A Fast Local Balanced Label Diffusion Algorithm for Community Detection in Social Networks

局部平衡标签扩散

Hamid Roghani and Asgarali Bouyer

Abstract—Community detection in large-scale networks is one of the main challenges in social networks analysis. Proposing a fast and accurate algorithm with low time complexity is vital for large-scale networks. In this paper, a fast community detection algorithm based on local balanced label diffusion (LBLD) is proposed. The LBLD algorithm starts with assigning node importance score to each node using a new local similarity measure. After that, top 5% important nodes are selected as initial rough cores to expand communities. In the first step, two neighbor nodes with highest similarity than others receive a same label. In the second step, based on the selected rough cores, the proposed algorithm diffuses labels in a balanced approach from both core and border nodes to expand communities. Next, a label selection step is performed to ensure that each node is surrendered by the most appropriate label. Finally, by utilizing a fast merge step, final communities are discovered. Besides, the proposed method not only has a fast convergence speed, but also provides stable and accurate results. Moreover, there is no randomness as well as adjustable parameter in the LBLD algorithm. Performed experiments on real-world and synthetic networks show the superiority of the LBLD method compared with examined algorithms.

Index Terms—Social networks, local community detection, Balanced label diffusion, local similarity

1 Introduction

The rapid growth of online social media has made the need of fast algorithms greater than ever for analyzing their information. In the massive data of these complex networks, in addition to accurate analysis, the execution speed is vital particularly for social network analysis [1], biological analysis [2] (e.g., protein-protein network), recommender systems [3], citation networks [4], and etc. Social media analysis has attracted special attention in the recent decade. In these networks, people are represented as nodes and their relations are shown as links or edges where most of the nodes are arranged in dense parts that are called communities or clusters. Each community is a group of nodes that have high density relationships inside themselves, but they have low connections with the rest of the network [5]. Uncovering communities helps to analyze the structure of communities, the role of nodes and their relations, and how the networks work in detailed [6]. However how adequate a community detection algorithm is, in terms of accuracy and time complexity in large-scale networks, still remains open.

Global methods need to consider information of all nodes in community discovering which makes them inefficient on large-scale networks due to their high time complexity. Local approaches try to utilize the local information of nodes such as degree, number of common neighbors, local paths, and local similarity measures with suitable efficiency on large-scale networks. The main idea behind local methods is defining local metrics and forming local communities by adding nodes and expanding the local communities. Some of the methods are based on label propagation of important nodes [7]–[9] whereas some others are based on identifying core nodes and expanding the local communities from inner side to outer side [6], [10], [11]. Since initial communities are strongly depended on identifying core nodes, these methods severely depend on accurate core detection techniques. Label diffusion is a new fast technique to expand initial communities from border nodes of communities to inside of them. It does not have issues such as identifying core nodes, assigning initial labels and some drawbacks of label propagation methods as well as improving execution speed of the algorithm with proper accuracy [12].

On the other hand, modularity-based methods [13], [14] only focus on the density of links and maximizing the gained modularity. However, the goal of community detection is not always maximizing the modularity, but also the aim is to find the real structure of the communities. In other words, community detection is not just a simple partitioning of a graph that ignores different roles of nodes [15]. In addition, label propagation-based algorithms as well as modularity-based methods provide unstable results on different executions. As mentioned above, some methods start community detection from core nodes whereas some of them [16] believe that discovering boundary nodes can help to surrender borders of communities to better identify communities. However, such methods are sensitive to identifying boundary nodes and since boundary nodes usually have overlapping nature, it is difficult to make decisions on assigning them to a distinct community. Such methods also have low accuracy with instable results.

• *The authors are with the Department of Information Technology and Computer Engineering, Azarbaijan Shahid Madani University, Tabriz 53751-71379, Iran. E-mail: {h.roghani, a.bouyer}@azaruniv.ac.ir.*

Manuscript received 18 Mar. 2021; revised 6 Mar. 2022; accepted 18 Mar. 2022. Date of publication 25 Mar. 2022; date of current version 1 May 2023.
(Corresponding author: Asgarali Bouyer.)
Recommended for acceptance by P. Tsaparas.
Digital Object Identifier no. 10.1109/TKDE.2022.3162161

Besides aforementioned issues, handling the big amount of data and proposing a community detection algorithm with acceptable quality and low time complexity is crucial for large-scale networks. Even though using local metrics is beneficial in reducing the time complexity, but designing fast algorithms that do not involve time consuming computations and have fast convergence with the required accuracy of results, is the most important effort. Local methods such as LPA [17], despite of its simple nature, can be time consuming on massive networks due to low convergence speed. Furthermore, some other methods such as Louvain [13] and Leiden [14] are faster than LPA, but their accuracy are not satisfactory. Most of these methods such as spectral and modularity-based approaches succeed when the network is sufficiently dense [18], but when communities of a network have a relatively low density and the number of external edges between communities is somewhat high, their quality significantly becomes lower than others [19], [20]. As a conclusion, due to the time complexity issues, only a small minority of community detection algorithms are capable to process large-scale networks in a reasonable time. To overcome these shortcomings, a local and fast algorithm with low time complexity and satisfactory quality on massive networks is proposed in this paper.

In the proposed method, balanced label diffusion is presented to address the label assigning problem of nodes. In this way, the communities are formed on both sides of the core nodes (inner side) as well as border nodes (outer side). Furthermore, to improve accuracy and stability of the algorithm, a new simple strategy is employed to discover rough cores (seed nodes of communities) without any time-consuming operations to form the main structure of communities. Finally, a fast merge method is adopted to merge small communities. The major contributions of this paper are summarized as follows:

- ✓ A new rough core detection method is proposed to uncover initial local communities to improve the stability and accuracy of the algorithm.
- ✓ A fast balanced label diffusion method with low time complexity is employed to form communities from both inner and outer sides to take advantage of both core-based and border-based methods.
- ✓ A new and fast merge method is adopted to obtain dense communities as far as possible.
- ✓ A new and efficient data structure is used to solve the memory problem of handling massive datasets.
- ✓ The effectiveness of the proposed method on small and large-scale networks is proved by several experiments on real-world and synthetic networks.

The rest of the paper is arranged as follows: in Section 2, related works are explained. In Section 3, details of the proposed method are discussed. Section 4 demonstrates several experiments to evaluate the accuracy and effectiveness of the proposed method in comparison with other algorithms. Finally, Section 5 concludes the proposed method and presents the future directions.

2 Related Work

Because of the importance of community detection as an essential tool for analyzing hidden information of networks, a wide range of algorithms with different features are studied. Recently, the main focus of researchers is on developing local methods, modularity-based methods, and non-negative matrix factorization-based methods [21]. Authors in [22] have used modularized non-negative matrix factorization and network structure embedding for community detection. Cavallari *et al.* proposed a novel community embedding framework for improving node embedding and community embedding for an efficient community detection [23]. Jia *et al.* suggested CommunityGAN, as a novel community detection framework that jointly solves overlapping community detection and graph representation learning [24]. Despite the novelty of the non-negative matrix factorization-based methods, most of new methods focuses on local heuristic to propose a low-time complexity algorithm.

Raghavan *et al.* proposed LPA [17] for the first time in 2007. In the LPA, by adopting a random order of nodes, each node iteratively updates its label to the label with highest frequency. But existence of random behavior in initial selection of nodes and tie break states makes the accuracy of the LPA instable and turns it into an unreliable method. Despite of its $O(m+n)$ time complexity, LPA could be inefficient on massive networks. K-shell decomposition and Bayesian network are two instances to solve the drawbacks of the LPA by assigning importance to nodes to specify a constant order of nodes which are proposed in [7], [8], respectively. Another LPA-based algorithm was proposed based on considering two semi-local concepts, called nodes and links strength concepts. In this algorithm a new approach was proposed for selecting initial nodes and label updating method to solve the problem of random behavior of the original LPA [25]. LP-LPA is another method for discovering communities by measuring the link strength value for edges and label influence value for nodes to avoid of random selection in tiebreak states. However, it needs a sorting step before starting label updating step [9]. Wang *et al.* proposed a node importance-based LPA (NI-LPA) [26]. They have combined Jaccard index, K-shell number, and signal propagation of nodes to assign more accurate importance score to nodes. Consequently, LPA-based methods are mostly based on defining importance metrics, sorting nodes based on their importance, and defining measures for tie break states. However, there are still problems such as accuracy, increasing time-complexity, and instability in most of the LPA-based algorithms.

The other class of local methods are based on discovering core nodes and expanding the community from inner to outer side. RTLCD, proposed by Ding *et al.* is a local method based on identifying core nodes [11]. In this approach after uncovering core nodes, local methods are adopted to expand communities. Berahmand *et al.* introduced ECES, a method for finding core nodes and expanding communities using local membership criterion [6]. CFCD, a core fitness based algorithm was proposed by Zhang *et al.* [27] which detects seed nodes of communities and expands them. Sun *et al.* proposed CDME, a core node expansion method based on Matthew effect for community detection [28]. LCDR is another local method based on ranking high importance nodes using Kirchhoff's idea or charge conservation [29]. Authors in [30] proposed a propagation based method

called FluidC that is inspired by the model of fluids expanding. Similar to CenLP algorithm [31], Tasgin *et al*. presented GCN method which is based on updating boundary nodes label [16]. Authors attempt to uncover communities by utilizing a benefit function and identifying border nodes. This method is based on a fast label propagation method but suffers from low accuracy and formation of monster communities. Zarezadeh *et al*. proposed an improved version of the GCN, which used a hybrid node scoring and synchronous label updating to improve the accuracy problem of the GCN [32]. Gujral *et al*. proposed a hierarchical agglomerative community detection (HACD) algorithm which integrates the local information in a graph with membership propagation to improve the efficiency of traditional hierarchical methods [33]. APAS is a new affinity propagation-based algorithm that detects communities by utilizing an adaptive asymmetric similarity matrix which could efficiently show the leadership probabilities between data points [34]. Authors in [35] proposed a compactness function to improve label propagation. After identifying the core nodes, weights are assigned to other nodes based on cores where label of nodes with higher influence will have higher priority which improves the propagation step.

Bouyer *et al*. by proposing the LSMD algorithm, showed that without finding core nodes and only by starting from low degree nodes, accurate communities can be obtained [12]. In this method, after grouping nodes based on their degrees, label assigning is performed in a multi-level diffusion way starting from lowest degree nodes and continuing to other levels of neighbors based on required condition.

Modularity-based methods are the other class of community detection algorithms that formulize the community detection problem into an optimization problem. Newman *et al*. suggested one of the first modularity-based algorithms in 2004 [36]. In this method, each pair of nodes resulting in highest modularity gain, are merged into one group. Clauset *et al*. proposed CNM algorithm, a modularity-based hierarchical agglomerative approach for community detection [37]. Later, an improved and fast method called Louvain was suggested by Blondel et al [13]. This heuristic approach consists of forming initial communities by merging nodes and initial communities to obtain higher modularity gain. Leiden [14] is a refined version of the Louvain algorithm to improve the drawbacks of the Louvain method and guaranties to obtain well connected communities. By adopting local moving strategy in Leiden, the execution speed is improved. Shang *et al*. proposed a new method by adopting node membership and community merging steps in [38]. At first, initial communities are identified and then are integrated with each other to form larger communities. In [39] authors proposed TJA-net, which combines k-nearest neighbors and the LPA algorithm to obtain initial communities and by adopting mutual membership of two communities they merge communities.

On the other hand, evolutionary methods are also used for community detection problem by adopting modularity as objective function. In [40] Lyu *et al*. suggested ELCD that uses BPSO to optimize the local communities. Authors in [41] presented a multi-objective algorithm based on discrete particle swarm optimization to improve optimization quality. Zhang *et al*. proposed RMOEA [42], a network reduction-based multi-objective evolutionary method to reduce the search space complexity of the previous works. However, it is concluded that evolutionary methods generally have high time and space complexity and their performance strongly degrades as the size of the network increases.

3 Proposed Algorithm

In this section, the proposed method is explained in details. The different parts of the proposed LBLD algorithm are presented in the following sub-sections.

Algorithm 1. LBLD Algorithm

Input: Neighboring list of nodes
Output: Final communities

1. Exclude nodes with degree 1 from step 2 until step 7.
2. Calculate nodes importance using Eq. (2) and sort them descending.
3. Initialize each node with label "0".
4. *Label_counter* ←1 and *diffusion_flag*← 0
5. Repeat step 5 for nodes with degree more than 1
   Select node i and its most similar neighbor j
   If ($similarity(i,j) == 0$) **Then** select neighbor with highest degree and assign same label to them
   Else if ($NI(i) > NI(j)$) Then Label (i) ← *Label_counter*
   Else Label (i) ← *Label_counter*, Label (j) ← *Label_counter*
   Label_counter ← *Label_counter*+1
6. **Select** 5% of nodes with *highest* importance and diffuse their label to the most similar common neighbors.
   For nodes participated in this step *diffusion_flag*←1
7. Perform balanced label diffusion step for nodes if *diffusion_flag* = 0
8. ***For all*** degree 1 nodes, assign neighbor's label to them
9. **Repeat** step 9 for all nodes (*label selection step*)
   - Select the most common community's label
   - Use Eq. (5) for tie break state.
10. Run fast **Merge Phase** (*Sub-section E*)
    end

Algorithm 1 shows the main steps of the LBLD algorithm. $similarity(i,j)$ shows the similarity between node i and node j which is computed using Eq. (1). $NI(i)$ denotes the node importance of node i which is calculated via Eq. (2) and $Label(i)$ denotes the label of node i. Since labels are generated, a variable such as *Label_counter* is used to generate labels by incrementing its value by 1 each time it is needed and "*diffusion_flag*" variable is used to show whether a node has diffused its label or not. "*Label_counter* ←1" is used for assigning a value to a variable. To better understand the steps of the algorithm the address to the source code of the LBLD algorithm is provided in Appendix A, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2022.3162161.

At the beginning of the algorithm, all of the nodes are initialized with label "0". Then, the labeling is started from label "1" for label diffusion as far as possible. When a new label is needed to be created, a new numerical label such as "2", "3", and so forth is generated to diffuse to the other group of nodes. Therefore, at each time a new label is

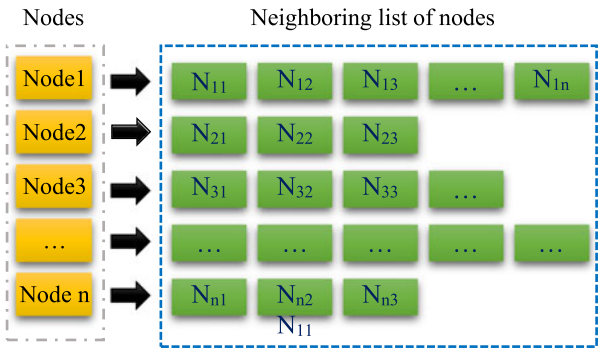


Fig. 1. Data structure used for implementing datasets. The keys are nodes of the graph and the value of each key is a list of neighboring nodes.

generated as: *New_label* = *current_label*+1. In addition, *diffusion_flag* is set to zero for every node. This flag indicates which nodes have diffused their labels to the other nodes. For a node *i* if *diffusion_flag* = 1, it means that the node *i* has previously participated in diffusion process and it cannot participate in balanced label diffusion step, again. In other words, each node can participate in the label diffusion step only once. Needless to mention that nodes with degree equal to 1 are not participated in any of the steps of the algorithm and only after performing balanced label diffusion step, nodes with degree 1 pick label from their neighbor at final steps. Since in social networks there are a lot of nodes with degree 1, this action saves computing time of the label diffusion process.

In this method, a combination of several concepts is used to propose an efficient algorithm. To design an efficient algorithm, initially a community detection algorithm should be imagined as a system that consists of different parts. So, each part does its task to provide efficient results for the system as a whole unit. This can be done by designing efficient data structures, new equations and efficient design of the steps of the algorithm. In this study, a neighboring list of nodes is applied which gives advantages of fast accessing to the neighbors of nodes and low usage of RAM space. Neighboring list of a network can be imagined as a linked list structure or dictionary structure in Python language that the key is the node and the list is all of the neighbors of that node. The length of the list for each key is equal to the degree of that node. Fig. 1 shows the adopted data structure for implementation of the LBLD algorithm.

Properties and dynamics of complex networks are studied to propose an efficient algorithm from different aspects. It is assumed that by performing community detection in the same way that communities are emerged in the real world, communities can be obtained much similar to the real communities. Inspired by this fact, first steps are designed to construct initial and rough cores of communities which gives stability and robustness to the algorithm and reduces labels variations in further steps. It has been also considered how new nodes are connected to the other nodes and the balanced label diffusion step is proposed. In the natural world, new individuals do not always attach to important elements, but although some of them are attached to low importance parts and some others are attached to the important individuals.

3.1 Step 1 of Label Diffusion: Select Most Similar Neighbor

In this step of the proposed method, using the power-law [5] and scale-free [43] features in complex networks, nodes are initially grouped with their most similar direct neighbors without considering the position of nodes and without any defined order. In a small-world network, neighbors of any given node are likely to be neighbors of each other. These nodes tend to be in a same community, because they are close to each other. Selecting the most similar neighbor means two nodes share some common neighbors. The nodes located in dense parts of the communities have more links with neighbors around them. Since these nodes are popular nodes, they have more relations with neighbors of their neighbors that shows they share more common interests. Nodes with high popularity are more capable of controlling other nodes and easily diffusing label to them. To summarize, the node that has higher similarity with its neighbors, can have more impact on its neighbors. Similarity relation is defined as follows.

$$Similarity_{i,j} = \frac{\left|N_i \cap N_j\right|}{\left|N_i \cap N_j\right| + \left|N_i \cup N_j\right|} \times \frac{\left|N_i \cap N_j\right|}{1 + |N_s - N_h|} \quad (1)$$

Where i and j are two adjacent nodes. N_i and N_j show neighbors of nodes i and 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. Node importance of node i ($NI(i)$) is calculated using (2).

$$NI\ (i) = \sum_{j \in N_i} similarity_{i,j} \quad (2)$$

The higher $NI(i)$ value, the more important a node is. Nodes are then sorted descending based on their importance value. It must be noted that Eq. (2) is not computed for nodes with degree 1 since each of them is only connected to one neighbor node and will receive its label in the last step and will not participate in diffusion steps. To prevent the process of repeatedly updating label of nodes, each node initially selects its most similar neighbor and at the end of this step, if there are changes in the label of nodes, a label updating process is performed for all nodes.

In this step, if the importance of selected neighbor is greater than the importance of the current node, then the current node and the neighbor node both will receive the same label, otherwise only the current node will receive a new label. If the similarity between selected node and its most similar neighbor is equal to zero, then the selected node will pick the label of the neighbor with the highest degree. In case of having label "0", a new label is generated and will be assigned to the selected node and its selected neighbor. By this way, the label of the important nodes is diffused to the nodes around them and nodes are gathered around important nodes. At the end of this step, except nodes with degree 1, all nodes have a label. Fig. 2 shows two consecutive operations of selecting the label of the most similar neighbor and then updating the label of nodes. Similarity scores between two neighbor nodes are written on the edges. Nodes with degree 1 are not labeled until last step. In Fig. 2a, nodes 0, 1, and 3 select the label of the most similar

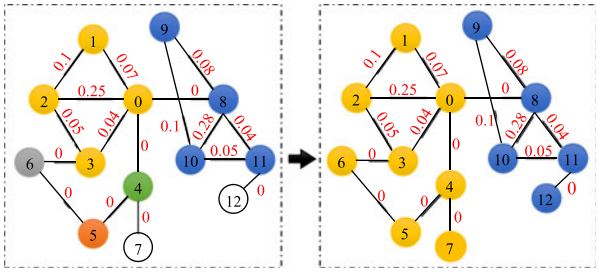


Fig. 2. Diffusing label between nodes and their most similar neighbors. (a) Selecting label of the most similar node. (b) Updating label allocation.

neighbor (node 2 is the most similar neighbor of nodes 0, 1, and 3). Initially, Node 4 selects label of node 0, node 5 selects label of node 4, and node 6 selects label of node 3 and they are shown with different colors. In (b) after updating label allocation order, nodes are assigned with the true labels. At last step, node 7 receives label of node 4 and label of node 11 is assigned to the node 12.

3.2 Step 2 of Label Diffusion: Find Rough Cores (Diffusion to Common Neighbors)

Before executing balanced label propagation step, it is needed to stabilize the main structure of the communities and labels of nodes to increase the probability of selecting right label in the next steps and to reduce the variety of the labels. After finishing this step, no new labels are generated which maintains the current state of the communities and prevents the algorithm from generating new wrong labels and further steps try to expand current communities and are considered as refinement steps. Having an initial knowledge about the communities before starting to diffuse labels to the whole of the network, would be beneficial and prevents the algorithm from the cold start situation which the structure of the communities is unknown and the algorithm should put much more effort on selecting right labels for detecting communities. Constructing initial rough cores plays a vital role in robustness and accuracy of the algorithm. In this step, a new method for identifying rough cores of communities is presented by analyzing the importance of nodes to create the main skeleton of the communities without any complex computations. Inspired by triadic closure property in networks [44], nodes forming triangles and connected nodes to these triangles show more tendency to be in the same community. Also, based on discussions in [12], [45], diffusing label to a group of nodes at once can be very applicable in reducing redundant checks of the algorithm and can significantly improve performance and execution time of the algorithm. Furthermore, at first steps of the algorithm, initial diffusion of labels by nodes to their similar neighbors with strong connections can improve the reliability in assigning label to other nodes in the next steps. This concept is used to diffuse label to all nodes in a triangle structure between nodes to select appropriate rough cores.

In this step, five percent of nodes with highest node importance value are selected as potential cores. For each of the selected core nodes, label of selected node and its most similar neighbor are assigned with same label. Also, their common neighbors receive label of the selected core node. Finally, the diffusion flag of the selected node, most similar neighbor, and all of their common neighbors will change to 1. This step is executed for all of the 5% selected rough cores. After finishing this step, almost the main skeleton of the communities is built. These communities with strong connections, have a significant impact on accuracy of label selection in the next steps.

It should be noted that by proposing this method, the aim is not to find all of the possible kernels, since the LBLD algorithm is not a method based on finding and expanding core nodes. Through experimental observations, it is found that five percent of nodes is enough to form desired communities with acceptable accuracy. In other words, the goal is not to completely identify kernel of communities, but it is to approximately find some rough cores to create skeleton of the communities to use them in the next steps of the algorithm. After constructing rough cores, balanced label diffusion is performed to expand initial communities from inner (by rough cores) and from outer side (by border nodes) to make more accurate and stable communities.

3.3 Balanced Label Diffusion

Updating order of the nodes significantly affect the formation of the communities. If the priority is given only to expansion of the label of the communities from the inner side to the outer side, the risk of incorrect label diffusing may increase. It also can result in forming monster communities by expanding the strong communities and merging small groups around it. A balanced approach is used to expand the communities from both inner and outer sides to prevent the formation of monster communities. Identifying border nodes is important to control the growth of the communities. It was confirmed that by starting from low degree nodes, accurate and stable community structures can be obtained [12]. However, in the proposed balanced diffusion method, a community not only is expanded from cores, but also it is grown from the border nodes sides. It can result in forming communities faster without detecting monster communities. Besides, since the initial structure of the communities is formed in the previous steps, it helps to diffuse labels of the communities more easily from the inner nodes to the outer nodes and wise versa.

This step is only performed one time for nodes with diffusion $flag = 0$. First, these nodes are sorted based on their importance (NI). Next, a new node is selected from the top of the list (with highest NI) and then another node is selected from the bottom of the list. In other words, one node is popped from the top of the list and the other node is popped from the bottom of the list. Each selected node changes its label to the most important label based on its neighbors. After updating label of these two nodes, the balance label diffusion method selects other two nodes from the top and bottom of the list. This selection procedure continues until there are no nodes remained in the list. Eq. (3) is used to update label of node i if it is a high importance node (selected from the top of the list) and Eq. (4) is used if node i is a low importance node (selected from the bottom of the list). Therefore, the importance of a community around node i is calculated by the equations (3) and (4):

$$C_{importance}\ (i, C_x) = \sum_{j \in N_i} NI(j)\ \ ,\ j \in C_x \tag{3}$$

$$C_{importance}\ (i, C_x) = \sum_{j \in N_i}^{\circ} (i) \times deg(j)\ \ ,\ j \in C_x \tag{4}$$

Where $C_{importance}$ indicates community importance of the community C_x, N_i is the neighboring set of node i, $NI(j)$ is node importance of node j, $\deg(j)$ shows degree of node j, and j belongs to the neighboring list of node i. These two equations are executed for all of the neighbors of node i which are located in different communities. Accordingly, the label of the community with highest score is assigned to node i. Needless to mention that after finishing this step, nodes with degree equal to 1 pick the label of their only neighbor. In other words, each degree 1 node only has one neighbor and it receives the label of its neighbor.

3.4 Label Selection Phase

Even though the previous steps attempt to discover accurate communities as far as possible, but the several labels may have been assigned incorrectly in the previous steps. Label selection phase is executed to ensure that each node is surrendered with the most suitable label. Since previous steps put a lot of effort on assigning appropriate labels to nodes, the first factor for updating is the frequency factor. Therefore, the selected node primarily updates its label to the label of the community with highest frequency among the neighbors. If multiple communities share the same frequency, then the effectiveness of each community is calculated by Eq. (5) to assign a label of community with highest $Effectiveness_{C_m}$.

$$Effectiveness_{C_m} = \prod_{j \in N_i} NI(j)\ ,\ j \in C_m \tag{5}$$

Where C_m is one of the communities with maximum frequency, and N_i is the neighbors of node i which have the label of community C_m and j is one of the selected neighbors of node i. Finally, the label of the community with highest effectiveness value is chosen for node i. This step is repeated once again in some conditions. In the proposed LBLD algorithm since each of the steps have a significant role on accuracy and robustness of the algorithm, and steps can correct slight mistakes of the previous steps, there is no need to execute the algorithm with a greater number of iterations which results in fast convergence of the proposed algorithm. In synthetic networks with very dense communities, executing this step with two iterations is enough to obtain stable and accurate results.

3.5 Fast Merge Phase

In community detection algorithms merge steps are executed to integrate some small and weak communities with appropriate communities around them. In fact, these small communities are part of the other communities but the algorithm has mistakenly isolated them. For this purpose, an efficient and fast approach is proposed to merge small communities. Unlike some other methods that use modularity measure with high time-consuming calculations for merging, the proposed merge method fully focuses on negotiations between important nodes to merge communities with ignoring many redundant operations. Merging communities at once and avoiding of checking each of the nodes individually, saves a lot of execution time for the LBLD algorithm. The merging steps are meaningful from the aspect that in real world, when leaders of two nations or groups agree on a subject, it means that other individuals are agreed too. Inspired by this fact the fast merge method is proposed. This step works as follows:

By excluding the largest community, the average size of the remained communities is calculated. Then, communities with size less than the average value are considered as small communities. Since the proposed merge step is based on comparing representative nodes of two communities, for each of the small communities, Eq. (6) is calculated to identify representative nodes.

$$RS_i = \ deg(i) + NI(i) \tag{6}$$

Where RS_i indicates the score of node i. The node with highest RS_i value is selected as representative node of its community. Then, a neighbor node j of representative node i with different label and highest RS_j score which has higher degree than the selected representative node, assigns its label to the selected representative node i and consequently two communities get merged at once. Otherwise, these two communities are not merged. Despite of simplicity, this merge method significantly is capable and fast.

3.6 Time Complexity Analysis of the LBLD Algorithm

The proposed algorithm consists of six main parts. At first, n_1 nodes with degree 1 are excluded and the algorithm is executed for the nodes with degree higher than 1. Therefore, n' nodes out of n nodes in the network are participated in the next steps of the algorithm, where $n' = \ n - n_1$. For n' nodes with average degree k, the time complexity for calculating nodes similarity is equal to $O(\frac{n'k^2}{2})$. Since $similarity_{i,j} = similarity_{j,i}$, the denominator is divided by 2. Time complexity for sorting n' nodes is proportional to $O(n' \log n')$. For n' nodes with average k neighbors, selecting most similar neighbor has $O(n'k)$ time complexity. At the next, for P selected nodes, the LBLD algorithm diffuses label to common neighbors which has $O(Pk^2)$ computational time, where $P\ = \ 0.05n'$, which is a very small number. Also, n'' is a part of n' nodes which participate in balanced label diffusion step where $n'' < n'$. So, this step has $O(n''k)$ time complexity. Finally, all of the degree 1 nodes are assigned with labels in $O(n_1)$. Therefore, the total time complexity for finishing the label selection step is $O(nk)$. In merge section, finding size of the communities needs $O(n)$ time complexity. For s small communities with n_s members, representative nodes are determined in $O(sn_s) < O(n)$. Total time complexity for merge is: $O\ (n + sn_s) = \ O(n)$. To conclude, total time complexity of the LBLD algorithm is: $O(nk)$.

4 Experiments

4.1 Experimental Settings

In this section, various experiments are applied on several real-world and synthetic networks to compare the

TABLE 1
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	14490	-
Collaboration	8361	15751	-
CA-HEPTH	9877	25985	-
PGP	10680	24316	-
Condmat-2003	31163	120029	-
Condmat-2005	40421	175691	-
Brightkite	58228	214078	-
DBLP	317080	1049866	13477
Amazon	334863	925872	75149
YouTube	1134890	2987624	8385
Orkut	3,072,441	117,185,083	6288363
LiveJournal	3997962	34681189	287512

performance of the proposed LBLD algorithm with the other popular and new algorithms. The LBLD method was programmed with Python 3.7. In addition, for CNM [37], Infomap [4], LPA [17], FluidC [30], Louvain [13], and Leiden [14] algorithms their Python implementations (CDLIB [46]) were used for experiments. From four different versions of the CFCD algorithm, CFCD2 is used as the strong version [27]. Also, results for RTLCD, CDME, and TJA-net algorithms are taken from [11], [28], and [39] respectively. Detailed information about the algorithms and their original implementation is presented in [47]. Moreover, the LSMD is implemented in MATLAB 2017. All the experiments are performed on a computer system with Core i5 processor (3.40-3.70 GHz), 12 GB RAM, and Python 3.7 on a Windows 10 operating system.

4.2 Datasets

To compare the performance of algorithms, 18 real-world networks are considered. Zachary's Karate club network with 34 nodes and LiveJournal with 3997962 nodes are the smallest and the largest used datasets, respectively. Details of real-world datasets are summarized in Table 1 where *N*, *M*, and *C* represent the number of nodes, number of edges, and number of communities, respectively. Detailed description of real-world datasets are available on SNAP project [48]. Likewise, to expand experiments LFR benchmark is used as synthetic networks [49]. The setting of LFR networks is shown in Table 2. Finally, Table 3 shows the different configurations of generated LFR networks.

4.3 Evaluation Metrics

Normalized mutual information (NMI), F-measure, and Modularity (Q) are three metrics that are used in this paper to evaluate the accuracy and quality of algorithms.

NMI [50] is a widely used measure to evaluate the accuracy of the obtained partitions where the ground-truth of the network is available. It measures the similarity between detected communities and real communities which is defined as follows.

TABLE 2
Parameters of 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
$\boldsymbol{\gamma}$	Node degree distribution exponent
$\boldsymbol{\beta}$	Community size distribution exponent

$$NMI\ (X,Y) = \frac{-2\sum_{i=1}^{C_X}\sum_{j=1}^{C_Y} C_{ij} log\left(\frac{C_{ij}N}{C_i C_j}\right)}{\sum_{i=1}^{C_X} C_i \log\left(\frac{C_i}{N}\right) + \sum_{j=1}^{C_Y} C_j \log\left(\frac{C_j}{N}\right)} \tag{7}$$

Where X shows the real communities and Y is a group of nodes identified by the algorithm. C_X shows 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}.

F-measure is another widely used metric to evaluate ground-truth datasets. It is defined as the harmonic mean of precision and recall [27]. This measure is mainly used in classification problems to measure the accuracy of methods in distinguishing true results from wrong ones. For a community, F-measure is defined as (10).

$$Precision\ = \frac{|C_D \cap C_R|}{|C_D|} \tag{8}$$

$$Recall\ = \frac{|C_D \cap C_R|}{|C_R|} \tag{9}$$

$$F - measure\ = \frac{2 \times precision \times recall}{precision + recall} \tag{10}$$

C_D indicates detected communities by the proposed method and C_R is the ground-truth communities which are defined in the network with ground-truth.

For networks without ground-truth, modularity measure [37] is used as the most well-known metric to evaluate quality of partitioning and density of detected communities.

TABLE 3
Properties of Generated LFR Networks

Network	N	*Min K*	*Max K*	*Min C*	*Max C*	$\boldsymbol{\gamma}$	$\boldsymbol{\beta}$	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

TABLE 4
NMI Results Obtained From Real-World Ground-Truth Datasets (Highest Values on Each Row are Bolded)

Datasets	CNM	Infomap	LPA	GCN	LCDR	FluidC	CDME	CFCD2	TJA-net	Louvain	Leiden	ECES(s = 1)	RTLCD	LSMD	LBLD
Karate	0.69	0.70	0.21	0.83	**1**	1	1	0.84	1	0.71	0.687	0.63	**1**	**1**	**1**
Dolphins	0.55	0.556	0.53	0.54	**1**	0.89	0.70	0.55	1	0.435	0.55	0.49	0.45	**1**	**1**
Polbooks	0.53	0.54	0.44	0.53	0.58	0.51	0.58	0.48	-	0.53	0.57	0.50	0.48	0.59	**0.60**
Football	0.74	0.90	0.85	0.87	0.90	0.89	0.93	0.83	0.927	0.85	0.85	0.75	0.51	**0.93**	0.91
DBLP	0.49	0.61	0.71	0.68	0.70	0.74	**0.75**	0.26	-	0.53	0.54	0.36	0.41	0.70	0.74
Amazon	0.88	0.42	0.93	0.90	0.69	0.90	0.96	0.72	-	0.83	0.86	0.58	0.72	0.95	**0.97**
YouTube	-	0.50	-	0.23	0.31	**0.72**	-	0.30	-	0.48	0.46	-	-	0.19	0.57
Orkut	-	-	-	-	-	-	-	-	-	-	-	-	-	0.27	**0.68**
LiveJournal	-	-	0.88	-	-	0.87	0.93	-	-	0.75	-	-	-	0.80	**0.934**

Modularity metric is defined as (11).

$$Q = \frac{1}{2m} \sum_{a,b} A_{ab} - \frac{deg_a \times deg_b}{2m} \times \delta(c_a, c_b) \tag{11}$$

Where m denotes number of edges in the network. A_{ab} is equal to 1 if there exists an edge between a and b in adjacency matrix A. $\delta(c_a, c_b)$ is the Kronecker delta. If both nodes a and b share same label, the value of $\delta(c_a, c_b)$ is equal to 1, otherwise it is equal to zero.

4.4 Experimental Results on Real-World Datasets

Obtained results from testing different algorithms on real-world datasets are evaluated with NMI, F-measure, and modularity metrics. NMI and F-measure results for RTLCD algorithm are taken from [11]. Also, NMI results of CDME are adopted from [28]. NMI results for LPA, FluidC, and Louvain for LiveJournal dataset are taken from [28]. Furthermore, most of the compared algorithms were not able to execute Orkut and LiveJournal datasets due to the time-complexity issues and lack of sufficient RAM. However, LBLD and LSMD algorithms are successfully executed in these datasets due to using a neighboring list structure in execution process.

4.4.1 NMI Evaluation

First, performance of the proposed LBLD method and other methods on 9 ground-truth datasets is evaluated based on NMI measure. Table 4 shows the obtained results for different algorithms evaluated with NMI. Top 5000 quality communities of ground-truth for DBLP, Amazon, YouTube, Orkut, and LiveJournal datasets are adopted for computing NMI and F-measure. Therefore, for evaluating results in these five datasets, only the nodes from detected communities which are presented in the original ground-truth file are selected. Since FluidC needs the number of communities as prior knowledge, the real number of communities in Table 1 is used as its parameter K.

It is revealed from Table 4 that the LBLD algorithm shows competitive results in comparison with other methods. In Karate and Dolphins datasets, the LBLD successfully discovers communities with NMI equal to 1. The LBLD also shows superior performance in Pollbooks, Amazon, Orkut, and LiveJournal datasets. In LiveJournal dataset, the LBLD achieves NMI result equal to 0.934, that is the highest quality ever seen on LiveJournal massive dataset. Furthermore, achieving NMI = 0.68 in Orkut dataset with 117 million of edges proves significant accuracy of the LBLD in comparison with the other algorithms. Needless to mention that despite of the results of the FluidC, it is not applicable for community detection in real-world networks because FluidC is not able to reveal the communities naturally and it depends on the predefine number of communities which is unknown in real-world networks. Furthermore, the LBLD can detect structure of communities in Amazon dataset with NMI = 0.97 that is the best result among all methods. In DBLP and YouTube datasets, second-best NMI results belong to the LBLD. In general, LBLD has the best and second-best rank in all of the networks with ground-truth.

4.4.2 F-Measure Evaluation

This measure shows the ability of an algorithm in identifying correct and incorrect results. Table 5 demonstrates F-measure results of different algorithms. It is obvious that six highest results from 9 experiments belongs to the LBLD. It was observed that, after performing rough core detection and balanced label diffusion step, almost a considerable number of nodes receive their permanent labels. One important point is that after rough core detection and second step of label diffusion, no new labels are generated and labels are reorganized between nodes during other steps of the algorithm. This can prove the impact of rough core detection and label diffusion to common neighbors which prevents the algorithm from generating and selecting wrong labels.

In Karate and Dolphins datasets, the LBLD algorithm discovers communities with 100% accuracy, so its precision, recall, and F-measure results are equal to 1. In Football dataset it has the highest value. In Amazon dataset, LBLD obtains NMI = 0.97 and F-measure = 0.90 that implies LBLD is close to real communities. However, CFCD2 method obtains NMI = 0.72 and F-measure equal to 0.94. LBLD is significantly efficient on YouTube, Orkut, and LiveJournal.

In LiveJournal dataset, obtaining F-measure equal to 0.94 implies the best quality of the algorithm in identifying true communities in a large-scale network. In addition, the LBLD outperforms all other algorithms in terms of F-measure in YouTube and Orkut networks. In DBLP, Amazon, and Polbooks datasets, second-best F-measure results belong to the LBLD. In DBLP, CFCD2 has obtained F-measure = 0.67 and NMI = 0.26, whereas the LBLD has F-measure = 0.65 and NMI = 0.74 which approximately is three times higher than CFCD2's NMI. It means that LBLD was

TABLE 5
F-measure Results Obtained From Real-World Ground-Truth Datasets (Highest Values on Each Row are Bolded)

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

more successful to discover partitions more similar to the real communities than CFCD2 algorithm.

4.4.3 Modularity Evaluation

In this section, algorithms are evaluated based on modularity in Table 6. Q and C show modularity value and the number of detected communities by the algorithm, respectively. Community detection groups similar nodes into same communities and aims to form dense groups of nodes. However, it is vital to mention that high value of modularity does not necessarily indicate efficiency and accuracy of the algorithm. For example, according to Table 6, LBLD in Karate dataset has revealed 2 actual communities with 100% accuracy (NMI, F-measure = 1) with modularity 0.3715, whereas Leiden with wrong detection of communities has obtained the highest modularity value equal to 0.42. Based on the ground-truth of Karate dataset, the actual modularity is 0.3715. As another example, in Dolphins dataset, LBLD with 2 communities has the lowest modularity but NMI = 1 with 100% detection accuracy, whereas Infomap and Leiden respectively detect 5 communities with modularity 0.53 and modularity 0.527 instead of the real modularity 0.378. The obtained modularity by the LBLD exactly is the same as the real modularity of Dolphins dataset based on the ground-truth. An interesting result of these experiments was that Leiden and Louvain had almost the best modularity in all datasets with ground-truth, while they had the worst results in terms of accuracy. Modularity experiments in this paper were conducted solely to show that the modularity criterion primarily is not a suitable measure for comparing and judging of algorithms.

By referring to the all of the ground-truth datasets mentioned in Tables 4 and 5, it becomes clear that our claim is true. In DBLP dataset, Leiden discovered only 208 communities, indicating that it has the worst performance among all of the compared methods while having the highest modularity. This significantly shows that Leiden has ignored a lot of communities and has merged all of them into 208 communities. The same conditions can be seen in Amazon dataset. In Amazon dataset, Leiden and Louvain methods both have highest modularity, but their F-measure meaningfully is low in comparison with LBLD and other methods. In addition, Infomap wrongly detects only 13 communities for Amazon which is far from the actual number of communities (75149), whereas its modularity is relatively close to the LBLD. To conclude, the aim of the community detection is not

TABLE 6
Experimental Results on Real-World Datasets Based on Modularity (Highest Values on Each Row are Bolded)

Datasets	CNM		Infomap		LPA		GCN		LCDR		Louvain		Leiden		LSMD		LBLD	
	C	Q	C	Q	C	Q	C	Q	C	Q	C	Q	C	Q	C	Q	C	Q
Karate	3	0.38	3	0.40	3	0.11	2	0.371	2	0.371	4	0.415	4	**0.42**	2	0.371	2	0.371
Dolphins	4	0.49	5	**0.53**	7	0.48	5	0.51	2	0.378	6	0.516	5	0.527	2	0.378	2	0.378
Polbooks	4	0.50	5	0.523	8	0.48	6	0.51	3	0.50	5	**0.526**	4	**0.526**	3	0.446	2	0.456
Football	6	0.57	11	0.60	9	0.55	12	0.57	9	0.60	9	**0.602**	9	**0.602**	12	0.58	13	0.58
Netscience	275	0.95	268	0.93	343	0.90	317	0.88	332	0.92	277	**0.96**	278	**0.96**	275	0.94	303	0.94
Power Grid	42	0.93	6	0.78	1406	0.60	678	0.64	473	0.79	41	0.93	42	**0.94**	446	0.793	341	0.82
CA-GRQC	415	0.82	377	0.84	991	0.75	781	0.72	691	0.75	392	**0.86**	393	**0.86**	456	0.77	561	0.79
Collaboration	668	0.80	618	0.84	1594	0.69	1876	0.70	1825	0.77	623	**0.85**	626	**0.85**	1421	0.72	1637	0.79
CA-HEPTH	538	0.72	458	0.74	1729	0.62	1344	0.60	993	0.627	475	**0.77**	478	**0.77**	586	0.63	761	0.70
PGP	191	0.85	3	0.13	2059	0.74	911	0.79	869	0.80	96	**0.88**	95	**0.88**	643	0.59	358	0.82
Condmat-2003	1240	0.68	945	0.68	4220	0.62	3387	0.61	3122	0.68	959	0.76	961	**0.77**	3502	0.57	2314	0.70
Condmat-2005	1437	0.65	1006	0.61	5145	0.59	3906	0.62	3691	0.64	1021	0.72	1026	**0.73**	3952	0.44	2565	0.66
Brightkite	1416	0.61	566	0.36	5644	0.57	2342	0.62	2436	0.44	742	0.68	678	**0.69**	2384	0.29	1251	0.60
DBLP	3078	0.73	531	0.81	43190	0.65	22589	0.69	19405	0.69	198	0.82	208	**0.83**	17280	0.65	18394	0.70
Amazon	1463	0.87	13	0.78	37428	0.72	29731	0.74	21749	0.77	232	**0.93**	382	**0.93**	34304	0.68	15501	0.80
YouTube	-	-	945	0.69	-	-	40183	0.64	25914	0.50	7365	0.716	4039	**0.73**	9636	0.42	25169	0.45
Orkut	-	-	-	-	-	-	-	-	-	-	-	-	-	-	8661	-	12306	-
LiveJournal	-	-	-	-	-	-	-	-	-	-	-	-	-	-	97213	-	103493	-

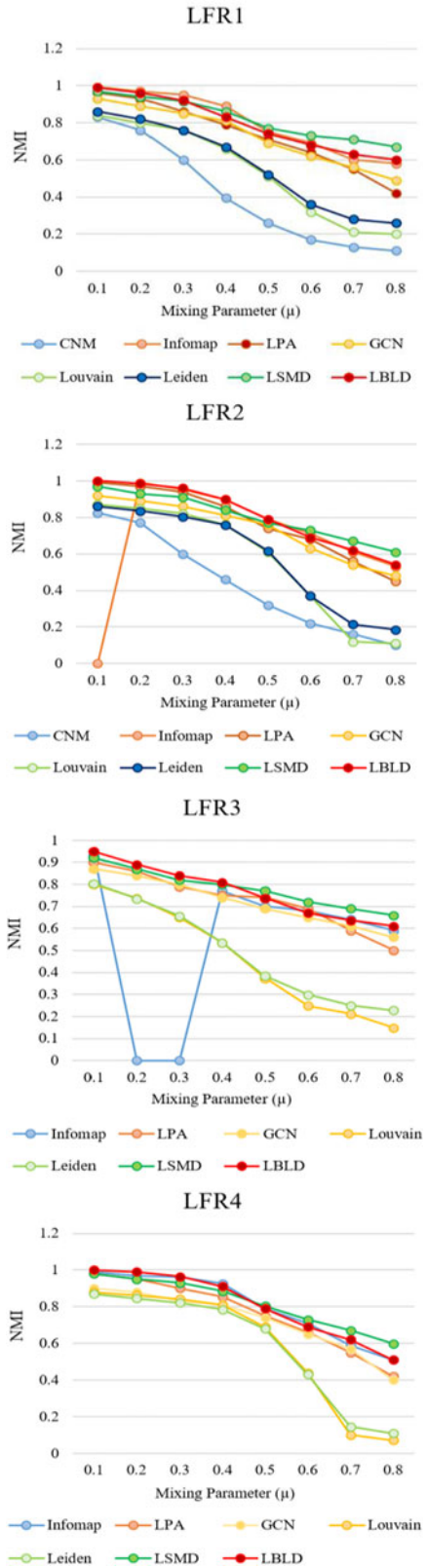


Fig. 3. Comparison of examined algorithms on 4 different LFR networks.

obtaining higher values of modularity, but it is expected to reveal the real structure of the communities. Since LBLD is inspired from the natural model of forming communities, its detected communities are closer to the real communities. Also, during experiments, LBLD and LSMD algorithms showed the nearest results to the real modularity of the networks with ground-truth. Other advantage of the LBLD algorithm is its stability, whereas other methods such as LPA, GCN, Louvain, and Leiden are instable methods in discovering communities.

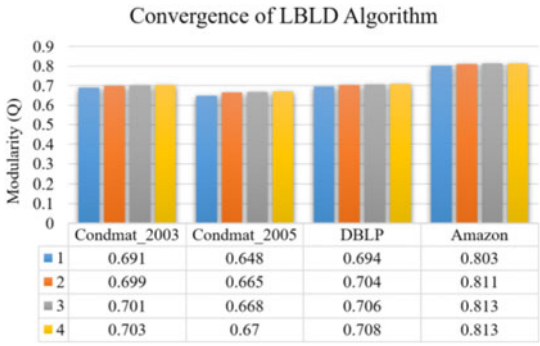
4.5 Experimental Results on LFR Datasets

In this section, algorithms are examined on LFR networks. As mixing parameter (μ) increases, network gets more fuzzier and resolution of the network is decreased. Accuracy and robustness of the algorithms are tested for high values of μ. Fig. 3 shows obtained NMI results on four LFR datasets. It can be seen that the best results belong to the LBLD and LSMD algorithms. Infomap fairly presents instable results in some LFRs. Due to the random walk nature of Infomap, unexpected behaviors in both LFR2 and LFR3 networks was observed. In Figs. 3b and 3c, starting with $\mu = 0.1$ where the network has the highest resolution, Infomap fails to find true communities and obtains NMI≅0. Unlike the Infomap, the LBLD shows meaningful changes to variations of mixing parameter. This is because of the designing system of the LBLD that at initial steps by forming rough cores, constructs the main communities and even when the value of the mixing parameter increases, LBLD by the help of the rough core detection step and balanced label diffusion, can find more accurate communities. Following the LSMD, the proposed LBLD method has the second place of robustness between other methods.

On the other hand, modularity-based methods obtain unacceptable and poor performance on LFR networks with low density due to the existence of small communities and suffering from the resolution limit problem. It is obvious that most of the algorithms can perform well on dense networks ($\mu \leq 0.4$), but when it comes to networks with relatively low density, only LBLD, LSMD, and GCN methods are able to maintain high accuracy. Since real-world networks are not heavily dense networks and mostly are considered as sparse networks, so modularity-based methods also do not show efficient performance in such networks, too. CNM, Louvain, and Leiden algorithms compared to the other methods, especially with LBLD and LSMD, show significant sensitivity to density and number of the communities of network and extremely have low efficiency and accuracy. Also, these three methods fail to compete with other methods on high resolution LFR networks with low values of μ. GCN algorithm due to the inability on allocating right labels to the border nodes as the network gets fuzzier and the number of the links between communities increases, has significant decreases in accuracy. This is the advantage of the LBLD algorithm that besides forming communities from the outer side (from low importance nodes), it expands rough cores from the inner side and totally gives stability to the algorithm and prevents diffusion of wrong labels from other communities.

4.6 Convergence Evaluation of the LBLD Algorithm

The LBLD algorithm shows efficient convergence speed during experiments. Label selection step is the only iterative part of the proposed method. According to the observations, label selection step mostly affects the border nodes to select the final labels. Thanks to the rough core detection and diffusion steps, nodes located in dense parts which



	Condmat_2003	Condmat_2005	DBLP	Amazon
1	0.691	0.648	0.694	0.803
2	0.699	0.665	0.704	0.811
3	0.701	0.668	0.706	0.813
4	0.703	0.67	0.708	0.813

Fig. 4. Modularity results obtained from execution of LBLD with different number of iterations on four real-world datasets.

have more connections with each other and are near to the kernels of the communities, have chosen stable labels in previous steps. Therefore, label selection step has a fast convergence speed and it is repeated with a maximum of two iterations in some datasets. Fig. 4 shows the modularity results obtained from executing the proposed method with four different number of iterations for label selection step. Results prove that the proposed LBLD method even with 1 iteration of label selection step can provide acceptable results. According to Fig. 4 convergence relatively starts after the first iteration is performed. For example, in Condmat_2003, there is only 0.012 difference in modularity when LBLD is executed with 1 and 4 iterations. To conclude, LBLD can provide competitive results by performing 1 or maximum of 2 iterations.

4.7 Evaluation of Execution Time

One of the main goals of proposing the LBLD algorithm is to find communities with a fast execution time. Besides the accuracy, it is expected from the algorithm to provide quality results on a reasonable time. Execution time of an algorithm depends on its time complexity, space complexity, type of operations used in the algorithm, type of compiler, etc. For example, the LPA despite of its low time complexity can be inefficient on large networks due to convergence problem and continuously checking direct neighbors of all nodes during the execution. Table 7 lists execution time of tested algorithms in several datasets. Algorithms are executed for different values of mixing parameter in LFR datasets and their average execution times are reported. Due to the performance issues of MATLAB compared to Python, it is not fair to compare LSMD (written in MATLAB) with other algorithms. For instance, the Louvain algorithm for YouTube dataset is executed in 2959(s) in MATLAB, whereas it is executed in 53(s) in its C++ version. The results of the Leiden and Louvain algorithms in these experiments are based on their C++ versions. Of course, Python version of Louvain is also six-folds slower than its C++ versions. Experiments on LFR datasets showed that there is an interesting performance issue in Leiden and Louvain.

Since the LBLD algorithm does not extremely depend on the structure of the networks and the number of communities, and it only depends on the number of nodes, the execution time of the LBLD algorithm relatively is same for different values of μ for a specific configuration of LFR dataset. However, it is not true for Leiden and Louvain. These two methods extremely depend on the resolution of the network and the number of communities and their performance considerably varies from a specific LFR network with various μ parameter. For instance, Leiden algorithm executes Amazon network with 334,000 nodes and 925,872 edges in 13 seconds whereas it finishes an LFR network with 200,000 nodes and about 900,000 edges in 18 seconds which is smaller than Amazon dataset in terms of number of nodes and edges. The analysis of these datasets revealed that Leiden severely depends on the minimum and maximum number of the nodes within communities. The previous LFR dataset is generated with the following parameters: $min\ K = 10$, $Max\ K = 25$, $Min\ c = 300$, and $Max\ c = 1000$.

To deeply understand and to expand experiments, another LFR network with 200000 nodes and about 690000 edges and a configuration with $min\ K = 5$, $Max\ K = 25$, $Min\ c = 100$, and $Max\ c = 600$ is generated. It is found that Leiden executed this dataset in 10 seconds. In both configurations, Leiden has the worst NMI and F-measure results. To conclude, Leiden does not show a meaningful behavior on networks and extremely depends on the size of the communities. Moreover, the Louvain algorithm showed considerable reactions to the variation of mixing parameter in LFR datasets. For example, in LFR4, when $\mu = 0.1$, the execution time is equal to 359 seconds, whereas for $\mu = 0.8$, Louvain terminates after 5107 seconds.

According to Table 7, the best execution time in all of the datasets belongs to the LBLD. Needless to mention that the LBLD outperforms other methods with a considerable difference. Second-best execution time belongs to Infomap, but there are significant differences in execution times between LBLD and Infomap. It is necessary to mention that according to [51], [52] 86% of Infomap algorithm is implemented with C++. CNM algorithm was not capable of completing YouTube dataset after 14 hours of execution. The LBLD algorithm especially shows an efficient performance on LiveJournal and Orkut datasets whereas other methods were not able to execute them. Efficient data structure of the LBLD as well as its fast nature give a great advantage to algorithm to easily execute on massive networks with millions of nodes and edges. Despite of near-linear time complexity in most of the compared methods, they are not able to finish community detection process on large-scale networks due to memory usage and other problems. From these experiments, it is revealed that Leiden does not have minimum quality in community detection on large-scale networks.

4.8 Evaluation of Memory Usage

As it is mentioned before, the LBLD algorithm uses an appropriate neighboring list structure to easily perform large-scale networks with low RAM usage and fast discovering communities. To compare the efficiency of the LBLD algorithm based on used neighboring list structure against the constructing graph structure, an experiment was performed on evaluating the memory usage of two mentioned structures based on a small RAM (12GB). Table 8 shows the obtained results for this experiment. The parameter R represents the amount of RAM space used for reading the datasets or constructing graph and parameter T represents the total RAM space used by the whole algorithm. As the results show, there are significant

TABLE 7
Execution Times of Examined Algorithms on Real-World and LFR Datasets (All Mentioned Times are in Seconds)

	CNM	Infomap	LPA	GCN	Louvain	Leiden	LSMD	LBLD
Condmat-2003	585	4	15	33	10	1	23	**2**
Condmat-2005	1043	6	33	73	15	2	38	**3**
Brightkite	3427	10	60	171	19	2	42	**6**
LFR1	311	6	46	37	51	2	31	**3**
LFR2	4038	14	161	108	112	6	78	**10**
LFR3	-	32	1496	121	459	9	94	**13**
DBLP	53757	46	2824	332	212	12	182	**20**
Amazon	18784	59	2488	355	128	13	161	**18**
LFR4	N/A	89	N/A	652	1432	45	523	**69**
YouTube	N/A	N/A	N/A	13569	498	53	609	311
LiveJournal	N/A	N/A	N/A	N/A	N/A	N/A	7061	**1347**
Orkut	N/A	N/A	N/A	N/A	N/A	N/A	30292	**13925**

TABLE 8
The Amount of RAM Usage By Two Different Data Structures (All Mentioned Values are in Megabytes)

Datasets	Proposed structure	Graph structure
DBLP	R: 242 T: 401	R: 798 T: 813
Amazon	R: 235 T: 404	R: 737 T: 764
YouTube	R: 612 T: 1022	R: 2081 T: 2172
LiveJournal	R: 3682 T: 5386	R: - T: -
Orkut	R: 8774 T: 10994	R: - T: -

differences in usage of RAM between two structures. In DBLP and Amazon datasets, constructing graph takes about two times more RAM space than the implemented neighboring list structure. The difference in RAM usage is much more obvious in YouTube dataset. Adopting graph structure uses three times more RAM space than the adopted structure in reading the dataset in YouTube dataset. In other words, the graph structure prevents us from constructing graph for large datasets such as LiveJournal and Orkut since it does not feet into the 12 GB of RAM space.

5 Conclusion

In this paper a local community detection algorithm with low time complexity based on label diffusion is proposed. Inspired by the fact that in the real-world people try to connect with important people and tend to be grouped with ones with similar ethics, the LBLD algorithm assigned nodes with their most similar neighbors to the same communities. A new way of constructing rough cores is used to effectively detect some initial seed nodes and adopted balanced label diffusion to address the mentioned problems in expanding communities. Finally, dense communities were obtained after utilizing a merge method. To demonstrate accuracy and efficiency of the proposed LBLD algorithm, it is tested on real-world and synthetic datasets with a wide variation range of nodes. The results proved superior performance and accuracy of the proposed method in large-scale networks as well as small networks. Running times showed that LBLD has much better running time than other methods due to its efficient data structure and lack of time-consuming operations. Highlights of the proposed method can be summarized as follows: LBLD has extremely fast convergence, robustness and stability, and lack of random nature as well as lack of adjusting parameter.

There remain several improvements to be investigated as future works. The proposed method has a great potential to be run in parallel. Computing nodes' score takes most of the execution time of the algorithm and since they are independent jobs, parallel execution can help to reduce execution time of the algorithm and improves its scalability. Also, improving label diffusion strategies can help to fast convergence of the algorithm and proposing new methods for rough core detection and adopting more accurate measures can severely impact the accuracy of the algorithm.

References

[1] L. Freeman, "The development of social network analysis," *Study Sociol. Sci.*, vol. 1, 2004, Art. no. 687.

[2] J. Chen and B. Yuan, "Detecting functional modules in the yeast protein–protein interaction network," *Bioinformatics*, vol. 22, no. 18, pp. 2283–2290, 2006.

[3] P. K. Reddy, M. Kitsuregawa, P. Sreekanth, and S. S. Rao, "A graph based approach to extract a neighborhood customer community for collaborative filtering," in *International Workshop Databases Networked Information Systems*, Berlin, Germany: Springer, 2002, pp. 188–200.

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

[5] S. Fortunato, "Community detection in graphs," *Phys. Rep.s*, vol. 486, no. 3-5, pp. 75–174, 2010.

[6] K. Berahmand, A. Bouyer, and M. Vasighi, "Community detection in complex networks by detecting and expanding core nodes through extended local similarity of nodes," *IEEE Trans. Comput. Social Syst.*, vol. 5, no. 4, pp. 1021–1033, 2018, doi: 10.1109/TCSS.2018.2879494.

[7] Y. Xing, F. Meng, Y. Zhou, M. Zhu, M. Shi, and G. Sun, "A node influence based label propagation algorithm for community detection in networks," *Sci. World J.*, vol. 2014, 2014, Art. no. 627581.

[8] X.-K. Zhang, J. Ren, C. Song, J. Jia, and Q. Zhang, "Label propagation algorithm for community detection based on node importance and label influence," *Phys. Lett. A*, vol. 381, no. 33, pp. 2691–2698, 2017.

[9] K. Berahmand and A. Bouyer, "LP-LPA: A link influence-based label propagation algorithm for discovering community structures in networks," *Int. J. Modern Phys. B*, vol. 32, no. 06, 2018, Art. no. 1850062.

[10] T. Zhang and B. Wu, "A method for local community detection by finding core nodes," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining*, 2012, pp. 1171–1176.

[11] X. Ding, J. Zhang, and J. Yang, "A robust two-stage algorithm for local community detection," *Knowl.-Based Syst.*, vol. 152, pp. 188–199, 2018.

[12] A. Bouyer and H. Roghani, "LSMD: A fast and robust local community detection starting from low degree nodes in social networks," *Future Gener. Comput. Syst.*, vol. 113, pp. 41–57, 2020, doi: 10.1016/j.future.2020.07.011.
[13] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Statist. Mechanics, Theory Experiment*, vol. 2008, no. 10, vol. 2008, Art. no. P10008.
[14] V. A. Traag, L. Waltman, and N. J. van Eck, "From louvain to leiden: Guaranteeing well-connected communities," *Sci. Rep.*, vol. 9, no. 1, pp. 1–12, 2019.
[15] S. Garruzzo and D. Rosaci, "Agent clustering based on semantic negotiation," *ACM Trans. Auton. Adaptive Syst.*, vol. 3, no. 2, pp. 1–40, 2008.
[16] M. Tasgin and H. O. Bingol, "Community detection using boundary nodes in complex networks," *Phys. A, Statist. Mechanics Appl.*, vol. 513, pp. 315–324, 2019.
[17] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Phys. Rev. E*, vol. 76, no. 3, 2007, Art. no. 036106.
[18] R. R. Nadakuditi and M. E. Newman, "Graph spectra and the detectability of community structure in networks," *Phys. Rev. Lett.*, vol. 108, no. 18, 2012, Art. no. 188701.
[19] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, "Inference and phase transitions in the detection of modules in sparse networks," *Phys. Rev. Lett.*, vol. 107, no. 6, 2011, Art. no. 065701.
[20] F. Radicchi, "Driving interconnected networks to supercriticality," *Phys. Rev. X*, vol. 4, no. 2, 2014, Art. no. 021014.
[21] L. Yang, X. Cao, D. Jin, X. Wang, and D. Meng, "A unified semi-supervised community detection framework using latent space graph regularization," *IEEE Trans. Cybern.*, vol. 45, no. 11, pp. 2585–2598, Nov. 2015.
[22] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 203–209.
[23] S. Cavallari, V. W. Zheng, H. Cai, K. C.-C. Chang, and E. Cambria, "Learning community embedding with community detection and node embedding on graphs," in *Proc. ACM Conf. Inf. Knowl. Manage.*, 2017, pp. 377–386.
[24] Y. Jia, Q. Zhang, W. Zhang, and X. Wang, "Communitygan: Community detection with generative adversarial nets," in *Proc. World Wide Web Conf.*, 2019, pp. 784–794.
[25] K. Berahmand and A. Bouyer, "A link-based similarity for improving community detection based on label propagation algorithm," *J. Syst. Sci. Complexity*, vol. 32, no. 3, pp. 737–758, 2019.
[26] T. Wang, S. Chen, X. Wang, and J. Wang, "Label propagation algorithm based on node importance," *Phys. A, Statist. Mechanics Appl.*, 2020, Art. no. 124137.
[27] J. Zhang, X. Ding, and J. Yang, "Revealing the role of node similarity and community merging in community detection," *Knowl.-Based Syst.*, vol. 165, pp. 407–419, 2019.
[28] Z. Sun *et al.*, "Community detection based on the matthew effect," *Knowl.-Based Syst.*, vol. 205, 2020, Art. no. 106256.
[29] S. Aghaalizadeh, S. T. Afshord, A. Bouyer, and B. Anari, "A three-stage algorithm for local community detection based on the high node importance ranking in social networks," *Phys. A, Statist. Mechanics Appl.*, vol. 563, 2021, Art. no. 125420.
[30] F. Parés *et al.*, "Fluid communities: A competitive, scalable and diverse community detection algorithm," in *Proc. Int. Conf. Complex Netw. Appl.*, 2017, pp. 229–240.
[31] H. Sun *et al.*, "CenLP: A centrality-based label propagation algorithm for community detection in networks," *Phys. A, Statist. Mechanics Appl.*, vol. 436, pp. 767–780, 2015.
[32] M. Zarezade, E. Nourani, and A. Bouyer, "Community detection using a new node scoring and synchronous label updating of boundary nodes in social networks," *J. AI Data Mining*, vol. 8, no. 2, pp. 201–212, 2020.
[33] E. Gujral, E. E. Papalexakis, G. Theocharous, and A. Rao, "Hacd: Hierarchical agglomerative community detection in social networks," in *Proc. IEEE 29th Int. Workshop Mach. Learn. Signal Process.*, 2019, pp. 1–6.
[34] S. Taheri and A. Bouyer, "Community detection in social networks using affinity propagation with adaptive similarity matrix," *Big Data*, vol. 8, no. 3, pp. 189–202, 2020.
[35] W. Zhang, R. Zhang, R. Shang, and L. Jiao, "Weighted compactness function based label propagation algorithm for community detection," *Phys. A, Statist. Mechanics Appl.*, vol. 492, pp. 767–780, 2018.
[36] M. E. Newman, "Fast algorithm for detecting community structure in networks," *Phys. Rev. E*, vol. 69, no. 6, 2004, Art. no. 066133.
[37] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Phys. Rev. E*, vol. 70, no. 6, 2004, Art. no. 066111.
[38] R. Shang, S. Luo, Y. Li, L. Jiao, and R. Stolkin, "Large-scale community detection based on node membership grade and sub-communities integration," *Phys. A, Statist. Mechanics Appl.*, vol. 428, pp. 279–294, 2015.
[39] R. Shang, H. Liu, L. Jiao, and A. M. G. Esfahani, "Community mining using three closely joint techniques based on community mutual membership and refinement strategy," *Appl. Soft Comput.*, vol. 61, pp. 1060–1073, 2017.
[40] C. Lyu, Y. Shi, and L. Sun, "A novel local community detection method using evolutionary computation," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3348–3360, Jun. 2021.
[41] M. Gong, Q. Cai, X. Chen, and L. Ma, "Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition," *IEEE Trans. Evol. Computation*, vol. 18, no. 1, pp. 82–97, Feb. 2014.
[42] X. Zhang, K. Zhou, H. Pan, L. Zhang, X. Zeng, and Y. Jin, "A network reduction-based multiobjective evolutionary algorithm for community detection in large-scale complex networks," *IEEE Trans. Cybern.*, vol. 50, no. 2, pp. 703–716, Feb. 2020.
[43] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Rev. Modern Phys.*, vol. 74, no. 1, 2002, Art. no. 47.
[44] M. S. Granovetter, "The strength of weak ties," *Amer. J. Sociol.*, vol. 78, no. 6, pp. 1360–1380, 1973.
[45] H. Roghani, A. Bouyer, and E. Nourani, "PLDLS: A novel parallel label diffusion and label Selection-based community detection algorithm based on spark in social networks," *Expert Syst. Appl.*, vol. 183, 2021, Art. no. 115377, doi: 10.1016/j.eswa.2021.115377.
[46] G. Rossetti, L. Milli, and R. Cazabet, "CDLIB: A python library to extract, compare and evaluate communities from complex networks," *Appl. Netw. Sci.*, vol. 4, no. 1, 2019, Art. no. 52.
[47] "CDLIB community discovery algorithms," Jul. 2019. [Online]. Available: https://cdlib.readthedocs.io/en/latest/reference/cd_algorithms/node_clustering.html
[48] "SNAP project," 2020. [Online]. Available: http://snap.stanford.edu/data/index.html
[49] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Phys. Rev. E*, vol. 78, no. 4, 2008, Art. no. 046110.
[50] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Statist. Mechanics, Theory Experiment*, vol. 2005, no. 9, 2005, Art. no. P09008.
[51] "Implementation of infomap algorithm," Dec. 2020. [Online]. Available: https://github.com/mapequation/infomap
[52] "Infomap project description," Apr. 2020. [Online]. Available: https://pypi.org/project/infomap/

Hamid Roghani received the MSc degree in information technology engineering from the Faculty of Computer Engineering and Information Technology, Azarbaijan Shahid Madani University, Tabriz, Iran, in 2020. His current research interests include social networks analysis, signal processing, machine learning, and distributed computing with Spark.

Asgarali Bouyer received the PhD degree in computer science from Universiti Teknologi Malaysia (UTM), in 2011. He is an associate professor with the Faculty of Computer Engineering and Information Technology, Azarbaijan Shahid Madani University, Tabriz, Iran. His current research interests include complex network analysis, social media mining, and data mining and its applications. He has published more than 70 papers in refereed journals and conferences.