

Scaling Density-Based Community Detection to Large-Scale Social Networks via MapReduce Framework

Muhammad Abulaish^a, Ishfaq Majid Bhat^b and Sajid Yousuf Bhat^{c,*}

^a *Department of Computer Science, South Asian University, Delhi, India, E-mail: abulaish@sau.ac.in*

^b *Department of Information Technology, Central University of Kashmir, J&K, India, E-mail: bhatishfaq005@gmail.com*

^c *Department of Computer Sciences, University of Kashmir, J&K, India, E-mail: bhatsajid@uok.edu.in*

Abstract. Community detection from networks is one of the long standing and challenging tasks in the field of complex network research. Detection of communities poses numerous challenges in terms of their overlapping and hierarchical nature, dynamics of networks and underlying communities, scalability of detection algorithms on large scale networks to mention a few. Traditional community detection methods are not readily scalable to large networks mainly due to the computation of global network metrics. This paper presents a novel scalable overlapping community detection approach for large scale networks by presenting a MapReduce framework based implementation of a density-based local community detection method. The method is divided in two stages where the first stage uses a MapReduce approach to identify a mutual-core connected subgraph of the underlying network. The second stage uses an existing connected component detection method, implemented via MapReduce, to identify connected components in the mutual-core connected subgraph generated in the first stage. A community is then taken as the union of the core-nodes in a connected component and the respective density-based neighborhood of each core-node in the connected component. The resulting approach is among the first scalable overlapping community detection methods proposed in literature.

Keywords: Community Detection, Overlapping Community, Connected Components, MapReduce, Large-Scale Networks

1. Introduction

Network systems are ubiquitous in nature and society, and form the basic structure representing interactions among various related entities. Some important network systems include (i) real-world social networks like human proximity networks, friendship networks, terror networks, and crime/gang networks, (ii) biological networks like protein-protein interaction networks and gene regulatory networks (iii) computer and computer-generated networks like Internet and WWW (iv) online social networks like Facebook, Twitter, and LinkedIn (v)

financial networks like banking transaction networks (vi) road networks (vi) power-grid networks, and (vii) telecommunication networks like mobile call graphs.

Social Network analysis (SNA) is a multi-disciplinary field dedicated to the analysis and modelling of relations and diffusion processes between various objects of network structures found in nature and society, and other information/knowledge processing entities. The aim of SNA is to understand how the behaviour and interaction of such entities translate to large-scale social network systems. SNA is one of the important techniques which finds significant application in anthropology, physics, mathematics, biology, communication networks, economics, geography, and computing. The application of SNA includes but not

*Corresponding author. E-mail: bhatsajid@uok.edu.in

limited to transport planning, dealing with organized crime in adversary networks, community detection and tracking, sentiment analysis and opinion mining, node influence analysis, link prediction, and highlighting functionally coherent groups of proteins and predict the level of functional homogeneity within these protein groups or communities.

One of the important challenges in the field of SNA addressed in the existing literatures and along the lines of which the current proposal is directed, is the *detection of communities and tracking their temporal evolution*. Actors in a social network tend to form groups and the task of community detection is to identify such groups (communities) through the study of network structures and topologies. Community detection from social networks is a long standing and challenging task because communities can often be closely related to the functional units of a system, e.g., they may represent functionally coherent groups of proteins, people discussing similar topics and sharing an ideology (e.g. relating to a political party, government initiatives, election campaigns and so on), criminals linked to a particular crime, group of people involved in money laundering in a financial transaction network, web pages related to similar topics and so on. Identification of such groups, clusters, or communities in a social network forms the basis for many other social network analysis tasks. Further, communities may overlap and have a hierarchical structure. Moreover, community structures change with the changing dynamics of the underlying network. These properties further complicate the process of community detection. It thus becomes important to devise novel methods to address community detection challenges in a unified manner.

This paper presents a novel approach to solve the scalability issue of earlier overlapping community detection methods for large networks. It provides a distributed implementation of a density-based overlapping community detection method proposed in [2] via Hadoop MapReduce framework [16]. The proposed approach is divided in two stages wherein first stage reduces the network to a density-based mutual-core connected subgraph. The second stage uses a distributed connected component finding algorithm [25] to extract connected components from this subgraph which are then finally used to represent the communities.

The rest of the paper is organised as follows. Section 2 presents a brief review of literature on existing parallel and distributed implementations of

community detection algorithms. This section also presents significant differences of closely related methods with the proposed approach. Section 3 presents detailed description on the implementation of proposed approach. Some preliminary results are summarized in section 4. Section 5 finally concludes the paper.

2. Related Works

Although many techniques and methods for detecting communities from social networks have been proposed in literature [4,5,6,7,8,9,10,11,12,36], none of them aims to address all the multifaceted challenges inherent in the field, but instead deal with a single objective. Identifying communities poses numerous challenges which include dealing with their overlapping and hierarchical nature [30,31,35], tracking their evolution in dynamic networks [29,2,40], and scaling the community detection methods for large-scale social networks, besides dealing with directed and weighted properties of the underlying networks. A comprehensive summary and review of various challenges and solutions to the community detection problem is presented in [40]. It is also often difficult to have consensus on the evaluation methods of identified community structures with the underlying ground truth [32,33,34]. Recently, community detection methods based on both structure and semantics/content of online social networks (like [44]) have also been proposed. Methods which include both the structure and content related data for detection and analysis of communities in online social networks may help in providing important insights on concepts related to the influence and diffusion in online social networks. In this line of work, [42,43] have highlighted the significance of communities and their dynamic behavior in online social networks for collaborative decision making and opinion formation.

Presently, huge size of social networks has created a great need for scalable community detection methods. Earlier methods fail to address this issue in their present state and very few possess characteristics that can be exploited for scaling the methods via parallelization and distributed computing, which is the backbone for addressing scalability issues over big data. Traditional community detection methods based on modularity optimization [22,36] and hierarchical clustering methods [23] are generally considered non-scalable to large-scale networks as they are based

on global network metrics and thus difficult to parallelize. Some of the recent parallel algorithms proposed for detecting disjoint communities include [13,14,15]. These methods are mainly based on shared memory architectures, and multi-core processors and GPUs. The MapReduce framework supports a parallel model for addressing big data challenges [16]. It involves software to be written using two important functions – *map* and *reduce*. The *map* function processes a sub-problem for input data and emits intermediate <key, value> pairs, whereas *reduce* function combines values associated with the same key and produces the final output. Figure 1 demonstrates the basic processing paradigm of MapReduce framework.

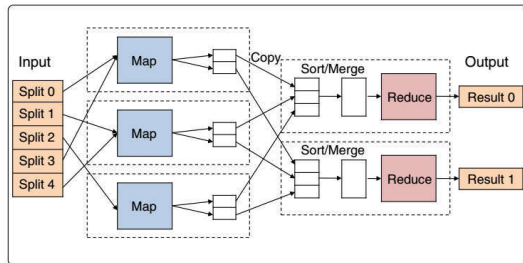


Fig. 1. Basic processing architecture of MapReduce framework

The popularity of MapReduce for processing big data has failed to effectively promote its usage for community detection in large-scale social networks because it is considered unsuitable for iterative graph algorithms [17]. However, recently, some research related to community finding from large-scale social networks using MapReduce framework appeared in [18,19,20,21,22]. Some of these methods are based on label propagation approaches derived from detecting connected components from large-scale networks. Other methods like [21] are based on modularity optimization and compute betweenness centrality of nodes and edges which requires higher communication between the computation nodes. This creates huge scope of improvement and highlights the importance of designing local MapReduce solutions for community detection problem and that is what this paper aims to offer. The method proposed in this paper is along the lines of approach presented in [22], wherein the first stage of the method establishes a similarity score between each pair of adjacent nodes and then updates the topology by removing or keeping edges based on a threshold on the similarity metric. The final stage identifies communities by

finding connected components in the edge filtered network based on label propagation. However, the method proposed in this paper significantly differs from the method of [22] in terms of distance function for measuring similarities between pairs of nodes, identification of overlapping communities vs disjoint communities, scalability to directed/undirected and weighted/unweighted networks vs only undirected and unweighted networks.

3. Proposed Approach

The main objective of this work is to scale the previous work of the authors [2], a novel density-based overlapping community detection method, using a distributed computation approach implemented over Hadoop via MapReduce framework. Being a density-based method, wherein communities are found by expanding a density function over local node neighbourhoods in a cascading fashion, this approach is naturally scalable to large-scale networks via parallelization and exploiting computation locality.

The proposed approach is based on a local density based distance function, which can be efficiently parallelized via MapReduce framework. The dual-layered distance function is a generalized function for every combination of directed/undirected and weighted/unweighted networks. Considering the social network as a graph $G = (V, E_w)$, where V is the set of nodes representing users and $E_w \subseteq V \times V$ is the set of weighted and directed links between the users, the proposed distance function is defined as follows.

Layer-1 It should be noted that the distance is measured only between those node pairs that are directly linked and have reciprocating interactions as they are expected to be less distant (more similar). In case of undirected networks, each link is considered to be reciprocating by treating it as a set of two oppositely directed links with the same weight as the original link. Given these considerations, the first layer of the distance function for two reciprocating nodes p and q is represented by equation 1, where V_p and V_q are sets of nodes to which nodes p and q have outgoing links/interactions respectively, and V_{pq} is the set of common nodes to which both p and q have outgoing links in the network.

In equation 1, $\Delta(p, q)$ represents the *layer-2* of the distance function which will be discussed shortly and η ($0 < \eta \leq 1$) is an input parameter which specifies the

$$dist(p, q) = \begin{cases} \Delta(p, q) & \text{if } |V_{pq}| > (\eta \times \min(|V_p|, |V_q|)) - 1 \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

resolution at which communities need to be identified. In simple terms, the first layer of the distance function ensures that the distance between two reciprocating nodes p and q is computed at the second layer only if the fraction of their commonly interacted nodes forms a significant fraction of the total outreached nodes of either p or q , i.e., $\min(|V_p|, |V_q|)$. Otherwise, the distance between p and q is taken as 1 (maximum).

Layer-2 The second layer of the distance function takes the intensity of interactions between nodes (link weights) into consideration. It is based on the assumption that if a node p has outgoing links (interactions) to a node q and a set of nodes V_{pq} (to which q also has outgoing links) then the similarity/distance between p and q can be measured in terms of proportion of response from q and nodes in V_{pq} to the interactions of p and vice versa. Formally, the response of node q and the nodes in V_{pq} to the interactions of node p is measured as the average of per-node reciprocated interactions (edge weights) of q and the nodes in V_{pq} towards p , represented by $\delta(p, q)$, as given in equation 2, where $I_{\overleftarrow{pq}}$ represents the amount of reciprocated interactions (weight) between two nodes p and q , i.e., minimum of the amount of interactions from p to q and q to p .

$$\delta(p, q) = \begin{cases} \left(\frac{\sum_{s \in V_{pq}} (I_{\overleftarrow{ps}}) + I_{\overleftarrow{pq}}}{|V_{pq}| + 1} \right) & \text{if } I_{\overleftarrow{pq}} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

Finally, symmetric distance between two nodes p and q , $\Delta(p, q)$, is taken as the maximum of their mutual directed-response (or *minimum of their mutual directed-response*) values normalized by their respective total weight of outgoing interactions (represented by $I_{\overrightarrow{p}}$ and $I_{\overrightarrow{q}}$ respectively) in the interaction graph, as given in equation 3.

The dual-layer distance function thus defined measures the amount of maximum average reciprocity among two nodes p and q and their common neighbors, provided the overlap of their neighbors is significant. Smaller values of $\Delta(p, q)$ represent higher response

$$\Delta(p, q) = \begin{cases} \min \left(\frac{\delta(p, q)^{-1}}{I_{\overrightarrow{p}}}, \frac{\delta(q, p)^{-1}}{I_{\overrightarrow{q}}} \right) & \text{if } \delta(p, q) > 0 \wedge \delta(q, p) > 0 \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

between nodes p and q and translates to more closeness between p and q .

Given a distance measure, we need to specify a neighborhood threshold to mark the boundary for any given node as required by density-based methods. However, instead of manually specifying the threshold value, we determine a *local neighborhood threshold* for a node p as the average per-receiver reciprocated interaction score of p with all its outreached neighbors. Formally, the local neighborhood threshold of a node p (ε_p) is defined using equation 4, where V_p represents the set of nodes to which p has out-links, $I_{\overleftrightarrow{p}}$ represents the number of reciprocated interactions of a node p (i.e., $\sum_{q \in V_p} \min(I_{\overrightarrow{pq}}, I_{\overleftarrow{qp}})$), where $I_{\overrightarrow{pq}}$ represents the number/weight of interactions from node p to node q , and $\frac{I_{\overleftrightarrow{p}}}{|V_p|}$ represents the average number of reciprocated interactions between p and all other nodes in V to which p has out-links. The denominator $I_{\overrightarrow{p}}$ represents the total count of outgoing interactions of node p and it normalizes the value of ε_p in the range $[0, 1]$.

$$\varepsilon_p = \begin{cases} \left(\frac{I_{\overleftrightarrow{p}}}{|V_p|} \right)^{-1} & \text{if } |V_p| > 0 \wedge I_{\overleftrightarrow{p}} > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Based on the distance function $dist(p, q)$ and local neighborhood threshold ε_p , we define a *local ε_p -neighborhood* of a node p as the subset of p 's out-linked nodes (i.e., V_p) with which its distance is less than or equal to ε_p . Formally, the local ε_p -neighborhood of a node p can be defined using equation 5.

$$N_p = \{q : q \in V_p \wedge dist(p, q) \leq \varepsilon_p\} \quad (5)$$

In simple terms, we can state that N_p contains those neighbors of p in the network that have a significant topological overlap with p and an above-average interaction intensity with p in the network neighborhood. This approach thus aims to find areas

of higher structural density (than the surrounding) to constitute a community.

Formally, for a given resolution fraction (η), a density-based community is realized by the following two key definitions.

Definition 1 (Core node) A node $p \in V$ having non-zero reciprocated interactions with any of its neighbor(s) in V_p is defined to be a core node if its local ε_p -neighborhood contains at least μ_p (local minimum-number-of-points threshold for p) of nodes in V_p , as given in equation 6, where $\mu_p = \eta \times |V_p|$.

$$CORE_\eta(p) \Leftrightarrow |N_p| \geq \mu_p \tag{6}$$

Definition 2 (Mutual-cores) Two nodes $p, q \in V$ are mutual-cores if both p and q are core nodes, and p belongs to local ε_q -neighborhood and q belongs to local ε_p -neighborhood.

The basic aim is to find all maximal sets of connected core nodes such that for each pair of nodes in a maximal set there exists a chain of nodes v_1, v_2, \dots, v_n such that v_i and v_{i+1} are mutual-cores for all i ranging from $1, 2, \dots, n - 1$. The set of all such connected core nodes forms the *mutual-core connected maximal subgraph* (MCMS) of a community. A community is then defined as the union of an MCMS (backbone of the community) and local ε_p -neighborhoods of each core node p in the MCMS. The set of all such possible communities identified forms the community structure of an underlying network.

3.1. MapReduce Implementation

As mentioned earlier, the main objective of this paper is to create a parallelized implementation of our earlier community detection method in [2] over a distributed computation platform (Hadoop) via MapReduce framework. The community detection process is divided into two mainstages viz 1. Computation of density-based neighbourhoods and formation of mutual-core network, and 2. Identifying communities from the mutual-core network by connected component extraction using label propagation.

3.1.1. Stage 1: Computation of density-based neighbourhoods and formation of mutual-core network

The initialization process of this stage involves a mapper and reducer pair, which computes local metrics to establish the density-based local neighbourhoods of each node in the network as shown in figure 2.

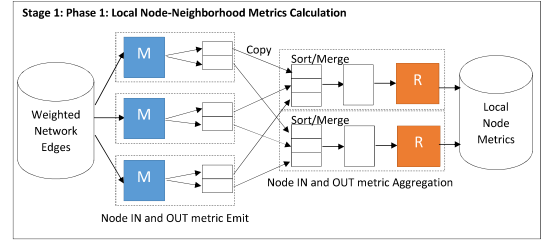


Fig. 2. MapReduce phase 1 for computing local metrics

The input to this phase is the network edges in the form of $\langle \text{source} \rangle \langle \text{destination} \rangle \langle \text{weight} \rangle$ tuple¹. For each record read by the mapper, it emits two records in the format shown in figure 3.

	Key		Value	
	node		out	in
Record 1	$\langle \text{source} \rangle$		destination:weight	
Record 2	$\langle \text{destination} \rangle$			source:weight

Fig. 3. Record format emitted by the mapper function of the first phase of Stage 1

The reducer function of this phase accepts this information in aggregated form for each node as shown in figure 4. This information is used by the reducer function to compute various local metrics for each node which include V_p , $I_{\vec{p}}$, $I_{\overleftarrow{p}}$ and ε_p as described in the first part of section 3.

For each node (record read by the reducer), the reducer emits this aggregated local information in the format shown in figure 5. The field *out_reciprocated* in the value of the emitted record for a node p is a set containing elements in the form $\langle q: I_{\overleftarrow{pq}} \rangle$ where q is a node in V_p and $I_{\overleftarrow{pq}}$ is the minimum of the weight of edges from p to q and q to p . However, if an edge

¹For undirected networks, each edge is represented by two records with swapped source and destination fields and may have different weights

	node	out	in
Example Record	Node ID	destination ₁ :weight ₁ , destination ₂ :weight ₂ , ... destination _n :weight _n	source ₁ :weight ₁ , source ₂ :weight ₂ , ... source _m :weight _m

Fig. 4. Records aggregated at the reducer function of the first phase of Stage 1

q to p does not exist, then I_{pq}^{\leftarrow} is taken as 0 for that q . Formally the set $out_reciprocated$ is represented as shown in equation 7.

Key		Value		
Node ID	$out_reciprocated$	I_p	ϵ_p	

Fig. 5. Record format emitted by the reducer function of the first phase of Stage 1

$$out_reciprocated = \left\{ \left(q : I_{pq}^{\leftarrow} \right) \mid q \in V_p \right\} \quad (7)$$

The second phase of this stage involves computing *layer-1* and *layer-2* components of the distance function as mentioned earlier to identify density-based neighbourhood of each node. The overview of this phase is shown in figure 7. The mapper function of this phase takes input from the reducer of the previous phase in the format shown in figure 5. For each node, it emits multiple records for ordered pairs of nodes in the format shown in figure 6.

Key		Value					
Node Pair < s,t >	$out_reciprocated$ of s	I_s	ϵ_s	$out_reciprocated$ of t	I_t	ϵ_t	

Fig. 6. Record format emitted by the mapper function of the second phase of Stage 1

To emit the records for a key node p , the mapper takes each node q from V_p only for which I_{pq}^{\leftarrow} is not equal to 0 and forms the key part of the record in the form $\langle s:t \rangle$ such that $s = \min(p,q)$ and $t = \max(p,q)$. The value part of the emitted record is half filled, with the local metrics of p from the previous reducer, for either s or t depending upon whether $s = p$ or $t = p$. For each node p , this mapper emits a maximum of $|V_p|$ number of records.

The reducer function of this phase aggregates these records on the key, thus completing the value part for each node pair emitted by the mapper. That means reducer has local metrics computed by the first phase

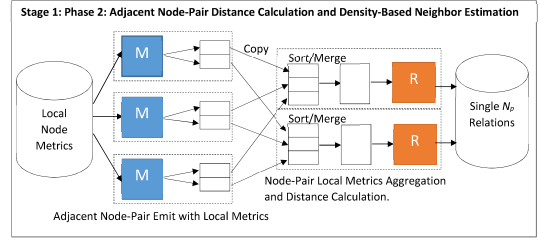


Fig. 7. MapReduce phase 2 for computing density-based neighbours

for both the nodes of the node pair which are sufficient to compute the distance between this pair of nodes as discussed in the first part of section 3. For each record thus read, the reducer computes the distance between the pair of nodes in the key part of the record from the information in the value part of the record. It emits 0, 1 or 2 records with key equal to the ID of one of the nodes in the node pair and the value equal to the ID of the other (along with $|V_p|$ of the key node) depending upon the following conditions: 1. Record with key = s and value = ' $t:|V_s|$ ' is emitted if $dist(s, t) < \epsilon_s$, and 2. Record with key = t and value = ' $s:|V_t|$ ' is emitted if $dist(s, t) < \epsilon_t$. Each record thus emitted represents that the node present in the value part of the record belongs to the density-based local neighbourhood of the node in the key part. To find the complete density-based local neighbourhood N_p of each node p as given in equation 5, these records simply need to be aggregated on the key which is done by the third MapReduce phase of this stage. In the third phase, the mapper only emits the records (identity mapper) for the reducer to aggregate N_p for each node p . Upon aggregation of N_p the reducer function uses equation 6, with the value of resolution parameter η passed as a configuration parameter, to determine whether the key node p is a core-node or not. If node p is found to be a core-node, the reducer performs two actions. First, it creates a separate file in HDFS named p with contents equal to the list of nodes in N_p (to be used later in the second stage). Second, the reducer takes each node q from N_p and emits records for the node pair p and q with key in the ordered form $\langle s:t \rangle$ such that $s = \min(p,q)$ and $t = \max(p,q)$ and value equal to 1. These records are emitted by the reducer for fourth and last MapReduce phase of this stage to identify the mutual-core network. The overview of this process is illustrated in figure 8.

The last MapReduce phase of this stage involves an identity mapper which simply emits the records from the previous reducer. The reducer function of this phase aggregates the value field of these records,

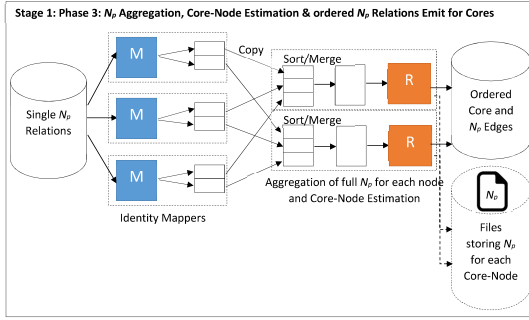


Fig. 8. MapReduce phase 3 for aggregating density-based neighbourhoods

wherein an aggregated value of 2 for a node pair represents a mutual-core relation between the two. For each mutual-core pair $\langle s:t \rangle$ thus found, the reducer emits a record with $key = s$ and $value = t$ thus forming the edges of the mutual-core network of the underlying social network as illustrated in figure 9.

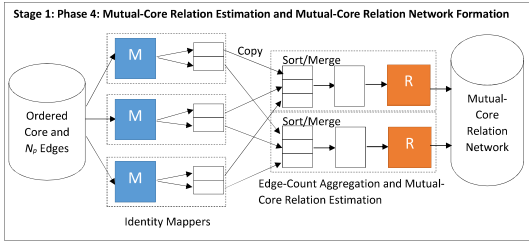


Fig. 9. MapReduce phase 3 for aggregating density-based neighbourhoods

3.1.2. Stage 2: Identifying communities from the mutual-core network by connected component extraction using label propagation.

The connected components of the mutual-core network constructed in the previous stage represent the MCMSs of various communities in the underlying network. For a connected component in the mutual-core network, its full community is identified by taking the union of density-based local neighbourhood N_p of each node p in the connected component. This becomes a trivial task once a connected component is extracted as the N_p of each core-node p has already been saved in separate files on the HDFS in the previous stage.

The important task in this stage thus is to extract the connected components from the mutual-core network. Many MapReduce framework based methods for extracting connected components from networks have been proposed in literature [24,25,26,27,28]. These

methods mainly vary in optimizations used for reducing the amount of communication generated between cluster nodes and the number of MapReduce rounds required to converge the result. Rastogi et al. [25] proposed *Hash-to-Min* MapReduce approach, which uses label propagation with logarithmic rounds to extract connected components. At the end of algorithm *Hash-to-Min*, the set of nodes C_v representing a connected component of which v is a member which satisfies the following property: If v_{min} is the smallest node of a connected component C , then $C_{v_{min}} = C$. For all other nodes v , $C_v = \{v_{min}\}$. The method is illustrated in figure 10.

At this stage, there are two alternative ways of representing and using the connected components. One way is to use the records emitted by *Hash-to-Min* in the form $\langle C_{v_{min}}, C \rangle$ wherein the key = $C_{v_{min}}$ is the smallest ID of the node in the connected component (representing the connected component ID) and value = C is the list of all the node IDs belonging to the connected component. In order to extract the complete communities, finally a single mapper function is used which reads a $\langle core-node_{min}, connected-component-list \rangle$ record and performs the following tasks for each such record:

1. Create a new directory on HDFS labelled with the connected-component ID i.e. $C_{v_{min}}$ in the key part of the record.
2. Move all the existing files containing the N_p of each core-node $p \in C$ in the value part of the record (already saved on HDFS in the previous stage) to the directory created in the previous step.

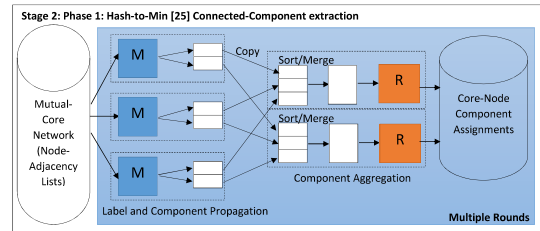


Fig. 10. Hash-to-Min [25] connected-component extraction

Alternatively, the last round of the *Hash-to-Min* method can be modified such that the last reducer also emits $C_{v_{min}} = \{v_{min}\}$. Thus the $\langle key, value \rangle$ format output of *Hash-to-Min* method becomes such that the key is a node ID (core-node in this case) and the value is the connected-component ID to which the node

belongs. Finally, to extract the complete communities, a single mapper function is used which reads a $\langle \text{core-node}, \text{connected-component ID} \rangle$ record and performs the following tasks for each such record:

1. Create a new directory on HDFS labelled with the connected-component ID in the value part of the record if it doesn't exist already.
2. Move the existing file containing the N_p of the core-node p in the key part of the record (already saved on HDFS in the previous stage) to the directory labelled with the connected-component ID in the value part of the record.

These two steps of the last mapper ensure that a community consisting of at least one pair of mutual-core nodes is represented by a separate directory containing files representing its core-nodes and each such file containing N_p i.e. the density-based local neighbourhood of each core-node p . Moreover, a community consisting of a single core-node p is simply represented by a separate file labelled p containing its N_p . These are those files which were not moved by the last mapper to any directory as the core-nodes represented by these files did not have mutual-core relations with any other core-nodes and thus did not appear in the mutual-core network as illustrated in figure 11.

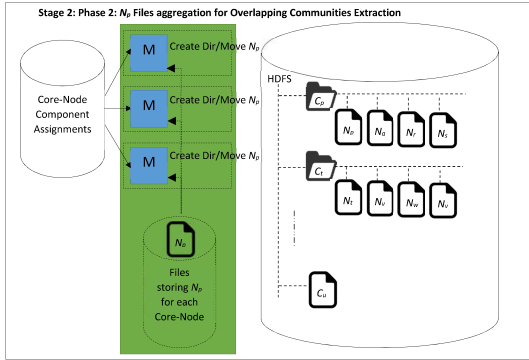


Fig. 11. Extracting communities by grouping N_p neighbourhoods of mutual-core nodes

4. Evaluation and Results

To present some basic characteristics of the proposed method, this paper uses two network datasets. One is a large interaction graph of the most popular online social network in Slovakia, namely,

Pokec [38]. It is a directed and unweighted network with 1632803 nodes and 30622564 edges. The second network is a relatively smaller web graph with 325729 nodes and 1497134 edges, where nodes represent pages from *University of Notre Dame* (domain *nd.edu*) and directed edges represent hyperlinks between them [39]. For generalization, a weight of 1 is assigned to each edge in both the networks. The results were generated on a two-node Hadoop cluster.

The quality of communities identified by the proposed community detection method has already been evaluated in [2] for a centralized implementation of the same. The distributed implementation of the method via MapReduce framework does not affect the nature of the communities detected as the base approach remains the same. Moreover, the heuristic approach for estimating a value for η is not implemented for the current distributed approach and thus η is required as input from the user. As a reminder, η is the resolution parameter that can take values in the range [0 1]. Higher values of η yield smaller denser communities and larger number of outliers whereas smaller values of η yield larger less-dense communities and lesser number of outliers.

For the current distributed implementation, η affects the computation overhead in terms of disk I/O operations, involved for creating files for core-nodes in the last phase of the first stage. It also dictates the overall complexity of the second stage for finding connected components as the size of input for this stage depends on the number of core-nodes identified in the first stage. Figure 12 illustrates the fraction of nodes identified by the proposed method as core-nodes in both the input networks at different values of η . It can also be observed from this figure that generally the number of core-nodes identified by the method is much smaller than the actual size of the network by a large magnitude. It can be thus concluded that the disk I/O overhead caused due to creation of core-node files in the first stage is a large magnitude less than as required for all nodes in input network.

Moreover, the number of MapReduce rounds in the first stage is fixed (i.e. four). The computation time for the first stage is thus totally dependent on the number of *Map* and *Reduce* functions executed in parallel and the communication overheads i.e. the structure of the cluster. An average time consumption of the *Mappers* and *Reducers* of the four phases of Stage 1 for the two input graphs is shown in figure 13.

From figure 13, it can be seen that the majority of the time required by the first stage is at phases 1 and

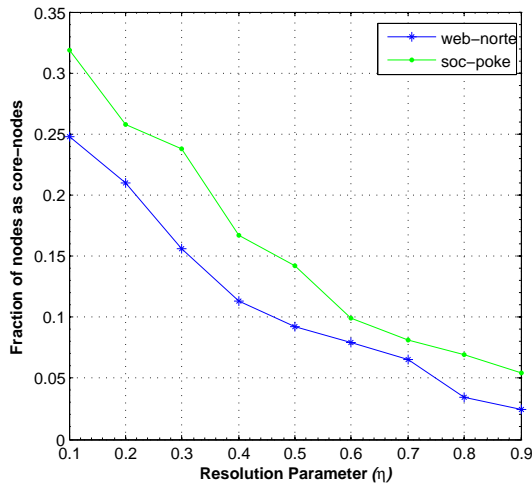


Fig. 12. Fraction of nodes identified as core-nodes from the two input graphs in Stage 1

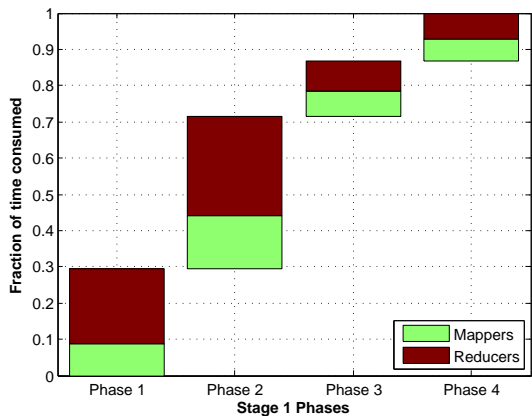


Fig. 13. Average time consumption of the Mappers and Reducers of Stage 1

2. That is where the neighbourhood aggregation and distance computation is performed. Moreover, most of the time is consumed at the *Reducer* stages due to the requirement of copying and sorting the data at various nodes of the cluster. The number of MapReduce rounds required by Stage 2, for identifying connected components, as reported in the base paper [25] is $O(\log n)$ where n is the number of nodes in the input graph. As demonstrated in figure 12, Stage 1 of the proposed method extracts a relatively smaller mutual-core network for Stage 2 to process. This further reduces the number of rounds required by Stage 2 compared to the number of rounds required in case the whole input network required processing in Stage 2. One of the obvious merits of a distributed implementation of an algorithm compared to its non-parallelized or non-

distributed implementation is achieved speedup. In this paper, the proposed MapReduce based distributed implementation of the centralized algorithm given in [2] also has an obvious speedup improvement which is directly proportional to the number of *Map* and *Reduce* nodes in the Hadoop cluster.

5. Conclusion and Future Work

We have presented a novel MapReduce framework based implementation of a density-based overlapping community detection method for large-scale social networks. The community detection process is divided into two main stages – (i) computation of density-based neighbourhoods and formation of mutual-core network, and (ii) identifying communities from the mutual-core network by connected component detection using label propagation. To do so, we have combined two state-of-the art methods, one the authors’ earlier work which is a density-based overlapping community detection method, and the other an optimized label propagation algorithm for connected component detection. The method is implemented on a two-node Hadoop cluster for evaluations. The resulting approach is among the first scalable overlapping community detection methods proposed in literature. Although this paper aims to address a challenging problem with a novel solution, the proposed solution can further be generalized to encompass other related problems. One of the directions along which the current work can be extended is to consider dynamic nature of networks and the underlying communities, as addressed in [2, 40]. Since the current paper is an adaption of the method proposed in [2], the authors aim to generalize the current method to large-scale dynamic networks by incorporating the adaptive community dynamics as proposed in [2].

References

- [1] Lam, C. (2010) Hadoop in action. Manning Publications Co.
- [2] Bhat, S. Y., & Abulaish, M. (2015) HOCTracker: Tracking the evolution of hierarchical and overlapping communities in dynamic social networks. *IEEE Transactions on Knowledge and Data Engineering*, 27(4), 1019-1013.
- [3] Bhat, S. Y., & Abulaish, M. (2013) Analysis and mining of online social networks: emerging trends and challenges. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 3(6), 408-444.

- [4] Newman, M. E., & Girvan, M. (2004) Finding and evaluating community structure in networks. *Physical Review E* 69, 026113.
- [5] Greene, D., Doyle, D., & Cunningham, P. (2010) Tracking the evolution of communities in dynamic social networks. In *Proceedings of the International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, Odense, Denmark, 176-183.
- [6] Lancichinetti, A., Fortunato, S., & Kertesz, J. (2009) Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3), 033015.
- [7] Zachary, W. W. (1977) An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4), 452-473.
- [8] Girvan, M., & Newman, M. E. (2002) Community structure in social and biological networks. In *Proceedings of the National Academy of Sciences*, 99(12), 7821-7826.
- [9] Lusseau, D., Schneider, K., Boisseau, O. J., Haase, P., Slooten, E., & Dawson, S. M. (2003) The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4), 396-405.
- [10] Burt, R. S. (1987) Social contagion and innovation: Cohesion versus structural equivalence. *American Journal of Sociology*, 92(6), 1287-1335.
- [11] Lancichinetti, A., & Fortunato, S. (2009) Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E* 80, 016118.
- [12] Cazabet, R., Amblard, F., & Hanachi, C. (2010) Detection of overlapping communities in dynamical social networks. In *Proceedings of the IEEE Second International Conference on Social Computing*, Minneapolis, MN, USA, 309-314.
- [13] Rytasareva, I., Chapman, T., & Kalyanaraman, A. (2014) Parallel algorithms for clustering biological graphs on distributed and shared memory architectures. *International Journal of High Performance Computing and Networking*, 7(4), 241-257.
- [14] Soman, J., & Narang, A. (2011) Fast community detection algorithm with GPUs and multicore architectures. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium*, Anchorage, AK, USA 568-579.
- [15] Bae, S. H., Halperin, D., West, J., Rosvall, M., & Howe, B. (2013) Scalable flow-based community detection for large-scale network analysis. In *Proceedings of the IEEE 13th International Conference on Data Mining Workshops*, Dallas, TX, USA, 303-310.
- [16] Dean, J., & Ghemawat, S. (2008) MapReduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1), 107-113.
- [17] Tian, Y., Balmin, A., Corsten, S. A., Tatikonda, S., & McPherson, J. (2013) From think like a vertex to think like a graph. In *Proceedings of the VLDB Endowment*, 7(3), 193-204.
- [18] Bahmani, B., Kumar, R., & Vassilvitskii, S. (2012) Densest subgraph in streaming and mapreduce. In *Proceedings of the VLDB Endowment*, 5(5), 454-465.
- [19] Li, Q., Wang, Z., Wang, W., Liu, Y., Wang, P., & Yu, T. (2011) LI-MR: a local iteration map/reduce model and its application to mine community structure in large-scale networks. In *Proceedings of the IEEE 11th International Conference on Data Mining Workshops*, Vancouver, BC, Canada, 174-179.
- [20] Moon, S., Lee, J. G., & Kang, M. (2014) Scalable community detection from networks by computing edge betweenness on mapreduce. In *Proceedings of the International Conference on Big Data and Smart Computing*, 145-148.
- [21] Shi, J., Xue, W., Wang, W., Zhang, Y., Yang, B., & Li, J. (2013) Scalable community detection in massive social networks using MapReduce. *IBM Journal of Research and Development*, 57(3/4), 1-12.
- [22] Zhuang, D. (2017) Modularity-based dynamic community detection. arXiv:1709.08350.
- [23] Fortunato, S. (2010) Community detection in graphs. *Physics Reports*, 486(3-5), 75-174.
- [24] Kiveris, R., Lattanzi, S., Mirrokni, V., Rastogi, V., & Vassilvitskii, S. (2014) Connected components in mapreduce and beyond. In *Proceedings of the ACM Symposium on Cloud Computing*, Seattle, WA, USA, 1-13.
- [25] Rastogi, V., Machanavajjhala, A., Chitnis, L., & Sarma, A. D. (2013) Finding connected components in map-reduce in logarithmic rounds. In *Proceedings of the IEEE 29th International Conference on Data Engineering*, Brisbane, QLD, Australia, 50-61.
- [26] Seidl, T., Boden, B., & Fries, S. (2012) CC-MR – finding connected components in huge graphs with MapReduce. In *Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, LNCS-7523, 458-473.
- [27] Lulli, A., Carlini, E., Dazzi, P., Lucchese, C., & Ricci, L. (2017) Fast connected components computation in large graphs by vertex pruning. *IEEE Transactions on Parallel & Distributed Systems*, 28, 760-773.
- [28] Abdolazimi, R., Naderi, H., & Sagharichian, M. (2017) Connected components of big graphs in fixed MapReduce rounds. *Cluster Computing*, 20(3), 2563-2574.
- [29] Palla, G., Barabasi, A. L., & Vicsek, T. (2007) Community dynamics in social networks. *Fluctuation and Noise Letters*, 7(3), L273-L287.
- [30] Palla, G., Derenyi, I., Farkas, I., & Vicsek, T. (2005) Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435, 814-818.
- [31] Palla, G., Tibely, G., Mones, E., Pollner, P., & Vicsek, T. (2015) Hierarchical networks of scientific journals. *Palgrave Communications* 1, Art. No. 15016.
- [32] Cherifi, C., & Cherifi, H. (2018) Evaluating community detection algorithms: a multidimensional issue. In *Proceedings of the NetSci Conference*, Paris, France.
- [33] Orman, G. K., Labatut, V., & Cherifi, H. (2012) Comparative evaluation of community detection algorithms: a topological approach. *Journal of Statistical Mechanics: Theory and Experiment*, 2012(8), P08001.
- [34] Jebabli, M., Cherifi, H., Cherifi, C., & Hamouda, A. (2015) Overlapping community detection versus ground-truth in AMAZON co-purchasing network. In *Proceedings of the 11th International Conference on Signal-Image Technology & Internet-Based Systems*, Bangkok, Thailand, 328-336.
- [35] Xie, J., Kelley, S., & Szymanski, B. K. (2013) Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys*, 45(4), 43:1-43:35.
- [36] Xie, J., Szymanski, B. K., & Liu, X. (2011). SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In *Proceedings of the IEEE*

- 11th International Conference on Data Mining Workshops, Vancouver, BC, Canada, 344-349.
- [37] Chen, M. (2015). Discovering community structure by optimizing community quality metrics, Doctoral Dissertation, Rensselaer Polytechnic Institute, Troy, NY, USA.
- [38] Takac, L., & Zabovsky, M. (2012). Data analysis in public social networks. In Proceedings of the International Scientific Conference & International Workshop on Present Day Trends of Innovations, Lomza, Poland, 1-6.
- [39] Albert, R., Jeong, H., & Barabasi, A. L. (1999) Internet: Diameter of the world-wide web. *Nature* 401, 130-131.
- [40] Azaouzi, M., Rhouma, D., & Romdhane, L. B. (2019) Community detection in large-scale social networks: state-of-the-art and future directions. *Social Network Analysis and Mining*, 9(1), 9-23.
- [41] Guo, C., Wang, J., & Zhang, Z. (2014). Evolutionary community structure discovery in dynamic weighted networks. *Physica A: Statistical Mechanics and its Applications*, 413, 565-576.
- [42] Dong, Y., Zha, Q., Zhang, H., Kou, G., Fujita, H., Chiclana, F., & Herrera-Viedma, E. (2018) Consensus reaching in social network group decision making: Research paradigms and challenges. *Knowledge-Based Systems*, 162, 3-13.
- [43] Dong, Y., Zhan, M., Kou, G., Ding, Z., & Liang, H. (2018) A survey on the fusion process in opinion dynamics. *Information Fusion*, 43, 57-65.
- [44] Azaouzi, M., & Romdhane, L. B. (2017) An evidential influence-based label propagation algorithm for distributed community detection in social networks. *Procedia Computer Science*, 112, 407-416.