Neural Computing and Applications (2023) 35:11073–11087
https://doi.org/10.1007/s00521-023-08284-8

ORIGINAL ARTICLE

KO: Modularity optimization in community detection

Furkan Öztemiz[1] · Ali Karcı[1]

Received: 18 August 2022 / Accepted: 6 January 2023 / Published online: 18 January 2023

Abstract
Many algorithms have been developed to detect communities in networks. The success of these developed algorithms varies according to the types of networks. A community detection algorithm cannot always guarantee the best results on all networks. The most important reason for this is the approach algorithms follow when dividing any network into communities (sub-networks). The modularity of the network determines the quality of communities in networks. It is concluded that networks with high modularity values are divided into more successful communities (clusters, sub-networks). This study proposes a modularity optimization algorithm to increase clustering success in any network without being dependent on any community detection algorithm. The basic approach of the proposed algorithm is to transfer nodes at the community boundary to neighboring communities if they meet the specified conditions. The method called KO (Karcı–Oztemiz) optimization algorithm maximizes the modularity value of any community detection algorithm in the best case, while it does not change the modularity value in the worst case. For the KO algorithm's test, in this study, Walktrap, Cluster Edge Betweenness, Label Propagation, Fast Greedy, and Leading Eigenvector community detection algorithms have been applied on three popular networks that were unweighted and undirected previously used in the literature. The community structures created by five community detection algorithms were optimized via the KO algorithm and the success of the proposed method was analyzed. When the results are examined, the modularity values of the community detection algorithms applied on the three different networks have increased at varying rates (0%, …,14.73%).

Keywords Community modularity · Community detection algorithms · Graph network · Modularity optimization

1 Introduction

Community detection algorithms perform clustering operations on networks. Identifying communities in the network provides many advantages and benefits, from pattern discovery to community management [1]. Due to the unique solution approaches of the developed community detection algorithms, different successes were exhibited in dividing the network. One of the most popular accepted metrics for measuring the clustering success of communities in a network is the modularity value [2]. We can say that the algorithm with a high modularity value is more successful in separating the network into communities [3]. In this study, a modularity optimization algorithm was proposed that rises the quality of clustering by increasing the modularity value independently of the network and the algorithm. The proposed algorithm performs transfers the nodes of the communities in the border regions to the neighboring communities as a result of the determined conditions. This algorithm is named KO (Karci & Oztemiz) Optimization Algorithm. The KO algorithm guarantees that the modularity value of any community detection algorithm does not change with the worst case and reaches the optimum value with the best case. All stages of the proposed algorithm were presented with an explanation. For the KO algorithm's test processes, in the study, Walktrap [4], Cluster Edge Betweenness [5], Label Propagation [6], Fast Greedy [7], Leading Eigenvector [8] community detection algorithms were implemented on weightless and directionless Complex [9], Zachary Kareta club [10] and Dolphins [11] networks. The clustering successes of algorithms on different networks were examined, and the results of their comparisons were included in this study. This way, the KO algorithm's success on different

✉ Furkan Öztemiz
furkan.oztemiz@inonu.edu.tr

[1] Department of Software Engineering, Inonu University, 44210 Battalgazi, Malatya, Turkey

networks and community detection algorithms has been tested.

In the literature, there are many studies on community detection methods, which include improving the modularity value of communities and developed based on modularity. The related works section provides brief information about the original studies in the literature.

2 Related works

When the studies are examined, some methods improve the community detection algorithms, while others focus on increasing the modularity value directly. These studies were tested on real and simulated networks, and their success and performance were determined. Algorithms with the Greedy approach are commonly used due to their performance and ability to provide successful solutions. In terms of performance, the speed-oriented FPMQA [12] method in online networks with the Greedy approach is a satisfactory algorithm. Furthermore, in studies aiming for high modularity [25], the vertex transport process was used to optimize community quality within specific criteria. The GRASP [28] method, which employs a matrix class to characterize the clustering family in the graph, outperformed well-known heuristics methods. Algorithms such as Bipartite [16] that only give successful results in private networks have been developed. Two novel fine-tuned [26] algorithms using the division and merge approach were compared to greedy approach algorithms and improved the modularity value in classical clique and LFR benchmark networks. Although many eigenvector-based studies do not achieve maximum modularity, there are studies that achieve above-average success [13, 17]. The modularity of the convolutional neural network-based approach [23] applied with the spatial eigenvector of the neighborhood matrix was higher than that of the basic deep learning methods. There are numerous studies in the literature that offer specific solutions to certain algorithms. The VLPA [14] and LPAm + [22] methods, which were developed specifically for the label propagation community detection algorithm, were designed to increase modularity. While VLPA provides success in high-dimensional networks, LPAm + accelerates network segmentation with the integration of multistep greedy agglomerative approach. The artificial bee colony-based MABC [29] algorithm, which is one of the popular metaheuristic methods, has been successful in real-world networks, except for private networks. Some metaheuristic-based methods require high memory and processing power for successful results. For this reason [18], the desired clear solution cannot be provided in large networks. Metaheuristic method performance evaluations are frequently encountered in the literature. Mod CSA [19] algorithm produced a more successful modularity result than the simulated annealing method in terms of accuracy and efficiency. While the smart local moving [24] method, developed with the approach known as local moving heuristic, shows success in small and medium networks, it does not give the same successful results in large networks. The causes of weak communities were investigated and optimized using the convex optimization framework [20] in studies on frameworks and schemas. In another study, modularity improvement studies based on machine learning were carried out on networks using the Louvain optimization scheme with the state-of-the-art technique [21]. Although the results are successful according to many algorithms in the literature, the network training process has a negative impact on performance. Mean field methods [27] produced successful results when compared to the results of the deterministic algorithm. Although the DSML-Mod and DSLM-Map methods [15] cannot achieve maximum modularity, they are effective for clustering in large graphs.

Many optimization methods using various solution methods have been presented in the literature to increase the modularity value. These methods are typically optimization studies performed on a specific algorithm or network [30, 31]. There is no stable method that can be applied to any community detection algorithm or network in general. This is a significant disadvantage. Because some methods produce successful results only in specific networks. Such algorithms are unlikely to be implemented in real-world networks. Furthermore, most methods either produce insufficient results or consume a significant amount of memory and time. Reaching the max modularity value of the results of the community detection methods and optimizing the modularity value to the maximum value was defined as the NP-Complete problem [32, 33]. The KO algorithm we presented aims to optimize the modularity values of any community detection algorithm in any type of network that is weightless and directionless. The KO algorithm does not perform community detection on its own. The presented algorithm's main goal is to improve the clustering quality of communities formed by any community detection algorithm in the literature. In this study, five community detection algorithms with different clustering techniques were chosen for the KO algorithm testing process. The time complexity of the KO optimization algorithm that we presented is $O(mn^2)$. Here, n represents the set of nodes and m represents the boundary nodes. The algorithm was created and visualized using the R programming language and the igraph library [34].

3 Material and method

The methods and processes applied in the study are summarized in 4 basic stages. In the image given in Fig. 1, information is shown about the process stages of the application in the study. As stated in Stage 1, three different networks were used in the study. Complex, Zachary Karate Club, and Dolphins networks are popular networks used in the literature [35–38]. The three networks are weightless and undirected graphs. Complex network consists of 70 nodes and 133 edge connections and contains special graphs such as linear, circular, star, and grid. Wayne Zachary picked up the Zachary Karate Club network in 1977 from a university's karate club members. It consists of 34 nodes and 78 edge relations. Each node represents one member of the club, while each edge represents the relation between the two members of the club. The Dolphin network is a social network of bottlenose dolphins observed between 1994 and 2001. While nodes refer to bottlenose dolphins living in Doubtful Sound in New Zealand, Edge correlations refer to the relationship between dolphins [39]. It consists of 62 nodes and 159 edge relations. Walktrap, Cluster Edge Betweenness, Label Propagation, Fast Greedy, Leading Eigenvector community detection algorithms given in Stage 2 were applied to three networks, and the modularity values of the networks were calculated according to the communities formed. In Stage 3, The Karcı & Oztemiz (KO) modularity optimization algorithm was applied, which we suggested in this study. Stage 3 was explained in detail in the next section. At this stage, it aims to increase the modularity values of the communities in the network by performing the transfer operations of the border nodes in line with the algorithm's criteria. The flowchart indicated in the image in Stage 3 is the minimalist situation of the KO algorithm flowchart given in Fig. 5. In Stage 4, the current state of the communities after the community optimization process was given visually.

3.1 Community detection algorithms

Community detection is used in many fields like identifying individuals with specific interests in social networks, discovering groups that perform manipulative transactions on the stock exchange, detecting intersections with similar profiles in transportation networks, etc. [40–42]. Many algorithms with different approaches have been developed for community detection in the literature [43]. These algorithms are evaluated under the headings indicated in Fig. 2 This study primarily aims to increase the modularity values of the algorithms under the headings of traditional and dynamic and to increase the quality of community classification. In addition, an algorithm under the headings of modularity-based algorithms was included in the analysis, and the results of the KO method under three headings were examined.

Walktrap algorithm firstly generates a transition probabilities matrix for each pair of nodes. Each element of this

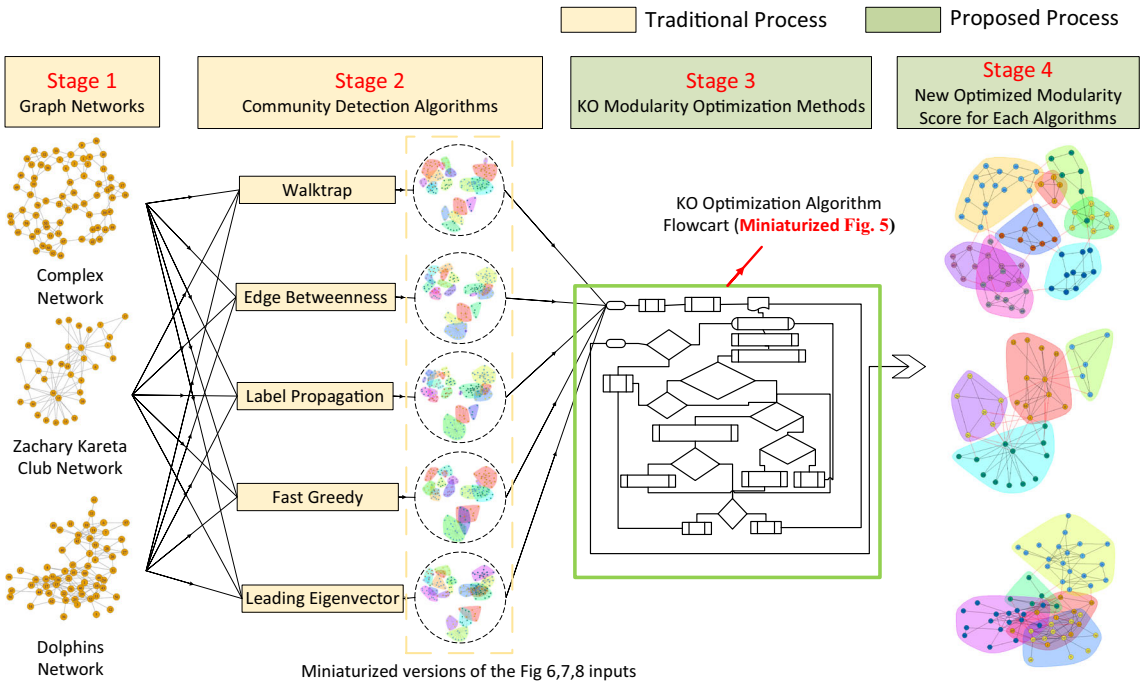


Fig. 1 Community modularity optimization process

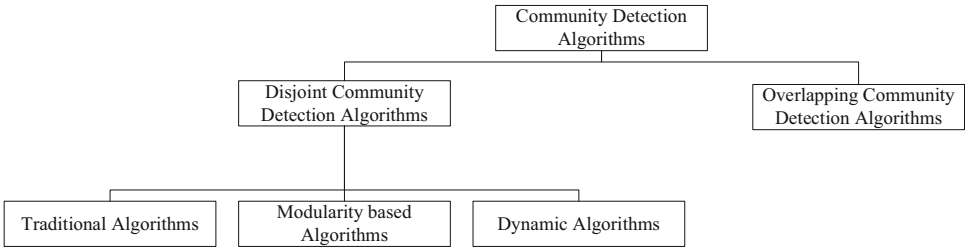


Fig. 2 Community detection algorithms classification headings [43]

matrix represents the probability of passing from a node to each given node with a random walk for a certain period of time. Passing probabilities are used to arrive at a measure of distance based on node degree or strength for each pair of nodes. This distance is then applied as a traditional hierarchical clustering technique (Ward's method). Communities are formed by minimizing the sum of the squares of the distances of each node from other nodes in its community [44]. The basic idea in this algorithm is: If two nodes are in the same community, they must both be the same distance from another third node with respect to the random walkway [45]. The formula of the Walktrap algorithm has been given in Eqs. 1 and 2.

$$D(i,j) = \sqrt{\sum_{n=1}^{N} \frac{\left(P_{in}^{m} - P_{jn}^{m}\right)^2}{d_n}} \tag{1}$$

$$D(C_k, C_{k'}) = \sqrt{\sum_{n=1}^{N} \frac{\left(P_{C_k n}^{m} - P_{C_{k'} n}^{m}\right)^2}{d_n}} \tag{2}$$

$P_{C_k j}^{m} = \frac{1}{N_k} \sum_{i \in C_k} P_{ij}^{m}$ is the probability of going from C_{k} to n_{j} in m steps ($n_j \notin C_k$). N is community and $D(i,j)$ distances are calculated for each pair [45].

Cluster Edge Betweenness algorithm is one of the first algorithms used to identify communities in networks [46]. This algorithm finds edges that are frequently between other nodes in the network, known as edge betweenness. It performs this operation based on betweenness centrality. For each network, the in-between edge value is calculated, and the edge with the highest value is subtracted. All edges affected by this removal have their edge betweenness values recalculated. This process iteratively repeats until there are no edges left. This status makes the algorithm significantly slower than other algorithms. Modularity is used to determine the optimum cut [5]. The modularity formula of the algorithm is given in Eq. 3.

$$Q = \sum_{i} e_{ii} - a_i^2 = Tre - \left|\left|e^2\right|\right| \tag{3}$$

a^i represents the edge ratio to which the nodes in the community i are connected. $Tr\ e$ specify the trace of this matrix ($Tre = \sum_i e_{ii}$).

Label Propagation is a desirable method because it is easy to implement, and the time complexity is linear [47]. Label Propagation Algorithm starts by giving each node a unique label, such as a character and number. In each iteration, each node replaces its label with the label of the node with the largest number of neighbors. If there is more than one maximum number of nodes, these nodes are changed randomly with the label of one of them. In this iterative process, dense node groups change their different tags to the same label, and nodes with the same label are grouped in the same community [48]. The label update formula is as in Eq. 4.

$$c_i = \underset{l}{\arg max} \sum_{j \in N^l(i)} w_{ij}. \tag{4}$$

When Formula 4 is examined, $N^l(i)$ represents the neighboring set of v_i labeled l. w_{ij} specifies the edge weight between v_{i} and v_{j} [48, 49].

The Fast Greedy algorithm starts with a subnet of connections between highly connected nodes. Next comes a hierarchical clustering algorithm. The algorithm then continues iteratively by joining neighboring communities with greedy approach. Each node moves into a community that maximizes its modularity functionality. As long as the modularity value of these collective communities increases, they continue to be combined [46, 50]. The formula of the algorithm is given in Eq. 5.

$$\frac{1}{2m} \sum_{vw} A_{vw} \delta(c_v, c_w) \tag{5}$$

A_{vw} represents the relationship between the $v$and w nodes. If there is a correlation, its value is 1. Otherwise, it is 0. The $\delta(i,j)$ function takes the value 1 if $i = j$;otherwise, it takes the value 0. c_{v} represents the community of node v.

Leading Eigenvector uses modularity to create the optimal community structure like the fast greedy algorithm [46]. To improve modularity, the algorithm first computes the first eigenvector of the modularity matrix before dividing the network into two communities. This process continues until there is no improvement in modularity anymore. The group separation formula of the algorithm is given in Eq. 6 [51]. s is a vector index. If i and j are in the same group, $\delta\left(g_i, g_j\right)$ is 1; otherwise, 0.

$$\delta(g_i, g_j) = \frac{1}{2}(s_i s_j + 1) \tag{6}$$

3.2 Modularity in community detection

Modularity is a quality function. It detects whether the network is divided with good quality [52]. Modularity measures community strength by comparing the fraction of edges within the community to fractions like this. The rationale is that a community should have more connections among themselves than a random collection of nodes [26]. Improving the modularity value is an NP-complete problem [33]. There is no deterministic algorithm that gives the highest modularity value. In the literature, there are algorithms that reach the max modularity value in specific networks and are applied by testing all possibilities [53]. Such algorithms do not enable community segregation as the network grows [32]. Practical algorithms are based on methods that give approximate results such as greedy, simulated annealing, or spectral optimization. Different approaches offer different balances between speed and accuracy [54]. The modularity Q of a network with n nodes and m connections is expressed mathematically as in Eq. 7 [55].

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{\mathrm{ij}} - \frac{k_i k_j}{2m}\right)\delta_{C_{\mathrm{i}}C_{\mathrm{j}}} \tag{7}$$

A is the neighborhood matrix of the network, k_{i} is the degree of node i, C_{i} is the community to which node i is assigned, and δ_{ij} is the Kronecker delta. The modularity measure in Eq. 7 only applies to unweighted and undirected graphs. The connections in the networks express the relationships at times, and the strength of the relationships at other times. In these cases, the modularity operation for weighted and directed graphs is performed as stated in Eq. 8.

$$Q = \frac{1}{|E|}\sum_{ij}\left(A_{ij} - \frac{k_i^{in} k_j^{out}}{|E|}\right)\delta_{C_i C_j}, \quad |E| = \frac{1}{2}\sum_{ij} A_{ij} \tag{8}$$

Here k_i^{in} denotes the input degree and k_j^{out} denotes the output degree. Q value close to 0 indicates that the fraction of edges within the community is poor, whereas a Q value close to 1 indicates that a mesh community structure approaches the maximum possible power [26].

3.3 KO modularity optimization algorithm (Proposed Methods)

The KO optimization algorithm aims to increase the modularity value independently of any network or community detection algorithm. In other words, the proposed method aims to improve the community quality of any community detection algorithm. In summary, the KO algorithm works by transferring the border nodes of communities in a network. The working logic of the algorithm was explained under several headings with sample figures. The working process of the algorithm was described in detail with a flow diagram and pseudocode.

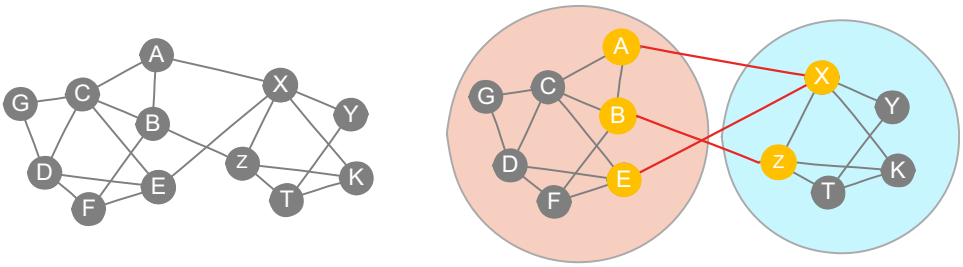
3.3.1 Separation of the graph into communities and detection of boundary nodes

The algorithm's first stage starts with applying any community detection method to the graph. An example graph structure is given in Fig. 3a Nodes that are connected to different communities are called boundary nodes. An example of community structure was shown in Fig. 3b. The nodes A, B, E, X, and Z indicated as yellow in Fig. 3b are the boundary nodes.

3.3.2 Transfer degrees of boundary nodes

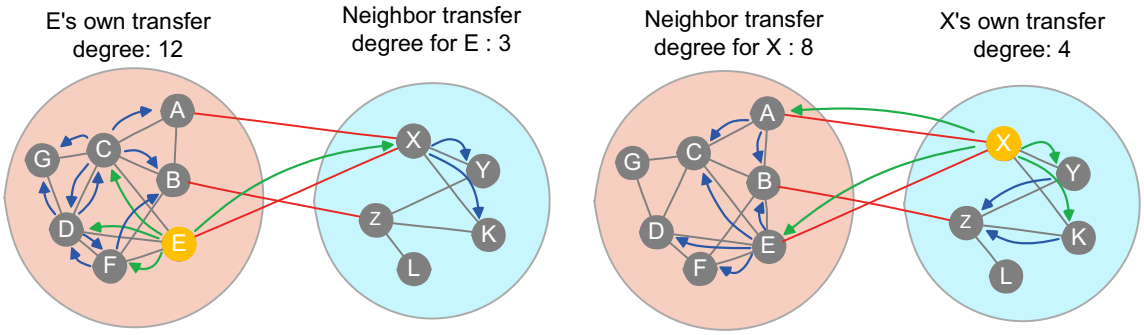
Transfer operations will be carried out in line with the transfer degrees of the boundary nodes. Figure 4 represents the edge relations between the two communities. Two transfer degrees are calculated for nodes A, B, E, X, and Z associated with these red edges within its own community and neighboring community. If the neighbor transfer degree of the boundary node is higher than its own transfer degree, it is added to the transfer list to transfer to the neighboring community. If its own transfer degree is less than or equal from the neighboring transfer degree, It remains in its own community. The transfer degree (own transfer degree) is the sum of edges that the related boundary node is connected to within two edge hops in its community. Neighbor transfer degree is the sum of edges the related boundary node is connected to within two edge hops in the neighbor community. The green arrows in Fig. 4 represent the one hop distance for the yellow node, and the blue arrows represent the two hop distance. If we examine the boundary node E in Fig. 4a, the transfer degree has been determined as 12, and the neighbor transfer degree has been determined as 4. Since node E's own transfer degree is higher than its neighbor's transfer degree, It is said to be highly committed to its community for node E and is not added to the transfer list. Node X in Fig. 4b has its own transfer degree of 4 and its neighbor transfer degree of 8. Because of node X has a stronger connection with the neighboring community, it is added to the transfer list.

The mathematical formula of transfer degrees is given in Eqs. 9 and 10. While $in(V_i)$ gives the own transfer degree of node$V$, $out(V_i)$ gives the neighbor transfer degree of node$V$. $N_{\mathrm{in}}(V_i)$ represent the node degree of node V in its



a Example graph structure **b** Boundary nodes

Fig. 3 Community structure and boundary nodes, **a** Example graph structure **b** Boundary nodes



a Transfer status for E **b** Transfer status for X

Fig. 4 Example transfer degree display **a** Transfer status for E **b** Transfer status for X

own community, $[N(V_j)]$ denotes the node degree of the neighbors of node V in its own community (except for links with node V). $N_{\text{out}}(V_i)$ refer to the number of nodes that node V is connected to in the neighboring community.

$$in(V_i) = \sum_{V_j \in Nin(V_i)} [N(V_j)] + N_{in}(V_i) \tag{9}$$

$$out(V_i) = \sum_{V_j \in Nout(V_i)} [N(V_j)] + N_{out}(V_i) \tag{10}$$

3.3.3 Creation of the transfer list and realization of the transfer

After calculating the transfer degrees of all border nodes, the boundary nodes to be added to the transfer list are determined. When adding boundary nodes to the transfer list, priority is given to the community and its nodes with a high total node degree in the network. In other words, communities with high correlation are added to the transfer list before low correlation communities. If there are boundary nodes to be added to more than one transfer list in the same community, priority is given to the node with the higher node degree. Transfer operations are performed according to the node order in the transfer list. Each node in the list is transferred to the community with the high neighbor transfer degree, and the modularity value is calculated. If the transfer increases the modularity of the network, the transfer is confirmed. If the transfer decreases the modularity value, the transfer operation is canceled, and the operation is taken back. This operation is performed for all boundary nodes in the transfer list.

3.3.4 Transfer list control

Boundary nodes whose transfer has been approved are removed from the transfer list. The transfer list is emptied, and a new transfer list is created in the next iteration by determining the boundary nodes again. As long as the transfer of at least one of the nodes in the transfer list is approved by making this check in each iteration, the approved nodes are removed from the transfer list, and the transfer list is emptied. In case of no transfer confirmation for the nodes in the transfer list in any iteration, the communities are checked. All nodes of the communities where more than half of the community members are on the transfer list are merged and act as a single node. The primary purpose of this process is to include small communities in large communities in a way that increases the modularity value. If the transfer of the community in which most of the members are on the transfer list to the neighboring community is approved, the transfer is carried out, and community members are removed from the transfer list. The execution of the KO algorithm continues with the next iteration. In any iteration, if the transfer of at least one of the nodes in the transfer list is not approved or the

Table 1 KO pseudocode

```
KO Modularity Optimization Algorithms
G : (V,E)                                                    // input graph
u : a node of graph
C <- Community Detection(G) // Any Community Detection Algorithm Cluster
Q <- Modularity(G,C)       // Modularity of G with respect to C community
Cu : Nodes at the community boundary
Un : Neighboring nodes of node u
CUn: Community of neighbor of node u
while( Cu != null ){
    Td <- TransferDegree(Cu)        // (Equation 9) Computing transfer values for border nodes
    Tun <- TransferDegree(Un)       // (Equation 10) Neighbour nodes transferring degrees
     // Sort boundary nodes by community size and node degrees in the transfer list
    if( Td > Tun)
        MultipleTransfers()
            TransferCount <- Count(NodeTransfer(Cu,Un))
            if( TransferCount > 1 )
                  if(Count(Maximum(Tun)) <= 1 )  // Are the number of nodes which have maximum
                                                 // transfering degrees, more than one?
                      MaxCommunity <- Max(TransferDegree).CUn
                      NodeTransfer(Cu,MaxCommunity) = True  // Add border node to neighbour node's
                                                            //community
                      Mod()
                           Qcurrent <- Modularity(G,C)
                       if(Qcurrent > Q )
                          NodeTransfer(Cu,MaxCommunity) = Accept; // Commit transfering.
                          Q <- Qcurrent
                       else
                          NodeTransfer(Cu,MaxCommunity) = Cancel; // Cancel transfering
             else if( TransferCount == 1)
                   TransferList <- Add (Cu) as ordered Array     //Add nodes to transfer list in order
                   NodeTransfer(Cu,MaxCommunity) = True
                   if( All nodes of C in TransferList )
                         Cu <- Combine(All nodes of C)   // Corresponding community is considered as unique
                         MultipleTransfers();
                   else
                      Mod();
             else
                 if(NodeDegree(Cu) == 1)   // Are number of corresponding nodes odd?
                     MultipleTransfers();
                 else
                     if(TransferCount == 0)
                         KO algorithm process complete
                     else
                         Pass Next Iteration
```

transfer of the communities whose majority of the members are on the transfer list is not approved, the algorithm stops working. The pseudocode, which includes all the stages of the KO optimization algorithm we have presented, is given in Table 1, and the flowchart is given in Fig. 5

4 Experimental results

The proposed KO modularity optimization algorithm was applied on three popular graphs with five community detection algorithms. These networks, known as Complex, Zachary Karate Club, and Dolphins in the literature, are weightless and undirected graphs. In Fig. 6, clustering modularity values of five community detection algorithms before and after applying the KO algorithm on the complex network were given. In addition, information on which nodes the KO algorithm takes to the transfer list in each iteration and which nodes are transferred are given. When Fig. 6 was examined, the modularity value of the Walktrap algorithm on the complex network has been determined as 0.670134. With the application of the KO algorithm, nodes 24, 37, 57, 56, and 12 were added to the transfer list in the first iteration. Nodes 37 and 57, indicated in red color in the transfer list, were transferred to neighboring communities

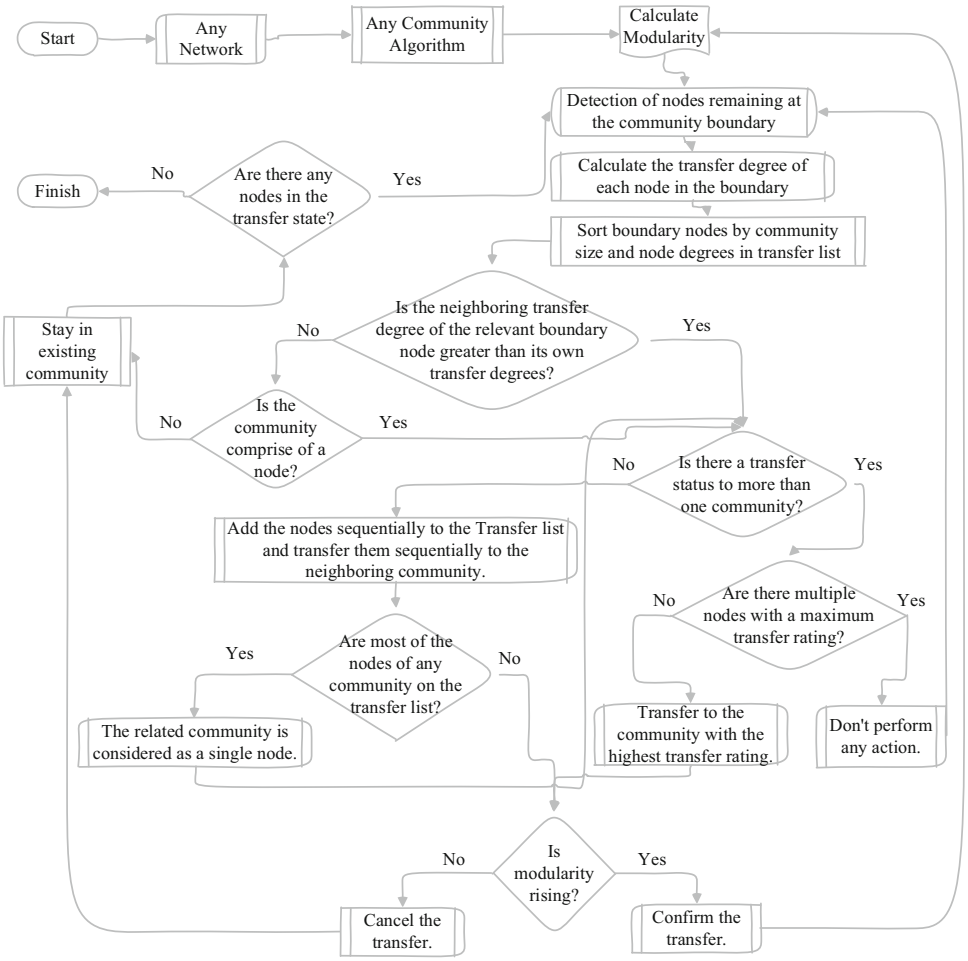


Fig. 5 KO algorithm flowchart

by obtaining transfer approval. After this transfer, the modularity value of the new community increased to 0.6863305. In the second iteration of the KO algorithm, nodes 24 and 12 were added to the transfer list. Since these two nodes do not receive transfer approval, the algorithm stops working.

The biggest optimization on the complex network has been on the communities created by the Leading Eigen Algorithm. The modularity value, which was 0.6074679 before the KO algorithm, increased to 0.6859065 after applying the KO algorithm. When we examine the results of the leading eigen algorithm. In the first iteration, nodes 56, 6, 44, 48, 20, 12, 57, 67, 49, 37, 38 were added to the transfer list, since the transfer of nodes 56, 6, 44, 48, 20, and 67 from these nodes increased the modularity value, the transfer was approved, and as a result of the first iteration, the modularity value increased to 0.6555486. In the 2nd iteration, the boundary nodes were redetermined, and the transfer list was created. The modularity value increased to 0.6714625 with the transfers of nodes 5, 47, and 12. In the third iteration, the modularity increased to 0.6772288 with the transfer of node 36. In the fourth iteration, nodes 49, 37, and 38 in the transfer list represent a community with three nodes. If more than half of the community is in the transfer list, which is one of the important criteria of the KO algorithm, the condition that the community is considered a node is applied. Since nodes 49, 37, and 38 are in the same community, they were transferred as a single node to the community with a high neighbor transfer degree. After this transfer process, the modularity increased to 0.6859065. In the next iteration, the algorithm finishes because no nodes can be added to the transfer list.

The KO modularity optimization results of the Zachary Karate club network are shown in Fig. 7. According to these results, the KO algorithm improved the communities formed by the Walktrap algorithm at the highest rate. The modularity value, which was 0.3532216 before the KO algorithm, increased to 0.4052433 after applying the KO algorithm. The slightest improvement has been in the communities created by the fast greedy algorithm. The modularity value of 0.3806706 obtained with the application of the fast greedy algorithm increased to 0.3813281 with the application of the KO algorithm. After the transfer

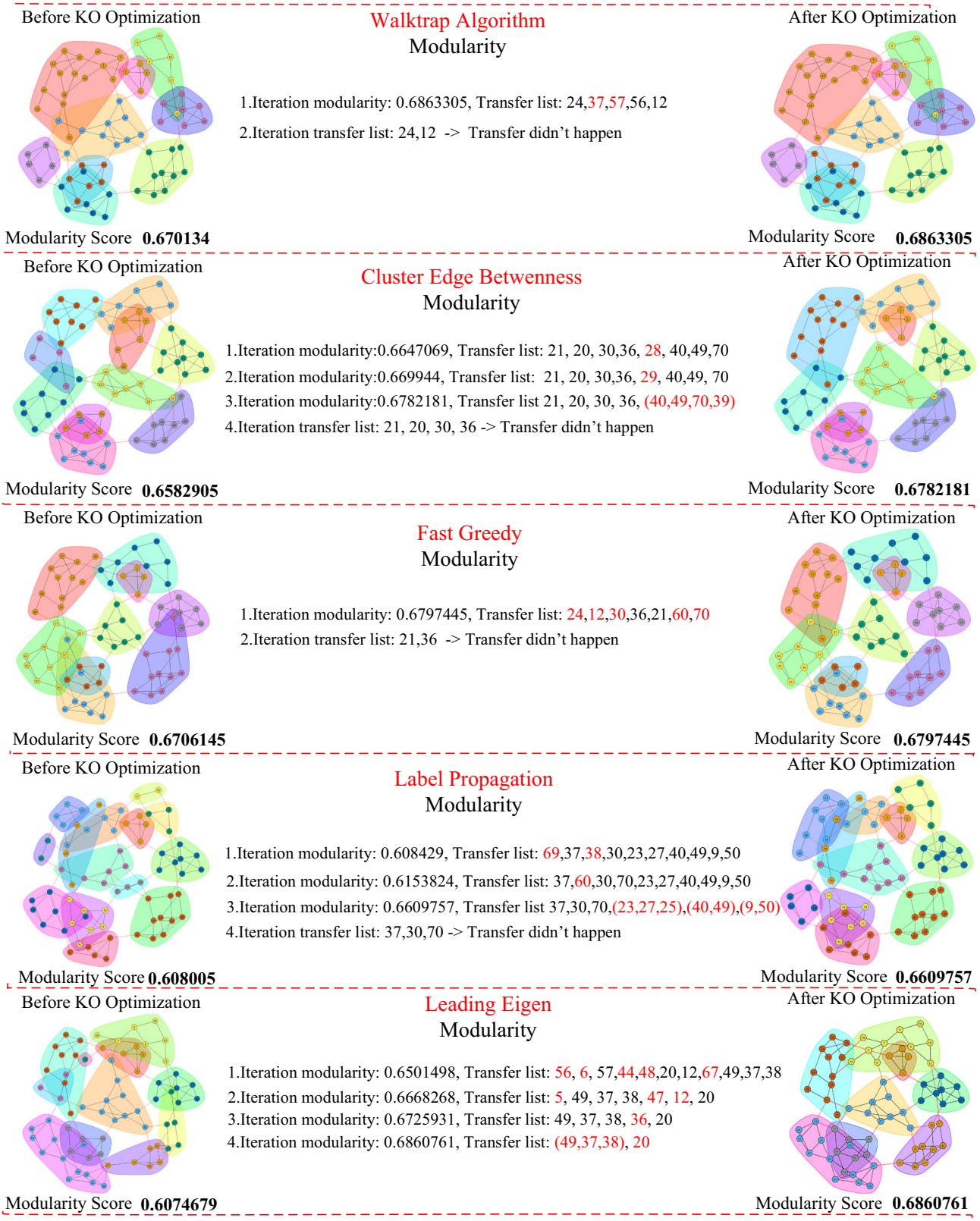


Fig. 6 KO modularity optimization results of Complex network

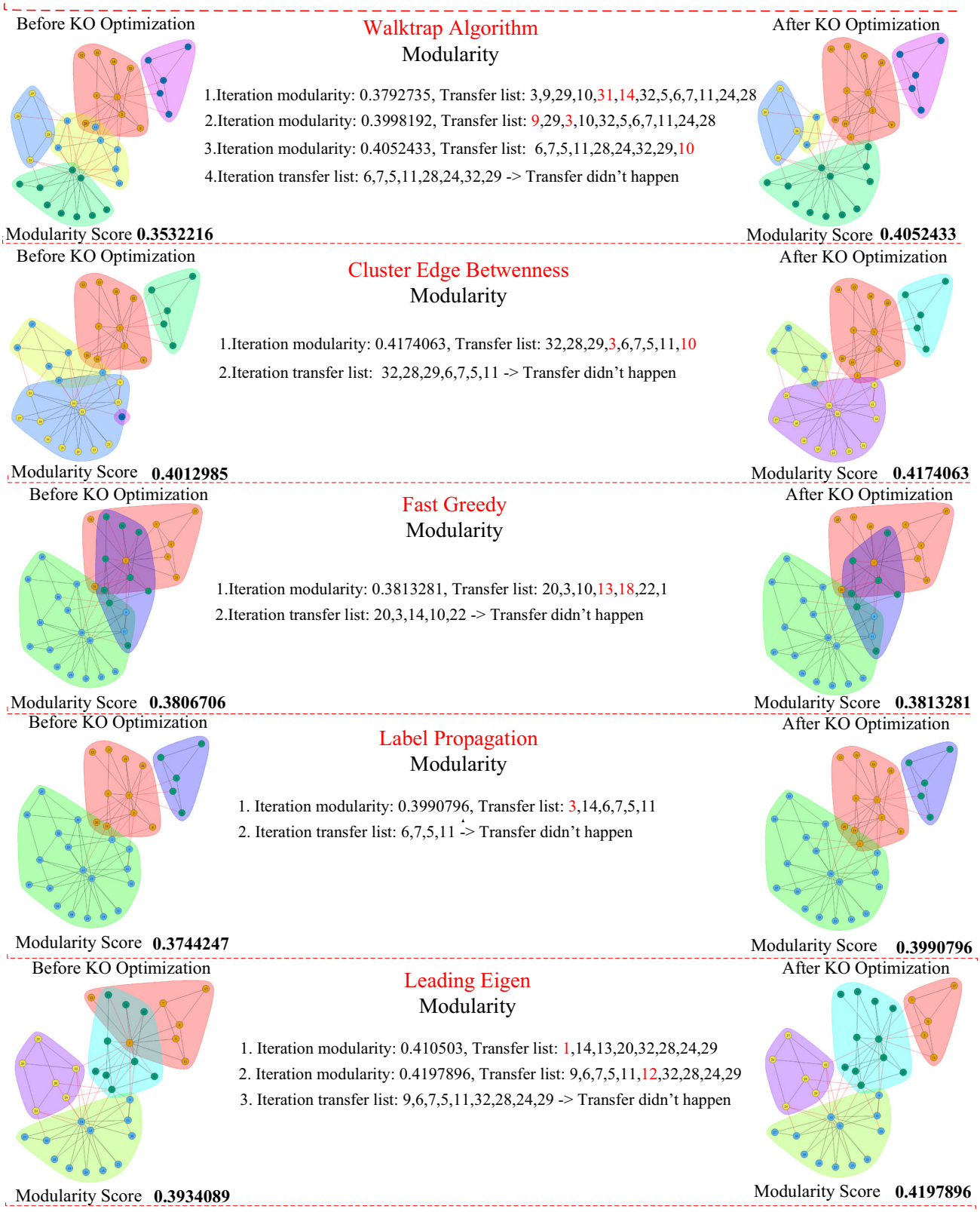


Fig. 7 KO modularity optimization results of Zachary Karate club network

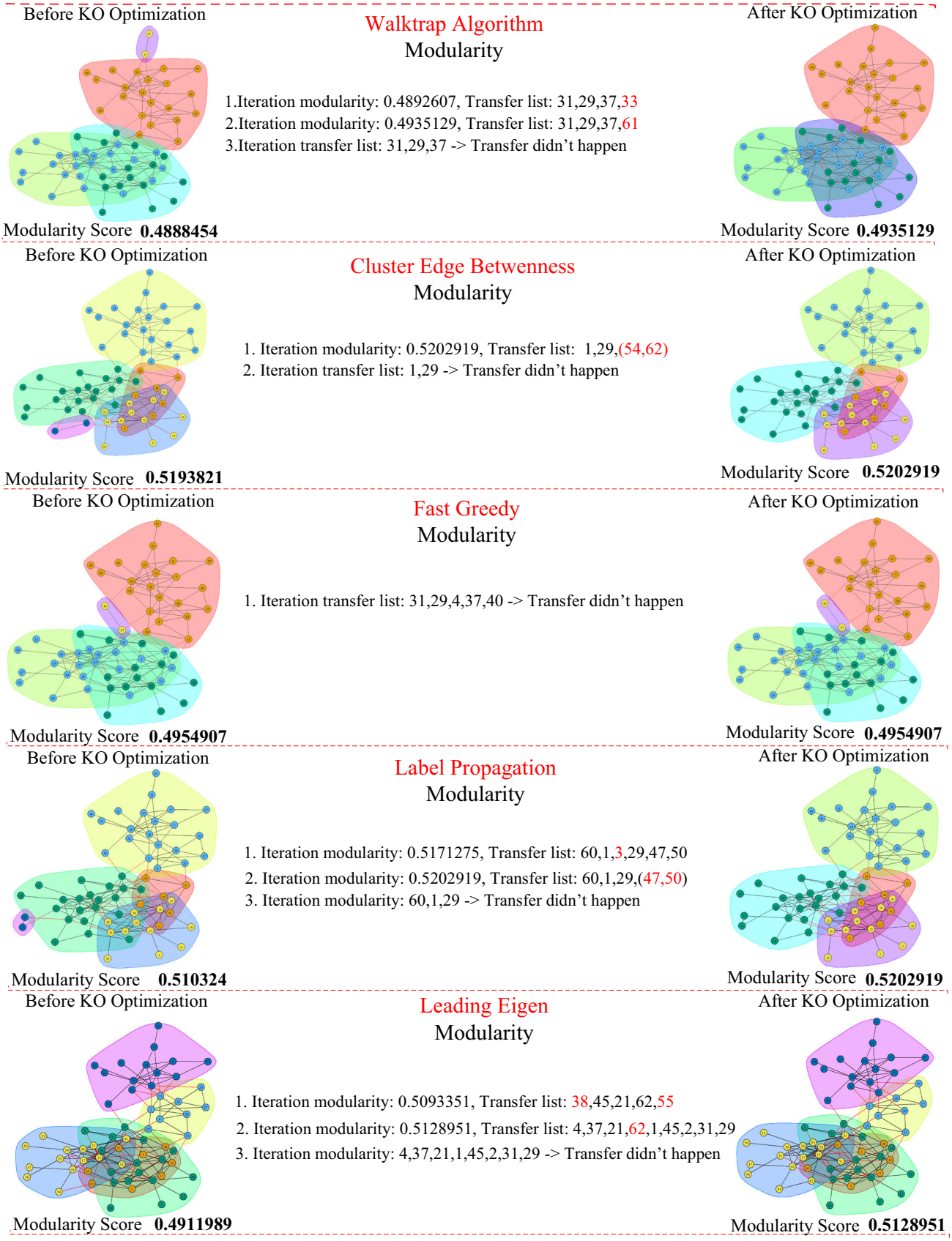
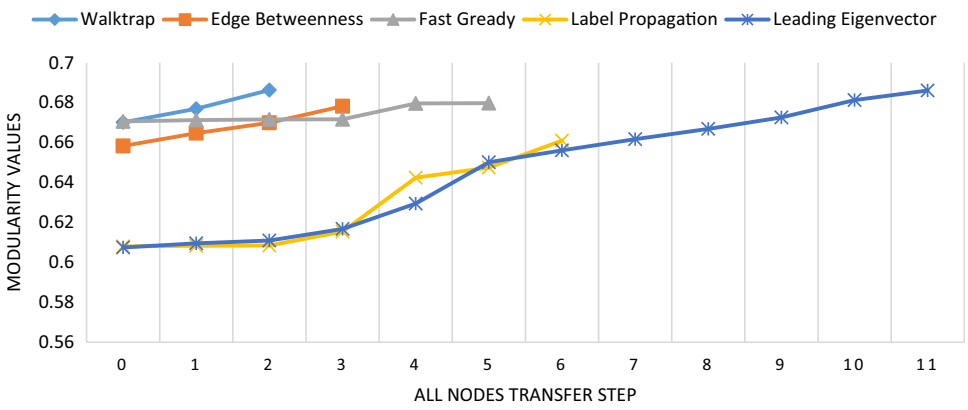
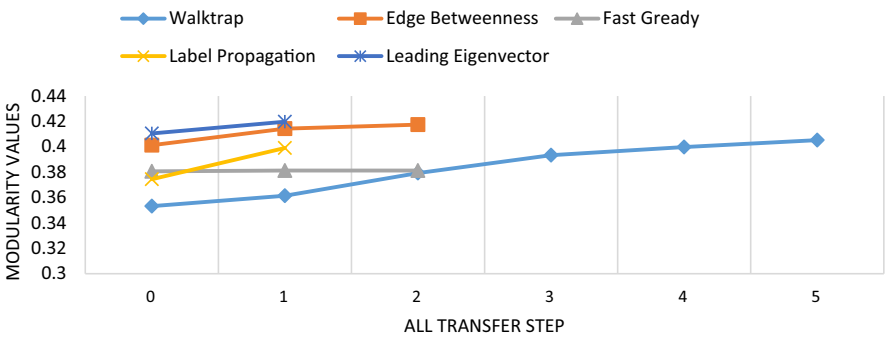


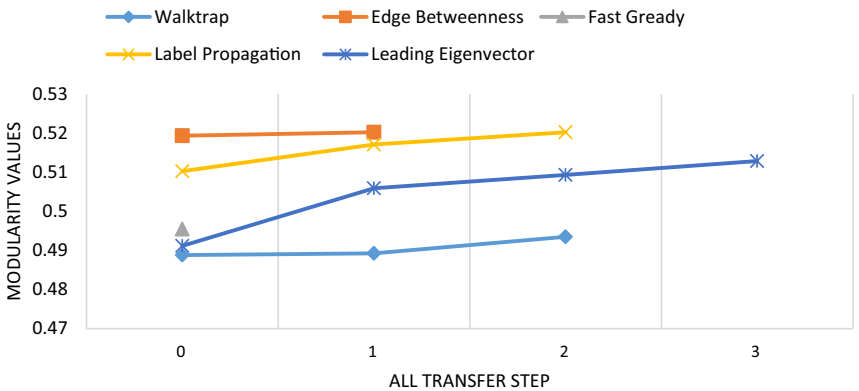
Fig. 8 KO modularity optimization results of Dolphins network



a Complex network's nodes transfer process



b Zachary karate club network's nodes transfer process

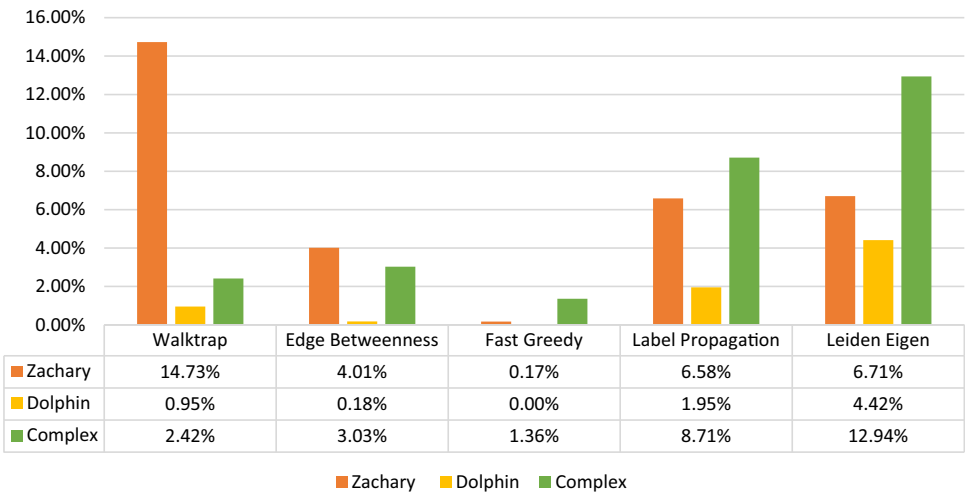


c Dolphin network's nodes transfer process

Fig. 9 The effect of the node transfer process on modularity in all networks

of nodes 13 and 18 in the first iteration, the algorithm stopped working in the second iteration.

In Fig. 8, the optimization results of the Dolphins network were given. When the results are examined, it is seen that the KO algorithm cannot improve the communities created by the Fast Greedy algorithm on the Dolphins network. In the first iteration, since the transfer of nodes 31, 29, 4, 37, and 40 in the transfer list was not approved, the transfer process wasn't carried out to increase the modularity value. The algorithm stops working after the first iteration and does not interfere with the network's community structure. The most significant improvement was in the community created by the Leiden Eigen algorithm. The modularity value of the Dolphin network on which the Leiden Eigen algorithm is applied is 0.4911989. After implementing the KO algorithm, the modularity value increased to 0.5128951. Nodes 38 and 55 in the first iteration and node 62 in the second iteration were



	Walktrap	Edge Betweenness	Fast Greedy	Label Propagation	Leiden Eigen
Zachary	14.73%	4.01%	0.17%	6.58%	6.71%
Dolphin	0.95%	0.18%	0.00%	1.95%	4.42%
Complex	2.42%	3.03%	1.36%	8.71%	12.94%

Fig. 10 KO algorithm modularity optimization rates

transferred. In the third iteration, since no transfer occurred, the algorithm finished its work.

In Fig. 9, the graph of the increase in the modularity values of the algorithms in three different networks after each node transfer was given. In Fig. 9a, the modularity increase trend of the algorithms on the Complex network was presented. When the graph is examined, 11 node transfers were carried out by applying the KO algorithm to the communities formed by the Leading Eigenvector algorithm. After these transfers, the modularity value increased from 0.6074679 to 0.6860761. Figure 9a depicts the graph of the modularity value that changes after each node transfer in the Complex network.

In Fig. 9b, the alteration of the modularity value after each node transfer in the Zachary karate club network was given. When the graph of the Walktrap algorithm is examined, the modularity value increased from 0.3532216 to 0.4052433 after transferring five nodes.

Figure 9c shows the changes in modularity value after each node transfer in the Dolphin network. When the graph of the Leading Eigenvector algorithm is examined, three nodes have been transferred, and the modularity has increased from 0.4911989 to 0.5128951.

In Fig. 10, after applying the KO optimization algorithm, the improvement rates in all the community detection algorithms and all networks applied in the study were given. When the results were examined, the KO algorithm increased the (modularity value) clustering quality of the Walktrap algorithm applied on the Zachary network by 14.73%. In another review, the KO algorithm increased the modularity value of the Leiden Eigen algorithm by 6.71 percent in the Zachary Karate club network, 4.42 percent in the Dolphin network, and 12.94 percent in the Complex network.

5 Results

The present study proposes an effective method to improve the clustering quality of any community detection algorithm applied to any graph. The proposed method aims to increase the modularity of community detection algorithms. According to the determined criteria, the method, which focuses on the transfer of nodes on the boundary of the communities on the graph, has a unique approach in this respect. All stages of the proposed method were given in detail in the study. On three well-known graphs previously used in the literature, communities were determined by applying Walktrap, Cluster Edge Betweenness, Label Propagation, Fast Greedy, and Leading Eigenvector community detection algorithms. These community structures were tried to be improved with the KO optimization algorithm. The algorithm's success was also emphasized in the results of the transfer operations for each iteration in the optimization process. In 14 of the 15 different experimental studies in which the KO algorithm was applied, an increase in the modularity value of the community was achieved in varying proportions from 0 to 14.73%. While the KO algorithm significantly increased the modularity results of the Label Propagation and Leading Eigenvector algorithms, it slightly improved the results of the Fast Greedy algorithm. Community detection algorithms in the literature have been developed with their own unique approaches to producing a solution to a specific problem. Therefore, community detection algorithms can't have optimal modularity in every network. At this point, the KO algorithm has eliminated a significant deficiency. It is a successful modularity optimization method that works independently of graph and community detection algorithms. We are working on some questions to improve the method we propose. To mention the important ones, can the time spent detecting the boundary nodes be reduced?

Studies continue on which types of graphs the KO algorithm gives more successful results. Can a preflight mechanism be added so that nodes whose transfers are not approved do not occupy a place in the Transfer list? It is anticipated that this control mechanism will reduce the memory and time consumed by the algorithm. The KO algorithm currently works on unweighted and undirected graphs. The algorithm is being tested in weighted and directed graphs, and it is expected that it will be published in the future.

Data Availability The datasets used in the study are presented in many different sources. Relevant data can be accessed at www.konect.cc/networks/ and www.kaggle.com (public repository). At the same time, the datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest. Researches are not related to human participants and/or animals.

References

1. Fortunato S, Hric D (2016) Community detection in networks: a user guide. Phys Rep 659:1–44. https://doi.org/10.1016/j.physrep.2016.09.002
2. Chakraborty T, Dalmia A, Mukherjee A, Ganguly N (2018) Metrics for community analysis: a survey. ACM Comput Surv 50(4):37. https://doi.org/10.1145/3091106
3. Kaur S, Singh S, Kaushal S, Sangaiah AK (2016) Comparatıve analysis of quality metrics for community detection in social networks using genetic algorithm. Neural Network World 26(6):625–641. https://doi.org/10.14311/NNW.2016.26.036
4. Akbari H (2021) Exploratory social-spatial network analysis of global migration structure. Social Networks 64:181–193. https://doi.org/10.1016/j.socnet.2020.09.007
5. Newman MEJ, Girvan M (2004) Finding and evaluating community structure in networks. Phys Rev E 69:026113. https://doi.org/10.1103/PhysRevE.69.026113
6. Mengdi L, Ying X (2022) Community detection via network node vector label propagation. Phys A: Statis Mech Appl 593:126931. https://doi.org/10.1016/j.physa.2022.126931
7. Labatut V, Balasque J-M (2012) Detection and interpretation of communities in complex networks. Pract Method Appl. https://doi.org/10.1007/978-1-4471-4048-1_4
8. Zarei M, Samani KA (2009) Eigenvectors of network complement reveal community structure more accurately. Physica A 388(8):1721–1730. https://doi.org/10.1016/j.physa.2009.01.007
9. Öztemiz F (2021) Karmaşık ağlarda hakim düğümlerin belirlenmesi için yeni bir yöntem. İnönü University Institute of Science Ph.D. Thesis
10. Chintalapudi SR, Prasad MHMK (2015) A survey on community detection algorithms in large scale real-world networks. In: 2nd international conference on computing for sustainable global development (INDIACom) 2015, pp. 1323–1327
11. Dickinson B, Hu W (2015) The effects of centrality ordering in label propagation for community detection. Social Networking 4:103–111. https://doi.org/10.4236/sn.2015.44012
12. Bu Z, Zhang C, Xia Z, Wang J (2013) A fast parallel modularity optimization algorithm (FPMQA) for community detection in online social network. Knowl-Based Syst 50:246–259. https://doi.org/10.1016/j.knosys.2013.06.014
13. Shırzad M, Feızı-Derakhshı M-R (2016) Communıty detection in social networks based on modularity optimization. Int J Adv Electron Comput Sci Vol. 3 (4)
14. Fang W, Wang X, Liu L, Wu Z, Tang S, Zheng Z (2022) Community detection through vector-label propagation algorithms. Chaos, Solitons & Fractals 158:456. https://doi.org/10.1016/j.chaos.2022.112066
15. Hamann M, Strasser B, Wagner D, Zeitz T (2018) Distributed graph clustering using modularity and map equation. Lect Notes Comput Sci 11014:688–702. https://doi.org/10.1007/978-3-319-96983-1_49
16. Souam F, Aïtelhadj A, Baba-Ali R (2014) Dual modularity optimization for detecting overlapping communities in bipartite networks. Knowl Inf Syst 40:455–488. https://doi.org/10.1007/s10115-013-0644-8
17. Newman MEJ (2006) Modularity and community structure in networks. Proc Natl Acad Sci 8577–8582(103):23. https://doi.org/10.1073/pnas.0601602103
18. Aloise D, Caporossi G, Hansen P, Liberti L, Perron S, Ruiz M (2012) Modularity maximization in networks by variable neighborhood search. Graph Partitioning Graph Clustering. https://doi.org/10.1090/conm/588/11705
19. Lee J, Gross SP, Lee J (2012) Modularity optimization by conformational space annealing. Phys Rev E 85:056702. https://doi.org/10.1103/PhysRevE.85.056702
20. Zhang XS, Wang RS, Wang Y, Wang J, Qiu Y, Wang L, Chen L (2009) Modularity optimization in community detection of complex networks. EPL (Europhysics Letters). https://doi.org/10.1209/0295-5075/87/38002
21. Hollocou A, Bonald T, Lelarge M (2019) Modularity-based Sparse Soft Graph Clustering. In: Proceedings of the Twenty-Second international conference on artificial intelligence and statistics in proceedings of machine learning research 89: 323–332. Available from https://proceedings.mlr.press/v89/hollocou19a.html
22. Liu X, Murata T (2010) Advanced modularity-specialized label propagation algorithm for detecting communities in networks. Physica A 389(7):1493–1500. https://doi.org/10.1016/j.physa.2009.12.019
23. Wu L, Zhang Q, Chen C, Guo K, Wang D (2020) Deep learning techniques for community detection in social networks. IEEE Access 8:96016–96026. https://doi.org/10.1109/ACCESS.2020.2996001
24. Waltman L, van Eck NJ (2013) A smart local moving algorithm for large-scale modularity-based community detection. Eur Phys J B 86:471. https://doi.org/10.1140/epjb/e2013-40829-0
25. Schuetz P, Caflisch A (2008) Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. Phys Rev E 77:046112. https://doi.org/10.1103/PhysRevE.77.046112
26. Chen M, Kuzmin K, Szymanski BK (2014) Community detection via maximization of modularity and its variants. IEEE Trans Comput Social Syst 1(1):46–65. https://doi.org/10.1109/TCSS.2014.2307458
27. Lehmann S, Hansen L (2007) Deterministic modularity optimization. Eur Phys J B 60:83–88. https://doi.org/10.1140/epjb/e2007-00313-2
28. Nascimento MCV, Pitsoulis L (2013) Community detection by modularity maximization using GRASP with path relinking.

Comput Oper Res 40(12):3121–3131. https://doi.org/10.1016/j.cor.2013.03.002
29. Aung TT, Nyunt TTS (2020) Modularity based ABC algorithm for detecting communities in complex networks. Int J Mach Learn Comput 10(2):323–329. https://doi.org/10.18178/ijmlc.2020.10.2.938
30. Xu G, Guo J, Yang P (2021) TNS-LPA: An improved label propagation algorithm for community detection based on two-level neighbourhood similarity. IEEE Access 9:23526–23536. https://doi.org/10.1109/ACCESS.2020.3045085
31. Yan M and Guoqiang C (2021) Label propagation community detection algorithm based on density peak optimization. In: 17th International conference on computational intelligence and security (CIS) pp 80–84. https://doi.org/10.1109/CIS54983.2021.00025
32. Brandes U et al (2008) On modularity clustering. IEEE Trans Knowl Data Eng 20(2):172–188. https://doi.org/10.1109/TKDE.2007.190689
33. Trajanovski S, Kuipers FA, Martín-Hernández J, Van Mieghem P (2013) Generating graphs that approach a prescribed modularity. Comput Commun 36(4):363–372
34. Igraph Library. https://igraph.org/. Accessed 19.08.2022
35. Öztemiz F, Karci A (2022) Bağlı Graflarda Etkili Düğümlerin Belirlenmesinde Yeni Bir Yaklaşım. Dokuz Eylül Üniversitesi Mühendislik Fakültesi Fen ve Mühendislik Dergisi 24(70):143–155. https://doi.org/10.21205/deufmd.2022247014
36. Cerdeira JO, Silva PC (2021) A centrality notion for graphs based on Tukey depth. Appl Math Comput 409:545. https://doi.org/10.1016/j.amc.2021.126409
37. Laassem B, Idarrou A, Boujlaleb L, Iggane M (2022) Label propagation algorithm for community detection based on Coulomb's law. Phys A: Statis Mech Appl 593:35435. https://doi.org/10.1016/j.physa.2022.126881
38. Acharya DB, Zhang H (2020) Community detection clustering via gumbel softmax. SN Comput Sci 1:262. https://doi.org/10.1007/s42979-020-00264-2
39. Konect. http://konect.cc/networks/. Accessed: 20.08.2022
40. Harenberg S, Bello G, Gjeltema L, Ranshous S, Harlalka J, Seay R, Padmanabhan K, Samatova N (2014) Community detection in large-scale networks: a survey and empirical evaluation. WIREs Comput Stat 6:426–439
41. Fortunato S (2010) Community detection in graphs. Phys Rep 486(3–5):75–174
42. Öztemiz F ve Karci A (2021) Topluluk Tespiti Yöntemi ile Ulaşım Ağında Verimli Yeşil Dalga Koridorlarının Belirlenmesi. Politeknik Dergisi ss. 1–1. https://doi.org/10.2339/politeknik.1074962
43. Javed MA, Younis MS, Latif S, Qadir J, Baig A (2018) Community detection in networks: a multidisciplinary review. J Netw Comput Appl 108:87–111. https://doi.org/10.1016/j.jnca.2018.02.011
44. Gates KM, Henry T, Steinley D, Fair DA (2016) A monte carlo evaluation of weighted community detection algorithms. Front Neuroinform 10:5435. https://doi.org/10.3389/fninf.2016.00045
45. Hoffman M, Steinley D, Gates KM, Prinstein MJ, Brusco MJ (2018) Detecting Clusters/communities in social networks. Multivariate Behav Res 53(1):57–73
46. Newman ME (2004) Fast algorithm for detecting community structure in networks. Phys Rev E 69(6):066133. https://doi.org/10.1103/PhysRevE.69.066133
47. Jokar E, Mosleh M (2019) Community detection in social networks based on improved label propagation algorithm and balanced link density. Phys Lett A 383(8):718–727. https://doi.org/10.1016/j.physleta.2018.11.033
48. Xing Y, Meng F, Zhou Y, Zhu M, Shi M, Sun G (2014) A node influence based label propagation algorithm for community detection in networks. Sci World J. https://doi.org/10.1155/2014/627581
49. Cordasco G, Gargano L (2010) Community detection via semi-synchronous label propagation algorithms. IEEE International Workshop on: Business Applications of Social Network Analysis (BASNA) pp. 1–8. https://doi.org/10.1109/BASNA.2010.5730298
50. Christensen AP, Garrido LE, Golino H (2020) Comparing community detection algorithms in psychological data: a Monte Carlo simulation. PsyArXiv. https://doi.org/10.31234/osf.io/hz89e
51. Newman MEJ (2006) Finding community structure using the eigenvectors of matrices. Phys Rev E 74:036104. https://doi.org/10.1103/PhysRevE.74.036104
52. Aktunc R, Toroslu IH, Ozer M and Davulcu H (2015) A dynamic modularity based community detection algorithm for large-scale networks: DSLM. (ASONAM '15) New York USA 1177–1183. https://doi.org/10.1145/2808797.2808822
53. Optimal Cluster. https://igraph.org/r/doc/cluster_optimal.html. Accessed: 29.07.2022
54. Danon L, Duch J, Díaz-Guilera A, Arenas A (2005) Comparing community structure identification. J Stat Mech 2005(9):P09008. https://doi.org/10.1088/1742-5468/2005/09/P09008
55. Botta F, del Genio C (2016) Finding network communities using modularity density. J Stat Mech: Theory Exp. https://doi.org/10.1088/1742-5468/2016/12/123402