# SbChain+: An Enhanced Snowball-Chain Approach for Detecting Communities in Social Graphs

Jayati Gulati, Muhammad Abulaish
Department of Computer Science
South Asian University
New Delhi, India

***Abstract*—In this paper, we present snowball-chain (SbChain+) approach, which is an improved version of SbChain community detection method in terms of precision with which communities are identified in a social graph. It exploits the topology of a social graph in terms of the connections of a node, i.e., its degree centrality, betweenness centrality and the number of links within its neighborhood defined by the local clustering coefficient. Two different functions have been used to identify neighbors for a given node. Hence, two approaches have been discussed with their pros and cons. In general, SbChain+ takes a social graph as an input and aims to identify communities around the core nodes in the underlying network. The core nodes are expected to have a high degree and have densely connected neighbors and guides in identifying cliques from the graph. The proposed approach takes its inspiration from snowball sampling technique and keeps merging the nodes with their neighboring nodes based on certain criteria to form snowballs. One of the functions discussed (SbChain+(i)) uses a hyperparameter, $\lambda$ for merging snowballs which further leads to the formation of communities. This hyperparameter also helps in achieving the desired level of coarseness in the communities, and it can be adjusted to fine tune the identified communities. While the second function (SbChain+(ii)) uses an average out degree function to merge snowballs. The *modularity* values are calculated at each level of the dendrogram formed by combining nodes and snowballs to decide an appropriate cut for community determination. SbChain+ is empirically evaluated using these two different functions over both real-world and LFR-benchmark datasets and results are evaluated on *modularity* and *normalized mutual information*. The aim of this study is to improve upon the previously discussed technique (SbChain) and to study the use of hyperparameter, i.e., the performance of a technique with or without the hyperparameter.**

***Keywords*—Clique; clustering; community detection; graph mining; snowball sampling; social network analysis**

## I. Introduction

Online social network is an ever-growing entity which can be modeled in the form of graphs (*aka* social graphs) with users or entities as the nodes and their interactions or relationships as weighted or unweighted edges [1]. Community detection is an application of studying social graphs that provides useful information about groups that might exist due to similar interests, occupation and so on. It is formally stated by Girvan and Newman as the *community detection* problem in [2]. Communities are expressed as a group of nodes that are coherent and are well-connected or have similar characteristics, and sparse connections with the other nodes or dissimilar characteristics with the rest of the nodes. Identifying communities enables an in-depth understanding of the arrangement of nodes and edges in a social graph, because they correspond to the entities and their respective relationships within a group or inter-groups. Hence, it can be useful in identifying highly cohesive sub-structures.

There are various approaches like density-based, hierarchical, and label propagation methods for community detection. The density-based approach aims to find core points in the network that have a high number of neighbors based on a predefined threshold value. It also identifies the isolated nodes or outliers in the same manner and then grows the communities. Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3] uses two external parameters, a *minpoint* threshold and a *neighborhood parameter* depending on which the results may differ. A density-based approach, called CMiner, is presented in [4] which finds overlapping communities using a new distance function derived from the average reciprocated interactions among nodes. The authors also proposed another approach in [5], called OCTracker, for finding overlapping communities using a density-based framework and tracking the various aspects of community evolution. Hierarchical approaches are another set of methods for community detection in social networks that may work either in a bottom-up manner or in a top-down manner. Bottom-up approach considers each node to be a separate community and combines nodes in an iterative manner to maximize modularity. On the other hand, top-down approach considers the entire network to be a single community and divides it in an iterative manner until the desired set of communities are obtained. However, some hierarchical approaches have found to have very high complexities depending upon the cost function to be optimized. In [6], the authors proposed a unified framework, called HOCTracker, which identifies hierarchical overlapping communities in social networks. Label propagation is another approach for community detection in social networks. The Label Propagation Algorithm (LPA) changes a node label to the label of its majority neighbors. However, since LPA uses local information, it gets stuck in local optima. Snowball-Chain (SbChain) is another community finding approach which works well when nodes find their best neighbors in the initial few iterations.

In this paper, we present two improved versions of our previous work Snowball-Chain (SbChain) [7], termed as Enhanced

Snowball-Chain (SbChain+(i) and SbChain+(ii)). It uses simple topological features of a graph, like degree centrality, betweenness centrality, clustering coefficient and average out degree function for detecting communities in undirected and unweighted social graphs, which is the major advantage of this SbChain+ when compared with other techniques in the current work. The idea behind SbChain+ is that the nodes having high centrality value and cliques among their neighbors may become good candidates to form communities. Therefore, the technique starts with the identification of seed nodes (i.e., the nodes having high degree and well-connected neighbors), it aims to identify their best scoring neighbors based on two different functions. Hence, two approaches based on two different functions for SbChain+ are discussed in the subsequent sections, with their pros and cons. One of them uses common neighbor merging strategy (function-i) and the other uses average out degree function (ODF) (function-ii). The nodes merge to form snowballs based on one of these functions. SbChain+(i) and SbChain+(ii) are compared with five other well established state-of-the-art community detection techniques, including Infomap [8], LPA [9], LPA (semi-synchronous) [10], Louvain [11], and SbChain [7]. Approaches like LPA is based on local node interactions and ignore the global information of nodes (like betweenness centrality etc.) in the graph. Whereas, SbChain+ considers all the information of the node by using local as well as global clustering coefficient. It can also be seen that according to [12] Louvain is unable to detect outliers unlike SbChain+. SbChain+ separates nodes with zero degree value before the algorithm begins its processing. The results reveal the effectiveness of the proposed SbChain+ for community detection in real-world social graphs when compared with these techniques. In short, the major contributions/enhancements in this work can be summed up by the following points:

1) Consideration of degree, betweenness centrality and normalized clustering coefficient for seed node identification leading to improvement in terms of community formation.
2) Two improved weight functions based on the concept of common neighbors using a hyperparameter and average ODF, respectively, to calculate the interaction intensity for a pair of nodes. The pros and cons of each of these functions is also studied.
3) An improved empirical validation of the proposed approach over both real-world and LFR-benchmark datasets in terms of identified number of communities, modularity (Q) and Normalized Mutual Information (NMI) for real-world datasets.

The rest of the paper is organized as follows. Section II presents a brief review of the existing literatures on community detection. Section III mentions the preliminary concepts used by the proposed approach. Section IV presents the functional details of our proposed SbChain+ method. Section V presents a discussion on hyperparameter tuning with pros and cons of both the functions used in SbChain+. Section VI presents details about the datasets, experimental settings, and an analysis of the experimental results. The complexity analysis of SbChain+ is mentioned in section VII. Finally, section VIII concludes the paper with future directions of research.

## II. Related Work

A lot of research in the field of community detection has been conducted in the past few years. In [13] and [14], review of existing community detection methods is presented. It divides the detection methods into probabilistic and deep learning categories. The traditional approaches utilize probability-based models for community identification, whereas complex networks are converted to lower dimensional data and worked upon by using deep learning methods. In this paper, we consider a classical approach for community detection that utilizes the parameters from the graph itself. Commonly used community detection methods for connected data are mainly based on Markov clustering algorithm, which uses a random walk process on the given network to identify communities in the form of clusters. The algorithm in [15] proposed a function-modularity intensity which uses network edges along with their weights for community evolution. Another common approach for identifying communities is implemented using hierarchy-based methods. The method in [16] is based on edge removal. It proposes to eliminate the edges having a high score calculated in terms of betweenness centrality and identifies optimized community based on the modularity values. A similar work is presented in [17] which combines nodes that maximize modularity in an agglomerative hierarchical order. It begins with assuming each node as a community and keeps combining nodes until highest value of modularity is achieved. Another work in [18] used spectral clustering along with global maximization of the modularity function.

Density-based approaches like Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [3] and Ordering Points To Identify Cluster Structure (OPTICS) [19] have also been proposed in literature for community detection. DBSCAN uses two user-defined parameters called as the *minimum point* threshold and *neighborhood radius*. OPTICS uses two distance measures – *core-distance* and *reachability-distance* to consider core as well as the points that lie inside the high-density clusters. Inspired from DBSCAN, a Structural Clustering Algorithm for Networks (SCAN) was proposed in [20]. It aims to detect hubs and outliers along with the communities. However, it also requires two parameters, namely, a minimum similarity threshold parameter and minimum number of neighbors. However, it does not provide any details about parameter settings. A popular overlapping community detection approach is Clique Percolation Method (CPM) which is based on growth of communities using $k$-cliques [21]–[23], wherein communities are defined by maximal union of adjacent $k$-cliques. The adjacency of cliques is decided by the number of common nodes between them [22].

Label Propagation Algorithm (LPA) [9] follows another community detection approach, which changes the label of each node to the most frequently occurring label in its neighborhood. The process continues until all the nodes are updated and no more changes can be made to the labels. LPA has been an inspiration for several other works in community detection. For example, authors in [24] proposed CenLP (Centrality-based Label Propagation) algorithm which considers weighted networks for community detection. They proposed a function to calculate the centrality of a node and its similarity

with the neighboring nodes. Another work in [25] creates a compactness function based on a node's weight which is calculated based on two factors: (i) common number of neighbors between a given node and its surrounding nodes, and (ii) degree of node under consideration. The neighboring communities are defined based on high degree nodes and their high degree neighbors. Communities are identified using weighted compactness function value between a node and its neighboring community. An adjustment strategy is devised to achieve an improved accuracy. Other studies in community detection (e.g., [26]) used the score of immediate neighbors of a node to decide its label. Evolutionary algorithm discussed in [27] also finds the community structure based on modularity maximization. The communities formed are merged based on a function calculated via intra-community and intercommunity links.

A study in [28] finds connected components that form a preference network, leading to the formation of communities. The preference network is formed by finding preference nodes using their spread capabilities. It identifies the highest number of overlapping neighbors between a given selector node and its one-path length neighbors. This is calculated using the `gossip` algorithm proposed in [29]. Another similarity-based approach was proposed in [30] which is called Community Detection Algorithm based on Structural Similarity (`CDASS`) and works in two phases. In phase one, the edges bearing a low similarity value are removed, leading to formation of several disconnected components in the network. The components are consolidated eventually to form a set of communities. The second phase identifies optimal communities from the previous phase to give the required results using an evaluation function. The function used in the second phase is realized from different structural parameters of the nodes in the given network. Another work that uses local graph information is discussed in [31] called Flow Propagation Algorithm (`FlowPro`). It can compute the community of exactly one node by using the flow based on edge weights. Each node stores half of the receiving flow and the process continues until there is no flow left to be circulated.

A few deep learning-based techniques are presented in [32], [33]. In [32], a weighted path matrix having path length two is created. It helps in identifying similarity among a node and its neighbors with path length of two or less. Further, a deep sparse autoencoder and k-means clustering algorithm is used to identify the communities. The work in [33] uses an existing technique to design an encoder for identification of communities and their respective nodes. It uses a dual decoder for unsupervised community detection. Another work in [34] uses graph compression technique for analysis of huge networks. The probability of a node to become a seed is calculated using two parameters, quality, and density of the nodes. Finally, the number of communities and initial seed set is recognized using these parameters. A work in [35] develops a framework called Seed Expansion with generative Adversarial Learning (`SEAL`) uses a graph neural network that uses sequential decision process and is trained via policy gradient. It works on the concept of discriminator and generator, the former identifies fake or real communities. While the latter fits in features of communities the real communities.

In [36], a genetic algorithm for feature selection to find communities is discussed. Features are identified and then classified into clusters based on community detection approaches. Next step employs a genetic algorithm that to pick up features based on a novel operation. A local community detection process in [37] works in two parts, where a core detection stage identifies communities based on modularity density. The next stage is the extension stage identifies coherent communities based on Jaccard coefficient.

Our proposed `SbChain+` method is inspired from the aforesaid similarity-based approaches which first find the seed nodes based on certain criteria and then search for highly connected nodes in the neighborhood. This leads to formation of snowballs, which keep expanding until no more nodes can join, eventually leading to the formation of communities. `SbChain+` is compared with both `LPA` and `Infomap` that utilize the concept of random walks and decompose the network into groups based on probability flow. It is also compared with `Louvain`, in which communities are grown in a hierarchical manner by adding nodes that lead to gain in modularity, marking the first phase of the community identification process. Thereafter, weights of the links belonging to a particular community are summed up to complete the second phase. Finally, first and second phases are iteratively repeated until the community formation process converges and a maximum modularity value is achieved.

## III. Preliminaries

For a graph $G(V, E)$ represented by $V = \{v_1, v_2, ...v_n\}$ as a set of $n$ nodes and $E = \{e_{ij} = (v_i, v_j) | v_i, v_j \in V \& \exists$ a link between $v_i$ and $v_j\}$ as a set of edges, the motive is to identify the seed or core nodes that form snowballs, and merging the snowballs/nodes finally to form communities. The notations used in the subsequent sections of this paper are briefly described in Table I.

TABLE I. Notations and their Brief Descriptions

| Notation | Description |
|---|---|
| $\mathcal{N}(v_i)$ | Set of immediate neighbors of a node $v_i$ |
| $k(v_i)$ | Degree centrality of $v_i$ |
| $b(v_i)$ | Betweenness centrality of $v_i$ |
| $LCC(v_i)$ | Normalized local clustering coefficient of $v_i$ |
| $\mathcal{N}_{best}(\mathcal{V})$ | A set of best neighbors for a given set $\mathcal{V} \subseteq V$, calculated by a score function |
| $s^{(n)}$ | A set of nodes of length $n$, called snowball |
| $\mathcal{N}(s^{(n)})$ | Neighbor set of $s^{(n)}$, given by $\mathcal{N}(v_1) \cup \mathcal{N}(v_2) \cup \ldots \mathcal{N}(v_n)$ |

For a graph $G$, the `SbChain+` algorithm initiates by sorting the nodes in non-increasing order based on their scores, which is generated by equation (1), as explained in the following definition.

**Definition 1.** *(Score). The score for a given node $v_i$ is calculated based on its normalized clustering coefficient, degree and betweenness centrality, as formally presented in equation* (1).

$$score(v_i) = (LCC(v_i) + k(v_i) + b(v_i))/3.0 \quad (1)$$

The degree, betweenness centrality and clustering coefficient parameters are used in this study because they provide con-

nectedness of a node with its neighboring nodes, and connectedness of the neighboring nodes, respectively. The degree centrality for a node is the fraction of nodes it is connected to in the graph. Betweenness centrality of a node is given by the sum of the fraction of all-pairs shortest paths that pass through it. The clustering coefficient also provides information about clique formation within the neighbor set of a node. All the three parameters are normalized to bring them on the same scale.

It should be noted that equation (1) works differently for a seed node $v_i$ and a snowball $s^{(n)}$. For $v_i$, the individual value of all the parameters is added, while for a snowball value $s^{(n)}$, the individual values for each node comprising the snowball are picked and divided by the number of nodes in the snowball as given by equation (2).

$$score(s^{(n)}) = score(v_1) + .... + score(v_n)/\|s^{(n)}\| \tag{2}$$

The nodes are selected one at a time in non-increasing order of their score to grow and form communities. Considering $v_i$ be the first selected node, the approach proceeds by finding the best neighboring node denoted by $\mathcal{N}_{best}(v_i)$ from $\mathcal{N}(v_i)$. This node $\mathcal{N}_{best}(v_i)$ is the one having the highest value of score given by equation (1). After the first round of iteration, many nodes combine with their best neighbor and their scores are updated by equation (2). However, it should be noted that a node $v_i$ combines with $\mathcal{N}_{best}(v_i)$ on conditions defined by two different functions as mentioned below.

**Definition 2.** *(Function-i) According to this function $v_i$ combines with $\mathcal{N}_{best}(v_i)$ if the degree of overlap among their neighbor set, i.e, cardinality of the overlapping set obtained by taking the intersection of its own neighbor set and that of the neighbor set of $v_i$ is higher than the hyperparameter $\lambda$ among all the neighbors of $v_i$, given by equation* (3)*.*

$$weight = \frac{\left\|\mathcal{N}(v_i) \cap \mathcal{N}(\mathcal{N}_{best}(v_i))\right\|}{min\{\|\mathcal{N}(v_i)\|, \|\mathcal{N}(\mathcal{N}_{best}(v_i))\|\}} > \lambda \tag{3}$$

**Definition 3.** *(Function-ii) According to this function $v_i$ combines with $\mathcal{N}_{best}(v_i)$ if the value of average ODF formed by the subgraph of their neighbors is less than the individual average ODF of $v_i$ with $\mathcal{N}(v_i)$ and $\mathcal{N}_{best}(v_i)$ with $\mathcal{N}(\mathcal{N}_{best}(v_i))$ as given by equations* (4) *and* (5)

$$avgODF(\mathcal{S}(v_i)) >= avgODF(\mathcal{S}(s^{(n)})) \tag{4}$$

$$avgODF(\mathcal{S}(\mathcal{N}_{best}(v_i))) >= avgODF(\mathcal{S}(s^{(n)})) \tag{5}$$

The nodes are bound to follow *non-redundant node strategy* when the process of community detection starts. According to this strategy, a node $v_i$ merges with only its prime neighbor given by $\mathcal{N}_{best}(v_i)$ in the current iteration. The same is applicable for $\mathcal{N}_{best}(v_i)$ as well, as it cannot join other nodes/snowballs in the same iteration, i.e., both these nodes are not allowed to join other nodes in the same iteration. This strategy leads to formation of mutually exclusive communities. When nodes join with their best node from the respective neighborhood set, they form snowballs as given by definition 4.

**Definition 4.** *(Snowball). A snowball $s^{(n)}$ is set of connected components formed by enumerating nodes contained in it, where $n$ is the cardinality of the set. It is formed either by merging a node $v_i$ with $\mathcal{N}_{best}(v_i)$ or by joining two or more snowballs.*

It is pertinent to note that the superscript $n$ signifies the number of elements in a snowball. Hence, there can exist many snowballs with a common value of $n$. Nonetheless, they can be distinguished by the elements contained in the set, as these elements are mutually exclusive. These snowballs ($s^{(n)}$) form a subgraph with their immediate neighbors (neighbors of the nodes that are contained in the it).

The set of neighbors for a snowball depicted by $\mathcal{N}(s^{(n)})$ is defined by the union of neighbor set of each node contained in $s^{(n)}$, i.e., $\mathcal{N}(v_1), \mathcal{N}(v_2), ..., \mathcal{N}(v_n)$. A snowball can combine any of the existing snowballs by a given condition which calculates the common nodes among the existing snowball and the newly formed snowball (formed by merging the two snowballs). A snowball is allowed to join any one of the existing snowballs, the one which has the maximum common neighbors with the current snowball. This process keeps continuing until no further snowballs can combine, and the final result is the community set as mentioned in definition 5.

**Definition 5.** *(Community set). A set of community may comprise of a single node or snowballs or both, that cannot be merged any further and have maximum value of (modularity) among all the iterations.*

## IV. Proposed SbChain+

The functional details of the proposed approach for finding communities, called Enhanced **S**now**b**all-**Chain** or SbChain+ are presented in this section, and it is designed for a simple graph, i.e., an undirected and unweighted graph. The inspiration for the approach comes from agglomerative hierarchical clustering which operates in a bottom-up manner, starting with single nodes as individual communities. These nodes expand to form snowballs by finding highly connected neighboring nodes. Snowballs keep adding nodes to form clique-like structures. The snowballs keep expanding till the criteria is met and until the convergence is fulfilled, i.e., the set of communities for a given iteration are identical to the community set from the previous iteration. The community set with the highest value of modularity is the final set of community returned by the algorithm, among all the calculated values from all the iterations.

### A. SbChain+ Algorithm

SbChain+ given by algorithm 4 commences by storing the structural properties of all the nodes, like their respective neighbors, local clustering coefficient, and degree, betweenness centrality, each of which is represented as a set. These sets are stored in the form of key-value pairs by the name of $\mathcal{N}, LCC, k$ and $b$, respectively, where the key is defined by the node and the value varies with the corresponding set values. The loop keeps running for $|V|$, where $V$ represents

**Algorithm 1:** $bestNeighbor(\mathcal{V}, \mathcal{N}(\mathcal{V}), score)$

**Input :** A set $\mathcal{V} \subseteq V$, neighbor list $\mathcal{N}(\mathcal{V})$, $score$ of each node/snowball in the current iteration
**Output:** Best neighbor set of $\mathcal{V}$ i.e, $\mathcal{N}_{best}(\mathcal{V})$

$maxScore \leftarrow 0$
$\mathcal{V}' \leftarrow \emptyset$
**if** $\|\mathcal{N}(\mathcal{V})\| = 0$ **then**
  **return** $\emptyset$
**foreach** $v \in \mathcal{N}(\mathcal{V})$ **do**
  **foreach** *snowball in score* **do**
    // *snowball* is a set of nodes
    **if** *v is a part of snowball* **then**
      $\mathcal{V}' \leftarrow snowball$
    **else**
      continue
    **if** $score(\mathcal{V}') > maxScore$ **then**
      $maxScore \leftarrow score(\mathcal{V}')$
      $\mathcal{N}_{best}(\mathcal{V}) \leftarrow \mathcal{V}'$
**return** $\mathcal{N}_{best}(\mathcal{V})$

**Algorithm 2:** $neighborApproval(i)(\mathcal{N}(\mathcal{V}_1), \mathcal{N}(\mathcal{V}_2), \lambda)$

**Input :** Neighbor set $\mathcal{N}(\mathcal{V}_1)$, neighbor set $\mathcal{N}(\mathcal{V}_2)$, $hyperparameter\lambda$
**Output:** Snowball $s^{(n)}$

$s \leftarrow \emptyset$
// s is a snowball
**if** $\frac{\|\mathcal{N}(\mathcal{V}_1)\&\mathcal{N}(\mathcal{V}_2)\|}{min(\|\mathcal{N}(\mathcal{V}_1)\|, \|\mathcal{N}(\mathcal{V}_2)\|)} >= \lambda$ **then**
  $n \leftarrow \|V_1 \cup V_2\|$
  Add $\langle \mathcal{V}_1, \mathcal{V}_2 \rangle$ to $s^{(n)}$
**return** $s^{(n)}$

the node set in graph $G$. However, it exits the loop when two consecutive iterations result in identical set of communities.

Before the iterations start, an initial score $score^{(1)}$ and neighbor list $\mathcal{N}$ is calculated for each node $v_i$ as a preprocessing step in algorithm 4. The score is calculated as per equation (1) or (2) and signifies the influence of a node/snowball in the graph, and calculated using the connections among its neighbor (given by LCC) and its own connections with the other nodes (given by degree and betweenness centrality). The flag value for each node is set to 0 for every iteration, so that each node can merge once in every iteration with its best neighbor. As each iteration $i$ proceeds, the nodes (or snowballs) represented by the set $\mathcal{V}_j$ from $score^{(i)}$ are arranged in non-increasing order. Each $\mathcal{V}_j$ is checked for a flag value, if the value is set, it means that the given $\mathcal{V}_j$ has merged with some nodes/snowball to form another snowball in the current iteration, it is not processed

**Algorithm 3:** $neighborApproval(ii)(\mathcal{N}(\mathcal{V}_1), \mathcal{N}(\mathcal{V}_2))$

**Input :** Neighbor set $\mathcal{N}(\mathcal{V}_1)$, Neighbor set $\mathcal{N}(\mathcal{V}_2)$
**Output:** Snowball $s^{(n)}$

$s \leftarrow \emptyset$
// s is a snowball
**if** $avgODF(\mathcal{N}(\mathcal{V}_1)) >= avgODF(\mathcal{N}(\mathcal{V}_1) \cup \mathcal{N}(\mathcal{V}_2))$ *and* $avgODF(\mathcal{N}(\mathcal{V}_2)) >= avgODF(\mathcal{N}(\mathcal{V}_1) \cup \mathcal{N}(\mathcal{V}_2))$ **then**
  $n \leftarrow \|V_1 \cup V_2\|$
  Add $\langle \mathcal{V}_1, \mathcal{V}_2 \rangle$ to $s^{(n)}$
**return** $s^{(n)}$

**Algorithm 4:** SbChain+(G, $\lambda$)

**Input :** A graph $G(V, E)$ and threshold $\lambda$
**Output:** Final community set $C$, $Q$, $NMI$

$\forall v_i$, calculate $\mathcal{N}(v_i), score^1(v_i)$
$maxQ \leftarrow -1, m \leftarrow |E|, sScore \leftarrow score^1$
**for** $i \leftarrow 1$ *to* $|V|$ **do**
  Arrange $score^i$ in non-increasing order
  $\forall v_i$, $flag(v_i) \leftarrow 0$
  **foreach** $\mathcal{V}_j \in score^i.keys$ **do**
    // $\mathcal{V}_j$ is a set of nodes or snowballs
    **if** $flag(\mathcal{V}_j) = 1$ **then**
      continue
    $\mathcal{V}' \leftarrow bestNeighbor(\mathcal{V}_j, \mathcal{N}(V_j), score^i)$
    // $\mathcal{V}'$ is $\mathcal{N}_{best}(V_j)$
    **if** $\mathcal{V}' = \emptyset$ **then**
      Add $\mathcal{V}_j$ to community
      continue
    **if** $flag(\mathcal{V}') = 1$ **then**
      continue
    $s^{(n)} \leftarrow neighborApproval(i)(\mathcal{N}(\mathcal{V}_j), \mathcal{N}(\mathcal{V}'), \lambda)$
    **if** $s^{(n)} = \emptyset$ **then**
      continue
    $maxInter \leftarrow 0$, $flag(\mathcal{V}_j), flag(\mathcal{V}') \leftarrow 1$
    $sScore(s^{(n)}), setflag \leftarrow 0$
    **foreach** $s \in s^{(n)}$ **do**
      $sScore(s^{(n)}) \leftarrow sScore(s^{(n)}) + sScore(s)$
    $score^i(s^{(n)}) \leftarrow \frac{sScore(s^{(n)})}{\|n\|}$
    **for** $j \leftarrow 1$ *to* $|comm|$ **do**
      $weight \leftarrow \frac{\|s^{(n)}\&comm(j)\|}{min(\|s^{(n)}\|, \|comm(j)\|)}$
      **if** $weight > maxInter$ **then**
        $maxInter \leftarrow weight$
        $setflag \leftarrow 1$, $saveIndex \leftarrow j$
      **else**
        $counter \leftarrow counter + 1$
      **if** $j = |comm|$ *and* $setflag = 1$ **then**
        $comm(j) \leftarrow comm(j) \cup s^{(n)}$
        $sScore(comm(j)) \leftarrow sScore(comm(j)) + sScore(s^{(n)})$
        $score^{i+1} \leftarrow \frac{sScore(comm(j))}{\|sScore(comm(j))\|}$, $score^i \leftarrow \frac{sScore(s^{(n)})}{\|n\|}$
        $score^i.pop(\mathcal{V}_j), score^i.pop(\mathcal{V}'), score^{i+1}.pop(s^{(n)})$
    **if** $counter = |comm|$ **then**
      $counter \leftarrow 0$
      Add $s^{(n)}$ to comm
      $score^i, score^{i+1} \leftarrow \frac{sScore(s^{(n)})}{\|sScore(s^{(n)})\|}$
      $score^i.pop(\mathcal{V}_j), score^i.pop(\mathcal{V}')$
  Copy keys from $score^i$ to $score^{i+1}$ and $comm$ which were not updated
  $Q \leftarrow Modularity(m, comm, E), NMI \leftarrow NMI(comm, GT)$
  **if** $maxQ < Q$ **then**
    $maxQ \leftarrow Q, maxNMI \leftarrow NMI$
  **if** $score^i.keys = score^{i-1}.keys$ **then**
    break
**return** $community, maxNMI, maxQ$

any further and the iteration continues with the next set in the order. Hence, as per the non-redundant node strategy, the algorithm jumps to the next best $\mathcal{V}_j$. The best neighbor for a node $\mathcal{V}_j$ is represented by $\mathcal{N}(\mathcal{V}_j)$ is returned by the algorithm 1. It should be noted that both $V_j$ and $\mathcal{N}_{best}(\mathcal{V}_j)$ are sets; therefore, they can contain more than one node. If these sets contain more than one node then they are called a snowball. Hence, it is represented as a set $\mathcal{V}'$. It should be noted that if $\mathcal{V}'$ is an empty set, therefore, $\mathcal{V}_j$ is a isolated node and forms a community of its own. Next, $\mathcal{V}'$ is also verified further for non-redundant node strategy, by checking its respective flag value. Further, a node joins its best neighbor based on two neighbor approval functions as discussed further.

*1)* $neighborApproval(i)$*:* It is should be noted that we discuss two functions for approval of the best neighbor for a given node as given by algorithm 2 and algorithm 3. According to algorithm 2 called $neighborApproval(i)$, both the sets, i.e., $\mathcal{N}(\mathcal{V}_1)$ and $\mathcal{N}(\mathcal{V}_2)$, are checked for overlapping criteria using the hyperparameter $\lambda$. This parameter lays the minimum value of overlap that should exist for two snowballs to combine. If their *weight* given by (1) is greater than or equal to the $\lambda$ value, then the sets merge to form a snowball $s^{(n)}$ which is returned by the algorithm else it returns an empty set.

*2)* $neighborApproval(ii)$*:* In algorithm 3 $neighborApproval(ii)$, the average out degree function of the original two sets given by $\mathcal{N}(\mathcal{V}_1)$ and $\mathcal{N}(\mathcal{V}_2)$ and their union $(\mathcal{V}_1 \cup \mathcal{V}_2)$ is calculated as per equation (6) given by [38].

$$avgODF(C) = \frac{1}{n_C} \sum_{u \in C} \frac{|\{(u,v) \in E : v \notin C\}|}{k_u} \tag{6}$$

It gives the average of the number of outgoing edges for each community node as compared to the total edges incident on that node. Therefore, if the value of $avgODF$ is smaller for the combined snowball (than the initial two sets), $s^{(n)}$ is returned else an empty set is returned.

Considering $neighborApproval(i)$ in algorithm 4, if a snowball $s^{(n)}$ is returned then the flag for all the nodes in the snowball is set to one. Therefore, these nodes cannot combine with other nodes in the current iteration, leading to formation of disjoint communities. Further, the score of the snowball is calculated by taking an average of the score of the elements in the snowball. It should be noted that a new snowball is allowed to merge with an existing snowball on a criteria stating that they have maximum overlap among all the existing snowballs, given by a parameter called $weight$ as mentioned in step 15. A snowball is allowed to join only one existing snowball, i.e., the one with the maximum common nodes. $score^i$ and $score^{i+1}$ are updated to include the average score values for of snowball and merged snowball, respectively. If the new snowball does not combine with any existing snowball, then it forms a community of its own as per step 34. The nodes that did not merge with other nodes/snowballs are copied from $score^i$ to $score^{i+1}$ and $comm$. In every iteration, modularity and NMI (using ground truth GT) are calculated. This process continues till two consecutive iterations return an identical community set. The final set of communities is returned along with the respective modularity and NMI values, as per the algorithm has the maximum modularity value among all the iterations.

## V. Discussion

The hyperparameter ($\lambda$) used in this study determines the limit of common nodes that should exist between two subgroups to merge them to form a snowball. This parameter is decided upon empirically, but it is guided by the ratio of edges to nodes in the social graph. It is also given by half of the average degree ($k_{avg}$) of the nodes, as given in Table XI. It can be observed from this table that $k_{avg} > 19$ for all LFR datasets. This signifies that the nodes are densely connected in the graph and can merge on a low $\lambda$ value. Setting a high value of $\lambda$ would result in joining of all the nodes, and eventually leading towards the formation of a single community. Therefore, it is observed that the results are best around $\lambda \leq 0.6$. Similarly, for lower values of average degree (i.e., $k_{avg} < 5$), a higher percentage of overlap is required because the neighbors (or $k(v_i)$) of two nodes/subgroups to be merged can be very low. The low degree nodes can easily be merged at lower $\lambda$ values, leading to the formation of a single community in the entire dataset. Hence, based on our experiments, the suitable value for the $\lambda$ hyperparameter is, $\lambda > 0.6$.

It should be noted that `SbChain+(i)` outperforms `SbChain+(ii)` in terms of the quality of the identified communities. The reason for this is that the $\lambda$ parameter allows a node or snowball to join other nodes or snowballs because the overlap among them keeps increasing with each iteration. Whereas, average ODF in case of `SbChain+(ii)` has stricter rules for merging. Therefore, the overlap among snowballs is not as much as in the case of former technique, i.e., the growth of the communities is comparatively slower in latter. The drawback associated with the former technique is that, although the $\lambda$ value is guided by the ratio of number of edges to the number of nodes, there is a scope of error associated with it. While, the shortcoming of `SbChain+(ii)` is that it produces average results since it allows a node to merge with its best neighbor based on increasing the density of edges inside a community than outside.

`SbChain+` improves upon `SbChain` by changing the the seed function to include various centralities, discussing two neighbor finding functions (with and without parameters). It can be seen that both `SbChain+(i)` and `SbChain+(ii)` outperform `SbChain` in terms of the quality of the identified community. Although, `SbChain+` shows comparable or good results for real-world datasets, it needs to be checked for large networks.

## VI. Experimental Setup and Results

This section describes the results and their analysis obtained by applying `SbChain+` approach with two different functions, $neighborApproval(i)$ and $neighborApproval(ii)$ on various datasets, namely, `SbChain+(i)` and `SbChain+(ii)`, respectively. The efficacy of these two techniques is verified using six real-world datasets and two sets of computer-generated Lancichinetti Fortunato Radicchi (LFR) benchmark datasets having 1K and 5K nodes, respectively. The details of the datasets are presented in the following subsections. Modularity (Q) and Normalized

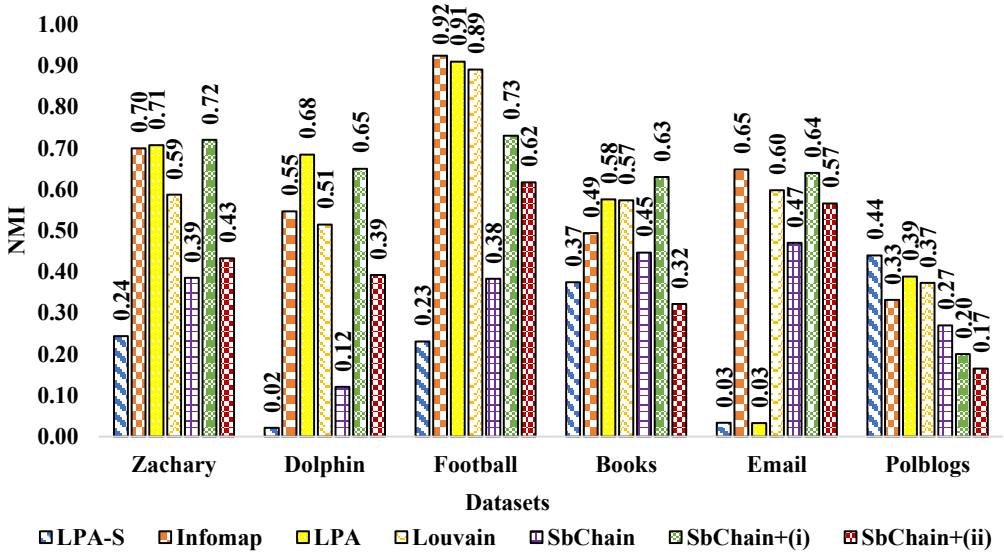


Fig. 1. Visualization of the performance comparison results of SbChain+ with state-of-the-art methods over real-world datasets in terms of NMI.

Mutual Information (NMI) are used to assess the quality of communities formed by SbChain+ and the other techniques. It should be noted that the values marked with * represent a very small number and is rounded off to 0.00.

TABLE II. STATISTICS OF REAL-WORLD NETWORKS

| Dataset | #Nodes | #Edges | #Communities |
|---|---|---|---|
| Karate | 34 | 78 | 2 |
| Dolphin | 62 | 159 | 2 |
| Polbooks | 105 | 441 | 3 |
| Football | 115 | 613 | 12 |
| Email [39] | 1005 | 16064 | 42 |
| Polblogs [40] | 1490 | 16715 | 2 |

### A. Real-World Datasets

The real-world datasets used in our experiment are briefly summarized in Table II[1]. The performance evaluation and comparison results of SbChain+(i) and SbChain+(ii) with some of the state-of-the-art methods, including Infomap [8], Label Propagation Algorithm (LPA)) [9], LPA (semi-synchronous) [10], Louvain [11], and our previously developed SbChain, in terms of Normalized Mutual Information (NMI), and modularity (Q) are shown in Tables III and IV, respectively.

It can be observed from the Tables III and IV that SbChain+ performs significantly better or comparably most real-world datasets. The average NMI produced by SbChain+(i) is comparable to the average NMI by Infomap among all the techniques. It is pertinent to note that both SbChain+ techniques are seen to perform better than SbChain as shown in Fig. 1 and 2. It should also be noted that SbChain and LPA techniques produce different result every time they are run.

[1]http://www-personal.umich.edu/~mejn/netdata/

### B. Synthetic Datasets

As described in [41], Lancichinetti-Fortunato-Radicchi (LFR) benchmark networks are used to generate synthetic datasets by tuning different parameters. In our experiments, LFR datasets with 1K and 5K nodes are generated using the parameter values presented in Tables V and VI, respectively. These datasets are denoted as LFR-1K and LFR-5K datasets, respectively. Out of these parameters, $\mu$ is the mixing parameter and defines the number of connections with neighbors in other communities. The value of $\mu$ is set within the range of $[0.1, 0.5]$ for LFR-1K and LFR-5K, and the step-size is changed at an interval of $0.1$ for both LFR-1K and LFR-5K datasets. It controls the percentage of edges between communities. By changing the $\mu$ values in the given range, different datasets are created and accordingly named as LFR-1K.1 – LFR-1K.5 and LFR-5K.1 – LFR-5K.5. Values upto $0.5$ are considered for $\mu$ parameter as the modular structure of a community becomes fuzzy beyond this value.

The empirical evaluation and comparison results of SbChain+ with the aforementioned techniques in terms of NMI, and modularity (Q) are presented in Tables VII-VIII for LFR-1K dataset, and in Tables IX-X for LFR-5K dataset. These tables present the varying $\mu$ in the range of $[0.1, 0.5]$ for LFR-1K and LFR-5K datasets, with the respective NMI, and Q for all the approaches.

It can be seen that our technique gives average results in most of the cases, The NMI for SbChain+ is LFR-1K as can be seen from Table VII is seen to outperform SbChain, LPA(SS). And, in general SbChain+(i) produces better results than SbChain+(ii). It is also seen that modularity values for LFR-1K from VIII are seen to be average, i.e., Infomap, Louvain and LPA produce a better modularity value than SbChain+. Fig. 3 and 4 show comparison charts for LFR-1K datasets in terms of NMI and modularity.

SbChain+(i) performs averagely in for LFR-5K datasets in terms of NMI and modularity, as seen from Tables

TABLE III. PERFORMANCE COMPARISON OF SBCHAIN+ WITH STATE-OF-THE-ART METHODS OVER REAL-WORLD DATASETS IN TERMS OF NMI

| Datasets | Community Detection Methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | Infomap | LPA(SS) | Louvain | LPA | SbChain | SbChain+(i) | SbChain+(ii) |
| Karate | 0.70 | 0.24 | 0.59 | 0.71 | 0.39(0.3) | 0.72(0.8) | 0.43 |
| Dolphin | 0.55 | 0.02 | 0.51 | 0.68 | 0.12(0.3) | 0.65(0.7) | 0.39 |
| Football | 0.92 | 0.23 | 0.89 | 0.91 | 0.38(0.8) | 0.73(0.7) | 0.62 |
| Polbooks | 0.49 | 0.37 | 0.57 | 0.58 | 0.45(0.6) | 0.63(0.7) | 0.32 |
| Email | 0.65 | 0.03 | 0.60 | 0.03 | 0.47(0.4) | 0.64(0.8) | 0.57 |
| Polblogs | 0.33 | 0.44 | 0.37 | 0.39 | 0.26(0.8) | 0.20(0.6) | 0.17 |

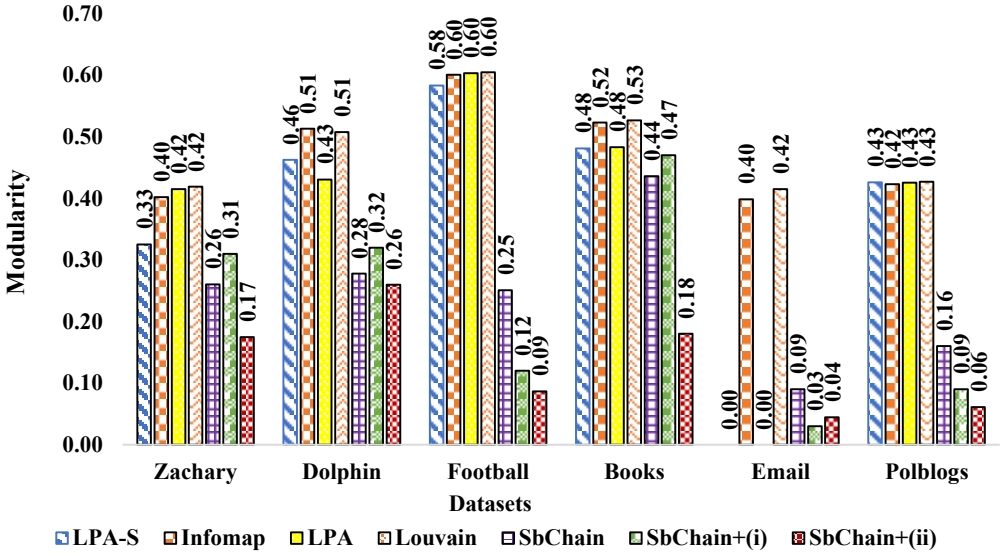


Fig. 2. Visualization of the performance comparison results of SbChain+ with state-of-the-art methods over real-world datasets in terms of modularity.

TABLE IV. PERFORMANCE COMPARISON OF SBCHAIN+ WITH STATE-OF-THE-ART METHODS OVER REAL-WORLD DATASETS IN TERMS OF MODULARITY (MOD.)

| Datasets | Ground truth | Community Detection Methods | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | Infomap | LPA(SS) | Louvain | LPA | SbChain | SbChain+(i) | SbChain+(ii) |
| Karate | 0.37 | 0.40 | 0.32 | 0.42 | 0.32 | 0.26 | 0.31 | 0.17 |
| Dolphin | 0.37 | 0.51 | 0.46 | 0.50 | 0.41 | 0.28 | 0.32 | 0.26 |
| Football | 0.55 | 0.60 | 0.58 | 0.60 | 0.58 | 0.25 | 0.12 | 0.09 |
| Polbooks | 0.41 | 0.52 | 0.48 | 0.52 | 0.48 | 0.44 | 0.47 | 0.18 |
| Email | 0.29 | 0.39 | 0.00* | 0.40 | 0.00 | 0.08 | 0.03 | 0.04 |
| Polblogs | 0.40 | 0.42 | 0.43 | 0.42 | 0.42 | 0.15 | 0.09 | 0.06 |

TABLE V. PARAMETERS USED TO GENERATE LFR-1K NETWORK

| Parameter | Value |
|---|---|
| Nodes (N) | 1000 |
| Average degree ($\langle k \rangle$) | 20 |
| Minimum community size ($c_{min}$) | 20 |
| Maximum community size ($c_{max}$) | 100 |
| Maximum degree ($k_{max}$) | 50 |
| Community size distribution exponent ($\beta$) | 1 |
| Degree distribution exponent ($\gamma$) | 2 |
| Mixing parameter ($\mu$) | [0.1, 0.5] |

TABLE VI. PARAMETERS USED TO GENERATE LFR-5K NETWORK

| Parameter | Value |
|---|---|
| Nodes (N) | 5000 |
| Average degree ($\langle k \rangle$) | 20 |
| Minimum community size ($c_{min}$) | 50 |
| Maximum community size ($c_{max}$) | 100 |
| Maximum degree ($k_{max}$) | 50 |
| Community size distribution exponent ($\beta$) | 1 |
| Degree distribution exponent ($\gamma$) | 2 |
| Mixing parameter ($\mu$) | [0.1, 0.5] |

TABLE VII. PERFORMANCE COMPARISON OF SBCHAIN+ WITH STATE-OF-THE-ART METHODS OVER LFR-1K DATASETS IN TERMS OF NMI

| Datasets | Community Detection Methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | Infomap | LPA(SS) | Louvain | LPA | SbChain | SbChain+(i) | SbChain+(ii) |
| LFR-1K.1 | 1.00 | 0.08 | 1.00 | 1.00 | 0.19(0.4) | 0.69(0.6) | 0.47 |
| LFR-1K.2 | 1.00 | 0.07 | 1.00 | 1.00 | 0.27(0.5) | 0.65(0.6) | 0.73 |
| LFR-1K.3 | 0.08 | 0.08 | 0.08 | 0.08 | 0.27(0.4) | 0.48(0.8) | 0.28 |
| LFR-1K.4 | 1.00 | 0.05 | 1.00 | 1.00 | 0.24(0.6) | 0.59(0.6) | 0.43 |
| LFR-1K.5 | 1.00 | 0.00 | 1.00 | 0.96 | 0.31(0.4) | 0.61(0.6) | 0.48 |

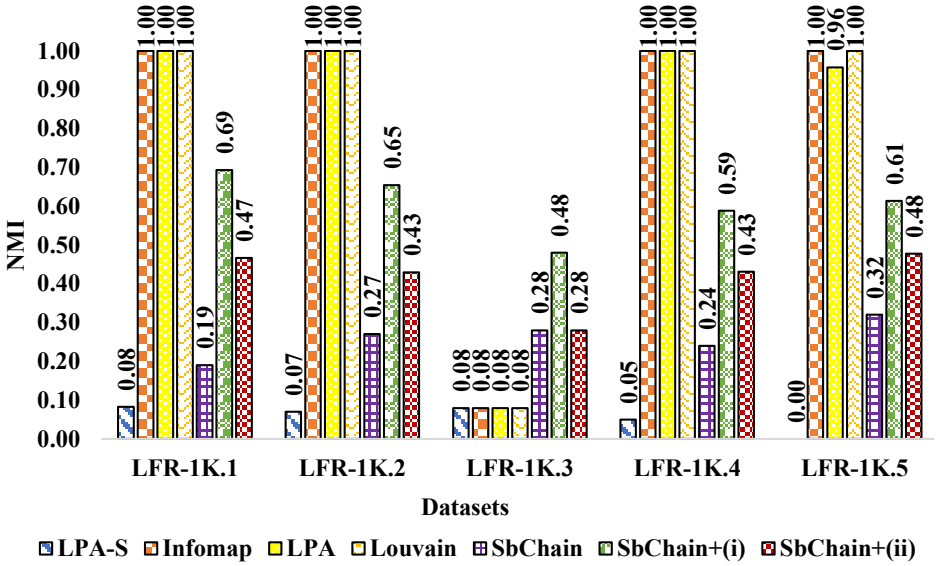


Fig. 3. Visualization of the performance comparison results of SbChain+ with state-of-the-art methods over LFR-1K datasets in terms of NMI.

TABLE VIII. PERFORMANCE COMPARISON OF SBCHAIN+ WITH STATE-OF-THE-ART METHODS OVER LFR-1K DATASETS IN TERMS OF MODULARITY

| Datasets | Community Detection Methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | Infomap | LPA(SS) | Louvain | LPA | SbChain | SbChain+(i) | SbChain+(ii) |
| LFR-1K.1 | 0.83 | 0.83 | 0.83 | 0.83 | 0.66 | 0.22 | 0.19 |
| LFR-1K.2 | 0.74 | 0.74 | 0.74 | 0.74 | 0.38 | 0.14 | 0.11 |
| LFR-1K.3 | 0.83 | 0.83 | 0.83 | 0.83 | 0.50 | 0.15 | 0.19 |
| LFR-1K.4 | 0.53 | 0.44 | 0.53 | 0.53 | 0.18 | 0.01 | 0.07 |
| LFR-1K.5 | 0.45 | 0.00 | 0.45 | 0.43 | 0.17 | 0.00 | 0.05 |

TABLE IX. PERFORMANCE COMPARISON OF SBCHAIN+ WITH STATE-OF-THE-ART METHODS OVER LFR-5K DATASETS IN TERMS OF NMI

| Datasets | Community Detection Methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | Infomap | LPA(SS) | Louvain | LPA | SbChain | SbChain+(i) | SbChain+(ii) |
| LFR-5K.1 | 1.00 | 0.12 | 1.00 | 1.00 | 0.15(0.7) | 0.76(0.6) | 0.55 |
| LFR-5K.2 | 1.00 | 0.13 | 1.00 | 1.00 | 0.37(0.8) | 0.71(0.6) | 0.55 |
| LFR-5K.3 | 1.00 | 0.13 | 1.00 | 1.00 | 0.38(0.7) | 0.67(0.6) | 0.56 |
| LFR-5K.4 | 1.00 | 0.10 | 1.00 | 1.00 | 0.38(0.7) | 0.66(0.6) | 0.53 |
| LFR-5K.5 | 1.00 | 0.09 | 0.98 | 1.00 | 0.39(0.7) | 0.66(0.6) | 0.53 |

IX and X, respectively. Although, it shows better results than LPA(SS), SbChain and SbChain+(ii). Fig. 5 and 6 show both the NMI and modularity values produced by all the techniques.

Therefore, it is seen that for real-world datasets SbChain+ is seen to perform better in terms of the identified communities as compared to the synthetic datasets. It should be seen that the average NMI produced by real-world datasets is at par with the other techniques results, and performs produces average results for LFR-1K and LFR-5K.

## VII. COMPLEXITY ANALYSIS

The best-case of the algorithm arises when all the nodes join different nodes/snowballs in each iteration, i.e., no node is left free in any iteration. This leads to a minimum of $log\ n$ iterations, where the number of nodes/snowballs also reduces to its half from the previous iteration. Therefore, a time complexity of $O(n)$ defines the best-case. The worst-case arises when only a single node joins another node/snowball in an iteration. This leads to $n-1$ iterations and the time complexity goes to $O(n^2)$. It is also evident that SbChain+ works well on both small and large real-world datasets in terms of the identified communities. However, on LFR datasets, it

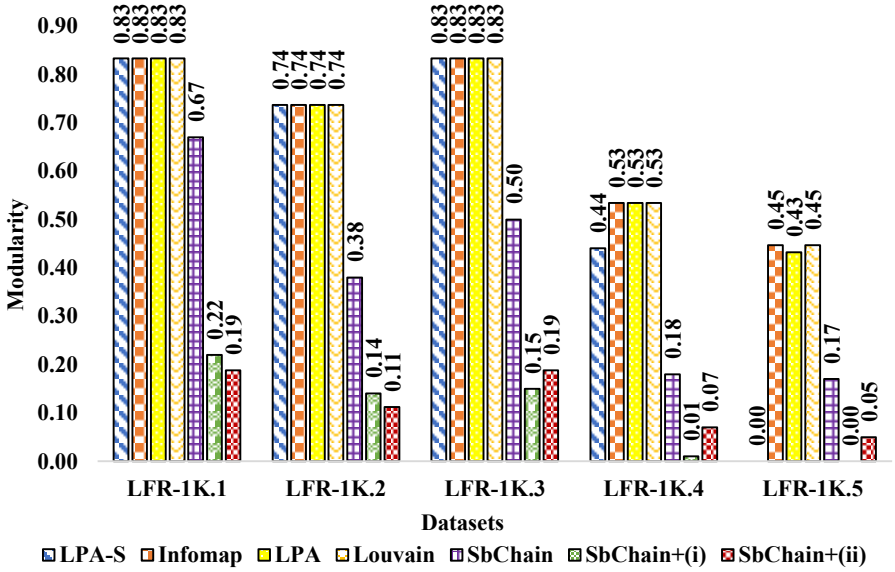


Fig. 4. Visualization of the performance comparison results of SbChain+ with state-of-the-art methods over LFR-1K datasets in terms of modularity.

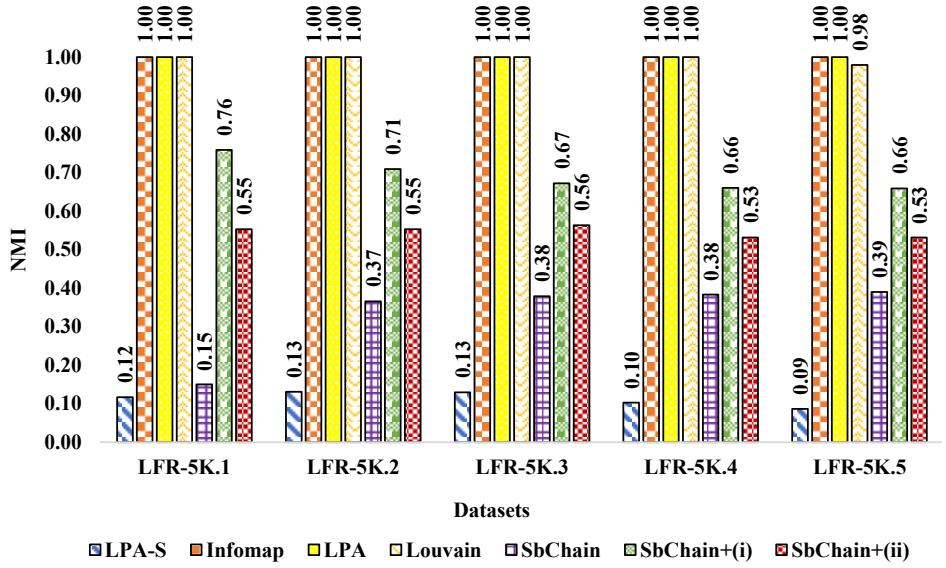


Fig. 5. Visualization of the averaged performance comparison results of SbChain+ with state-of-the-art methods over LFR-5K datasets in terms of NMI.

TABLE X. PERFORMANCE COMPARISON OF SBCHAIN+ WITH STATE-OF-THE-ART METHODS OVER LFR-5K DATASETS IN TERMS OF MODULARITY

| Datasets | Community Detection Methods | | | | | | |
|---|---|---|---|---|---|---|---|
| | Infomap | LPA(SS) | Louvain | LPA | SbChain | SbChain+(i) | SbChain+(ii) |
| LFR-5K.1 | 0.88 | 0.88 | 0.88 | 0.88 | 0.39 | 0.23 | 0.09 |
| LFR-5K.2 | 0.78 | 0.78 | 0.78 | 0.78 | 0.20 | 0.09 | 0.09 |
| LFR-5K.3 | 0.68 | 0.68 | 0.68 | 0.68 | 0.17 | 0.02 | 0.06 |
| LFR-5K.4 | 0.58 | 0.58 | 0.58 | 0.58 | 0.14 | 0.00* | 0.06 |
| LFR-5K.5 | 0.48 | 0.44 | 0.48 | 0.48 | 0.12 | 0.00* | 0.05 |

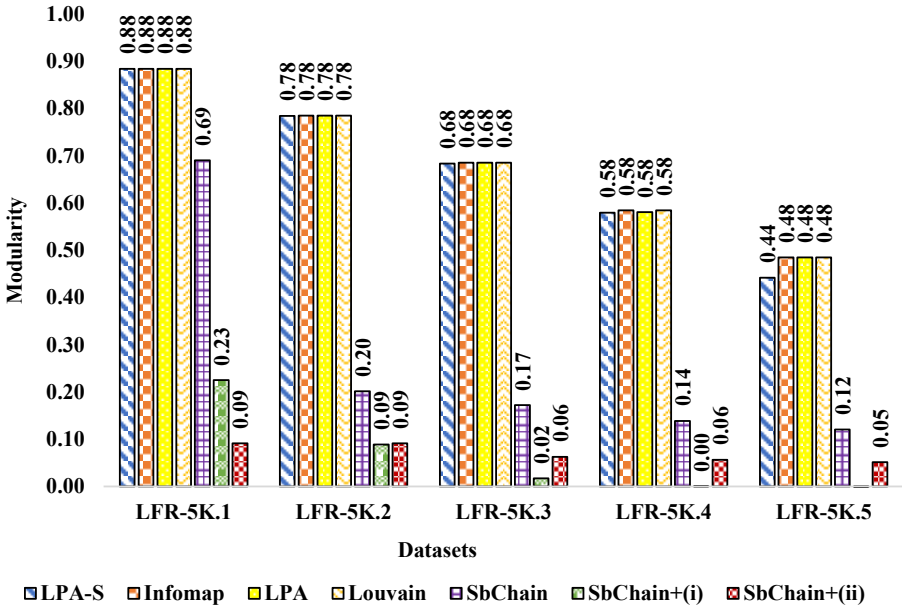


Fig. 6. Visualization of the performance comparison results of SbChain+ with state-of-the-art methods over LFR-5K datasets in terms of modularity.

TABLE XI. HYPERPARAMETER $\lambda$ VALUES

| Average degree | $\lambda$ |
|---|---|
| $< 5$ | $> 0.6$ |
| $> 10$ | $<= 0.6$ |

is seen to give average performance when compared to the best performing community detection methods. It should be noted that in comparison to LPA(SS) and SbChain, both, SbChain+(i) and SbChain+(ii) show better performance in terms of the identified communities.

## VIII. CONCLUSION AND FUTURE WORK

In this paper we have presented two approaches for enhanced snowball-chain approach, SbChain+ for detecting communities in social graph. In general, SbChain+ lays emphasis on finding nodes that have a high degree of interaction with its neighbors and a densely connected neighborhood. This reveals the core nodes, i.e., the nodes that may be a part of a clique and would further contribute towards formation of snowballs and eventually a community. SbChain+ improves over SbChain in terms of cardinality of the identified communities, NMI, and modularity. In SbChain+(i), this is achieved by changing the weight function, which is based on maximizing the intersection of the neighbors between two nodes using a $\lambda$ hyperparameter. The hyperparameter ($\lambda$) which defines the minimum overlap required for two snowballs to get merged for community formation. This parameter also helps in refinement of the communities – the higher the value of $\lambda$, higher is the cohesion. The low value of $\lambda$ gives higher coupling. On the other hand, SbChain+(ii) focusses on average ODF and does not use any hyperparameter and hence, detects better communities than SbChain. Whereas, SbChain focussed on maximizing both the common neighbors as well as the score between the nodes; by relaxing this criteria, the results are seen to be better than those given by SbChain.

Further, SbChain+ method can be extended and improved upon by making it a generic framework for finding communities closer to ground truth, in both simple and weighted/directed social graphs. Also, the technique can accommodate identifying dynamic communities based on the time-varying functions. This can be seen as a promising future direction of research.

## REFERENCES

[1] I. Himelboim, M. Smith, L. Rainie, B. Shneiderman, and C. Young, "Classifying twitter topic-networks using social network analysis," *Social Media + Society*, vol. 3, pp. 1–13, March 2017.

[2] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 99, no. 12, pp. 7821–7826, June 2002.

[3] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD), Portland, Oregon*, vol. 96, no. 34, August 1996, pp. 226–231.

[4] S. Y. Bhat and M. Abulaish, "A density-based approach for mining overlapping communities from social network interactions," in *Proceedings of the 2nd International Conference on Web Intelligence, Mining and Semantics (WIMS), Craiova, Romania*, June 2012, pp. 1–7.

[5] ——, "OCTracker: A density-based framework for tracking the evolution of overlapping communities in OSNs," in *Proceedings of 2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Istanbul, Turkey*, August 2012, pp. 501–505.

[6] ——, "HOCTracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 4, pp. 1019–1031, August 2014.

[7] J. Gulati and M. Abulaish, "A novel snowball-chain approach for detecting community structures in social graphs," in *Proceedings of the IEEE Symposium Series on Computational Intelligence (SSCI), Xiamen, China*, December 2019, pp. 2462–2469.

[8] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex networks reveal community structure," *Proceedings of the National Academy of Sciences (PNAS)*, vol. 105, no. 4, pp. 1118–1123, February 2008.

[9] U. N. Raghavan, R. Albert, and S. Kumara, "Near linear time algorithm to detect community structures in large-scale networks," *Physical Review E*, vol. 76, no. 3, pp. 036 106–(1–11), October 2007.

[10] G. Cordasco and L. Gargano, "Community detection via semi-synchronous label propagation algorithms," in *Proceedings of IEEE International Workshop on Business Applications of Social Network Analysis (BASNA), Bangalore, India*, January 2010, pp. 1–8.

[11] V. D. Blondel, J. L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, pp. P10 008–(1–12), January 2008.

[12] D. Singh and R. Garg, "Ni-louvain: A novel algorithm to detect overlapping communities with influence analysis," *Journal of King Saud University - Computer and Information Sciences*, July 2021.

[13] D. Jin, Z. Jin, P. Jiao, S. Pan, D. He, J. Wu, P. Yu, and W. Zhang, "A survey of community detection approaches: From statistical modeling to deep learning," *IEEE Transactions on Knowledge and Data Engineering*, pp. 1–22, January 2021.

[14] X. Su, S. Xue, F. Liu, J. Wu, J. Yang, C. Zhou, W. Hu, C. Paris, S. Nepal, D. Jin, Q. Z. Sheng, and P. S. Yu, "A comprehensive survey on community detection with deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, March 2022.

[15] P. G. Sun, L. Gao, and Y. Yang, "Maximizing modularity intensity for community partition and evolution," *Information Sciences*, vol. 236, pp. 83–92, March 2013.

[16] M. E. J. Newman and M. Girvan, "Finding and evaluating community structure in networks," *Physical Review E*, vol. 69, no. 2, pp. 026 113–(1–15), March 2004.

[17] M. E. J. Newman, "Fast algorithm for detecting community structure in networks," *Physical Review E*, vol. 69, no. 6, pp. 066 133–(1–5), June 2004.

[18] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the Society for Industrial and Applied Mathematics International Conference on Data Mining (SIAM-ICDM), Newport Beach, California*, April 2005, pp. 274–285.

[19] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proceedings of ACM Sigmod Record, New York, United States*, vol. 28, no. 2, June 1999, pp. 49–60.

[20] X. Xu, N. Yuruk, Z. Feng, and T. A. Schweiger, "SCAN: A structural clustering algorithm for networks," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, USA*, August 2007, pp. 824–833.

[21] T. S. Evans, "Clique graphs and overlapping communities," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 12, pp. 12 037–12 057, December 2010.

[22] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the overlapping community structure of complex networks in nature and society," *Nature*, vol. 435, no. 7043, pp. 814–818, June 2005.

[23] H. Shen, X. Cheng, K. Cai, and M.-B. Hu, "Detect overlapping and hierarchical community structure in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 8, pp. 1706–1712, April 2009.

[24] H. Sun, J. Liu, J. Huang, G. Wang, Z. Yang, Q. Song, and X. Jia, "CenLP: A centrality-based label propagation algorithm for community detection in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 436, pp. 767–780, May 2015.

[25] W. Zhang, R. Zhang, R. Shang, and L. Jiao, "Weighted compactness function based label propagation algorithm for community detection," *Physica A: Statistical Mechanics and its Applications*, vol. 49, pp. 767 –780, February 2018.

[26] M. Tasgin and H. O. Bingol, "Community detection using boundary nodes in complex networks," *Physica A: Statistical Mechanics and its Applications*, vol. 513, pp. 315–324, February 2019.

[27] S. Bilal and M. Abdelouahab, "Evolutionary algorithm and modularity for detecting communities in networks," *Physica A: Statistical Mechanics and its Applications*, vol. 473, pp. 89–96, February 2017.

[28] M. Tasgin and H. O. Bingol, "Community detection using preference networks," *Physica A: Statistical Mechanics and its Applications*, vol. 495, pp. 126–136, August 2018.

[29] P. G. Lind, L. R. da Silva, J. S. A. Jr, and H. J. Herrmann, "Spreading gossip in social networks," *Physical Review E*, vol. 76, no. 3, pp. 036 117–(1–10), October 2007.

[30] F. D. Zarandi and M. K. Rafsanjani, "Community detection in complex networks using structural similarity," *Physica A: Statistical Mechanics and its Applications*, vol. 503, pp. 882–891, August 2018.

[31] C. Panagiotakis, H. Papadakis, and P. Fragopoulou, "Flowpro: A flow propagation method for single community detection," in *Proceedings of the 11th IEEE Consumer Communications and Networking Conference (CCNC), Las Vegas, NV*, January 2014, pp. 17–22.

[32] S. Li, L. Jiang, X. Wu, W. Han, D. Zhao, and Z. Wang, "A weighted network community detection algorithm based on deep learning," *Applied Mathematics and Computation*, vol. 401, pp. 126 012–126 020, July 2021.

[33] D. He, Y. Song, D. Jin, Z. Feng, B. Zhang, Z. Yu, and W. Zhang, "Community-centric graph convolutional network for unsupervised community detection," in *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence (IJCAI), Virtual, Montreal*, August 2021, pp. 3515–3521.

[34] X. Zhao, J. Liang, and J. Wang, "A community detection algorithm based on graph compression for large-scale social networks," *Information Sciences*, vol. 551, pp. 358–372, November 2020.

[35] Y. Zhang, Y. Xiong, Y. Ye, T. Liu, W. Wang, Y. Zhu, and P. S. Yu, "SEAL: Learning heuristics for community detection with generative adversarial networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA*. Association for Computing Machinery, August 2020, pp. 1103–1113.

[36] M. Rostami, K. Berahmand, and S. Forouzandeh, "A novel community detection based genetic algorithm for feature selection," *Journal of Big Data*, pp. 1–27, January 2021.

[37] K. Guo, X. Huang, L. Wu, and Y. Chen, "Local community detection algorithm based on local modularity density," *Applied Intelligence*, pp. 1238–1253, January 2022.

[38] G. W. Flake, S. Lawrence, and C. L. Giles, "Efficient identification of web communities," in *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), Boston, USA*, August 2000, pp. 150–160.

[39] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Transactions on Knowledge and Discovery of Data*, pp. 2–41, March 2007.

[40] L. A. Adamic and N. Glance, "The political blogosphere and the 2004 U.S. election: Divided they blog," in *Proceedings of the 3rd International Workshop on Link Discovery, Chicago, Illinois*. Association for Computing Machinery, August 2005, pp. 36–43.

[41] A. Lancichinetti, S. Fortunato, and F. Radicchi, "Benchmark graphs for testing community detection algorithms," *Physical Review E*, vol. 78, no. 4, pp. 046 110–(1–5), November 2008.