal structure is the difference between the cluster density d_C and the network density d_G. This means that any subgraph with two or more nodes and a density greater than the overall network density can represent a potential cluster. However, this definition does not distinguish between a complete subgraph with a few nodes and one with many nodes in the same network. Moreover, any pair of connected nodes would be considered an ideal cluster under this measure. This issue can be addressed by incorporating the number of nodes in the subgraph, multiplying it by the difference between the cluster and network densities, i.e., $n_C\,(d_C - d_G)$. To ensure that small changes in cluster density or the number of nodes are clearly reflected, an exponential function can be applied. The final expression for the internal quality of a cluster C, denoted as $EQ^+(C)$, is:

$$EQ^+(C) = \begin{cases} e^{n_C\,(d_C - d_G)}, & n_C \neq 1, \\ 0, & n_C = 1. \end{cases}$$

The connection of cluster C to other clusters in the network can be characterized by the number of external edges l_C, i.e., the number of edges connecting nodes within the cluster to nodes in other clusters. The acceptable number of external edges depends largely on the context of the clustering and the desired outcome. To consider this, alongside the number of external edges, a parameter r is introduced to regulate the impact of external edges on the overall cluster quality. Additionally, the number of nodes in the cluster must be considered to distinguish between two clusters with the same number of external edges but different numbers of nodes. Naturally, a cluster with more nodes is expected to have more external edges connecting it to the rest of the network. Therefore, the value that measures the relationship of cluster C to other clusters, denoted $EQ^-(C)$, depends on the number of external edges per node $\frac{2l_C}{n_C}$ and is calculated as:

$$EQ^-(C) = e^{r\frac{2l_C}{n_C}}.$$

The difference $EQ^+(C) - EQ^-(C)$ represents the total E-quality function of a cluster C, while the overall E-quality function of a partition is the sum of the quality values of all clusters in the partition, i.e.,

$$EQ(\mathcal{P}) = \sum_{C \in \mathcal{P}} [EQ^+(C) - EQ^-(C)].$$

Assuming that the clusters in the partition $\mathcal{P}$ are not singletons, the E-quality function of the partition is given by:

$$EQ(\mathcal{P}) = \sum_{C \in \mathcal{P}} \left[e^{n_C\left(\frac{2m_C}{n_C(n_C-1)} - \frac{2m}{n(n-1)}\right)} - e^{r\frac{2l_C}{n_C}} \right].$$

To test the proposed e-quality function and compare it with modularity, a *Generic Variable Neighborhood Search* method was developed in [15]. This method maximizes a given quality function F, which measures the quality of the partition. A network partition into disjoint clusters is encoded as an array x of length n. Integer value at position i indicates the cluster label assigned to node v_i.

During the shaking phase, four types of neighborhoods are used:

- $\mathcal{N}_1(x,k)$: Contains solutions obtained by transforming x such that nodes from k clusters become singleton clusters.
- $\mathcal{N}_2(x,k)$: Contains solutions where k clusters in x are divided into two new clusters each.
- $\mathcal{N}_3(x,k)$: Contains solutions obtained by merging k pairs of adjacent clusters into single clusters.
- $\mathcal{N}_4(x,k)$: Contains solutions where k nodes are moved from one cluster to a neighboring cluster.

The pseudocode for the generic VNS method is presented in Algorithm 4.

Algorithm 4: Generic VNS-based Heuristic for Finding Communities in a Network

Input: Network G, quality function F, parameter $k_{\max}$
Output: Membership vector x_{cur}

$x_{\text{cur}} \leftarrow \text{RandomSolution}()$
$x_{\text{cur}} \leftarrow \text{LocalSearch}(x_{\text{cur}})$
$k \leftarrow 0$
while *stopping condition not satisfied* **do**
  $\mathcal{N} \leftarrow \text{RandomFrom}(\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4)$
  $x_{\text{temp}} \leftarrow \text{Shaking}(x_{\text{cur}}, \mathcal{N}, k)$
  $x_{\text{temp}} \leftarrow \text{LPA}(x_{\text{temp}})$
  if $F(x_{temp}) > F(x_{best})$ **then**
    $x_{\text{cur}} \leftarrow x_{\text{temp}}$; $k \leftarrow 0$
  if $k > k_{max}$ **then**
    $k \leftarrow 0$
  $k \leftarrow k + 1$

The computational results on artificial networks show that the E-quality function avoids creating overly small or excessively large communities. On real-world networks, it detects more reasonable and expected communities compared tomodularity.

In [16,22], it was demonstrated that the E-quality function effectively avoids the issues of merging or splitting clearly defined clusters, even in scenarios affected by the resolution limit of modularity based methods. By analyzing artificial networks with known community structures, the authors demonstrate that maximizing the E-quality function consistently identifies expected and reasonable clusters. As shown in Figure 7, the two clusters identified by VNS using the E-quality function are identical to the groups described by Zachary in [54] (Figure 1).

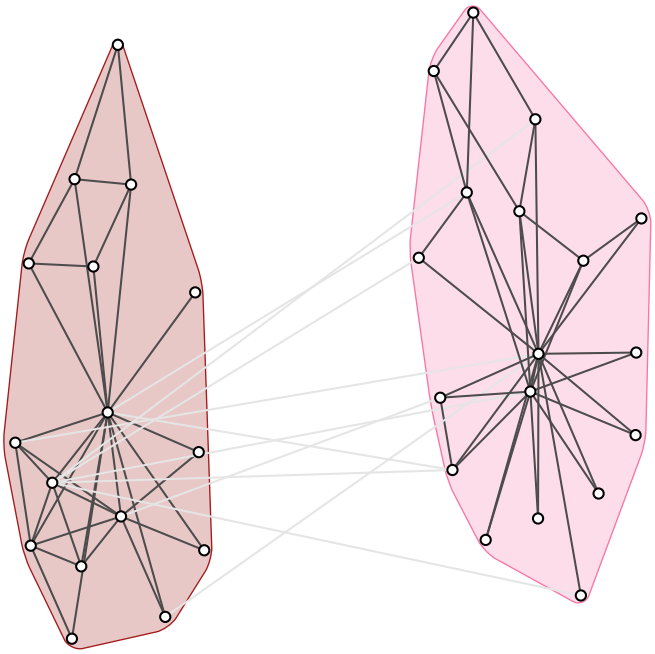


FIGURE 7. Karate club graph clustered using VNS with E-quality

3. Conclusion

Developing methods for graph clustering is of great importance for understanding the dynamics and evolution of complex systems that appear across various domains. Identifying clusters can contribute to better visualization and provide insights into the properties of individual nodes and their roles within the network. For instance, certain nodes within a cluster may connect the cluster to the rest of the network, while others may contribute to the control and stabilization of the cluster.

Due to the diversity of contexts in which clustering can be applied and the goals it aims to achieve, there is no precise universal definition of a cluster. Consequently, various approaches are applied in designing clustering methods, as described in the second chapter. One approach that has attracted significant attention involves defining a measure to evaluate the quality of a partition and constructing methods to find the partition that maximizes the value of the defined quality measure. This approach reduces the clustering problem to a combinatorial optimization problem, making a wide range of mathematical optimization techniques available for solving it.

The VNS method has been successfully applied to solve graph clustering using various partition quality measures, demonstrating its flexibility and effectiveness in identifying well-defined clusters across diverse network types. A significant contribution to the development of diverse VNS-based graph clustering methods

[14, 15, 17, 18, 46] has been made by professor Nenad Mladenović, the conceptual creator of the VNS method. Furthermore, a generic VNS-based method has been developed, designed to optimize any given quality measure. This adaptability allows the method to address different clustering objectives and apply customized measures specifically designed for particular applications, making it a robust and flexible tool for complex network analysis.

References

[1] A. Arenas, A. Fernandez, S. Gomez, *Analysis of the structure of complex networks at different resolution levels*, New J. Phys. **10**(5) (2008), 053039.

[2] A. Bačević, N. Vilimonović, I. Dabić, J. Petrović, D. Damnjanović, D. Džamić, *Variable neighborhood search heuristic for nonconvex portfolio optimization*, The Engineering Economist **64**(3) (2019), 254–274.

[3] A. Djenić, N. Radojičić, M. Marić, M. Mladenović, *Parallel VNS for bus terminal location problem*, Applied Soft Computing **42** (2016), 448–458.

[4] A. Duarte, L. F. Escudero, R. Martí, N. Mladenovic, J. J. Pantrigo, J. Sánchez-Oro, *Variable neighborhood search for the vertex separation problem*, Comput. Oper. Res. **39**(12) (2012), 3247–3255.

[5] A. Lancichinetti, S. Fortunato, *Limits of modularity maximization in community detection*, Phys. Rev. E (3) **84**(6) (2011), 066122.

[6] A. Miyauchi, Y. Kawase, *Z-score-based modularity for community detection in networks*, PLOS ONE **11**(1) (2016), 0147805.

[7] A. L. Barabási, *Network Science*, Cambridge University Press, 2016.

[8] B. Bollobás, *A probabilistic proof of an asymptotic formula for the number of labelled regular graphs*, Eur. J. Comb. **1**(4) (1980), 311–316.

[9] B. H. Good, Y. A. De Montjoye, A. Clauset, *Performance of modularity maximization in practical contexts*, Phys. Rev. E (3) **81**(4) (2010), 046106.

[10] C. Avanthay, A. Hertz, N. Zufferey, *A variable neighborhood search for graph coloring*, Eur. J. Oper. Res. **151**(2) (2003), 379–388.

[11] C. A. R. Pinheiro, *Community detection to identify fraud events in telecommunications networks*, SAS SUGI Proceedings: Customer Intelligence, 2012.

[12] D. Aloise, G. Caporossi, P. Hansen, L. Liberti, S. Perron, M. Ruiz, *Modularity maximization in networks by variable neighborhood search*, Graph Partitioning and Graph Clustering **588** (2013), 113–127.

[13] D. Džamić, B. Cendić, M. Marić, A. Denić, *Solving balanced multi-weighted attribute set partitioning problem with variable neighborhood search*, Filomat **33**(9) (2019), 2875–2891.

[14] D. Džamić, D. Aloise, N. Mladenović, *Ascent–descent variable neighborhood decomposition search for community detection by modularity maximization*, Ann. Oper. Res. **272**(1–2) (2019), 273–287.

[15] D. Džamić, J. Pei, M. Marić, N. Mladenović, P. M. Pardalos, *Exponential quality function for community detection in complex networks*, Int. Trans. Oper. Res. **27**(1) (2020), 245–266.

[16] D. Ž. Džamić, *Some properties of E-quality function for network clustering*, Yugosl. J. Oper. Res. **31**(1) (2021), 65–74.

[17] E. Carrizosa, N. Mladenović, R. Todosijević, *Sum-of-squares clustering on networks*, Yugosl. J. Oper. Res. **21**(2) (2011), 157–161.

[18] E. Carrizosa, N. Mladenović, R. Todosijević, *Variable neighborhood search for minimum sum-of-squares clustering on networks*, Eur. J. Oper. Res. **230**(2) (2013), 356–363.

[19] F. Chung, L. Lu, *Connected components in random graphs with given expected degree sequences*, Ann. Comb. **6**(1) (2002), 125–145.

[20] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, D. Parisi, *Defining and identifying communities in networks*, Proc. Natl. Acad. Sci. USA **101**(3) (2004), 2658–2663.

[21] F. Vidojević, A. Džamić, D. Džamić, M. Marić, *A novel artificial intelligence search algorithm and mathematical model for the hybrid flow shop scheduling problem*, J. Big Data **12**(1) (2025), 23.
[22] F. Vidojević, D. Džamić, M. Marić, *E-function for fuzzy clustering in complex networks*, IPSI BgD Trans. Internet Res.**18** (2022), 17–21.
[23] G. Ren, X. Wang, *Epidemic spreading in time-varying community networks*, Chaos **24**(2) (2014), 023116.
[24] H. Lou, S. Li, Y. Zhao, *Detecting community structure using label propagation with weighted coherent neighborhood propinquity*, Physica A **392**(14) (2013), 3095–3105.
[25] H. Zhang, X. Chen, J. Li, B. Zhou, *Fuzzy community detection via modularity guided membership-degree propagation*, Pattern Recognit. Lett. **70** (2016), 66–72.
[26] J. Brimberg, D. Urošević, N. Mladenović, *Variable neighborhood search for the vertex weighted k-cardinality tree problem*, Eur. J. Oper. Res. **171**(1) (2006), 74–84.
[27] J. Chen, H. Zhang, Z. H. Guan, T. Li, *Epidemic spreading on networks with overlapping community structure*, Physica A **391**(4) (2012), 1848–1854.
[28] J. Creusefond, T. Largillier, S. Peyronnet, *On the evaluation potential of quality functions in community detection for different contexts*, International Conference and School on Network Science, 111–125, 2016.
[29] J. Eustace, X. Wang, Y. Cui, *Community detection using local neighborhood in complex networks*, Physica A **436** (2015), 665–677.
[30] J. Li, X. Wang, J. Eustace, *Detecting overlapping communities by seed community in weighted complex networks*, Physica A **392**(23) (2013), 6125–6134.
[31] J. Su, T. C. Havens, *Fuzzy community detection in social networks using a genetic algortihm*, IEEE International Conference on Fuzzy Systems, 2039–2046, 2014.
[32] M. Chen, T. Nguyen, B. K. Szymanski, *A new metric for quality of network community structure*, ASE Human Journal **2**(4) (2013), 226–240.
[33] M. Zanin, P. Cano, J. M. Buldú, O. Celma, *Complex networks in recommendation systems*, Proceedings of the 2nd WSEAS International Conference on Computer Engineering and Applications, 120–124, 2008.
[34] M. E. J. Newman, M. Girvan, *Finding and evaluating community structure in networks*, Phys. Rev. E (3) **69**(2) (2004), 026113.
[35] M. J. Barber, J. W. Clark, *Detecting network communities by propagating labels under constraints*, Phys. Rev. E (3) **80**(2) (2009), 026129.
[36] N. Mladenović, *A variable neighborhood algorithm – A new metaheuristic for combinatorial optimization*, 12th Optimization Days, 1995.
[37] N. Mladenović, M. Dražić, V. Kovačevic-Vujčić, M. Čangalović, *General variable neighborhood search for the continuous optimization*, Eur. J. Oper. Res. **191**(3) (2008), 753–770.
[38] N. Mladenović, P. Hansen, *Variable neighborhood search*, Comput. Oper. Res. **24**(11) (1997), 1097–1100.
[39] N. Nikolić, M. Čangalović, I. Grujičić, *Symmetry properties of resolving sets and metric bases in hypercubes*, Optim. Lett. **11**(6) (2017), 1057–1067.
[40] N. S. Mitić, S. N. Malkov, J. J. Kovačević, G. M. Pavlović-Lažetić, and M. V. Beljanski, *Structural disorder of plasmid-encoded proteins in bacteria and archaea*, BMC bioinformatics **19**(1) (2018), 1–18.
[41] P. Pons, M. Latapy, *Post-processing hierarchical community structures: Quality improvements and multi-scale view*, Theor. Comput. Sci. **412**(8–10) (2011), 892–900.
[42] P. Wu, L. Pan, *Multi-objective community detection based on memetic algorithm*, PLOS ONE **10**(5) (2015), e0126845.
[43] P. Xanthopoulos, A. Arulselvan, V. Boginski, P. Pardalos, *A retrospective review of social networks*, Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining, 300–305, 2009.
[44] P. G. Sun, *Community detection by fuzzy clustering*, Physica A **419** (2015), 408–416.

[45] S. Cafieri, P. Hansen, L. Liberti, *Edge ratio and community structure in networks*, Phys. Rev. E (3) **81**(2) (2010), 026105.
[46] S. Cafieri, P. Hansen, N. Mladenović, *Edge-ratio network clustering by variable neighborhood search*, Eur. Phys. J. B, Condens. Matter Complex Syst. **87**(5) (2014), 116.
[47] S. Fortunato, *Quality functions in community detection*, SPIE Fourth International Symposium on Fluctuations and Noise, 660108, 2007.
[48] S. Fortunato, *Community detection in graphs*, Phys. Rep. **486**(3) (2010), 75–174.
[49] S. Fortunato, D. Hric, *Community detection in networks: A user guide*, Phys. Rep. **659** (2016), 1–44.
[50] S. Fortunato, M. Barthelemy, *Resolution limit in community detection*, Proc. Natl. Acad. Sci. USA **104**(1) (2007), 36–41.
[51] S. Sobolevsky, R. Campari, A. Belyi, C. Ratti, *General optimization technique for high-quality community detection in complex networks*, Phys. Rev. E (3) **90**(1) (2014), 012811.
[52] U. N. Raghavan, R. Albert, S. Kumara, *Near linear time algorithm to detect community structures in large-scale networks*, Phys. Rev. E (3) **76**(3) (2007), 036106.
[53] W. F. Pan, B. Jiang, B. Li, *Refactoring software packages via community detection in complex software networks*, Int. J. Autom. Comput. **10**(2) (2013), 157–166.
[54] W. W. Zachary, *An information flow model for conflict and fission in small groups*, J. Anthropol. Res. **33**(4) (1977), 452–473.
[55] X. Liu, T. Murata, *Advanced modularity-specialized label propagation algorithm for detecting communities in networks*, Physica A **389**(7) (2010), 1493–1500.
[56] X. Wang, J. Li, *Detecting communities by the core-vertex and intimate degree in complex networks*, Physica A **392**(10) (2013), 2555–2563.
[57] Y. Zhao, E. Levina, J. Zhu, *Consistency of community detection in networks under degree-corrected stochastic block models*, Ann. Stat. **40**(4) (2012), 2266–2292.
[58] Z. Li, S. Zhang, R. S. Wang, X. S. Zhang, L. Chen, *Quantitative function for community detection*, Phys. Rev. E (3) **77**(3) (2008), 036109.