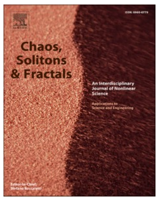
Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

journal homepage: www.elsevier.com/locate/chaos



LMFLS: A new fast local multi-factor node scoring and label selection-based algorithm for community detection

Huxiong Li [a], Samaneh Salehi Nasab [b], Hamid Roghani [c], Parya Roghani [d], Mehdi Gheisari [a,g,h,i,*], Christian Fernández-Campusano [e], Aaqif Afzaal Abbasi [f], Zongda Wu [a,*]

[a] Institute of Artificial Intelligence, Shaoxing University, Zhejiang, China
[b] Department of Engineering, Lorestan University, Aleshtar Campus, Aleshtar, Iran
[c] Department of Computer Engineering, Azarbaijan Shahid Madani University, Tabriz, Iran
[d] Department of Biology, Islamic Azad University Central Tehran Branch, Tehran, Iran
[e] Department of Electrical Engineering, University of Santiago of Chile (USACH), Santiago, 9170124, Chile
[f] Department of Earth and Marine Sciences, University of Palermo, Palermo, Italy
[g] Department of Computer Science and Engineering, Saveetha School of Engineering, Saveetha Institute of Medical and Technical Sciences, Chennai, 602105, India
[h] Department of Computer Science, Islamic Azad University, Tehran, Iran
[i] Department of R&D, Shenzhen BKD Co.LTD, Shenzhen, China

ARTICLE INFO

Keywords:
Community detection
Local similarity
Multi-factor scoring
Label selection

ABSTRACT

Community detection is still regarded as one of the most applicable approaches for discovering latent information in complex networks. To meet the needs of processing large networks in today's world, it is important to propose fast methods that have low execution time and fast convergence speed, while maintaining algorithmic accuracy. To overcome these issues, a fast local multi-factor node scoring and label selection-based (LMFLS) method with low time complexity and fast convergence is proposed. Node scoring step incorporates diverse metrics to better assess impact of nodes from different aspects and obtain more meaningful order of nodes. In second step, to construct and stabilize initial structure of communities, an efficient label assignment technique based on the selection of the most similar neighbor is suggested. Moreover, two label selection strategies are proposed to significantly enhance the accuracy and improve convergence of the algorithm. During the label selection step, each node in graph tends to choose the most appropriate label based on a multi-criteria label influence from its surrounding nodes. Finally, by utilizing a novel merge method, small group of nodes are merged to form the final communities. Meanwhile, since drug repositioning is one of the popular research fields in therapeutics, to extend the application of the proposed algorithm in practical context, the LMFLS algorithm is applied on Drug-Drug network to find potential repositioning for drugs. Thorough experiments are conducted on both actual real-world networks and synthetic networks to assess the algorithm's performance and accuracy. The findings demonstrate that the proposed method outperforms state-of-the-art algorithms in terms of both accuracy and execution time.

1. Introduction

Complex systems such as social networks [1], information systems [2], Protein-Protein interaction [3], drug-drug or disease-disease network [4], citation networks [5], recommender systems [6], and etc. in real-world have the potential to be modeled with graphs and complex networks, such that the nodes are adopted to show the elements and the relationships between them are demonstrated with links which can be weighted or unweighted. Modeling complex systems with graphs makes it possible to investigate the interactions between entities using existing tools such as community detection. The term community in a network consists of nodes that form a tightly connected group within a subgraph and sparsely connected with nodes of other communities. Community detection is regarded as an efficient tool for revealing latent information by grouping similar entities in the same communities [7].

Identifying hidden groups within networks is crucial for various

* Corresponding authors.
E-mail addresses: 2019000060@usx.edu.cn (H. Li), salehinasab.s@lu.ac.ir (S.S. Nasab), h.roghani@azaruniv.ac.ir (H. Roghani), mehdi.gheisari61@gmail.com (M. Gheisari), christian.fernandez@usach.cl (C. Fernández-Campusano), zongda1983@163.com (Z. Wu).

https://doi.org/10.1016/j.chaos.2024.115126
Received 30 May 2024; Accepted 3 June 2024
0960-0779/© 2024 Published by Elsevier Ltd.

applications, including community structure analysis, detection of influential nodes, and network function exploration. For instance, social network analysis allows for the discovery of user communities, influential users, and their relationships [8]. Similarly, Drug-Disease networks can reveal potential drug repurposing candidates [9], protein-protein networks can elucidate protein functions [10], and recommender systems can leverage customer segmentation to suggest relevant products [6].

While community detection in social networks is a well-studied area with constant innovation, efficiently identifying communities in terms of accuracy, convergence speed, stability and robustness, execution time, and so forth in complex network, continues to be unresolved. Existing methods based on global metrics, though informative, become impractical in networks with massive number of nodes due to their demanding computational requirements (e.g., Newman et al.'s edge-betweenness with $O(n^3)$ time complexity [11]). In contrast, methods exists which utilize local information of nodes to measure the similarity and proximity between them [12] without regarding the total information of the graph. Primarily, local methods involve computing the local distance or identifying common neighbors between two adjacent nodes. Consequently, the time complexity of these methods for n nodes is proportional to $O(nk^2)$. This computational efficiency renders them well-suited for large-scale networks.

The Label Propagation Algorithm (LPA) stands as a popular method for detecting communities within networks which relies on the propagation of labels among nodes [13]. While LPA has nearly linear time complexity, it exhibits low accuracy and instability in community detection due to weaknesses related to random node selection and label updating. Variants like NIBLPA [14], LPA-Intimacy [15], LPA-NI [16], CenLP [17], LINSIA [18], and etc. improve accuracy by considering one criterion for node importance and label influence. The findings indicate that accounting for node importance and label influence can enhance the accuracy of the algorithm. However, drawbacks remain, including instability and slow convergence for large networks. Also, adopting multiple steps for the tie break operation, will add extra complexity to the algorithm which impacts their running time.

The reasons above prevent these kinds of methods to be executed on large networks. Another point that should be concerned is that adopting one criterion for assigning importance to nodes would not be adequate for efficiently evaluating them, since nodes, especially core nodes, have more different features that a good combination of them can better reflect their importance. If the importance of the nodes is effectively distinguished from one another, it will aid the algorithm in selecting the correct labels without requiring multiple tiebreak steps. As an instance, a node exhibiting a high degree, is not necessarily a core node. A node with high degree but low similarity with its neighbors, may probably have low K-shell score. In Fig. 1, node P, despite its relatively high degree, resides in shell 1 due to the absence of common neighbors. To address this limitation, considering the sum of common neighbors between nodes, along with other factors, can enhance the accuracy of selecting important nodes—a primary objective of this research.

Core expansion-based approaches aim to identify core nodes by considering their features and expanding initial communities from seed nodes outward [19,20]. However, accurately determining core nodes demands additional effort, including defining precise measures, selecting an appropriate number of seed nodes, and establishing similarity and merging criteria for integrating nodes into the initial community which can be a challenging task. Also, various community detection methods, including Louvain [21], Leiden [22], and etc. have primarily focused on maximizing modularity gain within communities. However, it is essential to recognize that community detection aims not only to optimize modularity but also to reveal the genuine structure of communities. The task involves more than simple network partitioning; it must consider the diverse roles of nodes [21]. Unfortunately, modularity-based approaches suffer from significant limitations, such as disregarding topological information, node similarity, trapping in local maxima, greedy

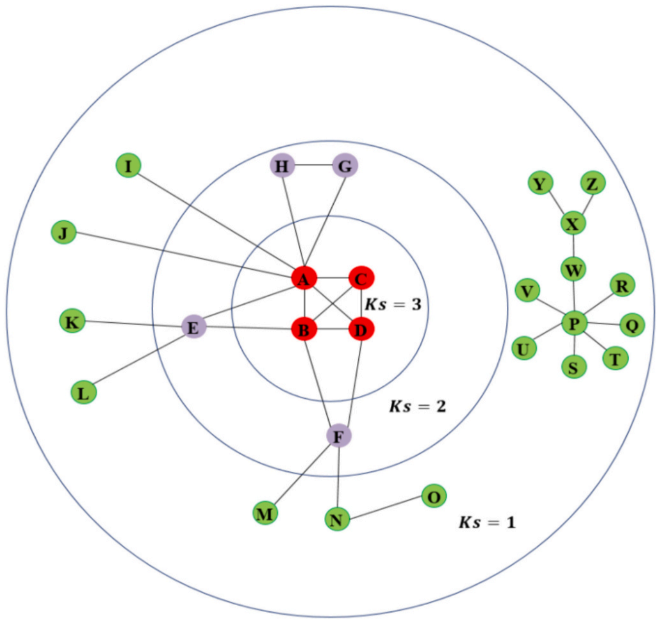


Fig. 1. A network with three shells.

nature, and varying node importance, leading to inaccurate results. Despite numerous improvements, the accuracy of these methods remains unchanged. For example, experiments cited in [22] demonstrate no significant difference in accuracy between the Leiden and Louvain algorithms.

To improve the defects of the mentioned approaches above, label diffusion-based methods are suggested which can be considered as a new fast technique which diffuse label to a batch of nodes at a same time if they satisfy the desired conditions. These community detection methods mitigate the limitations associated with core node identification, initial label assignment, and certain issues encountered in label propagation approaches. Furthermore, it enhances the computational efficiency of the algorithm while preserving accuracy [23]. However, they require strict control conditions to prevent incorrect label diffusion, since a group of nodes are assigned with label at the same time and incorrect label assignment will result in distribution of the wrong label to other nodes around them. Handling large-scale networks with numerous nodes and achieving satisfactory community partitioning is challenging due to the network's sheer size and time complexity constraints. To address this, precise design of community detection algorithms is crucial. While the LPA and its improved versions offer a straightforward approach to community detection with linear time complexity, their performance can be hindered on large networks due to slow convergence. The iterative nature of the algorithms, which involves examining the labels of all neighboring nodes at each step, can lead to significant computational overhead, particularly when dealing with extensive datasets. This limitation can render LPA impractical for large-scale network analysis. Experiments in [23,24] prove this fact. While alternative methods like the Leiden [25] and Louvain [26] algorithms offer faster detection capabilities compared to LPA-based approaches, their accuracy often falls short. In many cases, these methods tend to underestimate the true number of communities present in the network. This issue is mostly seen in large networks [23]. Most of the methods mentioned above succeed when the network is sufficiently dense [27].

Finally, to exemplify how a community detection algorithm like LMFLS can be utilized in practical scenarios, a case study on drug repositioning will be conducted. There are also high similarities between the LMFLS community detection algorithm and the problem of drug repositioning based on Drug-Drug complex network. In this case the corresponding concepts are as follows:

Network representation: Drug repositioning involves constructing a network where drugs are regarded as nodes similar to LMFLS algorithm, and their interactions are modeled with edges and the similarity score between them is indicated with weights which are used by LMFLS algorithm to group drugs based on their similarities. The LMFLS community detection algorithm can identify clusters of related drugs within this network by using the similarity between drugs and the weight of edges between them.

Similarity-based approaches: Community detection algorithms often rely on similarity metrics. In drug repositioning, the Drug-Drug associations are based on shared targets, biological pathways, chemical structure, and so forth. Similarity measures can help identify potential repurposing candidates.

Finding hidden associations: The LMFLS community detection algorithm reveals hidden patterns within complex networks. The community concept in the LMFLS method is regarded as a group of clustered drugs having similar efficacies. By identifying drug-drug communities, unexpected relationships between drugs may discover which is regarded as drug repositioning. For example, a drug used for one condition might be effective for a seemingly unrelated disease and can be grouped with other drugs.

Multi-modal data integration: Drug repositioning combines diverse data sources. Community detection algorithms handle multi-modal networks effectively. They integrate information from different domains, aiding in repurposing predictions.

Handling noisy data: Drug-Drug associations can be noisy due to incomplete evidence. Community detection algorithms are robust to noise and can extract meaningful structures.

Scalability and efficiency: Large-scale Drug-Drug networks require efficient algorithms. Community detection methods, especially the LMFLS algorithm handles large graphs, making them suitable for drug repositioning.

To solve the mentioned problems above, the primary contribution of this paper lies in introducing a fast local multi-factor node scoring and label selection-based algorithm, called LMFLS, with low time complexity and fast convergence which can be executed on weighted and unweighted networks and is significantly capable of being executed on massive networks. In order to improve the procedure of assigning importance to nodes, several factors are combined to evaluate nodes from different aspects. An improved version of H-index is proposed and is combined with the improved version of K-shell decomposition and other factors to assess the dominance power of nodes. Inspired by the natural way communities form in the real world, at first step of the algorithm each node is grouped with its most important neighbor, which stabilizes the structure of communities for the next steps which enhances the speed of convergence and the accuracy rate of the method. An efficient method for label selection is proposed where each node selects the most appropriate label around it based on combination of metrics. Also, by excluding nodes with degree 1 and presenting a new idea for preventing some of the nodes from updating their label in each iteration, the performance of the algorithm is considerably improved. To avoid performing tie break operation and to decrease running time of the suggested method, an efficient approach is adopted to better distinguish the difference of label influence. In addition to label selection step, a novel and efficient approach is proposed to integrate small group of nodes with neighbor communities. Finally, as an application, the LMFLS algorithm is applied to Drug-Drug network to explore new and potential drug repositioning candidates. The primary contributions of this research work can be succinctly outlined as follows:

- A comprehensive criterion which adopts multiple factors for assigning importance to nodes is proposed to evaluate nodes from diverse aspects to better distinguish and rank them.
- A new method for label assignment based on the most similar neighbor is suggested to form initial communities and to consolidate the core structure of detected communities for subsequent steps.
- An efficient label selection step with fast convergence and low computational cost is employed to select the most suitable label which avoids additional tie break operation.
- A new and fast merge method is presented based on the concept of integration of dominions.
- The LMFLS method is capable of being applied on weighted and unweighted networks.
- A streamlined data structure is introduced to address memory challenges when dealing with extensive networks.

In the subsequent sections, we delve into the existing related studies (Section 2), provide a comprehensive exposition of our novel algorithm (Section 3), discuss the empirical evaluation of our approach using both real-world and synthetic datasets, including the Drug-Drug network for drug repositioning (Section 4), and finally, conclude by outlining potential future directions (Section 5).

2. Related work

Given the crucial role of community detection in uncovering hidden patterns and relationships within networks, this section delves into a comprehensive examination of various algorithms. Recently, researchers have directed their primary attention toward the development of local methods especially label propagation, label diffusion, and modularity-based methods.

Modularity-based methods formulate the community detection problem as an optimization task, aiming to identify optimal partitions within a network by maximizing the density of connections within communities while minimizing the connections between them. Clauset et al. in [28] suggested CNM, a hierarchical method based on modularity by utilizing a max-heap structure to enhance the performance of previous methods. Later, Blondel et al. proposed Louvain method, an improved and fast version of modularity-based methods [26]. This heuristic algorithm involves creating initial communities by merging nodes, with the objective of achieving greater modularity gain. Leiden [25], a method based on modularity metric, modifies the existing limitations in Louvain and improves the running time by suggesting local moving strategy. Approaches suggested in [12,29] share similarities and employ a two-stage process for community detection. At first, a similarity criterion is used to assign weights to edges. Next, the algorithm removes edges from graph which have a weight less than the defined threshold. The algorithm then iteratively merges these communities if doing so improves the overall modularity value.

Raghavan et al. was the first one who suggested the Label Propagation Algorithm (LPA) [13]. This approach employs a random node order and iteratively assigns labels to nodes based on the most frequent label within their neighborhood. K-shell decomposition, Bayesian network, and the notion of different influence of nodes on each other are instances which are suggested to address the shortcomings of the LPA by defining an order for selection of nodes based on their importance and adopting label influence concept to improve label assignment task [14–16]. Authors in [30] introduced a node importance-based variant of the label propagation algorithm. The Jaccard similarity metric along with signal propagation strategy is suggested to enhance the method of assigning importance to nodes. Authors introduced a novel community detection method that leverages boundary nodes and label propagation [31]. However, this approach faces challenges such as low accuracy, formation of large communities, and slow convergence. Li et al. introduced DS-LPA, a robust community detection algorithm that combines Density Peak Clustering and Label Propagation [32]. By improving local density calculations and considering information transmission power, DS-LPA outperforms other algorithms in terms of both accuracy and efficiency, especially for large datasets. An efficient algorithm for large data sets, introduced by [33], constructs a weighted graph by combining node attributes and topological structure. Node centrality is determined using Laplacian centrality, and the process of selecting and updating nodes is refined by incorporating their influence.

To the best of our knowledge, label diffusion-based methods, due to their novel nature, have been less frequently presented by other researchers. Bouyer et al. introduced the LSMD algorithm, which identifies accurate communities without explicitly pinpointing core nodes [24]. LSMD starts from low-degree nodes, using a multi-step label diffusion strategy. Nodes are grouped by degree, and labels propagate through a multi-level diffusion process based on specific conditions. Roghani et al. introduced PLDLS, a Spark-based parallel community detection algorithm. It combines label diffusion with a novel label selection method, enhancing accuracy and scalability [34]. To enhance the accuracy and the strategy of label diffusion, an approach based on balanced diffusion of labels with fast running time was proposed in [22]. Obtaining rough cores of communities besides adopting balanced label diffusion strategy for constructing communities from central parts as well as border regions of communities, can result in obtaining more accurate communities.

Core expansion-based approaches are another category of local methods which involves discovering a group of nodes as core or seed nodes, and then extending initial communities. RTLCD is suggested based on the idea of discovering central nodes of communities [19] and adopting local metrics to expand initial communities. Authors put forward CFCD algorithm [35] which involves discovering seed nodes and expanding initial group of nodes. The ECES algorithm, suggested by Berahmand et al. is a method based on expansion of communities by utilizing a membership index [36]. CDME is another approach for community detection based on expansion of core nodes which is inspired by Matthew effect [37]. Pares et al. suggested a community detection algorithm based on label propagation, called FluidC, which leverages the model of fluid dynamics to discover communities [38]. You et al. introduced a novel community discovery method that involves identifying kernel nodes, propagating labels, and merging discovered communities [39].

2.1. Drug repositioning

Since in this paper heterogeneous biological network of drugs is adopted for modeling interaction between drugs, it is crucial to be familiar with databases as they are utilized to extract data and construct the interaction networks. Masoudi et al. conducted a comprehensive investigation of essential databases, along with their advantages and disadvantages, that are used to extract diverse information about drug repositioning [40]. Authors in [9] constructed a heterogeneous weighted drug-disease network by incorporating the relationship of Disease-Gene and Drug-Target. Groza et al. [41] created a drug–gene network using DrugBank and applied projection to generate a drug–drug similarity network. Then they applied modularity-based clustering to identify the potential candidate drugs for repurposing. A new method of analyzing the interaction of drugs based on the Drug-Drug network by utilizing community detection algorithm is proposed in [42]. The results of their study reveal functional drug categories and intricate relationships between them. Sia et al. [43] introduced an innovative community detection algorithm, founded on the removal of negatively curved edges, and tested this novel approach to analyze drug-drug networks generated from DrugBank. Koss et al. [44] used the patients' perspectives reflected in social media, to construct a network of drugs and diseases and employ the Leiden algorithm for community detection to identify potential drugs for repositioning.

3. The proposed method

This section presents a detailed explanation of the proposed LMFLS method, outlining its various steps in subsequent subsections. To illustrate the practical application of the algorithm, its steps are demonstrated through a case study using a real-world dataset.

3.1. The proposed method

The pseudo-code of Algorithm 1 outlines the core functionality of the LMFLS algorithm. In the Algorithm 1, *ID* of nodes is a numerical number such as 1, 2, 3, and etc. which indicates the name or the ID of a node in graph, *is_dense_flag* is used to show that if a node is located in a dense part or not. The total importance of node *i* is shown by *TotalImportance(i)*. *Label (i)* and *Label (i)* ← *j* shows label of node *i* and the label assignment action, respectively. The most similar neighbor of node *i* is demonstrated by *mostSimilarNeighbor (i)*. For a detailed understanding of the implementation of the algorithm, the source code of the LMFLS algorithm is provided in Appendix A.

After performing improved K-Shell decomposition step, nodes having a degree of 1 are disregarded and do not participate in any stage of the algorithm. It should be noted that the execution of the merge step is optional. According to the fact that in complex networks a significant proportion of nodes have a degree of 1, adopting this technique prevents the algorithm from wasting time on performing computations on nodes with degree 1 which do not have any impact on the procedure of the algorithm. By applying this technique and utilizing the *is_dense_flag* concept to exclude nodes situated in dense regions of the graph, the algorithm enhances its execution time, especially for large-scale graphs. Then ID of nodes is assigned as the initial label of nodes which indicates that each node belongs to a separate community at the beginning of the algorithm.

Algorithm 1. LMFLS Algorithm.

```
Input: Neighboring list of nodes
Output: Final communities
1.  Exclude nodes with degree 1 from step 2 until step 8.
2.  Calculate total importance of all nodes using Eq. (6), then sort them
    descending.
3.  Assign ID of nodes as their initial label, and is_dense_flag ← 0
4.  perform step 4 for nodes with degree more than 1.
Select node $i$ and its most intimate neighbor $j$ using Eq. (7)
If (TotalImportance(i) <= TotalImportance(j)) Then Label ($i$) ← $j$
Else select intimate neighbor $k$ of node $j$
  - if ($i == k$) Then Label ($i$) ← $i$, Label ($j$) ← $i$
  -Else if ($i \neq k$): If (TotalImportance(i) >= TotalImportance(k)) Then Label ($i$)
← $i$, Label ($j$) ← $i$, Label ($k$) ← $i$
-Else Then Label ($i$) ← $k$, Label ($j$) ← $k$, Label ($k$) ← $k$
5.  Update label assignment of step 4
6.  Perform fast label selection step for nodes that is_dense_flag == 0
7.  Perform final label selection step for nodes with degree more than 1
-Label($i$) ← label(mostSimilarNeighbor(i))
8.  Perform fast Merge step
```

By adopting a meaningful combination of several concepts, an efficient algorithm is proposed which evaluates properties of nodes from a new point of view. In the context of designing an efficient algorithm, one should conceptualize a community detection as a composite system comprising distinct components. Each of these components performs its designated task, contributing to the overall efficiency of the system. Subsequent steps then utilize the outcomes from preceding steps, aiming to enhance the overall performance. This can be achieved through the development of efficient data structures, novel metrics for node importance, and a streamlined algorithm design. This study employs a neighbor list data structure to address the challenges associated with processing large-scale networks [84]. This approach facilitates rapid access to neighboring nodes, resulting in efficient processing times and optimized utilization of RAM resources. From a conceptual standpoint, the neighbor list structure is analogous to a dictionary in the Python programming language. In this structure, node serves as the key, and the associated value is a list containing all neighboring nodes. Besides this, since the proposed method is adaptable, capable of handling both weighted and unweighted network structures, the structure can maintain the weight of edges between nodes.

Unlike other approaches, such as label propagation or modularity-based methods, which redundantly repeat the same actions multiple times without significantly enhancing the performance of the algorithm, this paper takes a different approach. Rather than persistently iterating identical steps, a set of distinct steps is employed, mostly each executed either once, or in some situations with a maximum of two repetitions. For example, one step is regarded as a structure stabilizer (node attraction step), the next step is based on label selection which is executed with maximum of 2 iterations (fast label selection step), then final label checking step is adopted to correct mistakes in label assignment (final label selection step) which is executed with maximum of 2 iterations, a merge step is executed to integrate small communities to improve wrong partitioning of previous steps (fast merge step), and finally label assignment to degree 1 nodes is performed. It is postulated that by conducting community detection in a manner akin to how communities naturally emerge in the real world, communities that closely resemble those found in actual social structures will be formed. Inspired by this fact, by suggesting node attraction step, initial robust communities are emerged that gives stability and robustness to the algorithm which results in prevention of generating new communities by the algorithm and reducing variation of labels in further steps. It has also been considered how individuals in the real world select the most appropriate group for themselves, and the label selection step is designed.

3.1.1. Node importance

Node importance is a concept that quantifies the influence or significance of a node in a complex network. Nodes may have different impacts on the network such as forming central parts of the communities, spreading information, controlling the flow, maintaining the connectivity, and so forth. According to the power-law [7], there are only a few nodes that have higher influence and there are a lot of nodes which are under the influence of the important nodes. These important nodes which are usually called core nodes, have more than one property that can be investigated to better distinguish them from other nodes. Nodes with high influence typically exhibit strong control over adjacent nodes (have higher influence on them), possess a large number of edges, high similarity sum, maintain extensive links with the second-degree neighbors, and are located in regions with high density.

To accurately evaluate the significance of nodes within a graph, a multifaceted approach is employed, utilizing multiple metrics derived from distinct conceptual frameworks. Each metric is designed to attribute higher scores to nodes of greater importance, enabling their classification as core nodes or peripheral nodes. Relying solely on a single metric may result in the assignment of identical scores to nodes with varying degrees of influence. For instance, nodes P and U in Fig. 1 exhibit contrasting degrees but possess the same shell score when evaluated using a singular metric. This hinders algorithmic efficiency in label assignment. To address this limitation, we propose the combination of multiple metrics to comprehensively assess node importance. By leveraging a diverse array of perspectives, this approach provides a more nuanced and accurate evaluation of node significance. In the proposed approach, an improved K-shell metric, an improved H-index, along with the degree of nodes, the number of common neighbors, the edge weight, and other factors are combined to develop a comprehensive metric capable of effectively evaluating the role of nodes within a network.

K-shell decomposition methodology is employed to analyze structural characteristics of a network. This analysis enables the identification of the most central nodes within a network based on their interconnectedness. The K-shell decomposition method assigns significance to nodes contingent upon their placement within the network hierarchy, which in turn reflects the depth of their integration. This metric provides a suitable foundation for initializing label allocation from central nodes toward peripheral nodes. Furthermore, augmenting this method with the degree centrality of nodes enhances the discriminatory capabilities of the analysis. Notwithstanding the significance of K-shell values in selecting suitable nodes for label assignment, certain nodes with elevated K-shell scores may still exhibit limited ability to effectively propagate labels. Additionally, situations may arise where nodes with dissimilar roles, influence, or importance within the network share the same improved K-shell score due to possessing identical degree and shell scores. However, nodes with equivalent degree or shell numbers do not inherently possess equal importance. To illustrate this concept, consider a scenario involving two nodes, A and C in Fig. 3, each with the same degree and situated within the same shell 3. According to Eq. (2) while their improved K-shell scores would be identical due to their shared degree, their actual roles and capabilities may differ.

To effectively differentiate between such nodes, a supplementary node assessment layer is proposed that examines both the node's degree and the degree distribution pattern of its neighboring nodes. This is accomplished through the improved H-index metric. The H-index serves as a complementary metric to improved K-shell. Within the context of networks, H-index gauges the degree of a node while simultaneously considering the quality of those connections. It encapsulates both the breadth (connection count) and depth (connection quality) of a node's influence. It examines the pattern of neighbor degree distribution, assessing whether a node is surrounded by significant nodes with extensive connectivity to others. By combining these two metrics, we are able to capture the importance of nodes both in terms of their structural position (as determined by the K-shell) and degree of node and the model of degree distribution of the neighbors around a node (as measured by the H-index).

K-shell is responsible for distinguishing core and peripheral nodes based on their location within the graph whereas H-index supplements K-shell by differentiating nodes with equivalent K-shell scores, adding an extra layer of node evaluation that considers the distribution and concentration of important nodes around a given node. A node with a high K-shell value indicates its presence in a densely connected region, while a high H-index signifies its adjacency to important nodes with extensive connections. This ensures effective label propagation due to the favorable degree distribution patterns of neighboring nodes. By combining improved K-shell and H-index, nodes with identical improved K-shell scores acquire distinct importance levels based on

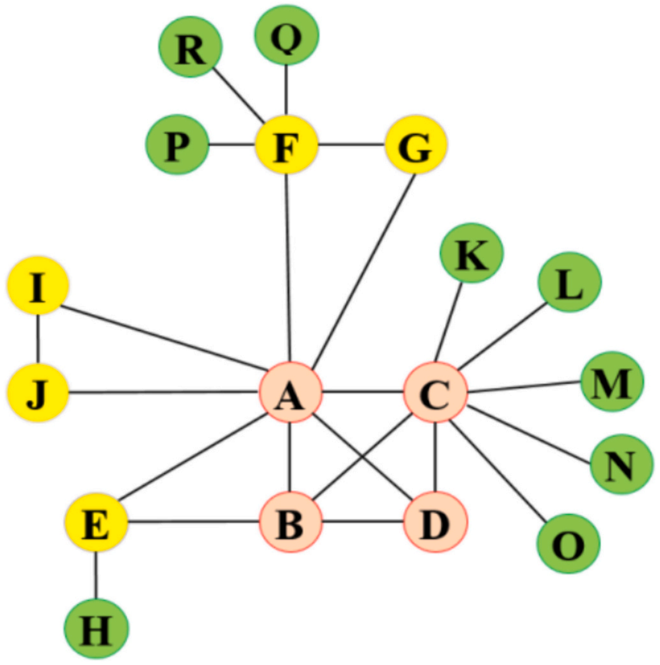


Fig. 3. A sample graph to show complementary role of H-index and K-shell methods. Nodes colored with green, yellow, and pink belong to shell 1, 2, and 3 respectively.

their neighboring nodes and the neighbors' degrees making them suitable starting points for label assignment. In conjunction with the total edge weight and common neighbors shared between the target node and its neighbors, these metrics assess nodes from multiple perspectives.

- Improved K-Shell Score:

K-shell decomposition is a method used to assign importance to nodes based on the position of node within the network hierarchy. [45]. The denser the location of a node, the higher its shell score is. A K-shell within graph G is a maximal connected subgraph where every vertex has a degree of at least k. The K-shell value of a specific node i, denoted as (i), signifies that i belongs to a K-shell but not to any (k + 1)-shell. The K-shell decomposition method is commonly employed to discern core and periphery nodes. As discussed before, K-shell score is not able to fully distinguish the importance of nodes and can assign same score to nodes which in fact have different roles. Since degree centrality represents how well-connected a node is in the network, it would be an appropriate choice to combine degree centrality [46] with K-shell score, so that the drawback of the original K-shell can be improved. Degree centrality and improved K-shell score of a node is defined in Eq. (1) and Eq. (2), respectively.

$$DC(i) = \frac{degree(i)}{n-1} \tag{1}$$

Where $DC(i)$ shows the degree centrality of node i, and $degree(i)$ and n represent the degree of node i and the total number of nodes in the graph, respectively.

$$Ks_{improved}(i) = Ks(i) + (Ks(i) \times DC(i)) \tag{2}$$

Where $Ks(i)$ shows K-shell score of node i, which is multiplied to degree centrality to give a more comprehensive evaluation of node and to improve its score proportional to the degree centrality of the node. The K-shell score of a node is an interesting measure that lies somewhere in between local and global methods. It is partially local because it considers the immediate neighborhood of a node, and it is global since the K-shell represents a subgraph of the entire network. Combining K-shell with degree centrality, benefits from both advantages of them. According to Fig. 1, the score of nodes P and U based on their original K-shell score is equal to 1. By adopting Eq. (2), the final score of nodes P and U is equal to 1.28 and 1.04, respectively.

$$Dominance(i) = H-index_{improved}(i) + Ks_{improved}(i) + \left[\frac{\sum_{j=1}^{|N_i|} \left(|CN_{i,j}| + W_{e_{ij}} + 2 \right) + (Ks_{improved}(i) + degree(i))}{|N_j| + 1} \right] \tag{5}$$

- Improved H-index score:

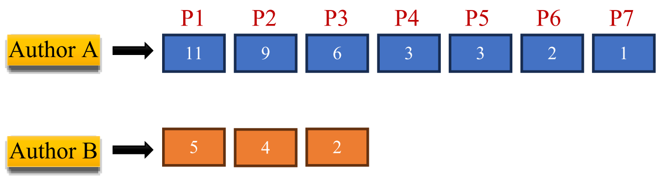


Fig. 2. An example of computing H-index for two authors. Author A, with H-index 3, has 7 published papers, and Author B, with H-index 3, has 3 published papers.

The H-index is an author-centric metric employed to evaluate both research output and citation influence [47]. It represents the maximum value of h such that an author has published at least h papers, each cited at least h times. Despite its widespread use, the H-index has limitations. For instance, consider Fig. 2 that two authors, A and B, with the same H-index of 3. However, Author A has written 7 articles, while Author B has only 3. To address this, an improved version of H-index is proposed that can be calculated via Eq. (3) which is able to better distinguish nodes according to the degree of neighbors. This equation is used to calculate the score of each cell to decide whether we should select the next index in the array or terminate the traversing of the array.

$$Cell_{Score(k)} = [V\_k] + \left([V\_k] \times \left[\frac{|V|}{100} + DC(i) + \frac{k}{|V|} \right] \right) \tag{3}$$

Where $Cell_{Score(k)}$ represents the score of the k-th index in the sorted vector V, $[V\_k]$ shows the value of the k-th cell, $|V|$ is the length of the vector, and k indicates the index of the cell in vector V. If the $Cell_{Score(k)}$ of the cell is greater than the k, then the next cell is selected to be checked. Upon closer examination, the index numbers of primary cells in a vector start low. However, as the index increases, meeting the criteria becomes more challenging due to higher article citation requirements. To address this, $\frac{k}{|V|}$ is added to assist with citation counts in long vectors. The improved H-index is calculated via Eq. (4).

$$H-index_{improved}(i) = k - 1 \tag{4}$$

Where, k is the index of cell that the condition is not satisfied.

- Node Dominance:

The concept of dominance power of a node i is defined as the influence of that node on its direct and indirect neighbors, as well as the ability of a node to access the neighbors of its neighbors. This reflects that the dominant node has more common neighbors with its neighbors and is regarded as a popular and influential node which can better assign its label to the nodes around it. The dominance power of a node is defined as Eq. (5).

Where N_i and N_j are the neighboring list of the nodes i and j, $|CN_{i,j}|$ and $W_{e_{ij}}$ demonstrate the common neighbors between nodes i and j, and the weight of the edge connecting these two nodes respectively, and they are added with 2, to participate the nodes i and j themselves when calculating their common neighbors. The term $(|CN_{i,j}| + W_{e_{ij}} + 2)$ considers the information between nodes i and j such as their common neighbors and the strength of the link between two nodes. Furthermore, every time node i is compared with its neighbor, $(Ks_{improved}(i) + degree(i))$ is considered to better distinguish the final score of nodes and to participate in the influence of node i.

To make it more understandable, consider the sample graph illustrated in Fig. 3, comprising 18 nodes. Two specific nodes, designated as A and C, exhibit identical degree centrality values of 0.44 and an improved K-shell score of 4.32. To demonstrate the efficacy of the improved H-index in distinguishing between nodes with identical K-shell scores, only the H-index and K-shell components of Eq. (5) for nodes A and C are calculated. The sorted array of neighbor degrees for node A is as follows: [2–5,8], corresponding to nodes C, F, B, D, E, G, I, and J, respectively. Given that the first three elements (nodes C, F, and B) possess degree values exceeding their respective positions in the array, we commence the calculation of the H-index from the fourth element, node D, which has a degree of 3. This approach underscores the complementary nature of the H-index and K-shell metrics in evaluating the significance of nodes within a network.

$$\mathrm{Cell}_{\mathrm{Score(D)}} = 3 + \left(3 \times \left[\frac{8}{100} + 0.44 + \frac{4}{8}\right]\right) = 6.06$$

$$\mathrm{Cell}_{\mathrm{Score(E)}} = 3 + \left(3 \times \left[\frac{8}{100} + 0.44 + \frac{5}{8}\right]\right) = 6.43$$

$$\mathrm{Cell}_{\mathrm{Score(G)}} = 2 + \left(2 \times \left[\frac{8}{100} + 0.44 + \frac{6}{8}\right]\right) = 4.54$$

So, the improved H-index of node A is equal to 5. For node C, the sorted array of neighbor degrees is as follows: [1,3,4,8], corresponding to nodes A, B, D, K, L, M, N, and O, respectively. Similar to node A, since the first three elements (nodes A, B, and D), obviously have degree values greater than the index of their position, the H-index is calculated from the fourth cell, node K with degree 1.

$$\mathrm{Cell}_{\mathrm{Score(K)}} = 1 + \left(1 \times \left[\frac{8}{100} + 0.44 + \frac{4}{8}\right]\right) = 2.02$$

Since 2.02 is not greater than the position index 4, so the H-index of node C will be 3. The calculation of the improved H-index for nodes A and C clearly demonstrates its complementary role to the K-shell metric in evaluating the importance of nodes within a network. Despite possessing identical improved K-shell scores of 4.32, nodes A and C exhibit distinct H-index values: 5 for node A and 3 for node C. This differentiation highlights the ability of the improved H-index to capture nuances in node influence that may not be evident from the K-shell score alone. The sum of the improved K-shell score and H-index score further underscores this distinction, with node A receiving a higher combined score (9.32) compared to node C (7.32). As illustrated in Fig. 3, node A indeed exhibits greater potential for propagating labels due to its connections with influential neighbors. This observation further supports the complementary role of the H-index in assessing node importance and enhancing the accuracy of label propagation algorithms.

- Total importance:

In real life, in addition to the personal importance of a person, the importance of a person also comes from its friends [48]. For this end, the total importance of a node consists of personal dominance and neighbors' importance which can be calculated via Eq. (6). So, there is a high possibility for a person with high dominance score to affect others with less importance. In this relation, neighbors with degree 1 are not considered, since they do not have any impact on any of the nodes.

$$Importance_{Total}(i) = Dominance(i) + \sum_{j=1}^{|N_i|} \frac{Dominance(j)}{Dominance_{average}} \times |CN_{i,j}| \quad (6)$$

Where $Importance_{Total}(i)$ indicates the total importance of node i based on personal importance and neighbors' importance of node i. $Dominance(i)$ and $Dominance(j)$ respectively represent the dominance of nodes i and j. $|CN_{i,j}|$ demonstrates the number of common neighbors between two nodes i and j. $Dominance_{average}$ is the average of dominance of neighbors of node i. Unlike the other methos which adopt a threshold as a coefficient to regulate the participation degree of neighbors, in this method the degree of involvement of each neighbor is depended on the portion of importance it takes among the neighbors. By adopting this technique only neighbors which share common neighbors with node i, will have influence on it. After calculating Eq. (6), nodes are sorted descending which defines the order of the node selection in the next steps.

3.1.2. Node attraction step

A prevalent challenge encountered in community detection is the "cold start" problem, characterized by an initial lack of knowledge regarding the underlying community structure. In this scenario, algorithms commence with each node assigned to its own separate community, devoid of any prior information. This necessitates a greater computational effort to identify true community labels, thereby hindering convergence. To mitigate this issue, an initial step is recommended that aims to establish the fundamental community structure based on principles observed in real-world networks. This approach leverages natural patterns of community formation to provide a more informed starting point for the community detection process. This step ensures that no new labels are generated in the next steps, which preserves the current state of the communities and avoids the creation of erroneous labels. In the following steps, the algorithm expands the communities by selecting the most suitable label for each node. Initially, each node identifies its most closely connected neighbor by considering both the number of shared neighbors and the edge weight between them, calculated using Eq. (7).

$$MSN(i) = \mathrm{argmax}_{j \in N_i} |CN_{i,j}| + W_{e_{ij}} \quad (7)$$

Where $MSN(i)$ represents the most similar neighbor of node i, N_i shows the list of neighbors of node i. $|CN_{i,j}|$ demonstrates number of shared neighbors and $W_{e_{ij}}$ shows the edge weight between nodes i and j, respectively. A neighbor node with the highest obtained score from Eq. (7) is selected as the most similar or most intimate neighbor of node i. Nodes are then grouped with their most similar direct neighbor without considering the position of nodes and without any defined order.

The label assignment process operates by comparing the total importance of node i with its most similar neighbor, denoted as M. If the total importance of node i is lower than that of node M, node i is assigned the label of node M. Otherwise the algorithm proceeds to identify the most similar neighbor of node M, designated as W. In the scenario where nodes W and i are identical, the label of node i is assigned to both node M and node i. If nodes W and i differ, the algorithm evaluates two conditions: If node i exhibits a higher total importance than node i, nodes i, M, and W are assigned the label of node i. Conversely, if node W possesses a greater total importance, the label of node W is assigned to all three nodes. Fig. 4 illustrates two consecutive operation of the node attraction step, in which each node chooses the label of its most similar neighbor and then updates its own label accordingly. The weights of the edges between adjacent nodes are indicated on the edges.

In Fig. 4(a), nodes 1, 2, 3, 8, and 9 are assigned with the label of their most similar neighbor, node 4. Initially, the label of node 1 is assigned to node 5, because it has one common neighbor with node 1 and one with node 4, but the edge weight between nodes 1 and 5 is higher than the

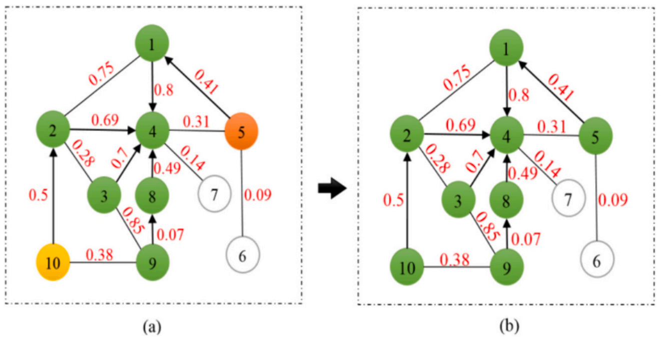


Fig. 4. The procedure of node attraction step. (a) Each node selects the label of its most similar neighbor. (b) Updating the allocation of labels.

edge weight between node 4 and node 5. Similarly, the label of node 2 is assigned to node 10, because it has a higher edge weight with node 2 than with node 4. Initially, nodes 5 and 10 are depicted with distinct label colors. Following the adjustment of the label allocation order, as illustrated in Fig. 4(b), the nodes are assigned with their correct labels. Since the labels of nodes 1 and 2 are changed to 4, nodes 5 and 10 also update their labels to 4. At the end of the algorithm, node 7 receives the label of node 4 and node 6 receives the label of node 5.

3.1.3. Fast label selection step

Community formation is heavily influenced by the sequence of nodes. The previous step facilitates the fast label selection step, as it avoids the cold start state and enables the nodes to select the appropriate label with more confidence. The suggested approach is neither purely based on label diffusion nor purely based on label propagation. It is not in label diffusion-based methods since no labels are diffused, and it is not label propagation-based, as no label is propagated. In this step, based on the previous knowledge, nodes only select the label of the existing communities around them. This step is performed for nodes with *is_dense_flag* equal to zero. After executing this step once, *is_dense_flag* is changed to 1 for a portion of nodes which are located in dense parts. By adopting this technique, a significant time is saved by avoiding of checking dense nodes which it is unlikely that their label will change. This step operates based on the order of the list of nodes sorted by their total importance.

The label selection process depends on the communities around the target node. If the target node has only one distinct community in its neighborhood, this implies that the node is an important node which is located in a dense region, or it is a node with low degree such as 2, 3, 4, and etc. that is completely surrendered. In this case, the target node is excluded and maintained to update its label in the final label selection step. In both scenarios, fully surrendered nodes update their label only one time based on the label of the most similar neighbor. It is worth noting that executing this step with a maximum of two iterations is sufficient. There may be a situation where a node may belong to multiple communities due to its position at the boundary of different groups or due to the erroneous partitioning of a single community into several subgroups, resulting in a dense node being influenced by multiple communities. Since the previous steps have successfully diffused the labels of the influential nodes throughout the network, the label frequency is used to determine the community membership of the node. If the 90 % of the most frequent label has a greater value than the second most frequent label, the label of the community with maximum frequency is allocated to the selected node and its *is_dense_flag* is set to 1. If none of the aforementioned conditions are satisfied, the final condition, label influence, is evaluated via Eq. (10). For each of the surrounding communities of the target node, the sum of the importance values of the nodes in that community (Eq. (8)) and the weighted similarity (Eq. (9)) between them and the target node are computed:

$$Importance_{(i,C_x)} = \sum_{j\in N_i, j\in C_x}^{|N_i|} Importance_{Total}(j) \tag{8}$$

$$WeightedSimilarity_{(i,C_x)} = \sum_{j\in N_i, j\in C_x}^{|N_i|} \frac{|N_i \cap N_j| \times W_{e_{ij}}}{1 + |N_s - N_h|} \tag{9}$$

$$CommunityInfluence_{C_x} = \left(Importance_{(i,C_x)} \times Freq_{C_x} \times Importance_{Total}(i)\right) + WeightedSimilarity_{(i,C_x)} \tag{10}$$

Where nodes i and node j are two nodes connected to each other, N_i and N_j representing their respective sets of neighboring nodes. $Importance_{(i,C_x)}$ indicates the influence of the community C_x on node i and the total importance for neighbor node j in community C_x is indicated by $Importance_{Total}(j)$. In Eq. (9), $WeightedSimilarity_{(i,C_x)}$ denotes the importance of the community C_x based on similarity score of two neighbor nodes i and j and the weight (strength) of the links connecting them to node i. $|N_i \cap N_j|$ and $W_{e_{ij}}$ show the number of shared nodes and the edge weight between node i and node j, respectively. For node i and j, if $|N_i| \geq |N_j|$, N_s refers to N_j and N_h refers to N_i. However, if $|N_i| < |N_j|$, N_s is set with N_i and N_h is set with N_j. To the best of our knowledge, no one has used such diverse combination of metrics to evaluate the score of the communities around a node from different aspects. According to Eq. (10), the influence of a community is evaluated based on the importance of nodes, as well as the frequency of label of the community. Besides these metrics, the importance of the origin of the label is added. To the best of our knowledge, no previous work has used the importance of the origin of the label. After applying Eq. (10) to all of the communities around node i, finally the community with the maximum score allocates its label to node i.

3.1.4. Final label selection

Although the previous steps adopt various metrics to maximize the accuracy of the algorithm, some wrong label assignments are inevitable. The final label selection phase is a fast process that ensures that each node receives the most suitable label. Since the previous steps have already assigned appropriate labels to the nodes, there is no need to adopt complex computations in this step. Consequently, any node with a degree exceeding one will adjust its label to correspond with the label of the neighbor possessing the largest quantity of mutually connected nodes among the neighbors. If multiple neighbors have the same number of similarities, then the first one is selected.

3.1.5. Fast merge step

Generally, the purpose of merge steps in community detection is to combine weak communities that are likely to be part of a strong community but have been erroneously separated by the algorithm. A novel and efficient approach is proposed to merge small communities in a fast manner. In contrast to methods that rely on the time-consuming modularity criterion, which neglects the relationship and similarity between communities, the method suggested for merging communities streamlines the process by focusing solely on negotiations between key community members, eliminating unnecessary steps. The merging steps are motivated by the real-world phenomenon of two governments, nations, or empires that seek to merge with each other. They usually attempt to be neighbors first, and then form a larger group by integrating into one entity. Moreover, instead of the rulers communicating directly, they often send an ambassador or a representative to negotiate on their behalf. The ambassador with the highest score and the ruler with the most influence assimilates the smaller government with themselves. When two ambassadors of two nations agree to merge, it implies that the other members are also in agreement. Based on this observation, the community merging step is designed to merge communities at once, which avoids the need for the algorithm to check each node individually. The suggested merging method proceeds as follows:

Communities with size less than the average size of the communities are supposed as small communities. The origin of the label of a community is considered as the ruler of that community. Each ruler then choses an ambassador among its neighbors based on three metrics which are adopted via Eq. (11). The neighbor with highest obtained score from Eq. (12), is selected as the ambassador of that nation or group.

$$NAS(j) = \sum_{k=1}^{|N_j|} |CN_{j,k}| + \sum_{k=1}^{|N_j|} W_{e_{jk}} + degree(j) \tag{11}$$

$$NA(C_i) = \mathrm{argmax}_{j\in N_i} NAS(j) \tag{12}$$

Where $NAS(j)$ shows the nation ambassador score of node j, $|N_j|$ represents the number of neighbors of node j, $|CN_{j,k}|$ and $W_{e_{jk}}$ demonstrate the number of common neighbors and the weight of the edge between nodes j and k, respectively. Finally, $degree(j)$ is used to show the degree of node j. In Eq. (12), $NA(C_i)$ indicates the ambassador of nation C_i. As it is apparent from Eq. (11), an efficient combination of metrics is adopted to select a node which is superior from diverse aspects. An ambassador should have high number of common interests with its neighbors and the strength of this ties should be high. Also, this node should have more connections than other nodes, which all of them are gathered in Eq. (11). Then, a neighbor node k of ambassador node j with different label and highest total importance is selected. If the total importance of the ruler of the selected community is greater than the ruler of the target community C_i, and they share common nodes with each other, then two communities have the sufficient conditions to be merged with each other. Despite of its simplicity, the merge method is significantly capable and fast.

3.1.6. Label assignment to degree 1 nodes

After finishing the execution of the aforementioned steps, finally, for all nodes with a degree of 1, the algorithm chooses the label of the single node adjacent to them. The exclusion of nodes with degree 1 during the entire algorithm execution does not affect the accuracy, but rather significantly decreases the runtime and avoids the need for checking them in each iteration.

3.2. Computational complexity analysis of LMFLS algorithm

LMFLS is structured around six main key steps: calculating node importance, node attraction step, fast label selection, final label selection, merge step, and finally label assignment step for nodes with degree 1. At the beginning of the algorithm, only K-shell decomposition is executed for all nodes. The computational cost of the initial iteration of the K-shell decomposition algorithm is expressed as $O(n+m)$, with n denoting the number of nodes and m representing the number of edges in the graph. Following the removal of n_1 nodes with a degree of 1, the network comprises $n' = n - n_1$ nodes, which continue to be involved in the subsequent phases of the algorithm. Computing degree centrality of nodes is performed in $O(n')$. The execution of the improved H-index on the n' nodes, characterized by an average degree of k, exhibits a time complexity of $O(n' \times klogk)$. The similarity of n' nodes is computed in $O\left(\frac{n'k^2}{2}\right)$. Also, the computational cost of calculating total node importance is equal to $O(n'k)$. Sorting n' nodes requires $O(n'logn')$ time complexity. The node attraction step, involving the selection of the most similar neighbor for each of the n' nodes, exhibits a time complexity of $O(n'k)$. The execution of the fast label selection step with one iteration for n' is completed in $O(n'k)$. The final label selection step is executed with $O(n'k)$ time complexity. The time required to determine the number of members of communities within the merge step is $O(n')$. Representative nodes for small communities with n_s members are identified in $O(sn_s) < O(n')$. The merge step exhibits an overall time complexity of $O(n' + sn_s)$, which is asymptotically equivalent to $O(n')$. This is the reason why the proposed merge step is fast and efficient. Labeling all nodes with a degree of 1 is completed with a time complexity of $O(n_1)$. Therefore, the LMFLS algorithm's overall time complexity is $O(n+m)$.

Table 1
The data structure adopted for implementation of the LMFLS algorithm. The keys indicate ID of nods and the value for each key is a list containing ID of neighbors and the edge weight between neighbor and the key node.

Key	Value
1	List (11: [1], 15: [1], 16: [1], 41: [1], 43: [1], 48: [1])
2	List (18: [1], 20: [1], 27: [1], …,37: [1], 42: [1], 55: [1])
3	List (11: [1], 43: [1], 45: [1], 62: [1])
…	…
62	List (3: [1], 38: [1], 54: [1])

Table 2
A sample of order of nodes in Dolphins dataset and their obtained score. *DC* stands for degree centrality.

Node ID	DC	K-shell	H-index	Dominance	Total importance
15	0.196	7.18	10	172.03	221.32
52	0.163	6.98	6	183.31	215.32
46	0.180	7.08	10	142.23	182.73
38	0.180	7.08	10	137.84	175.04
34	0.163	6.98	9	134.68	174.29
…	…	…	…	…	…
56	0.032	2.06	2	5.60	5.60

3.3. Example

To enhance the understanding of the LMFLS algorithm, a detailed example based on the Bottlenose Dolphin community in Doubtful Sound [50] is provided. The Dolphins network is made up of 62 interconnected individuals, linked by 159 relationships, forming two distinct groups. Table 1, shows an example of the data structure adopted for implementing the LMFLS algorithm. Since the Dolphins dataset is unweighted, each edge between two adjacent nodes is initially assigned with weight 1.

According to the algorithm, after computing K-shell step, nodes with degree 1 are excluded. Table 2, shows the list of nodes and their score obtained from calculating degree centrality, improved K-shell, improved H-index, node dominance, and finally total node importance which are calculated from Eqs. (1), (2), (4), (5), and (6) repetitively.

In the next step, each node selects the most similar neighbor according to Eq. (7). The resulting pairs of nodes are presented in Table 3. *MSN* and *NMSN* stand for the most similar neighbor of the target node and the most similar neighbor of the most similar neighbor of the target node, respectively.

According to Table 3, since $Importance_{Total}(15) \geq Importance_{Total}(34)$ and *NMSN* of node 34 is node 15 itself, both of the nodes are assigned with the label of node 15. As another example, node 2 is selected as the target node. Node 55 is the most similar neighbor of node 2. Since $Importance_{Total}(2) \geq Importance_{Total}(55)$, then the most similar neighbor of node 55 should be checked. Because $Importance_{Total}(58) \geq Importance_{Total}(2)$, then *Label (2) ← 58, Label (55) ← 58,* and *Label (58) ← 58*. The updated network is shown in Fig. 5. According to Fig. 5, after the execution of the node attraction step, 6 communities are formed. The green community is detected with a high accuracy. It is worth noting that communities labeled as "Red", "Purple", "Blue", "Dark Blue", and "Orange" are, in fact, one community. However, due to their dense connections with nodes around them, they are initially divided into several communities and the next steps of the algorithm will correct them. At the next, the label selection step is performed. Table 4 highlights nodes situated in areas of high connectivity within the graph based on label assignment of previous step and *is_dense_flag* of these nodes is equal to 1. The Dominant label column displays the label of the community that is the most frequently observed around the target node. The origin of each community is shown along with the color of the community. The origin of each community is shown

Table 3
An example of selected target nodes and their most similar neighbor along with the most similar neighbor of the most similar neighbor of the target node.

Target node	MSN	NMSN
15	34	15
52	46	52
46	52	46
38	15	34
34	15	34
…	…	…
2	55	58
56	52	46

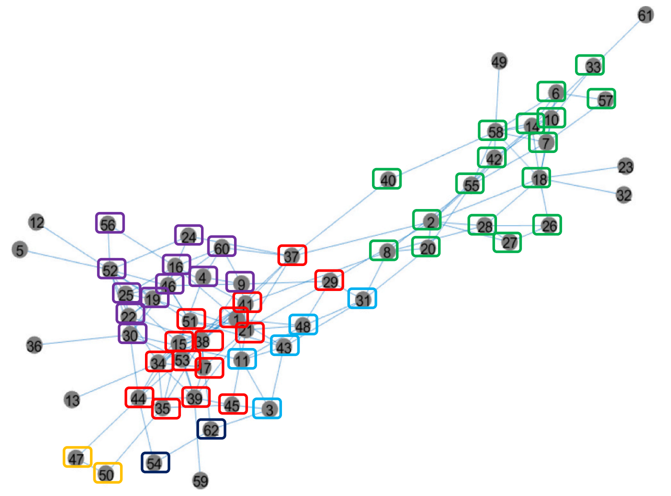


Fig. 5. Updated state of the Dolphins network after performing node attraction step.

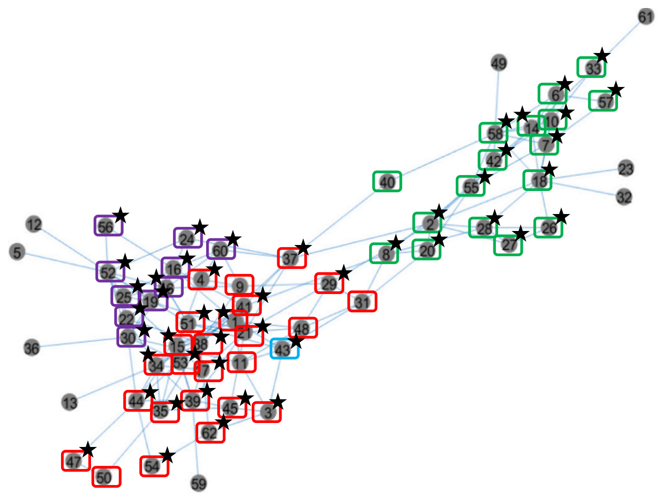


Fig. 6. Performing 1 iteration of fast label selection step. Nodes such as 31, 9, 48, 11, 3, etc. have selected the label of red community and only node 43 is remained isolated. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
A sample list of nodes which are located in dense regions in Dolphins network. The first number in each of the parenthesis indicates the origin of that community and the second number shows the frequency of that community around the target node.

Node ID	Dominant label
58	🟩58
18	🟩58
39	🟥15
14	🟩58
45	🟥15
15	(15,10), (52,2) → 🟥15
52	(52,7), (15,1) → 🟪52
46	(52,9), (15,2) → 🟪52
38	(15,7), (52,3) → 🟥15
34	(15,8), (52,1) → 🟥15

along with the color of the community.

As it is listed in the Table 4, nodes 58, 18, 39, etc. are nodes which are surrendered with one type of label, and nodes such as 15, 52, 46, 38, 34, and others are associated with more than one type of community.

Fig. 6 presents the new updated state of the network. By performing one iteration of the label selection part of the algorithm, nodes such as 31, 9, 48, 11, 3, 4, 62, 50, 47, and 54 selected a new community (Red community). Only node 43, despite changing its *is_dense_flag* to 1, has not properly updated its label. The black-colored star demonstrates the nodes with *is_dense_flag* equal to one. According to the results, only nodes 48, 9, 31, 11, 50, and 40 are not flagged as dense. If the fast label selection step is executed with two iterations, only nodes 31 and 40 would not be flagged as not dense. This indicates that nodes 31 and 40 are precisely in boundary sections. Table 5 shows the procedure of label selection for nodes 48, 9, 31, 11, 50, 40. Community influence is calculated via Eq. (10). The equation is designed optimally to check communities from different aspects, as there are considerable differences between scores. As a result, there is no need to perform a tie-break step. The final label selection is executed to correct some slight mistakes and to join isolated communities to their neighbor communities. Only node 43 is updated in this step, since other nodes are correctly assigned with the label of the communities around them. Table 6 shows an example of the procedure for assigning the label of the similar and closest neighbor. The updated network is visualized in Fig. 7.

Table 5
Selecting the most appropriate label for nodes which are not flagged as dense.

Node ID	Community influence	Selected community
48	(15, 118159), (43, 20671)	15
9	(52, 145568), (15, 214122)	15
31	(58, 16982), (15, 41248), (43,3152)	15
11	(15, 38255), (43, 8342), (52,31489)	15
50	(15, 5990), (50, 55)	15
40	(14, 9634), (58, 21219)	58

Table 6
An example of the procedure of selecting the label of most similar neighbor. The first number in each of the parenthesis indicates the neighbor of target node and the second value shows the number of common neighbors between target node and its neighbor.

Node	(Neighbor Id, Number of common neighbors)	Selected label
43	List [(1,2), (3,1), (11,3), (31,1), (48,3), (51,0)]	Label(11) → 15
31	List [(8,1), (20,1), (29,1), (43,1), (48,2)]	Label(48) → 15
29	List [(2,0), (9,1), (21,2), (31,1), (48,2)]	Label(21) → 15
9	List [(4,1), (21,1), (29,1), (38,1), (46,2), (60,2)]	Label(46) → 52
…	…	…

According to the Table 6, node 43 has the highest common neighbors with node 11 and the label of node 11 which is 15 (Red community), is assigned to node 43. According to the results, the Red labeled, Purple labeled, and Green labeled communities have 24, 12, and 17 members, respectively. The mean community size before the merge operation is 17.66. Since the Green and Purple communities have a size <17.66, they are regarded as small communities. Table 7 shows the information of small communities such as the ruler and the ambassador of the community, and the community with which the small community is merged. Based on the information presented in Table 7, the ruler of the Purple community is node 52 and node 46 is the ambassador of this community. Among its neighbors, node 38 with the highest total node importance of 175, is selected.

Since node 38 has a different label than node 46, and since the total importance of the ruler of the Purple community is less than the total importance of the ruler of the Red community, and nodes 52 and 15 share common neighbors 25 and 51, then both Purple and Red

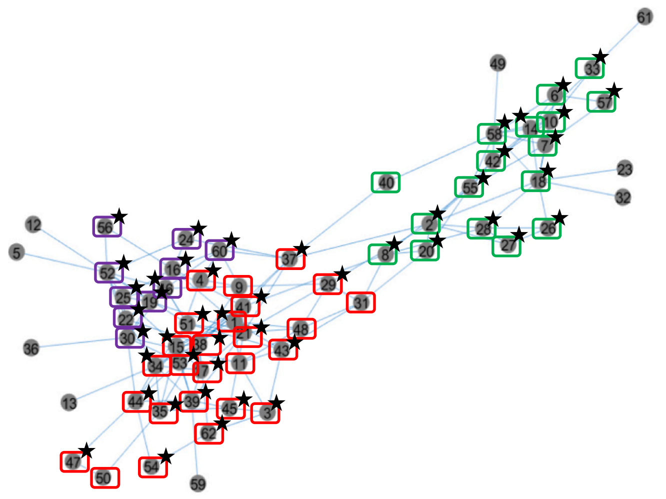


Fig. 7. The updated network after performing final label selections step on the network presented in Fig. 7.

Table 7
Information about the selected small communities. Each community has a numerical identifier which is regarded as the origin of that community which has the highest importance. SNA indicates the selected neighbor of the ambassador.

Community ID	Ruler	Ambassador	SNA	Merged
🟪Purple (52)	52	46 (score:56)	38	🟥 (🟥Red community)
🟩Green (58)	58	14 (score:44)	18	🟥

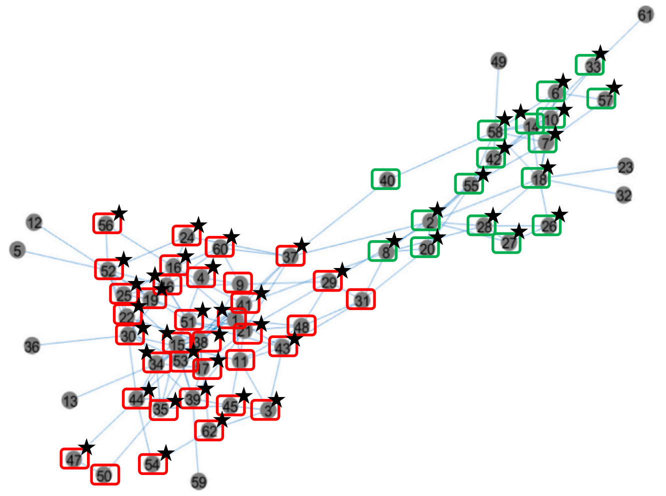


Fig. 8. The updated network after completing the fast merge step. The Red and Purple communities are merged with each other. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

communities are integrated with each other. The Green community, due to the fact that the important neighbor of ambassador is node 18, which has the same label as node 14, does not have the sufficient condition to be merged with other communities. Fig. 8 shows the updated network. Finally, label assignment to nodes with degree 1 is performed. The final state of the network with two communities with detection accuracy 100 % is visualized in Fig. 9.

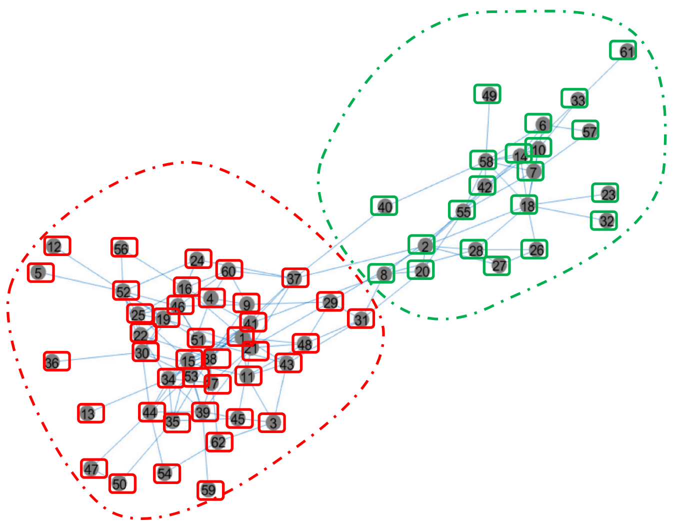


Fig. 9. The final state of Dolphins network with two detected communities with 100 % detection accuracy.

Table 8
Properties of real-world datasets.

Dataset	*N*	*M*	*C*
Zachary's Karate Club	34	78	2
Dolphins	62	159	2
U.S Political Books	105	441	3
Football	115	613	12
Net-science	1589	2742	–
Power Grid	4941	6594	–
CA-GRQC	5242	14,490	–
Collaboration	8361	15,751	–
CA-HEPTH	9877	25,985	–
PGP	10,680	24,316	–
Condmat-2003	31,163	120,029	–
Condmat-2005	40,421	175,691	–
DBLP	317,080	1,049,866	13,477
Amazon	334,863	925,872	75,149
YouTube	1,134,890	2,987,624	8385
Orkut	3,072,441	117,185,083	6,288,363
LiveJournal	3,997,962	34,681,189	287,512

4. Experiments

A thorough evaluation of the proposed LMFLS algorithm is conducted in this section, comparing its performance with other leading and novel algorithms across a diverse range of real-world and synthetic networks of varying sizes. Python version 3.7, is used for implementation of the LMFLS algorithm. The experiments employ Python implementations of several community detection algorithms: Louvain [26], FluidC [38], CNM [28], LPA [13], Infomap [5], and Leiden [25]. We use CFCD2, considered the most accurate version [35], from among four variations of the CFCD algorithm. Results for RTLCD and CDME algorithms are sourced from [19,37], respectively. A comprehensive overview of the algorithms and their original implementations can be found in [51]. The LSMD [24], NIBLPA [14], and LPA-Intimacy [15] are programmed in MATLAB 2017 and python implementation of the GCN algorithm [31] is adopted. The experiments were carried out on a computer system with a Core i5 processor (3.70 GHz), 12 GB of RAM, and using Python version 3.7 on Windows 10.

4.1. Real-world networks

A comparison of algorithm performance was conducted using 17 real-world networks. Table 8 summarizes the details of the real-world datasets, where N denotes the number of nodes, M the number of

Table 9
Parameters of the LFR benchmark.

Parameters	Description
N	Number of nodes
Min K	Minimum degree of nodes
Max K	Maximum degree of nodes
μ	Mixing parameter
Min c	Number of nodes within the smallest community
Max c	Number of nodes within the biggest community
γ	Node degree distribution exponent
β	Community size distribution exponent

Table 10
Properties of the generated LFR networks.

Network	N	*Min K*	*Max K*	*Min C*	*Max C*	γ	β	μ
LFR1	50,000	5	15	10	50	2	1	0.1–0.8
LFR2	100,000	7	20	10	100	2	1	0.1–0.8
LFR3	200,000	5	20	20	100	2	1	0.1–0.8
LFR4	500,000	10	25	20	200	2	1	0.1–0.8

edges, and C the number of communities. A comprehensive description of the real-world datasets is accessible through the SNAP project [52].

4.2. Synthetic networks

The LFR[1] benchmark [53] is employed as a synthetic dataset generator, utilizing a range of parameters to create networks with specific characteristics. The mixing parameter (μ) significantly influences the interconnectedness of nodes within the model. As its value approaches 1, the network becomes increasingly densely connected, making it considerably harder to distinguish distinct communities. The LFR benchmark's configuration is detailed in Table 9, while Table 10 presents the characteristics of the generated datasets.

4.3. Evaluation metrics

This paper employs three widely used metrics, namely Normalized Mutual Information (NMI), F-measure, and Modularity (Q), to assess the detection accuracy of examine methods. When ground-truth of networks is accessible, the NMI metric serves as a widely accepted standard for measuring the precision of identified communities [54]. The accuracy of the detected communities is evaluated by comparing them to the real communities, thus assessing the degree of similarity between the two sets. The NMI score can be adopted via Eq. (13).

$$NMI(X,Y) = \frac{-2\sum_{i=1}^{C_X}\sum_{j=1}^{C_Y} C_{ij} log\left(\frac{C_{ij}N}{C_iC_j}\right)}{\sum_{i=1}^{C_X} C_i log\left(\frac{C_i}{N}\right) + \sum_{j=1}^{C_Y} C_j log\left(\frac{C_j}{N}\right)} \tag{13}$$

Where X demonstrates the real communities and Y is a group of nodes identified by the algorithm. C_X represents the number of communities in the ground-truth and C_Y is the number of discovered communities and N is the number of nodes. C denotes the confusion matrix where the rows are the real communities and columns are the detected communities. C_{ij} represents the number of common nodes between the real community i in set X with the detected community j in set Y. C_i shows the sum of the row i in the matrix C_{ij} and C_j shows the sum of the column j in the matrix C_{ij}.

The F-measure, a widely employed metric in evaluating ground-truth datasets, offers a comprehensive assessment of accuracy by harmonizing precision and recall [35]. In the context of community detection, the F-measure is calculated according to Eq. (14). A value closer to 1 indicates a higher degree of accuracy, signifying the method's effectiveness in identifying communities with precision.

$$Precision = \frac{|C_D \cap C_R|}{|C_D|} \tag{14}$$

$$Recall = \frac{|C_D \cap C_R|}{|C_R|} \tag{15}$$

$$F - measure = \frac{2 \times precision \times recall}{precision + recall} \tag{16}$$

Where C_D indicates the detected communities and C_R represents the real communities based on the ground-truth information. However, when ground truth is absent, the modularity measure [28] serves as a prominent metric to gauge the effectiveness of community detection algorithms in identifying dense and well-structured communities which is calculated by Eq. (17).

$$Q = \frac{1}{2m}\sum_{a,b} A_{ab} - \frac{deg_a \times deg_b}{2m} \times \delta(c_a, c_b) \tag{17}$$

Where, a and b represent two distinct nodes within the network, while m shows the total number of connections (edges) present. The adjacency matrix A is used to determine the existence of an edge between nodes a and b, where A_{ab} equals 1 if such a connection exists and 0 otherwise. The degrees of nodes a and b, denoted by deg_a and deg_b, respectively, represent the number of connections each node has. Furthermore, c_a and c_b represent the labels assigned to nodes a and b, respectively. The Kronecker delta, symbolized as $\delta(c_a, c_b)$, is a function that evaluates to 1 if both nodes a and b share the same label and 0 if they have different labels.

4.4. Experimental results on real-world datasets

This empirical study encompasses a comprehensive range of community detection algorithms, encompassing distinct categories such as label propagation, label diffusion, modularity optimization, core node expansion, and random walk-based approaches. The primary objective is to provide a comprehensive comparative analysis across different methods. The NMI results for LPA, FluidC, and Louvain on the LiveJournal dataset are adopted from [37].

Most of the compared algorithms were unable to execute Orkut and LiveJournal datasets due to time-complexity issues and lack of sufficient RAM (due to adopting graph data structure in their implementation). However, among the diverse methods evaluated, only the LMFLS, LBLD, and LSMD algorithms successfully completed execution on these datasets. This success can be attributed to their utilization of a neighboring list structure and their efficient time complexity during execution.

The LMFLS algorithm was executed with a maximum of two iterations for both the fast label selection and the final label selection steps. While these represent the maximum iteration limits, in most instances, the LMFLS algorithm converged and terminated within a single iteration.

Given that FluidC necessitates prior knowledge of the number of communities, the actual number of communities, as presented in Table 8, was utilized as the parameter K for this algorithm. It is important to note that the datasets used in this study do not provide weights for edges. Therefore, the weights are considered to be 1. However, since the drug and disease networks, which are conducted as an application of the LMFLS in the next section, are weighted networks, the LMFLS is designed in such a way to support the weight of edges.

4.4.1. NMI evaluation on real-world datasets

Table 11 presents results obtained for different algorithms evaluated with NMI. Furthermore, to enhance the visibility of the community

[1] Lancichinetti–Fortunato–Radicchi

Table 11
NMI results obtained from real-world ground-truth datasets (highest values on each row are bolded).

Datasets	*CNM*	*Infomap*	*LPA*	*NIBLPA*	*LPA-Intimacy*	*GCN*	*LCDR*	*FluidC*	*CDME*	*CFCD2*	*Louvain*	*Leiden*	*ECES(s = 1)*	*RTLCD*	*LSMD*	*LBLD*	*LMFLS*
Karate	0.69	0.70	0.21	0.8	1	0.83	1	1	1	0.84	0.71	0.687	0.63	1	1	1	1
Dolphins	0.55	0.556	0.53	0.50	0.63	0.54	1	0.89	0.70	0.55	0.435	0.55	0.49	0.45	1	1	1
Polbooks	0.53	0.54	0.44	0.53	0.54	0.53	0.58	0.51	0.58	0.48	0.53	0.57	0.50	0.48	0.59	0.60	0.60
Football	0.74	0.90	0.85	0.72	0.86	0.87	0.90	0.89	0.93	0.83	0.85	0.85	0.75	0.51	0.93	0.91	0.90
DBLP	0.49	0.61	0.68	0.46	0.61	0.68	0.70	0.74	0.75	0.26	0.53	0.54	0.36	0.41	0.70	0.74	0.746
Amazon	0.88	0.42	0.93	0.60	0.63	0.90	0.69	0.90	0.96	0.72	0.83	0.86	0.58	0.72	0.95	0.97	0.97
YouTube	–	0.47	–	–	–	0.23	0.31	0.72	–	0.30	0.48	0.46	–	–	0.19	0.57	0.501
Orkut	–	–	–	–	–	–	–	–	–	–	–	–	–	–	0.27	0.68	0.665
LiveJournal	–	–	0.88	–	–	–	–	0.87	0.93	–	0.75	–	–	–	0.80	0.934	0.94

detection results of the LMFLS algorithm, Fig. 10 presents a visualization of the communities identified in Karate, Dolphins, Polbooks, and Football datasets, achieving NMI scores of 1, 1, 0.60, and 0.90, respectively. As it is implied from Table 11, the proposed LMFLS has obtained the highest NMI score in 5 out of 9 datasets. The LMFLS algorithm, applied to both Karate and Dolphins networks, demonstrated perfect community identification. This is evidenced by an NMI score of 1, indicating an exact match between the algorithm's detected communities and the ground truth. The LMFLS algorithm achieved the third highest accuracy with an NMI of 0.90 in Football network, which is only 0.03 lower than the highest result obtained by the LSMD algorithm. It should be noted that LSMD employs several controlling conditions, such as edge clustering coefficient and nodes' influence on each other when assigning labels. In contrast, LMFLS, despite its simple yet accurate and robust nature, was able to achieve an NMI of 0.90 and very close to the NMI of the LBLD.

The results indicate that the LMFLS algorithm exhibits superior performance in large-scale networks such as DBLP, Amazon, LiveJournal, and Orkut, achieving competitive results compared to other methods like LBLD. Furthermore, achieving NMI score equal to 0.94 and 0.665 in LiveJournal and Orkut datasets, respectively, is an obvious evidence of efficiency of the LMFLS algorithm on large networks. Besides this, the highest NMI scores in Amazon and LiveJournal belongs to the LMFLS. In DBLP dataset, LMFLS has obtained the second-best NMI only with 0.004 difference from the highest result. While the FluidC algorithm exhibits acceptable results, its applicability to real-world network community detection is hindered by its inherent limitations. FluidC struggles to identify communities organically, relying instead on a pre-defined number of communities.

This dependence on a predetermined value, which is often unknown in real-world scenarios, restricts its effectiveness in practical applications. Leiden and Louvain algorithms despite of achieving highest modularity results in most of the datasets listed in Table 13, were not successful in obtaining higher NMI results than the LMFLS. The proposed LMFLS algorithm provides stable results on different executions with the same number of iterations, whereas Louvain, Leiden, LPA, Infomap, GCN, NIBLPA, and so forth can obtain unstable results on different executions. The LBLD algorithm employs robust and accurate techniques to identify rough cores and uses an efficient balanced label diffusion approach, resulting in high accuracy in large datasets such as: Amazon, YouTube, Orkut, and LiveJournal. Despite its simple design compared to the LBLD, the LMFLS is capable of achieving competitive results with the LBLD algorithm in most of the experiments.

4.4.2. *F-measure evaluation on real-world datasets*

To evaluate the accuracy score of each method, the F-measure is computed for each identified community. Subsequently, the average F-measure score is computed for each dataset. Table 12 presents a comprehensive summary of F-measure results, specifically for the algorithms under consideration on ground-truth networks. The LMFLS algorithm was completely successful in identifying communities with 100 % accuracy in Karate and Dolphins datasets, resulting in an F-measure of 1. In the Karate network, after that node attraction step of the LMFLS was executed, an NMI of 1 was obtained, indicating the accuracy of the node attraction step.

It was observed that after performing node attraction and one iteration of the fast label selection steps, a significant proportion of nodes were assigned with their final labels, and the next steps were executed to reorganize and improve the results. Table 11 shows that the CFCD2 algorithm has a lower NMI than the LMFLS and LBLD algorithms (with a difference of 0.09) in the Polbooks dataset. Interestingly, Table 12 reveals a significantly higher F-measure for the CFCD2 algorithm compared to both LMFLS and LBLD algorithms. However, it's important to acknowledge that the implementation of CFCD2 is not publicly available, and the results presented are based on data from the original publication. Consequently, a degree of uncertainty might exist in the

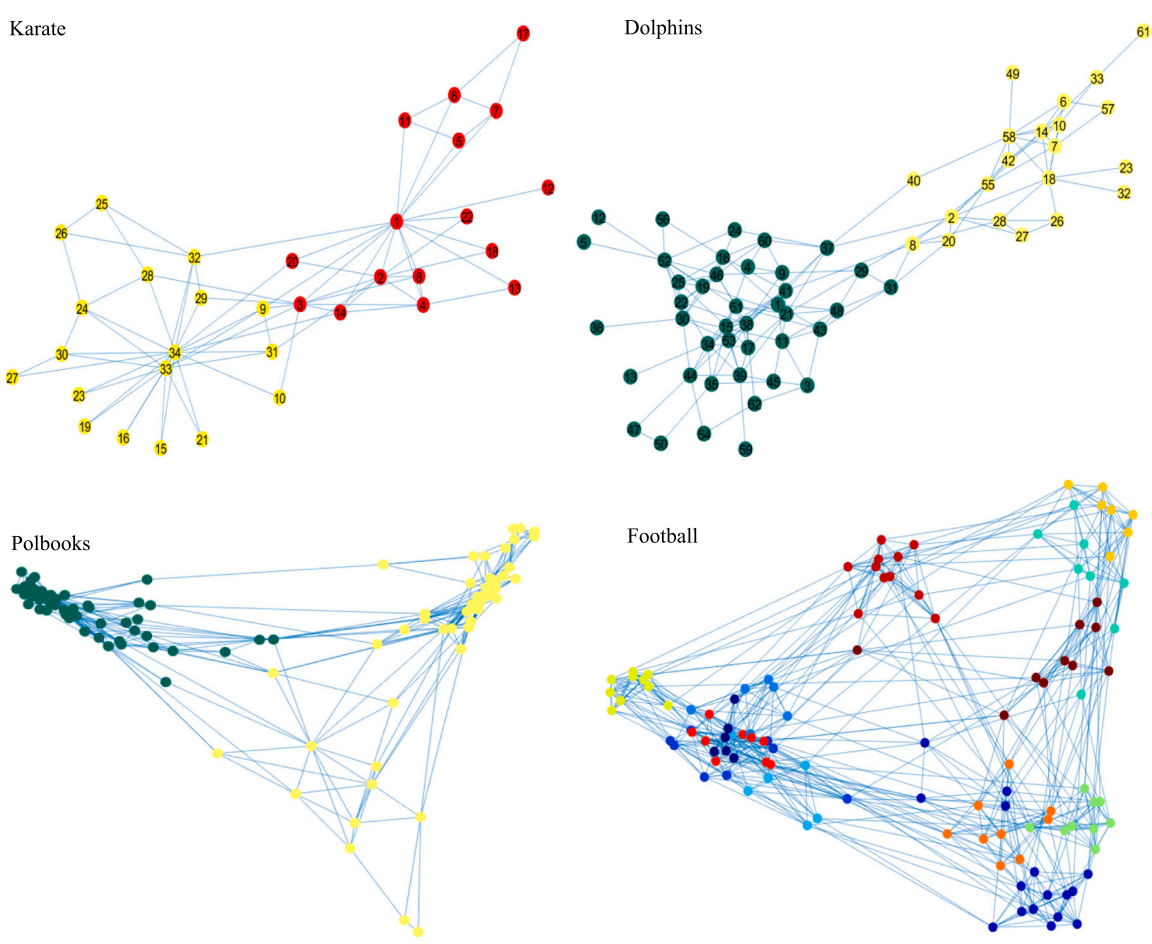


Fig. 10. Visualization of the detected communities by LMFLS algorithm in Karate, Dolphins, Polbooks, and Football networks.

Table 12
F-measure results obtained from real-world ground-truth datasets (highest values on each row are bolded).

Datasets	*CNM*	*Infomap*	*LPA*	*NIBLPA*	*LPA-Intimacy*	*GCN*	*LCDR*	*FluidC*	*CFCD2*	*ECES(s = 1)*	*RTLCD*	*Louvain*	*Leiden*	*LSMD*	*LBLD*	*LMFLS*
Karate	0.81	0.87	0.42	0.39	1	0.95	1	1	0.97	0.55	1	0.84	0.81	1	1	1
Dolphins	0.49	0.62	0.62	0.39	0.43	0.29	1	0.97	0.88	0.39	0.73	0.61	0.64	1	1	1
Polbooks	0.55	0.45	0.48	0.53	0.61	0.38	0.66	0.59	0.76	0.51	0.66	0.54	0.61	0.70	0.68	0.68
Football	0.57	0.85	0.86	0.47	0.68	0.76	0.90	0.82	0.87	0.63	0.60	0.87	0.87	0.91	0.916	0.91
DBLP	0.18	0.19	0.43	0.40	0.45	0.44	0.57	0.48	0.67	0.33	0.37	0.15	0.15	0.49	0.65	0.654
Amazon	0.38	0.01	0.76	0.66	0.71	0.80	0.90	0.72	0.94	0.37	0.74	0.20	0.27	0.84	0.90	0.94
YouTube	–	0.16	–	–	–	0.11	0.21	0.41	0.32	–	–	0.10	0.10	0.13	0.61	0.559
Orkut	–	–	–	–	–	–	–	–	–	–	–	–	–	0.19	0.66	0.67
LiveJournal	–	–	–	–	–	–	–	–	–	–	–	–	–	0.89	0.94	0.94

reported CFCD2 results. This is because a strong F-measure typically corresponds to a high NMI score.

Based on the observed results, it is evident that the LMFLS and LBLD algorithms achieved the highest score of NMI and F-measure within the Polbooks network compared to all other methods investigated. The results of the CFCD2 algorithm in the DBLP dataset reveal a contradiction. CFCD2 obtained an NMI of 0.26, which is considerably weak, whereas LMFLS obtained an NMI of 0.746. This indicates that LMFLS was more successful in discovering partitions that are more similar to the real communities than the CFCD2 algorithm. Despite the weak NMI, CFCD2 obtained the highest F-measure (F-measure = 0.67), which is not entirely reliable, since there is a 0.46 difference between NMI and F-measure. If an algorithm obtains NMI = 0.26, it means that there is a 0.74 dissimilarity between the detected communities and the real communities.

Furthermore, in the Amazon dataset, the CFCD2 method exhibits a noteworthy performance, attaining an NMI score of 0.72 and an F-measure score of 0.94. Therefore, it can be said that the best result of F-measure in the DBLP is achieved by LMFLS and LBLD algorithms. LMFLS is also significantly efficient on large networks used in experiments such as: Amazon, Orkut, and LiveJournal, achieving highest F-measure among the other methods indicating that the detected communities by LMFLS are much closer to the real communities than those detected by other methods. It is noteworthy that, despite achieving acceptable NMI scores in the DBLP, Amazon, and YouTube datasets, the Leiden and Louvain methods exhibit significantly lower F-measure scores in these datasets. The challenges stem from the inherent characteristics of Leiden and Louvain algorithms, which exhibit an excessive inclination toward community amalgamation, compounded by the absence of reliable metrics for discerning meaningful community boundaries. Table 13

Table 13
Experimental results on real-world datasets based on modularity (highest values on each row are bolded).

Dataset	LPA		NIBLPA		LPA-Intimacy		CNM		LCDR		GCN		Infomap		Louvain		Leiden		LSMD		LBLD		LMFLS	
	C	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*	*C*	*Q*
Karate	3	0.11	5	0.4	2	0.3715	3	0.38	2	0.371	2	0.371	3	0.40	4	0.415	4	0.42	2	0.371	2	0.371	2	0.371
Dolphins	7	0.48	5	0.43	4	0.5	4	0.49	2	0.378	5	0.51	5	0.53	6	0.516	5	0.527	2	0.378	2	0.378	2	0.378
Polbooks	8	0.48	4	0.55	3	0.48	4	0.50	3	0.50	6	0.51	5	0.523	5	0.526	4	0.526	3	0.446	2	0.456	2	0.456
Football	9	0.55	8	0.5	10	0.60	6	0.57	9	0.60	12	0.57	11	0.60	9	0.602	9	0.602	12	0.58	13	0.58	13	0.56
Netscience	343	0.90	371	0.87	310	0.88	275	0.95	332	0.92	317	0.88	268	0.93	277	0.96	278	0.96	275	0.94	303	0.94	309	0.935
Power Grid	1406	0.60	1349	0.62	515	0.71	42	0.93	473	0.79	678	0.64	6	0.78	41	0.93	42	0.94	446	0.79	341	0.82	438	0.82
CA-GRQC	991	0.75	954	0.72	607	0.73	415	0.82	691	0.75	781	0.72	377	0.84	392	0.86	393	0.86	456	0.77	561	0.79	593	0.80
Collaboration	1594	0.69	2369	0.70	1758	0.70	668	0.80	1825	0.77	1876	0.70	618	0.84	623	0.85	626	0.85	1421	0.72	1637	0.79	955	0.78
CA-HEPTH	1729	0.62	1611	0.61	949	0.68	538	0.72	993	0.627	1344	0.60	458	0.74	475	0.77	478	0.77	586	0.63	761	0.70	928	0.70
PGP	2059	0.74	1864	0.74	625	0.76	191	0.85	869	0.80	911	0.79	3	0.13	96	0.88	95	0.88	643	0.59	358	0.82	554	0.81
Condmat-2003	4220	0.62	3586	0.51	2691	0.67	1240	0.68	3122	0.68	3387	0.61	945	0.68	959	0.76	961	0.77	3502	0.57	2314	0.70	1872	0.684
Condmat-2005	5145	0.59	2360	0.25	2938	0.59	1437	0.65	3691	0.64	3906	0.62	1006	0.61	1021	0.72	1026	0.73	3952	0.44	2565	0.66	1985	0.64
DBLP	43,352	0.65	30,986	0.61	12,097	0.70	3078	0.73	19,405	0.69	22,589	0.69	531	0.81	198	0.82	208	0.83	17,280	0.65	18,394	0.70	23,559	0.71
Amazon	37,428	0.72	44,036	0.67	22,549	0.74	1463	0.87	21,749	0.77	29,731	0.74	13	0.78	232	0.93	382	0.93	34,304	0.68	15,501	0.80	22,600	0.82
YouTube	–	–	–	–	–	–	–	–	25,914	0.50	40,183	0.64	968	0.69	7365	0.716	4039	0.73	9636	0.42	25,169	0.45	18,838	0.47
Orkut	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	8661	–	12,306	–	18,489	–
LiveJournal	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	97,213	–	103,493	–	119,125	–

reveals that the Leiden algorithm has erroneously identified 208, 382, and 4039 communities for the DBLP, Amazon, and YouTube datasets respectively. These findings represent a significant deviation from the actual community structure, indicating a substantial misidentification of community boundaries. As evident from Tables 11, 12, and 13, the LMFLS and LBLD algorithms are two methods that are more successful in obtaining high accuracy.

4.4.3. Modularity evaluation on real-world datasets

Table 13 contains a comprehensive evaluation of the algorithms' performance across 17 real-world networks, utilizing modularity as the primary assessment metric. The table denotes the modularity value (Q) and the corresponding number of detected communities (C) for each dataset. Although obtaining dense communities is the goal of community detection, it is crucial to acknowledge that a high modularity value, while indicative of a well-structured community, does not unequivocally guarantee the algorithm's efficiency and accuracy. As an instance, as indicated by Table 11 and Table 12, the LMFLS algorithm in the Karate dataset has accurately identified two communities with 100 % accuracy and a modularity of 0.3715. Conversely, while Leiden achieved the highest modularity score of 0.42, it concurrently exhibited a significant misidentification of community boundaries, raising concerns about the accuracy of its results despite its high modularity value.

Based on the established ground-truth of the Karate dataset, the actual modularity score is determined to be 0.3715. Furthermore, the LMFLS algorithm has identified two communities out of two communities with the lowest modularity in Dolphins network. However, it has achieved 100 % detection accuracy with NMI = 1. This outcome signifies that the proposed LMFLS algorithm has effectively identified the true underlying community structure and accurately assigned members to their respective communities. Therefore, the modularity value attained by the LMFLS algorithm precisely matches the true modularity of the Dolphins dataset, as established by the ground-truth, indicating a perfect alignment between the algorithm's output and the actual community structure. In contrast, Infomap and Leiden have detected five communities with modularity 0.53 and 0.527, respectively, instead of the actual modularity of 0.378. Leiden algorithm has detected five communities in the Dolphins dataset, while the actual number of communities is two. Leiden has identified small but dense communities, which has resulted in a higher modularity score than LMFLS. A comprehensive analysis of Table 13 reveals that the Leiden algorithm consistently achieves the highest modularity values among its counterparts. However, this superior performance in modularity is unfortunately accompanied by a less than satisfactory accuracy in community detection, highlighting a trade-off between these two critical metrics.

Furthermore, when examining the Polbooks dataset, the LSMD method exhibits the lowest modularity score but achieves the second-highest F-measure, suggesting a greater proficiency in uncovering the authentic community structure of the Polbooks network. The LMFLS algorithm demonstrates a marginal difference of 0.01 in modularity compared to LSMD, indicating that the modularity attained by the LMFLS algorithm closely approximates the true modularity of the Polbooks dataset. In contrast, the Leiden and Louvain algorithms have detected four and five dense communities, respectively, with a modularity score much higher than the actual one.

A thorough examination of the ground-truth datasets presented in Tables 8, 11, and 12 substantiates the validity of our claim. In the DBLP dataset, the Leiden algorithm identifies a mere 208 communities, signifying the poorest performance among all compared methods, despite its attainment of the highest modularity score. This observation highlights a critical discrepancy between modularity and the actual representation of community structure, underscoring the limitations of relying solely on modularity as an evaluation metric. The same conditions can be observed in the Amazon dataset. In the Amazon dataset, both Leiden and Louvain have the highest modularity scores, but their F-measure is significantly lower than the LMFLS and other methods.

Adding to the aforementioned observations, Infomap incorrectly detects a mere 13 communities for Amazon, a stark contrast to the true number of 75,149 communities, despite its modularity score being relatively close to LMFLS. This discrepancy underscores the importance of considering both modularity and community identification accuracy.

Throughout the experiments, the LMFLS, LBLD, and LSMD algorithms exhibited remarkable alignment with the actual modularity of networks with ground-truth, indicating their effectiveness in accurately capturing the underlying community structures. The LMFLS algorithm demonstrably exhibits superior stability compared to alternative methods. While other algorithms, including LPA, NIBLPA, LPA-Intimacy, GCN, Louvain, and Leiden, exhibit variability in community detection outcomes across different executions, the LMFLS algorithm consistently yields identical results for a given number of iterations, ensuring predictable and reliable community structure identification. It is understandable that LMFLS might not achieve a higher modularity score than algorithms like CNM, Louvain, and Leiden, as these algorithms explicitly optimize for modularity. In addition, LMFLS has obtained better modularity than LSMD and LBLD in datasets such as: CA-GRQC, DBLP, Amazon, and YouTube.

4.5. *Results of experiments conducted on LFR datasets*

To expand the experiments, four LFR datasets are incorporated alongside real-world networks. These LFR datasets were generated with varying values of the parameter μ, ranging from 0.1 to 0.8. While most algorithms exhibit consistent behavior for small μ values, their accuracy may diverge as μ increases. Methods demonstrating greater robustness maintain accuracy under these conditions. Analysis of Fig. 11 reveals that LMFLS, LBLD, and LSMD consistently yield the highest results. Generally, label propagation-based methods achieve second-best scores after the aforementioned algorithms. However, these methods exhibit reduced robustness, particularly at high values of μ. Notably, LPA-Intimacy stands out among label propagation-based approaches, demonstrating both higher accuracy and robustness compared to LPA and NIBLPA, but unfortunately, its design is not robust enough for high values of μ.

The Infomap algorithm exhibits instability in certain LFR networks. The observed behavior stems from the intrinsic stochasticity inherent to the Infomap algorithm. Notably, anomalous results were detected in both LFR2 and LFR3 network configurations. As depicted in Fig. 11(b) and Fig. 11(c), with an initial mixing parameter of 0.1, the Infomap algorithm attained an approximate NMI score of 0.79 whereas other examined methods obtain higher results than this. In contrast, the LMFLS algorithm responds meaningfully to variations in the mixing parameter. Even as the mixing parameter increases to 0.6, 0.7, and 0.8, LMFLS consistently identifies accurate communities when compared to other methods. Conversely, modularity-based methods demonstrate suboptimal performance in sparse LFR networks. The prevalence of small communities and the resolution limit problem hinder their effectiveness. While most algorithms obtain acceptable results in datasets with $\mu \leq 0.4$, only LMFLS, LBLD, LSMD, LPA-Intimacy, and GCN maintain high accuracy in networks with relatively low density. Given that real-world networks are often sparse rather than heavily dense, modularity-based methods also struggle in such scenarios. The GCN algorithm's accuracy is negatively impacted by the increasing fuzziness of networks with more inter-community links. This is attributed to the algorithm's inability to effectively label border nodes in such ambiguous network structures. The LMFLS algorithm leverages a dual approach: it employs precise metrics to assign importance to nodes and executes a node attraction step. By strategically grouping nodes around pivotal ones, LMFLS effectively constructs accurate communities even within fuzzy networks.

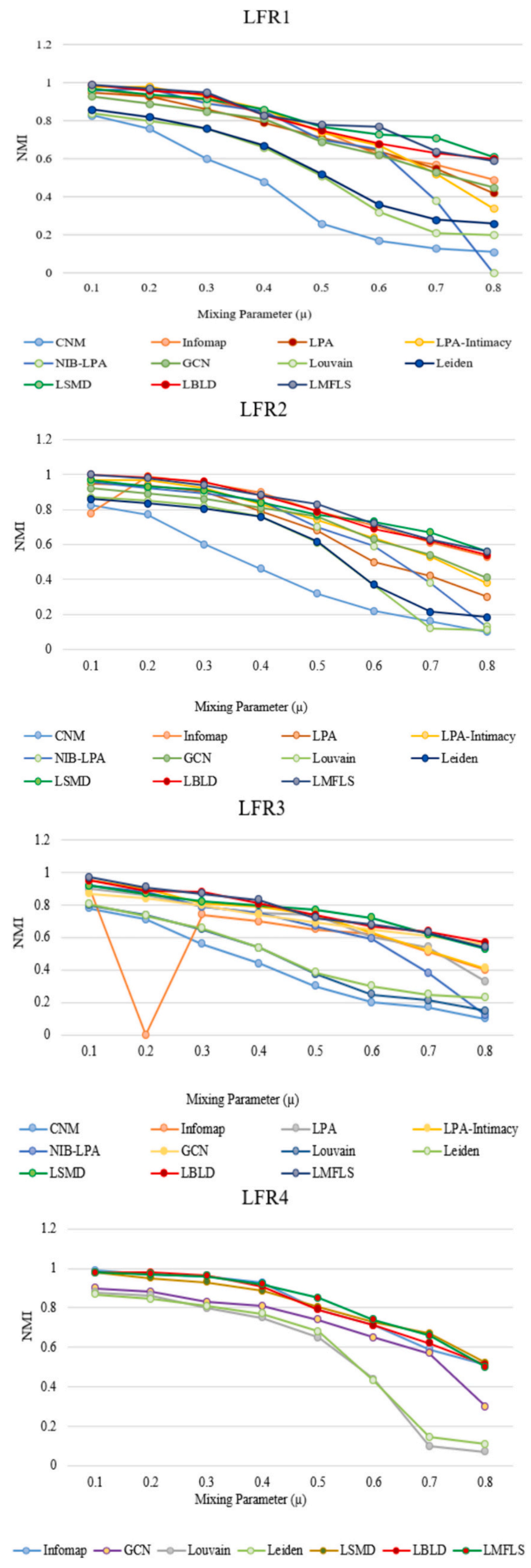


Fig. 11. NMI comparison of algorithms on 4 LFR networks.

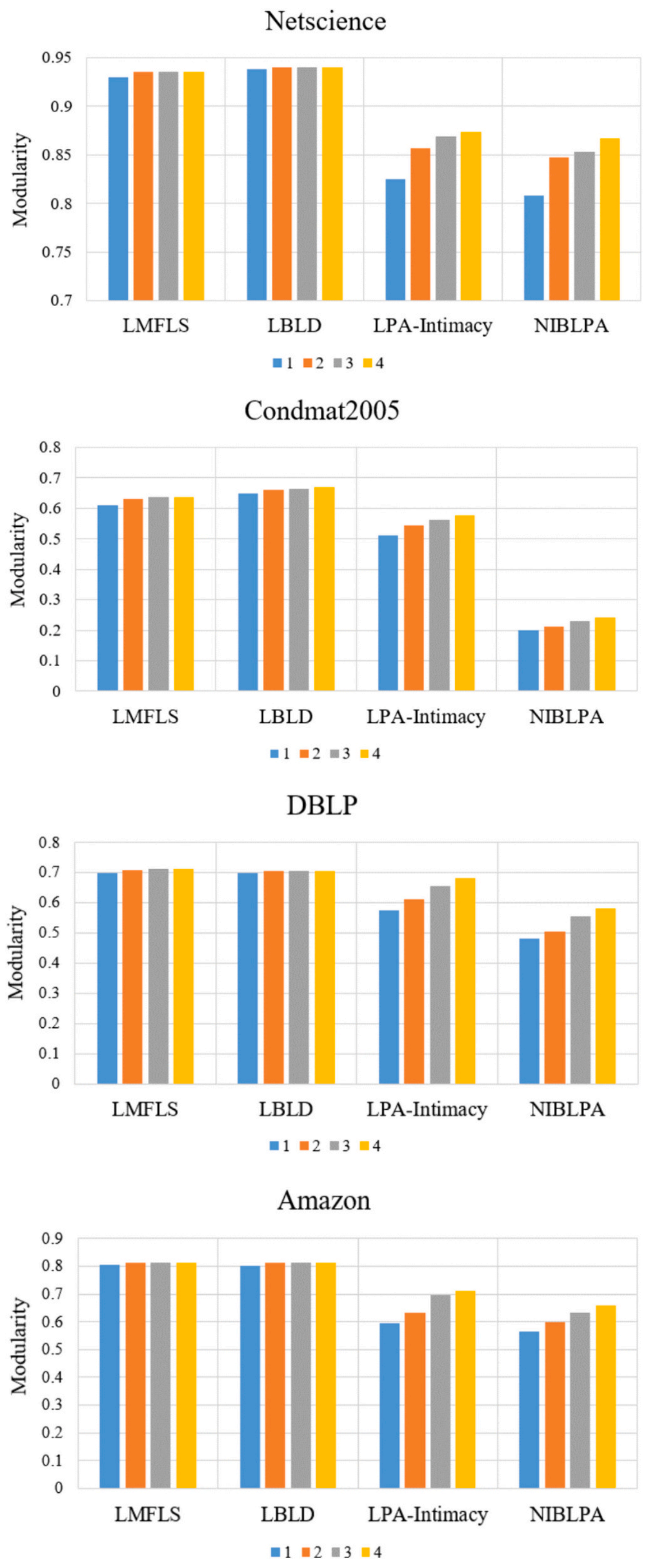


Fig. 12. Modularity results obtained from execution of LMFLS, LBLD, LPA-Intimacy, and NIBLPA with different numbers of iterations on four real-world datasets.

4.6. *Convergence analysis of the LMFLS*

To highlight the superiority of the LMFLS over label propagation-based methods, the convergence speed of the LPA-Intimacy and NIBLPA algorithms are compared to that of LMFLS. Notably, while LMFLS and LBLD converge within the first or second iteration, LPA-Intimacy and NIBLPA fail to converge even after ten iterations. LMFLS and LBLD exhibit fast convergence, with minimal observed improvement beyond the second iteration. In contrast, LPA-Intimacy and NIBLPA gradually improve their modularity results with increasing repetitions but do not converge. Fig. 12 presents a comparative analysis of modularity scores achieved by the proposed LMFLS algorithm, along with LBLD, LPA-Intimacy, and NIBLPA, across varying iteration counts and network scales (small, medium, and large). Notably, the LMFLS algorithm achieves final results after just one iteration in datasets such as Karate, Dolphins, Polbooks, and Football. Also, convergence tends to commence after the initial iteration. For instance, in the Netscience network, the LMFLS algorithm converges following the first iteration, with the result obtained in the fourth iteration being identical to that of the first.

The modularity value obtained when executing LMFLS with 1 iteration in Condmat_2003 differs from that obtained with 4 iterations by a mere 0.003. The LMFLS algorithm exhibited remarkable convergence properties even on extensive networks like DBLP and Amazon, achieving convergence after a single iteration. Furthermore, the difference in modularity between executing LMFLS with one iteration and four iterations was negligible, amounting to approximately 0.007.

4.7. *Evaluation of memory usage*

To evaluate the performance of the LMFLS algorithm, two implementations were developed. One implementation employed the

Table 14
The amount of RAM usage by different data structures (all mentioned values are in Megabytes).

Datasets	Proposed structure	Graph structure (NetworkX)
Karate	R: 0.26	R: 0.26
	T: 0.26	T: 0.26
Dolphins	R: 0.27	R: 0.27
	T: 0.27	T: 0.27
Polbooks	R: 0.28	R: 0.37
	T: 0.29	T: 0.50
Football	R: 0.31	R: 0.51
	T: 0.33	T: 0.54
Netscience	R: 1.3	R: 1.6
	T: 1.8	T: 2.8
Power	R: 3.7	R: 4.2
	T: 5.4	T: 8.1
CA-GRQC	R: 6.4	R: 7.3
	T: 8.1	T: 13.2
Collaboration	R: 7	R: 10
	T: 10	T: 17
CA-HEPTH	R: 11	R: 14
	T: 15	T: 24
PGP	R: 11	R: 14
	T: 14	T: 23
Condmat-2003	R: 49	R: 65
	T: 60	T: 102
Condmat-2005	R: 69	R: 94
	T: 84	T:140
DBLP	R: 421	R: 645
	T: 517	T: 906
Amazon	R: 361	R: 587
	T: 501	T: 850
YouTube	R: 975	R: 1997
	T: 1535	T: 2498
LiveJournal	R: 4990	R: N/A
	T: 8920	T: N/A
Orkut	R: 9258	R: N/A
	T:11988	T: N/A

suggested neighbor list structure, while the other constructed a graph using the NetworkX library in Python. Our evaluation focuses on memory usage within the constraints of a RAM with 12GB of capacity. The results of RAM usage for each of data structures are written in Table 14. The parameter *R* denotes the RAM allocation dedicated to dataset ingestion and graph construction, while parameter *T* represents the aggregate RAM consumption of the algorithm in its entirety. Based on the results obtained for small-scale datasets, the differential RAM usage between the proposed structure and the graph structure is not substantial. However, in all scenarios, the proposed structure utilizes less RAM space than the graph structure. The primary distinction between these two structures becomes more pronounced in large-scale networks. A marked discrepancy in RAM utilization emerged when comparing the graph and neighboring list data structures in network analysis. The neighboring list structure consistently demonstrated superior RAM efficiency, particularly within large-scale datasets. In the DBLP and Amazon datasets, constructing a graph required approximately double the RAM space compared to implementing the neighboring list structure. This disparity became even more pronounced with the YouTube dataset, where graph construction consumed three times more RAM than the neighboring list approach during dataset reading. The significant RAM demands of the graph structure ultimately hindered the analysis of large datasets such as LiveJournal and Orkut. Due to these constraints, constructing a graph within the available 12 GB of RAM proved infeasible. This analysis highlights the substantial advantages of the neighboring list structure in terms of RAM efficiency, particularly when handling datasets of considerable size. The neighboring list approach allows for the processing of larger networks without exceeding practical memory limitations.

4.8. *Evaluation of execution time*

The LMFLS algorithm is proposed with the primary objective of finding communities in a short time span, especially on large-scale networks. In addition to accuracy, the algorithm is expected to deliver high-quality results within a reasonable duration. While some community detection algorithms may achieve satisfactory accuracy in large-scale networks, their execution times often prove prohibitive. To address the growing need for efficient community detection, algorithms must demonstrate both definitive and consistent results while achieving rapid execution speeds. Table 15 provides a comprehensive analysis of the execution times for the algorithms evaluated across various datasets. A direct comparison between LSMD, NIBLPA, and LPA-Intimacy, implemented in MATLAB, and other algorithms would be inaccurate due to MATLAB's performance limitations compared to Python. While the Leiden algorithm, implemented through the CDLIB library, utilizes a significant portion of C++ and C code, contributing to its faster execution than LMFLS, the comparison remains inherently biased due to the underlying programming language differences.

Analysis of LFR datasets revealed a notable performance disparity between LMFLS and the Leiden and Louvain algorithms. While LMFLS exhibits consistent execution times across varying μ values for a given LFR dataset configuration, its performance is solely influenced by the number of nodes, independent of network structure or community count. In contrast, Leiden and Louvain demonstrate significant sensitivity to resolution and community count within the network, resulting in substantial performance fluctuations across different LFR networks with varying μ parameters. The observed performance inconsistencies between Leiden and Louvain are further exemplified by specific dataset analyses. In the LFR4 dataset, for instance, Louvain's execution time varied dramatically with changes in the μ parameter, ranging from 359 s ($\mu = 0.1$) to 5107 s ($\mu = 0.8$). Similarly, the Leiden algorithm exhibited unexpected behavior when comparing its performance on the Amazon network (334,000 nodes, 925,872 edges) and an LFR network with fewer nodes and edges (200,000 nodes, 900,000 edges). Despite the LFR network's smaller size, the Leiden algorithm required 18 s for completion, compared to 13 s for the Amazon network. This disparity underscores the algorithm's sensitivity to the minimum and maximum node counts within communities. These findings highlight the unpredictable nature of Leiden and Louvain algorithms, which are highly susceptible to community size variations and lack consistent performance across diverse networks.

According to the results presented in Table 15, it is evident that for the first four datasets, the execution time is <1 s. Consequently, no discernible difference can be observed among the performance of the examined methods. Despite the existence of substantial differences between the execution time of the proposed algorithm and other methods, this time is so minimal that it is not perceptible when executed on a computer. However, as the datasets increase in size, the time difference becomes more pronounced. In large datasets, any additional or non-optimal operation can cause a significant increase in the execution time of the algorithm due to the increased number of nodes. For instance, the difference in execution time becomes more apparent starting with the Collaboration datasets. Table 15 reveals that LMFLS and LBLD consistently exhibited the most efficient execution times across all datasets. These algorithms demonstrated remarkable

Table 15
Execution times of the examined algorithms on real-world and LFR datasets (All mentioned times are in seconds).

	CNM	Infomap	LPA	NIBLPA	LPA-Intimacy	GCN	Louvain	Leiden	LSMD	LBLD	LMFLS (NetworkX)	LMFLS
Karate	0.018	0.009	0.015	0.17	0.19	0.02	0.016	0.011	0.15	0.001	0.004	0.001
Dolphins	0.018	0.015	0.021	0.23	0.26	0.039	0.018	0.011	0.21	0.002	0.007	0.002
Polbooks	0.064	0.022	0.076	0.39	0.41	0.09	0.018	0.015	0.34	0.007	0.018	0.005
Football	0.078	0.026	0.11	0.52	0.53	0.12	0.022	0.015	0.47	0.009	0.029	0.007
Netscience	0.36	0.19	0.24	2.7	3.2	0.38	0.19	0.07	1.32	0.09	0.1	0.082
Power	1	1.2	1.7	12.3	14.4	1.9	1.8	0.098	3	0.18	0.37	0.15
CA-GRQC	14	1.31	2.3	17.2	20.6	4.6	1.73	0.19	4.8	0.30	0.9	0.25
Collaboration	18	1.72	2.63	25	26	4.9	2.2	0.28	5.25	0.36	0.81	0.32
CA-HEPTH	68	2.9	4.97	40	45	7.8	3.48	0.4	7.6	0.56	1.2	0.48
PGP	52	2.3	4.6	33	38	7.6	3.24	0.34	6.3	0.52	1.32	0.46
Condmat-2003	585	4	15	204	230	33	10	1	23	2	15	2
Condmat-2005	1043	6	33	298	385	73	15	2	38	3	26	3
LFR1	311	6	46	984	986	37	51	2	31	3	19	3
LFR2	4038	14	161	1588	1680	108	112	6	78	10	52	10
LFR3	19,835	32	1496	1810	1893	121	459	9	94	13	66	13
DBLP	53,757	46	2824	22,146	21,237	332	212	12	182	20	146	18
Amazon	18,784	59	2488	23,090	20,120	355	128	13	161	18	116	19
LFR4	N/A	89	N/A	N/A	N/A	652	1432	45	523	69	342	64
YouTube	N/A	N/A	N/A	N/A	N/A	13,569	498	53	609	311	2970	302
LiveJournal	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	7061	1347	N/A	1315
Orkut	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	30,292	13,925	N/A	14,228

competitive performance, achieving comparable execution times in most of the datasets. Notably, LBLD surpassed LMFLS in the Orkut dataset, while LMFLS outperformed other algorithms in the DBLP, LFR4, YouTube, and LiveJournal datasets. The substantial performance gap between LMFLS/LBLD and other methods is evident. While Infomap achieved the second-fastest execution times, significant differences in execution time persisted between LMFLS and Infomap. It is noteworthy that, as documented in [55,56], a significant portion (73 %) of the Infomap algorithm is implemented in C++, which could contribute to its performance advantage. These findings underscore the exceptional performance of LMFLS, particularly when compared to other established community detection algorithms. Despite running for an extended period of 15 h, the LPA-intimacy, LPA, NIBLPA, and CNM algorithms failed to complete their execution on the YouTube dataset, indicating significant performance limitations when dealing with this large-scale network.

The LMFLS algorithm exhibited exceptional efficiency in processing the LiveJournal dataset, outperforming other methods (excluding LSMD and LBLD) which were unable to complete the task. Furthermore, while the NetworkX implementation of LMFLS demonstrated slightly slower execution times compared to the neighboring list implementation across other datasets, this disparity highlights the efficiency of the proposed data structure. Despite the near-linear time complexity of most compared methods, their ability to handle large-scale networks was severely hampered by memory constraints and other performance bottlenecks. This underscores the significant impact of data structure design on the effectiveness of community detection algorithms, particularly in the context of large-scale datasets.

4.9. Comparison analysis of model design of examined algorithms

Based on the comprehensive evaluations conducted across a diverse range of datasets, it is evident that the primary competition lies between the LMFLS and LBLD algorithms. These two methods consistently demonstrate superior performance in comparison to alternative methods. Notably, LMFLS and LBLD exhibit comparable behavior across various evaluation metrics, including NMI, F-measure, modularity, convergence speed, execution time, and robustness to variations in the mixing parameter within the LFR datasets. To investigate these two methods, different aspects are compared:

Graph representation: The underlying reason for the comparable execution time of LMFLS and LBLD lies in their efficient design choices. Both algorithms employ a neighboring list representation for network implementation instead of constructing a graph, leading to a significant reduction in execution time. However, LBLD differs from LMFLS in that it does not store edge weights. Notably, the time complexities of both algorithms are similar because neither incurs computationally expensive operations such as calculating modularity gain, examining all neighboring nodes, or performing sorting operations with a time complexity of $O(nlogn)$. The time complexity of LBLD is $O(nk)$, while that of LMFLS is $O(n+m)$), which contributes to their comparable execution times. However, it is essential to emphasize that similar time complexities do not always translate to identical execution times. In addition to their time complexity, LBLD employs an efficient algorithmic design, similar to LMFLS. Rather than repeatedly executing the same steps and operations with a large number of repetitions, LBLD utilizes multiple steps with only one or two iterations. This efficient balanced label diffusion mechanism, followed by an enhanced label propagation step, enables LBLD to converge rapidly, akin to LMFLS. Consequently, the fast convergence of both algorithms results in their low execution times.

Model design: LBLD is a method based on label diffusion and encompasses several steps, including assigning importance to nodes, identifying rough cores and diffusing their labels to neighboring nodes, implementing balanced label diffusion, performing an improved label propagation, and finally employing a merging step. LMFLS, on the other hand, consists of steps such as assigning importance to nodes, incorporating a node attraction step, employing fast label selection, performing final label selection, and utilizing a merging step. A key shared characteristic of both methods is their initial phase, where nodes are prioritized based on their importance to ensure a consistent order of processing, albeit with significant differences. One of the primary reasons for LMFLS's superior performance and accuracy compared to LBLD lies in its adoption of a combination of metrics. Specifically, LMFLS leverages improved K-shell, improved H-index, common neighbors, edge weights, and node degrees to assign more accurate and comprehensive scores to nodes. This enables LMFLS to rank nodes more effectively, as the order of label updates significantly impacts the final accuracy. In contrast, LBLD relies on a single metric, which is the sum of an improved similarity metric between nodes and their neighbors. This approach may have limitations in distinguishing nodes and selecting an accurate order of nodes.

Both LBLD and LMFLS aim to establish initial communities and stabilize the fundamental structure of communities to facilitate subsequent steps and enhance accuracy. However, their approaches to contracting initial communities and stabilizing the structure differ significantly. After assigning importance scores to nodes, LBLD selects the top 5 % of nodes as rough cores and performs label diffusion to establish initial communities around core nodes and stabilize the structure. In contrast, LMFLS employs only a node attraction step for constructing initial communities and stabilizing the structure. While both methods share the same goal, LMFLS exhibits greater efficiency. By performing the node attraction step, all nodes receive labels, and the structure of communities is stabilized more effectively. This is because all nodes have labels and are clustered around their important and similar neighbors. LBLD, on the other hand, selects a limited number of nodes as cores and diffuses labels to their surrounding nodes which share common neighbors with them. Consequently, not all nodes receive labels, and the structure is not fully stabilized for balanced label diffusion, potentially leading to inaccuracies.

Furthermore, since LMFLS assigns labels to all nodes during the node attraction step, the subsequent label selection step is performed faster and with more confidence. This is because LMFLS has more prior knowledge and all nodes surrounding a given node have labels. In contrast, LBLD performs balanced label diffusion when not all nodes have labels, which contributes to LMFLS's superior accuracy compared to LBLD. Another factor contributing to LMFLS's faster execution time compared to LBLD is the difference in their approaches to label assignment. LBLD calculates common neighbors between core nodes and surrounding nodes during label diffusion, which can be computationally intensive. In contrast, LMFLS employs a more efficient strategy by selecting the most similar neighbor in the node attraction step. This optimization reduces the computational overhead and contributes to LMFLS's improved execution time.

Label assignment mechanisms: Moreover, in LBLD's balanced label diffusion step, all nodes are examined in each iteration. In contrast, LMFLS employs a more selective approach. Nodes located in dense regions and surrounded by a single label type, or nodes where the most frequent label is much greater than the second most frequent label, are excluded from further iterations. This optimization contributes to LMFLS's superiority in execution and convergence speed compared to LBLD. Furthermore, empirical evidence suggests that the combination of node attraction, fast label selection, and final label selection steps in LMFLS yields better results and faster execution than LBLD's approach of constructing rough cores, diffusing their labels, adopting balanced label diffusion, and implementing improved label propagation.

Another key differentiator lies in LMFLS's additional layer for label refinement (final label selection), whereas LBLD solely relies on balanced label diffusion and improved label propagation. A significant distinction in label assignment also contributes to LMFLS's superior performance and accurate label selection. In LBLD's balanced label diffusion, only a single metric is employed, namely the influence or importance of nodes within each community. LMFLS, on the other hand, leverages a more comprehensive set of metrics, including the sum of node importance, weighted similarity, label frequency, and the importance of the label's origin. This comprehensive approach enables LMFLS to effectively calculate the influence of labels. Additionally, LMFLS employs more refined strategies in label assignment, such as considering the *is_dense_flag*. These optimizations contribute to the accurate formation of communities by avoiding unnecessary label updates for every node, resulting in improved execution time. Furthermore, LMFLS's more stabilized structure of communities and comprehensive metric for label evaluation lead to a reduction in label updating variations, facilitating faster convergence. It is important to acknowledge that LBLD's balanced label diffusion step remains a highly efficient label assignment technique, enabling LBLD to achieve accurate results and fast convergence. However, the comprehensive metric combination and efficient step combination employed by LMFLS ultimately provide better performance compared to LBLD.

Other methods employed in the experiments adopt a graph structure, which significantly increases RAM usage and hinders the execution speed of the algorithms. These methods lack mechanisms to initially gather nodes around important nodes and stabilize the initial network structure. They commence from a "cold-state" where no nodes have labels, leading to increased effort in selecting true labels and performing tie-breaking steps in the initial stages. Consequently, their accuracy suffers, convergence is delayed, and execution time increases. Moreover, these methods lack strategies to avoid unnecessary label updates for certain nodes. Instead, they examine all nodes in each iteration, contributing to high execution times and slow convergence.

Additionally, most of the algorithms exhibit higher time complexity than LMFLS, or even if their time complexity is comparable, they employ time-consuming operations such as calculating modularity gain, verifying the labels of all neighbors in each iteration, and executing multiple tie-breaking steps. Their node evaluation mechanisms are also weak. For instance, NIBLPA, LPA-Intimacy, and LCDR solely rely on K-shell score, sum of influence, and inverse sum of node degrees to assign importance to nodes, respectively. Their label assignment strategies are equally weak, utilizing only a single metric such as the sum of K-shell scores or the sum of node importance within a community. LPA and LPA-based methods suffer from low accuracy and high execution times due to their inefficient label updating strategy, exhaustive node examination in each iteration, and delayed convergence. GCN attempts to identify border nodes using a single metric and updates border nodes to reduce execution time. However, accurately identifying border nodes is challenging, and GCN's weak metric hinders its accuracy. Additionally, it employs a suboptimal fitness function to assign nodes to communities or determine border node status.

Furthermore, community detection can be viewed as a system where steps should be harmoniously combined to enhance overall performance. However, the aforementioned methods merely repeat the same operations in multiple iterations without introducing significant improvements and do not have any strategy to refine mistakes of the previous steps, so iteratively performing same steps with same mistakes, do not enhance the performance of the algorithm. In contrast, LMFLS employs diverse steps and distinct layers with a single iteration to leverage the benefits of each concept. CNM, Louvain, and Leiden disregard the importance, similarity, and relationships between nodes, opting for a greedy merging approach. This results in lower accuracy and higher execution times due to time-consuming operations such as computing modularity for every node and community. LPA is plagued by numerous drawbacks, including randomness, disregard for node importance and similarity, instability, and a label updating strategy solely based on label frequency, which is an unreliable metric.

Table 16
Comparison of time complexity of different methods.

Algorithm	Time complexity
CNM [28]	$O(nlog^2n)$
LPA [13]	$O(m+n)$
NIBLPA [14]	$O(nlog\,n)$
LPA-Intimacy [15]	$O(nlog\,n)$
GCN [31]	$O(nk)$
Infomap [5]	$O(n(n+m))$
Louvain [21]	$O(nlogn)$
Leiden [22]	$O(nlogn)$
LSMD [24]	$O(nk)$
LBLD [22]	$O(nk)$
LMFLS	$O(n+m)$

Despite having similar or comparable time complexity to algorithms like LPA, GCN, LSMD, and LBLD, LMFLS exhibits superior execution time. This is attributed to its efficient data structure, optimized design that promotes fast convergence, and the absence of time-consuming operations. To provide a comprehensive comparison, Table 16 presents the time complexity of the algorithms mentioned in Table 15 whose execution times were evaluated. This analysis demonstrates that even methods with similar time complexity to LMFLS incur higher execution times due to the aforementioned issues in their model design.

4.10. Application of LMFLS algorithm in drug repositioning

Drug repositioning is an approach for identifying new therapeutic applications for existing medications [57]. This approach offers a compelling alternative to traditional drug development, which is often characterized by lengthy timelines, high research costs, and substantial resource requirements. One of the efficient computational methods to overcome the limitations of traditional drug discovery, involves modeling drug interactions as graphs, utilizing various metrics such as shared targets, biological pathways, chemical structures, etc. to capture diverse interactions between drugs. However, the initial form of this graph merely depicts these interactions without providing actionable insights. Community detection algorithms are suggested as efficient tools to extract valuable information and facilitate the clustering of drugs based their similar patterns. These cohesive communities represent functionally related sets of drugs, revealing hidden patterns within the network. By identifying these cross-community connections, the approach uncovers hidden associations that might be missed by traditional methods.

Based on the experimental results presented in preceding sections, it has been observed that the LMFLS algorithm exhibits superior accuracy and efficiency compared to other algorithms. In an effort to broaden its applicability in practical scenarios, this section will focus on the implementation of the proposed LMFLS on Drug-Drug network as a case study of LMFLS in real-world application. The aim is to identify clusters of drugs with analogous functionality, which are utilized to treat similar diseases, a process commonly referred to as drug repositioning. To conduct a comprehensive examination of various facets of drugs and to construct a more inclusive network that encompasses a broad spectrum of drug-related information, a multiplex graph structure is proposed. This structure will consist of four layers of drug similarity networks based on diverse concepts, thereby encapsulating drug similarities from multiple perspectives. Multiplex networks, a specific type of multilayer network, are characterized by multiple layers of nodes, each layer representing a unique type of relationship between those nodes [58]. These networks can be employed to model complex systems that have multiple types of interactions between their components.

5. Construction of drug networks

Given that drug relationships can be modeled using different concepts, such as sharing similar targets, having similar chemical structures, and so forth, they can be represented with multiplex graphs. Moreover, the adoption of multiplex graphs and the merging of their layers can address the issue of missing information. Finally, the LMFLS algorithm is applied to final integrated Drug-Drug network. The procedure for constructing the multiplex graph for drugs is detailed in the following sections.

1) Drug-Drug network based on Target similarity: The DrugBank database [59] serves as the primary knowledge database for obtaining comprehensive drug-related information. DrugBank is an extensive online resource that provides data on drugs and drug targets, along with other valuable drug-related information. Drugs that have received FDA approval are selected as the target drugs for examination. Information such as the drug name, DrugBank ID, drugs' target, and etc. are retrieved from the DrugBank XML file (version 5.1.10). This section aims to construct a Drug-Drug similarity graph according to their shared targets. The underlying premise of this approach is that drugs with similar targets tend to exhibit similar effects and behavior due to their analogous chemical structures and interactions with the body's proteins. Thus, by identifying drugs that share similar targets, it is possible to predict which drugs are likely to exhibit similar effects [60].

Following the construction of the Drug-Target bipartite network, a link prediction algorithm based on Jaccard similarity [61] is applied to the network to predict potential new links between drugs and targets. BiGraph [62], a Python package for link prediction in bipartite networks, is utilized for application on various constructed bipartite networks. A threshold of 0.7 is set for link prediction, meaning that predicted links with a weight lower than 0.7 are not added to the network. Subsequently, to obtain a Drug-Drug similarity graph according to target proximity, graph projection is performed with a threshold of 0.8. Fig. 13 illustrates the sample procedure of obtaining Drug-Drug similarity graph based on target of drugs.

2) Drug-Drug network based on Chemical structure similarity: The chemical similarity principle posits that if two molecules share similar structures, they are likely to exhibit similar bioactivities [63]. Consequently, the similarity between drugs as chemical information is incorporated into computational methods to enhance the accuracy of predictions. The Simplified Molecular Input Line Entry System (SMILES) is a chemical notation system that utilizes concise ASCII strings to represent the structures of chemical species [64]. The SMILES codes for drugs are adopted from PubChem database [65]. To compare the similarity of two molecules using SMILES code, the RDKit module [66] is utilized. Subsequently, the Tanimoto similarity coefficient [67] is employed to compare the similarity between the two molecules of drugs. The SMILES code of drugs was compared pairwise, and drugs exhibiting a chemical similarity >0.8 were connected to each other.

3) Drug-Drug network based on Pathway similarity: Despite lacking common targets, drugs can exhibit similar therapeutic effects against the same disease. This phenomenon arises from the interconnected nature of biological pathways, where different targets often contribute to the same pathway, ultimately influencing the disease process. This section elucidates the concept of pathway similarity, a measure quantifying the overlap in biological pathways affected by the molecular targets of pharmaceutical agents. Subsequently, a Drug-Drug network is generated based on the similarity of pathways associated with the targets of drugs. To establish associations between drug targets and biological pathways, the targets of drugs are first extracted from DrugBank. The KEGG database [68] is used to download associated pathways of proteins. Prior to constructing the Drug-Drug similarity network according to pathway similarity, a link prediction operation with a threshold of 0.7 is performed on the bipartite network of Protein-Pathway to predict potential new links. Then, based on Drug-Target associations and Protein-Pathway associations, a list of Pathways associated with the targets of each drug is created. The pathway lists of drugs are then compared pairwise using Salton similarity [69], and drugs with a pathway similarity >0.8 are connected via an edge. The similarity score between pathway sets of drugs is considered as the weight of edge connecting two drugs to each other.

$$similarity_{L1,L2}^{Salton} = \frac{|L_1 \cap L_2|}{\sqrt{|L_1| + |L_2|}} \tag{18}$$

Where L_1 and L_2 are two sets containing pathway information, respectively. $|L_1|$ and $|L_2|$ show the length of two sets L_1 and L_2.

4) Drug-Drug network based on Disease similarity: A drug is meticulously designed to interact with a specific target protein. If a drug's target is linked to a gene associated with the disease, the drug's action can significantly impact disease progression. Drugs that correspond to similar diseases exhibit commonality in their functions and efficacies within the human body. To achieve this, first a bipartite network is constructed comprising Drugs-Targets, linking drugs to their respective targets. Subsequently, a link prediction algorithm with a threshold of 0.7 is applied to predict potential links that can be established between drugs and targets. Next, the Disease Gene Network (DisGeNet), which captures Disease-Gene associations [70], is leveraged to extract information about disease-related genes. Link prediction algorithm is applied to the Disease-Gene bipartite network using a threshold of 0.7. Both the Drug-Target and Disease-Genes networks are utilized to connect drugs and diseases associated with the same genes or targets. Consequently, the new Drug-Disease bipartite graph reflects interconnected relationships. Finally, the link prediction algorithm with a threshold of 0.7 is applied within the bipartite network to predict novel associations between drugs and diseases.

Notably, our approach introduces a novel aspect: at each step, link prediction is utilized to generate potential connections between entities within the bipartite graph. To the best of our knowledge, while existing

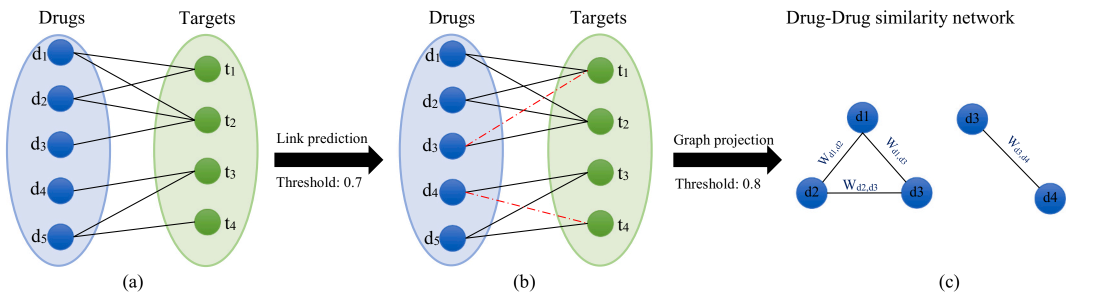


Fig. 13. The total procedure of obtaining Drug-Drug similarity network from projecting Drug-Target bipartite network. (a) Shows the associations between drugs and targets. (b) Drug-Target bipartite network after performing link prediction. Red-colored lines show the new predicted links between drugs and targets. (c) The final Drug-Drug similarity network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

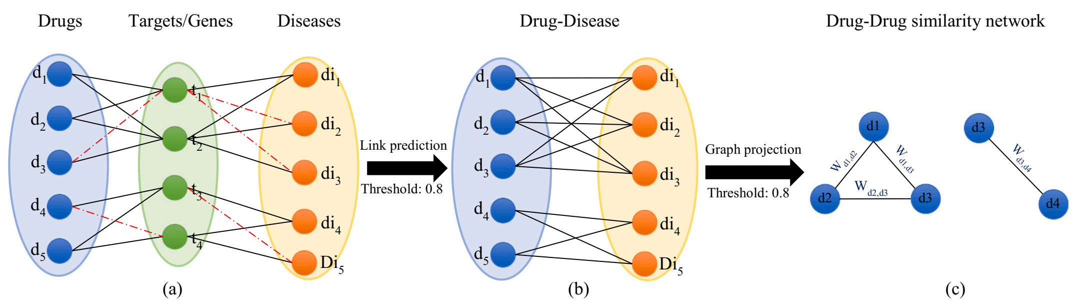


Fig. 14. The total procedure of obtaining Drug-Drug similarity network from projecting Drug-Disease bipartite network. (a) Shows the associations between Drug-Target and Disease-Genes. Black-colored and red-colored lines indicate the original and predicted links, respectively (b) Obtained Drug-Disease bipartite network based on shared Targets/Genes between drugs and diseases. (c) The final Drug-Drug similarity network after projecting Drug-Disease bipartite network. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

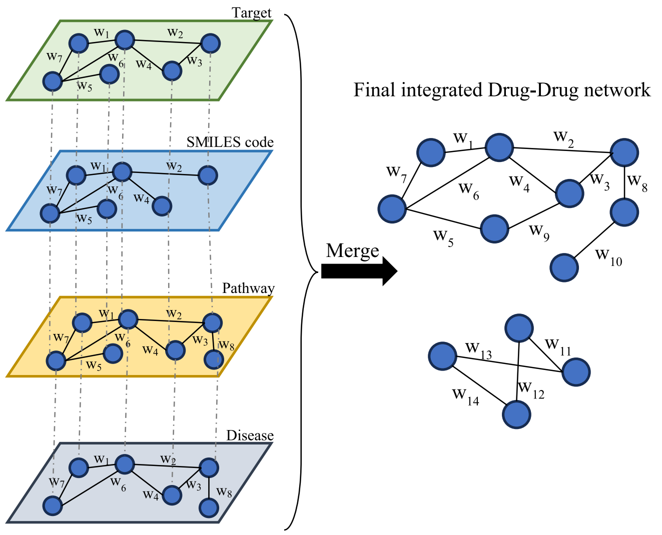
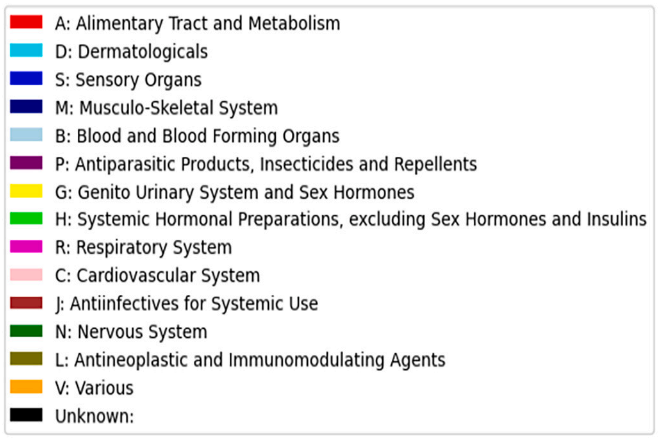


Fig. 15. The procedure of constructing a 4-layered multiplex network of Drug-Drug based on Target, SMILES code, Pathway, and Disease similarity metrics and merging them to obtain the final Drug-Drug network.



🟥	A: Alimentary Tract and Metabolism
🟦	D: Dermatologicals
🟦	S: Sensory Organs
🟦	M: Musculo-Skeletal System
🟦	B: Blood and Blood Forming Organs
🟪	P: Antiparasitic Products, Insecticides and Repellents
🟨	G: Genito Urinary System and Sex Hormones
🟩	H: Systemic Hormonal Preparations, excluding Sex Hormones and Insulins
🟪	R: Respiratory System
🟥	C: Cardiovascular System
🟥	J: Antiinfectives for Systemic Use
🟩	N: Nervous System
🟫	L: Antineoplastic and Immunomodulating Agents
🟧	V: Various
⬛	Unknown:

Fig. 16. 14 anatomical main groups based on first level of the ATC classification system.

works primarily focus on link prediction as a standalone technique, our method takes a more comprehensive approach. To construct the similarity network of drugs, the Drugs-Diseases bipartite graph is projected. Specifically, a threshold of 0.8 is applied and the weight of this projection represents the strength of the link between two drugs within the Drug-Drug network. Fig. 14 visually illustrates an instance of this procedure. In our endeavor to integrate the four layers of the Drug-Drug multiplex network into a cohesive whole, an approach centered around the average weight of edges is employed. Consider two nodes, A and B, connected in different layers. In the final integrated network, the weight of the edge between nodes A and B is determined as: $\frac{W^1_{e_{AB}}+W^2_{e_{AB}}+W^3_{e_{AB}}+W^4_{e_{AB}}}{4}$, where $W^1_{e_{AB}}$ indicates the weight of edge between nodes A and B in layer 1. Fig. 15 shows the concept of constructing a 4-layered multiplex network and obtaining the final Drug-Drug network. Next, the LMFLS algorithm is applied to the obtained Drug-Drug network, aiming to uncover communities of drugs with similar efficacies. Notably, drugs within the same community exhibit similar functionality and may be repurposed for similar diseases—a phenomenon known as drug repositioning.

6. Discussion

Fig. 17 depicts five instances of drug communities detected by the LMFLS algorithm. Given the absence of a consistent ground-truth for evaluating result accuracy, as since the objective is to augment new drugs to the group of the existing set of known drugs used for treating similar diseases, it would not be possible to evaluate results based on an exact ground-truth. Instead, to reveal the classification of drugs, the Anatomical Therapeutic Chemical (ATC) classification system is employed. By using the first level of the ATC codes of drugs as the ground-truth, drug classifications can be assessed more accurately. This investigation employed a network analysis approach to infer the potential therapeutic roles of the drug by examining its proximity to other drugs within established therapeutic classifications. Consequently, when examining Fig. 17, the majority of drugs sharing the same ATC code predominantly belong to the same therapeutic group. Other drugs grouped alongside them are considered potential candidates for drug repositioning. Fig. 16 depicts the 14 anatomical main groups based on the first level of the ATC classification system.

According to Fig. 17(a), the majority of drugs belong to Antineoplastic and Immunomodulating Agents. Since three out of six drugs have the same ATC code L, this group is labeled as L. Triptorelin, a potent GnRH agonist, effectively inhibits testosterone and estrogen synthesis, thereby finding utility in the management of advanced prostate cancer [71]. Similarly, Leuprolide, another GnRH agonist, plays a significant role in the therapeutic management of various conditions including endometriosis, uterine fibroids, central precocious puberty in pediatric patients, and advanced prostate cancer [72]. Both medications belong to the class of GnRH agonists, which exert their therapeutic effects by modulating hormonal levels within the body. Degarelix, a gonadotropin-releasing hormone (GnRH) receptor antagonist, is employed as a hormonal therapeutic agent in the management of advanced prostate cancer. Its mechanism of action involves inhibiting the production of

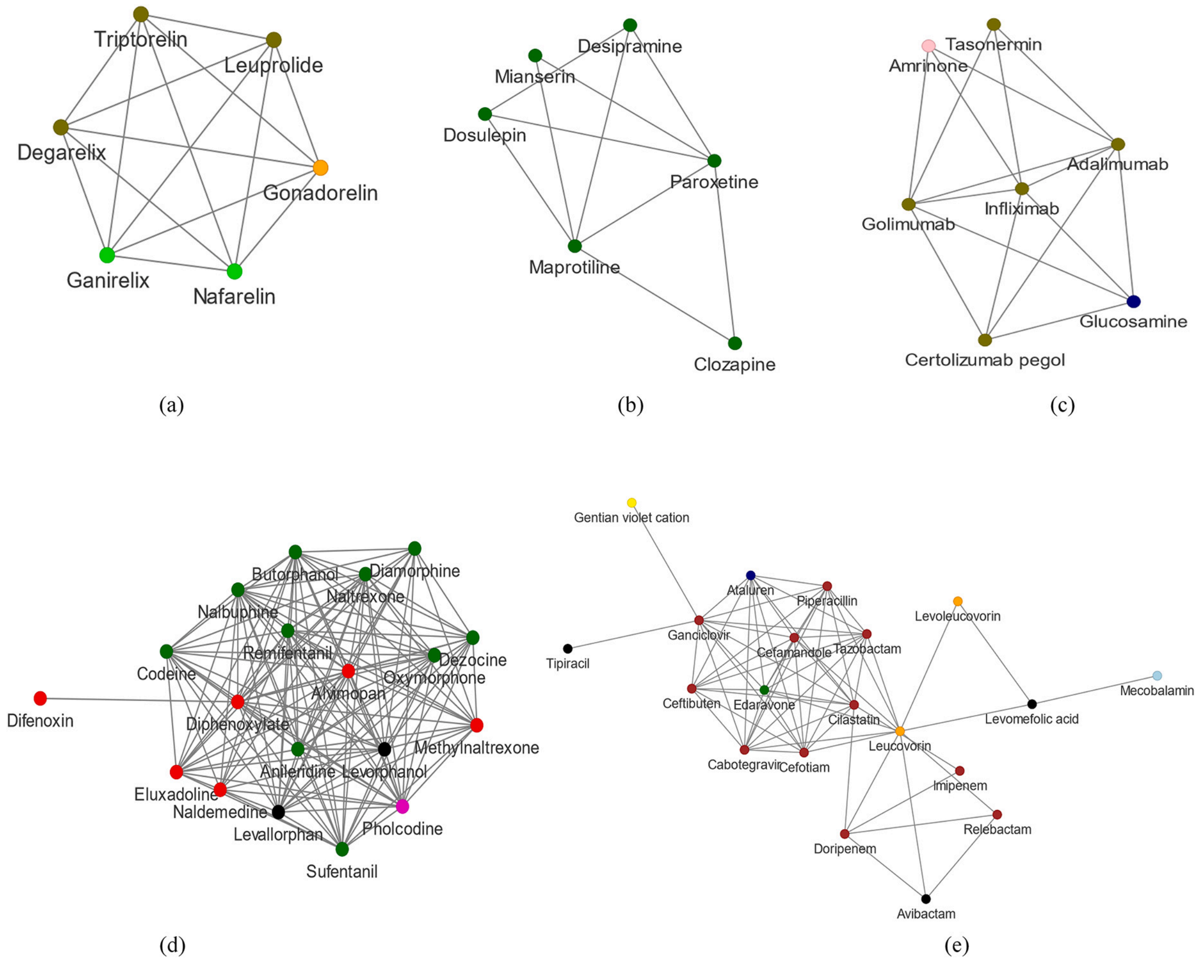


Fig. 17. Examples of discovered drug communities resulting from applying the LMFLS community detection algorithm to Drug-Drug network. Different colors are mapped according to the 1st-level of ATC sub-group based on Fig. 16.

testosterone [73]. According to the research explanation and use case, Triptorelin, Leuprolide, and Degarelix are prescribed for treatment of prostate cancer and for reducing levels of certain hormones in the body. These drugs are shown with the same olive color in Fig. 17(a).

The chemical similarity between Triptorelin and Leuprolide is 0.80, indicating that these two drugs are 80 % similar in their chemical structures, which proves that they have similar efficacies. Although Degarelix has a chemical similarity of 0.48 with Triptorelin, it has a similar mechanism of action with Triptorelin. Degarelix, a GnRH antagonist, exerts its therapeutic effect by selectively binding to GnRH receptors located within the pituitary gland. This binding action effectively inhibits the interaction between GnRH and its receptors, thereby suppressing the release of luteinizing hormone (LH) and follicle-stimulating hormone (FSH). The consequent reduction in LH and FSH levels leads to a significant decrease in testosterone production, achieving the desired therapeutic outcome. Therefore, Degarelix and Triptorelin have similar efficacies and can be suggested as repositioning to each other. Besides this, Gonadorelin is a gonadotropin-releasing hormone agonist (GnRH agonist) that is used in fertility medicine and to treat amenorrhea and hypogonadism [74]. Although Gonadorelin has a different ATC code (V with orange color), it is grouped in the same community as Triptorelin, Leuprolide, and Degarelix. Triptorelin, Leuprolide, and Gonadorelin exhibit a commonality in their classification as analogues of gonadotropin-releasing hormone (GnRH). Chemical similarity between Gonadorelin and Leuprolide and Triptorelin is equal to 0.84 and 0.83, respectively, which shows that they have very similar chemical structures. To better evaluate, the mechanism of action of Leuprolide and Gonadorelin is compared:

Leuprolide, a synthetic GnRH analogue, acts as a GnRH receptor agonist, eliciting an initial surge in the release of luteinizing hormone (LH) and follicle-stimulating hormone (FSH) from the pituitary gland. Similarly, Gonadorelin, a naturally occurring GnRH analogue, stimulates the pituitary gland to release LH and FSH, mimicking the physiological action of endogenous GnRH. As the comparison implies, Leuprolide and Gonadorelin have similar mechanism of action, which is regarded as strong evidence of their similarity. Nafarelin is a gonadotropin-releasing hormone agonist (GnRH agonist) that is prescribed for treatment of endometriosis, Uterine Fibroids, Transgender Hormone Therapy, and early puberty [75]. According to the result of analysis, Nafarelin is regarded as another repurposing candidate drug that is grouped with Triptorelin, Leuprolide, and Degarelix. As per the information provided, Nafarelin is classified in Systemic Hormonal Preparations, excluding Sex Hormones and Insulins, which is different from Antineoplastic and Immunomodulating Agents based on ATC codes. When the chemical structure of Nafarelin is compared with Triptorelin and Leuprolide based on their SMILES code, it has 0.89 (89 %) and 0.8 (80 %) similarity with these two drugs, respectively. This high similarity between two drugs implies that they have very similar chemical structures and despite being classified in two different drug categories based on ATC codes, drugs sharing a high degree of chemical structure have a very high probability of having similar efficacies and mechanisms of action. To investigate the shared targets and mechanism of action of Triptorelin and Nafarelin, a comparison is made between the two drugs:

Triptorelin [76]: Target: Triptorelin is a GnRH agonist, specifically targeting the hypothalamic-pituitary-gonadal (HPG) axis. Mechanism of Action: Triptorelin's initial administration induces a transient elevation in follicle-stimulating hormone (FSH), luteinizing hormone (LH), estradiol, and testosterone levels. This temporary surge may exacerbate symptoms in patients with advanced prostate cancer, including bladder outlet obstruction, bone pain, and hematuria. However, with sustained daily use, triptorelin persistently occupies the GnRH receptors, leading to a reversible downregulation of these receptors in the pituitary gland and desensitization of the pituitary Gonadotropes. This desensitization ultimately renders tissues and functions reliant on gonadal steroids inactive.

Nafarelin [75]: Target: Nafarelin is also a GnRH agonist, acting on the GnRH receptor. Mechanism of Action: Nafarelin, upon initial administration, stimulates the release of pituitary gonadotropins, namely luteinizing hormone (LH) and follicle-stimulating hormone (FSH), leading to a transient elevation in gonadal steroidogenesis. However, repeated dosing eliminates this stimulatory effect on the pituitary gland. Continuous daily administration of Nafarelin results in a reversible downregulation of GnRH receptors within the pituitary gland. As a result, both Triptorelin and Nafarelin are synthetic agonists of GnRH, affecting the HPG axis. They modulate gonadotropin secretion and play essential roles in managing conditions such as advanced prostate cancer and endometriosis. Ganirelix, which is grouped alongside Triptorelin, Leuprolide, and Degarelix, shares the same first-level ATC code (H) with Nafarelin. Ganirelix acetate is an injectable competitive gonadotropin-releasing hormone antagonist [77]. Its primary application lies in assisted reproduction, where it is used to regulate ovulation. Ganirelix functions by blocking the action of gonadotropin-releasing hormone (GnRH) upon the pituitary gland. In context of fertility treatment, Ganirelix plays a crucial role in hormone regulation for women undergoing assisted reproductive procedures. As it is obvious Ganirelix has similar treatments usage as Triptorelin, Leuprolide, Degarelix, and Nafarelin. Even they have same mechanism of action. Ganirelix is a gonadotropin-releasing hormone antagonist (GnRH antagonist) which is identical to Triptorelin, Leuprolide, Degarelix, Nafarelin. All of the aforementioned drugs participate in two common pathways: G alpha (q) signaling events and Hormone ligand-binding receptors. When two drugs share the same pathways based on their targets, it signifies that they both interact with similar biological pathways within the body. By having common targets, these drugs may exert similar effects or influence similar physiological processes.

As it is apparated, all of the drugs are common in targeting Gonadotropin-releasing hormone receptor (GNRHR), despite their differences in first-level ATC codes. The suggested approach for connecting drugs based on 4 different factors and utilizing the accurate detection of LMFLS algorithm, was successful to find a group of drugs with similar efficacy and mechanism of action with 100 % accuracy, despite of having different first-level ATC codes. Ganirelix, along with Triptorelin, Leuprolide, Degarelix, and Nafarelin, exhibits similar therapeutic usage. Importantly, these drugs share a common mechanism of action. Ganirelix acts as a gonadotropin-releasing hormone antagonist, akin to Triptorelin, Leuprolide, Degarelix, and Nafarelin. Based on the analysis results, three new drugs from other categories were identified as potential repositioning candidates for drugs within the category of Antineoplastic and Immunomodulating Agents.

As another instance, drugs grouped in Fig. 17(b) are analyzed as follows: Desipramine, a tricyclic antidepressant, exerts its therapeutic effects through inhibition of noradrenaline reuptake. Furthermore, it has been established that desipramine possesses analgesic properties, as evidenced by studies conducted in both animal models and human subjects [78]. Patients suffering from neuropathic pain often present with comorbid depression and anxiety, conditions known to be temporally linked to noradrenergic dysfunction within the locus coeruleus (LC) as pain transitions into a chronic state. Antidepressants are widely recognized as the first-line pharmacotherapeutic approach for neuropathic pain, with the LC emerging as a promising target for such therapy. Notably, desipramine has demonstrated efficacy in preventing or mitigating the noradrenergic impairment induced by neuropathic pain [79]. Mianserin is another antidepressant that is used to treat depression and anxiety. It is a weak inhibitor of norepinephrine reuptake and strongly stimulates the release of norepinephrine [80].

Dosulepin, an antidepressant, is typically reserved for patients who have not responded to alternative therapies due to its potential toxicity [81]. Maprotiline, another antidepressant, finds application in the management of depressive illness, bipolar disorder, and anxiety associated with depression. Its mechanism of action involves inhibiting neuronal norepinephrine reuptake and exhibiting some anticholinergic activity [82]. Paroxetine, a serotonin reuptake inhibitor, is indicated for the treatment of ymajor depressive disorder, panic disorder, social phobia, and premenstrual dysphoric disorder. A notable feature of paroxetine is its high potency and selectivity in inhibiting serotonin reuptake [83]. Clozapine, a second-generation antipsychotic drug, is employed in the treatment of treatment-resistant schizophrenia and for reducing the risk of suicide in schizophrenic patients [83]. Even though in Fig. 17(b), there was not any other drugs from different category of drugs than Nervous system, but the proposed method was very successful to group six drugs (with 100 % accuracy) with same first-level ATC code and similar efficacy. Based on the information provided in Table 17, all the drugs belong to the neurological drug group, inhibit the neuronal noradrenaline reuptake and are prescribed for similar diseases such as controlling depression and anxiety.

7. Conclusion

The paper proposed a fast local multi-factor node scoring and label selection-based algorithm, called LMFLS, with low time complexity and

Table 17
The comparison of target and pathway information of drugs in Fig. 17(b).

Drug	Target	Pathway
Desipramine	Sodium-dependent serotonin transporter, Alpha-2 adrenergic receptors, Histamine H1 receptor, Muscarinic acetylcholine receptor M1, Muscarinic acetylcholine receptor M2, Dopamine D2 receptor, …	Histamine receptors, Serotonin receptors, Norepinephrine, insulin secretion, Neurotransmitter clearance, Serotonin clearance from the synaptic cleft, Dopamine receptors, Carnitine metabolism, …
Mianserin	Histamine H1 receptor, Sodium-dependent serotonin transporter, Sodium-dependent dopamine transporter, Dopamine D2 receptor, …	Histamine receptors, Serotonin receptors, Dopamine receptors, MECP2 regulates neuronal receptors and channels, …
Dosulepin	Histamine H1 receptor, Muscarinic acetylcholine receptor M1, Sodium-dependent noradrenaline transporter, Sodium-dependent serotonin transporter, …	Histamine receptors, Serotonin receptors, Surfactant metabolism, Adrenoceptors, Acetylcholine regulates insulin secretion, Serotonin clearance from the synaptic cleft, …
Maprotiline	Sodium-dependent noradrenaline transporter, Histamine H1 receptor, Alpha-1 adrenergic receptors, Dopamine D2 receptor, …	Histamine receptors, Serotonin receptors, Surfactant metabolism, Adrenoceptors, Acetylcholine regulates insulin secretion, …
Paroxetine	Sodium-dependent serotonin transporter, Dopamine D2 receptor, Histamine H1 receptor, Serotonin Receptors, …	Histamine receptors, Serotonin receptors, Neurotransmitter receptors and postsynaptic signal transmission, Dopamine receptors, …
Clozapine	Dopamine D2 receptor, Dopamine D1 receptor, Histamine H1 receptor, Histamine H4 receptor, Glutathione S-transferase P, …	Histamine receptors, Serotonin receptors, Surfactant metabolism, Adrenoceptors, Dopamine receptors, Neutrophil degranulation, …

fast convergence. The idea of utilizing multiple factors for scoring nodes was suggested to overcome drawbacks of previous node scoring techniques and to investigate nodes from diverse aspects to better distinguish them based on their different importance in network. A new method of constructing main structure of communities and stabilizing them was proposed based on the natural way of forming communities in real world. Next, fast label selection was proposed to empower the algorithm to assign most appropriate label to nodes based on their local information. The implementation of a novel, accelerated merging technique facilitated the formation of dense communities. To comprehensively evaluate the accuracy, convergence speed, and efficiency of the proposed LMFLS algorithm, it was subjected to rigorous testing using a diverse collection of real-world and synthetic datasets encompassing a broad spectrum of node variations. The obtained results unequivocally demonstrated the superiority of the LMFLS algorithm in terms of accuracy, execution time, convergence speed, and efficient RAM utilization. Comparative analysis of execution times revealed that the LMFLS algorithm consistently outperforms existing methods. This superior performance is attributed to its efficient data structure, rapid convergence rate, and avoidance of computationally intensive operations. As a real-world application, the proposed method was applied on integrated Drug-Drug network constructed based on drugs target, pathway, chemical structure, and common relation of drugs with diseases. The analysis of the discovered drug communities revealed accurate and meaningful relations between drugs of one community, and new potential and meaningful repositioning candidates for drugs was discovered.

The proposed method exhibits significant potential for parallelization. This stems from the inherent independence of node score computations, a key component of the algorithm's execution. Furthermore, the exploration of novel node scoring metrics and label selection strategies holds the potential to considerably enhance both the accuracy and efficiency of the algorithm.

CRediT authorship contribution statement

Huxiong Li: Supervision. **Samaneh Salehi Nasab:** Conceptualization, Data curation, Validation, Writing – original draft, Writing – review & editing, Investigation, Software, Methodology. **Hamid Roghani:** Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Parya Roghani:** Formal analysis, Validation, Writing – original draft. **Mehdi Gheisari:** Project administration. **Christian Fernández-Campusano:** Funding acquisition. **Aaqif Afzaal Abbasi:** Writing – review & editing. **Zongda Wu:** Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

https://github.com/hamid-roghani/LMFLS

Acknowledgement

This work was supported by National Natural Science Foundation of China (Grant No. 62341208), Natural Science Foundation of Zhejiang Province (Grant Nos. LY23F020006 and LR23F020001). Moreover, this has been supported by Islamic Azad Unviersity, Iran with number 133713281361.

Appendix A. Implementation of the LMFLS algorithm

Supplementary data to this article (source code and datasets) can be found online at: https://github.com/hamid-roghani/LMFLS

References

[1] Freeman L. The development of social network analysis. Stud Sociol Sci 2004;1: 687.
[2] Flake GW, Lawrence S, Giles CL, Coetzee FM. Self-organization and identification of web communities. Computer 2002;35(3):66–70.
[3] Chen J, Yuan B. Detecting functional modules in the yeast protein–protein interaction network. Bioinformatics 2006;22(18):2283–90.
[4] Wang W, Yang S, Zhang X, Li J. Drug repositioning by integrating target information through a heterogeneous network model. Bioinformatics 2014;30(20): 2923–30.
[5] Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proc Natl Acad Sci 2008;105(4):1118–23.
[6] Reddy PK, Kitsuregawa M, Sreekanth P, Rao SS. A graph based approach to extract a neighborhood customer community for collaborative filtering. In: International workshop on databases in networked information systems. Springer; 2002. p. 188–200.
[7] Fortunato S. Community detection in graphs. Phys Rep 2010;486(3–5):75–174.
[8] Moody J, White DR. Structural cohesion and embeddedness: a hierarchical concept of social groups. Am Sociol Rev 2003:103–27.
[9] Wu C, Gudivada RC, Aronow BJ, Jegga AG. Computational drug repositioning through heterogeneous network clustering. BMC Syst Biol 2013;7:1–9.
[10] Rives AW, Galitski T. Modular organization of cellular networks. Proc Natl Acad Sci 2003;100(3):1128–33.
[11] Newman ME, Girvan M. Finding and evaluating community structure in networks. Phys Rev E 2004;69(2):026113.
[12] Saoud B, Moussaoui A. Node similarity and modularity for finding communities in networks. Phys A:Stat Mech Appl 2018;492:1958–66.
[13] Raghavan UN, Albert R, Kumara S. Near linear time algorithm to detect community structures in large-scale networks. Phys Rev E 2007;76(3):036106.
[14] Xing Y, Meng F, Zhou Y, Zhu M, Shi M, Sun G. A node influence based label propagation algorithm for community detection in networks. Sci World J 2014; 2014.
[15] Kong H, Kang Q, Liu C, Li W, He H, Kang Y. An improved label propagation algorithm based on node intimacy for community detection in networks. Int J Mod Phys B 2018;32(25):1850279.
[16] Zhang X-K, Ren J, Song C, Jia J, Zhang Q. Label propagation algorithm for community detection based on node importance and label influence. Phys Lett A 2017;381(33):2691–8.
[17] Sun H, et al. CenLP: a centrality-based label propagation algorithm for community detection in networks. Phys A:Stat Mech Appl 2015;436:767–80.
[18] Hao J, Chen P, Chen J, Li X. Multi-task federated learning-based system anomaly detection and multi-classification for microservices architecture. Futur Gener Comput Syst 2024;159:77–90. https://doi.org/10.1016/j.future.2024.05.006.
[19] Ding X, Zhang J, Yang J. A robust two-stage algorithm for local community detection. Knowl-Based Syst 2018;152:188–99.
[20] Zhang T, Wu B. A method for local community detection by finding core nodes. In: 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining. IEEE; 2012. p. 1171–6.
[21] Garruzzo S, Rosaci D. Agent clustering based on semantic negotiation. ACM Trans Autonom Adapt Syst 2008;3(2):1–40.
[22] Roghani H, Bouyer A. A fast local balanced label diffusion algorithm for community detection in social networks. IEEE Trans Knowl Data Eng 2023;35(6): 5472–84. https://doi.org/10.1109/TKDE.2022.3162161.
[23] Roghani H, Bouyer A. A fast local balanced label diffusion algorithm for community detection in social networks. IEEE Trans Knowl Data Eng 2022;35(6): 5472–84.
[24] Bouyer A, Roghani H. LSMD: a fast and robust local community detection starting from low degree nodes in social networks. Futur Gener Comput Syst 2020;113: 41–57. https://doi.org/10.1016/j.future.2020.07.011 (/12/01/2020).
[25] Traag VA, Waltman L, van Eck NJ. From Louvain to Leiden: guaranteeing well-connected communities. Sci Rep 2019;9(1):1–12.
[26] Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. J Stat Mech Theor Exp 2008;2008(10):P10008.
[27] Nadakuditi RR, Newman ME. Graph spectra and the detectability of community structure in networks. Phys Rev Lett 2012;108(18):188701.
[28] Clauset A, Newman ME, Moore C. Finding community structure in very large networks. Phys Rev E 2004;70(6):066111.
[29] Zarandi FD, Rafsanjani MK. Community detection in complex networks using structural similarity. Phys A:Stat Mech Appl 2018;503:882–91.
[30] Wang T, Chen S, Wang X, Wang J. Label propagation algorithm based on node importance. Phys A:Stat Mech Appl 2020;551:124137.
[31] Tasgin M, Bingol HO. Community detection using boundary nodes in complex networks. Phys A:Stat Mech Appl 2019;513:315–24.
[32] Li C, Chen H, Li T, Yang X. A stable community detection approach for complex network based on density peak clustering and label propagation. Appl Intell 2022; 52(2):1188–208.

[33] Berahmand K, Haghani S, Rostami M, Li Y. A new attributed graph clustering by using label propagation in complex networks. J King Saud Univ-Comput Inf Sci 2022;34(5):1869–83.
[34] Roghani H, Bouyer A, Nourani E. PLDLS: a novel parallel label diffusion and label selection-based community detection algorithm based on spark in social networks. Expert Syst Appl 2021:115377. https://doi.org/10.1016/j.eswa.2021.115377 (/06/11/2021).
[35] Zhang J, Ding X, Yang J. Revealing the role of node similarity and community merging in community detection. Knowl-Based Syst 2019;165:407–19.
[36] Berahmand K, Bouyer A, Vasighi M. Community detection in complex networks by detecting and expanding core nodes through extended local similarity of nodes. IEEE Trans Comput Soc Syst 2018;5(4):1021–33.
[37] Sun Z, et al. Community detection based on the Matthew effect. Knowl-Based Syst 2020;205:106256.
[38] Parés F, et al. Fluid communities: a competitive, scalable and diverse community detection algorithm. In: International conference on complex networks and their applications. Springer; 2017. p. 229–40.
[39] You X, Ma Y, Liu Z. A three-stage algorithm on community detection in social networks. Knowl-Based Syst 2020;187:104822.
[40] Masoudi-Sobhanzadeh Y, Omidi Y, Amanlou M, Masoudi-Nejad A. Drug databases and their contributions to drug repurposing. Genomics 2020;112(2):1087–95.
[41] Groza V, Udrescu M, Bozdog A, Udrescu L. Drug repurposing using modularity clustering in drug-drug similarity networks based on drug–gene interactions. Pharmaceutics 2021;13(12) (pp. 2117 %@ 1999-4923).
[42] Udrescu L, et al. Clustering drug-drug interaction networks with energy model layouts: community analysis and drug repurposing. Sci Rep 2016;6(1) (pp. 32745 %@ 2045-2322).
[43] Sia J, Jonckheere E, Bogdan P. Ollivier-ricci curvature-based method to community detection in complex networks. Sci Rep 2019;9(1):9800.
[44] Koss J, Bohnet-Joschko S. Social media mining of long-COVID self-medication reported by Reddit users: feasibility study to support drug repurposing. JMIR Form Res 2022;6(10):e39582.
[45] Sun PG, Miao Q, Staab S. Community-based k-shell decomposition for identifying influential spreaders. Pattern Recogn 2021;120:108130.
[46] Laghridat C, Essalih M. A set of measures of centrality by level for social network analysis. Proc Comput Sci 2023;219:751–8.
[47] Hirsch JE. An index to quantify an individual's scientific research output. Proc Natl Acad Sci 2005;102(46):16569–72.
[48] Cao C, Chen Z, Caverlee J, Tang L-A, Luo C, Li Z. Behavior-based community detection: Application to host assessment in enterprise information networks. In: *Proceedings of the 27th ACM international conference on information and knowledge management*, 2018, pp; 1977-1985.
[50] Lusseau D, Schneider K, Boisseau OJ, Haase P, Slooten E, Dawson SM. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. Behav Ecol Sociobiol 2003;54(4):396–405.
[51] CDLIB Community Discovery Algorithms. https://cdlib.readthedocs.io/en/latest/reference/cd_algorithms/node_clustering.html (accessed).
[52] SNAP project. http://snap.stanford.edu/data/index.html; 2020 (accessed).
[53] Lancichinetti A, Fortunato S, Radicchi F. Benchmark graphs for testing community detection algorithms. Phys Rev E 2008;78(4):046110.
[54] Danon L, Diaz-Guilera A, Duch J, Arenas A. Comparing community structure identification. J Stat Mech Theor Exp 2005;2005(09):P09008.
[55] Implementation of Infomap algorithm. https://github.com/mapequation/infomap (accessed).
[56] Infomap project description. https://pypi.org/project/infomap/ (accessed).
[57] Jarada TN, Rokne JG, Alhajj R. A review of computational drug repositioning: strategies, approaches, opportunities, challenges, and directions. J Chemother 2020;12:1–23.
[58] Kinsley AC, Rossi G, Silk MJ, VanderWaal K. Multilayer and multiplex networks: an introduction to their use in veterinary epidemiology. Front Vet Sci 2020;7:596.
[59] Wishart DS, et al. DrugBank 5.0: a major update to the DrugBank database for 2018. Nucleic Acids Res 2018;46(D1):D1074–82.
[60] Yıldırım MA, Goh K-I, Cusick ME, Barabasi A-L, Vidal M. Drug–target network. Nat Biotechnol 2007;25(10):1119–27.
[61] Jaccard P. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull Soc Vaudoise Sci Nat 1901;37:547–79.
[62] Bigraph Link Prediction Package. https://github.com/bi-graph/Bigraph (accessed).
[63] Lo Y-C, Torres JZ. Chemical similarity networks for drug discovery. Special Top Drug Discov 2016;1:53–70.
[64] Maggiora G, Vogt M, Stumpfe D, Bajorath J. Molecular similarity in medicinal chemistry: miniperspective. J Med Chem 2014;57(8):3186–204.
[65] Kim S, et al. PubChem 2019 update: improved access to chemical data. Nucleic Acids Res 2019;47(D1):D1102–9.
[66] Landrum G. Rdkit: Open-source cheminformatics software. 2016.
[67] Bajusz D, Rácz A, Héberger K. Why is Tanimoto index an appropriate choice for fingerprint-based similarity calculations? J Chemother 2015;7(1):1–13.
[68] Kanehisa M, Goto S, Furumichi M, Tanabe M, Hirakawa M. KEGG for representation and analysis of molecular networks involving diseases and drugs. Nucleic Acids Res 2010;38, no. suppl_1:D355–60.
[69] Ban Y, Liu Y, Yin Z, Liu X, Liu M, Yin L, Zheng W. Micro-Directional Propagation Method Based on User Clustering. Comput Inform 2024;42(6):1445–70. https://doi.org/10.31577/cai_2023_6_1445.
[70] Piñero J, et al. The DisGeNET knowledge platform for disease genomics: 2019 update. Nucleic Acids Res 2020;48(D1):D845–55.
[71] N. I. o. Diabetes, Digestive, and K. Diseases. LiverTox: clinical and research information on drug-induced liver injury. National Institute of Diabetes and Digestive and Kidney Diseases; 2012.
[72] Leuprolide. https://www.drugs.com/leuprolide.html (accessed).
[73] Degarelix. https://go.drugbank.com/drugs/DB06699 (accessed).
[74] Gonadorelin. https://www.mayoclinic.org/drugs-supplements/gonadorelin-intravenous-route-injection-route/description/drg-20067426 (accessed).
[75] Nafarelin. https://go.drugbank.com/drugs/DB00666 (accessed).
[76] Triptorelin. https://go.drugbank.com/drugs/DB06825 (accessed).
[77] Ganirelix. https://www.rxlist.com/ganirelix-drug.htm#description (accessed).
[78] Alba-Delgado C, et al. The function of alpha-2-adrenoceptors in the rat locus coeruleus is preserved in the chronic constriction injury model of neuropathic pain. Psychopharmacology 2012;221:53–65.
[79] Alba-Delgado C, Llorca-Torralba M, Mico JA, Berrocoso E. The onset of treatment with the antidepressant desipramine is critical for the emotional consequences of neuropathic pain. Pain 2018;159(12):2606–19.
[80] Mianserin. https://go.drugbank.com/drugs/DB06148 (accessed).
[81] Dosulepin. https://go.drugbank.com/drugs/DB09167 (accessed).
[82] Maprotiline. https://go.drugbank.com/drugs/DB00934 (accessed).
[83] Foster RH, Goa KL. Paroxetine: a review of its pharmacology and therapeutic potential in the management of panic disorder. CNS Drugs 1997;8(2):163–88.
[84] He H, Li X, Chen P, Chen J, Liu M, Wu L. Efficiently localizing system anomalies for cloud infrastructures: a novel Dynamic Graph Transformer based Parallel Framework. J Cloud Comput 2024;13(1):115. https://doi.org/10.1186/s13677-024-00677-x.