Knowledge and Data Engineering, IEEE Transactions on (Volume:PP , Issue: 99),16 June 2015.

FOCS: Fast Overlapped Community Search

between communities are known to be denser than the non-overlapped regions of the communities. However, most of the existing algorithms that detect overlapping communities assume that the communities are denser than their surrounding regions, and falsely identify overlaps as communities. Further, many of these algorithms are computationally demanding and thus, do not scale reasonably with varying network sizes. In this article, we propose FOCS (Fast Overlapped Community Search), an algorithm that accounts for local connectedness in order to identify overlapped communities. FOCS is shown to be linear in number of edges and nodes. It additionally gains in speed via simultaneous selection of multiple near-best communities rather than merely the best, at each iteration. FOCS outperforms some popular overlapped community finding algorithms in terms of computational time while not compromising with quality.

Index Terms—overlapping community search, social network, local heuristic, complex network

1 Introduction

A social network comprises a finite number of individuals and connections among them.A connection or tie usually links a pair of individuals based on their common interest, relationship through work, family, romance, friendship, partnership in crime etc. The complexity involved in the appearance and disappearance of such connections leads to the formation of some non-trivial topological structures. Moreover, these networks are often huge in size, which prevent the application of most of the traditional graph theoretic algorithms that do not scale well. Social network analysis intends to generate useful insights into such large, complex networks with the help of a range of novel and scalable computational methods.

In a social system, individuals tend to group with others who are like-minded or with whom they interact more regularly and intensely than others. This process leads to the formation of communities. In a community the participant actors are densely connected to each other, whereas nodes that belong to different communities do not interact much. Furthermore, actors with interests and purposes in different fields result in overlapped communities. Such overlapping communities are frequent in social graphs.

Identification of communities has many real life applications. For example, a telecom service provider might take additional measures to retain a consumer who has significant connectivity within a specific community. This is important because exit of such an important consumer might go viral and lead to an undesired shrinkage in the respective community [1]. Commercial web sites may market offers by eying increased sales within certain communities of individuals. Sometimes, a piece of information can be diffused easily into a community by informing a handful of its influential individuals. This, in fact, is quite evident today, in the way a locally published information is spread across a huge mass of socially connected people. Information about political views, social irregularities, natural calamities, important conferences, newly created media etc. diffuse quickly through social networks. There has been instances of identifying dubious communities involved in organized crime [2]. To summarize, community detection has diverse applications including the prediction of forthcoming events, activities or developments, business intelligence, campaign management, infrastructure management, churn prediction, etc.

Networks today, typically consist of nodes in millions and edges in billions. Mining useful information from such large-scale networks demands methods, which are fast, efficient and requiring information that are local to the nodes in consideration. Such methods, apart from being fast, are able to overcome the memory constraints. In this paper, we propose FOCS (Fast Overlapped Community Search) algorithm that searches for overlapped communities in large networks based on locally computed scores. The method has been applied to several large social and biological networks. The detected communities have been compared with respective ground-truth communities for the networks. FOCS has performed well in terms of both time and efficiency when compared with some popular overlapped community detection algorithms.

- *S. Bandyopadhyay and Garisha Chowdhary are with Machine Intelligence Unit, Indian Statistical Institute, Kolkata, India, 700108. E-mails: sanghami@isical.ac.in, garisha.ch@gmail.com*
- *D. Sengupta is with Computational and Systems Biology Group, Genome Institute of Singapore, 60 Biopolis St., Singapore, 138672. E-mail: senguptad@gis.a-star.edu.sg*

2 Related Work

The problem of community detection is to identify naturally existing groups of actors such that nodes within a group are densely connected with each other while being sparsely connected to the nodes that belong to different groups. Social communities, in topological terms, are nothing but graph clusters. However, graph-clustering methods, in general, are only applicable on networks which are way smaller than today's social networks. An excellent review of the community detection methods can be found in [3], [4] and [5].

Graph clustering is an optimization problem, which is computationally intractable. The advent of social networks and their impact on day to day life have rejuvenated this area of research with the demand of faster algorithms that scale well for large-sized graphs. A number of graph clustering approaches exist in literature [6]. Hierarchical methods, for example, usually optimize a global score such as conductance, spectral distance, modularity etc., and obtain disjoint clusters. In [7] Rosvall and Bergstrom created a partition of network into clusters based on compression of information on random walk taken on the network. Blondel et al. in [8] found hierarchical disjoint communities in massive networks based on modularity optimization.

As already mentioned, partitioning a graph into disjoint clusters is inapplicable in the context of social networks, since they are generally organized into overlapped communities. Secondly, communities in social graphs exist at both the vertical and horizontal tiers. For example, in corporate scenarios, communities naturally exist within a horizontal tier, a subunit assigned to a task. Concurrently, a committee formed to activate tasks up and down the vertical differentiation of organization exemplifies a community existing across horizontal tiers. Therefore, methods that find communities through hierarchical clustering of nodes [9], fail to capture such communities. This enforces the need of clustering methods that allow a node to be present in communities detected at all hierarchical levels.

Further, there are approaches that identify certain predetermined structures in the network such as: cliques [10], k-cores [11], and, n-cliques [12], as communities. These methods generally perform well but are computationally demanding, and restrictive at times. Other methods that start from a seed (node [9] or clique [10]) and expand until a certain score such as cut ratio, or conductance is decreased, fail to identify all existing communities in social network. This is because the number of outgoing edges from a community is many a times greater than the number of edges within the community, as can be seen around community B encircled with a dashed line in Figure 1. Most density based methods [13], are inapplicable to

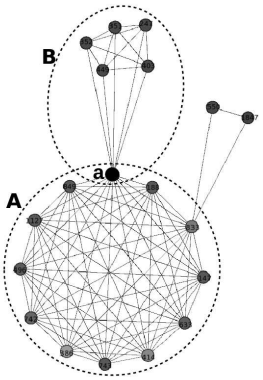


Fig. 1: Snapshot of a section of DBLP network

community detection in social network because of the same reason. Communities in social networks have rather denser overlapped regions when compared to non-overlapped portions of communities [14] [15].

In [16] [17] the idea of partitioning edges, instead of nodes, into communities has been explored. It allows a node with multiple edges to be assigned to multiple communities. These methods assume that the links are homogeneous, i.e., two individuals are connected via a single functionality or interest. This assumption violates the observed statistics that the likelihood of an edge between a pair of nodes increases with the number of communities they share [14]. There also exist several model based approaches to community detection including the block stochastic model [18] and another based on non-negative matrix factorization (BigClam) [19]. In these methods a given graph is considered to be a realization of the proposed statistical model. These methods are usually driven by a certain objective function. BigClam in particular takes care of the statistics as mentioned earlier in this paragraph. Additionally, BigClam has smaller time complexity as compared to other existing non-negative matrix factorization methods, because of improvement in objective function from l_2 norm to log-likelihood. As stated in [19], BigClam "achieves near linear running time". However, as evident from the results in Table 4, it is still unable to produce results within a reasonable time for very large networks. In addition, BigClam also requires the number of communities to be given as input.

Local optimization methods, on the other hand, achieve near optimum solution by optimizing a fitness function defined on parameters that describe local topological configurations. Reducing the community detection problem in social networks into a local optimization problem is more convincing. This is intuitive as the process of community formation is initiated by a participating individuals, in a manner more local than global. Success of such local optimization based approaches depends heavily on the considerations made while constructing the fitness functions. A re-

cent method, for example, defines the local fitness score to be the fraction of neighbors of a node that are within its community [30]. Several such fitness functions have already been defined in the literature [9] [23] [28] [31].

Label propagation algorithms (LPAs) in particular start by assigning each node a unique label and then propagate labels ensuring that a node receives one that maximum of its neighbors share [20] [21] [22] [23]. COPRA [23] modified the classical LPA [24] such that each node can retain multiple labels in order to find overlapped community structure. In SLPA [20] the occurrence frequency of labels received over consecutive iterations for each node is maintained while a sender (neighboring) node sends the most probable label. Such methodology also helps a node to decide upon its membership strength in each of its communities based on the probability information of received labels. While maintaining that each node must only hold labels that majority of its neighbors share, the LPAs emphasize on largest possible communities for each node, ignoring the small well connected communities among a minority of its neighbors. In real life scenario, however, one usually forms a small community with ones family members and close relatives, while many more larger ones with school/ college class mates. In Figure 1, for example, for the node marked a in the overlapped region of the two communities A and B, only a very small fraction of its neighbors, i.e., only 5 of 16, are in community B. LPAs, in this case, will assign only label A to node a resulting in two disjoint communities. Algorithm DEMON [25] extracts local network for each node, applies label propagation algorithm to each of them, and finally finds union of obtained communities to get overlapped community structure. The algorithm however suffers with the same limitation as an LPA.

Local spectral clustering based methods have also found application in overlapped community detection [26] [27]. These methods usually require an upper bound on the number of communities as input. They usually first approximately embed the graph in $d \ll n$ dimensions (where n is the number of nodes) using spectral clustering. Following this, the points in low dimensional space are clustered using simpler existing clustering methods. However, computation of eigenvalues/eigenvectors for spectral clustering are computationally expensive. Efforts to parallelize computation in MapReduce in [27] still show limited application in terms of scalability.

In [28] [29], the problem of overlapped community detection in social networks has been addressed using a game theoretic framework, where the dynamics of community formation have been captured as a strategic game. Here, each node, a selfish agent in disguise, selects the communities to join or leave, based on its definition of utility. Utility is usually a combination of gain and loss functions. In [28], for example, increase in modularity has been formulated as the gain function, whereas the number of communities a node joins is the input parameter to the loss function. There are other methods that solve community detection problem for social networks based on cost-benefit tradeoff [31]. They mostly add or remove nodes iteratively from a community, or merge communities, in order to improve the benefits, and reduce the costs incurred to a node. Many approaches among these impose the number of communities a node participates in as a restriction [18] [23] [28] [30] [32], which is not the case in real networks [14].

Although the aforementioned methods are simple and fast, they mostly find disjoint clusters. The ones that find overlapped clusters are mostly computationally demanding, and still restrictive. This makes them inapplicable to large scale real networks. FOCS, on the other hand is a fast algorithm that evolves on the basis of some locally computed scores to discover overlapped communities. It scales well over large sized social networks. It additionally gains in speed via simultaneous selection of multiple near-best communities rather than merely the best. This helps to save a number of iterations. Moreover, the communities detected by the method are not limited to a particular hierarchical level, rather are inclusive of all meaningful communities in the given network. Furthermore, the method is deterministic i.e., the results are not dependent on the sequence in which the nodes are considered. This is a problem in [9] [21] [23] [28] [29] [31] .

3 METHOD

3.1 Problem Definition

We are given an undirected, unweighted graph $G(V, E)$. The graph is assumed to be simple (without self loop or parallel edges).

The problem of community detection is to find family of subgraphs $S = \{S_i | S_i \subset V\}$ such that for any node v_j in a subgraph S_i, it is more *connected* in the subgraph S_i than in another subgraph S'_j. Here, $S'_j = (S_k | v_j \notin S_k \wedge S_k \in S)$ is any subgraph in family S not containing node v_j. Each subgraph $S_i \in S$ is a community.

For each node v_j, $\forall j \in \{1, 2, .., |V|\}$, let $S(v_j) = \{S_i | v_j \in S_i \wedge S_i \in S\}$ be the collection of communities containing node v_j. Further, let $S'(v_j) = S - S(v_j)$ be the collection of communities not containing node v_j. If each node v_j belongs to exactly 1, or no community at all, i.e., $|S(v_j)| \leq 1$, then it is called disjoint clustering, overlapped clustering otherwise. FOCS algorithm, proposed in this paper, explores overlapped clusters in a given graph.

3.2 Connectedness

As has already been mentioned in Section 3.1, for each node v_j, $\forall j \in \{1, 2, .., |V|\}$, v_j is more *connected* to

TABLE 1: Status of a node in a community on basis of *neighborhood connectedness* and *community connectedness* scores.

		community connectedness score	
		low	high
neighborhood connectedness score	low	less interest and low belongingness (not this community)	less interest but high belongingness (strong community with few neighbors)
	high	high interest but low belongingness (not accepted by community)	high interest and high belongingness (node is central to the community)

any community in $S(v_j)$ than any of the communities in $S'(v_j)$. Consequently, we say, v_j is equally well *connected* to all the communities in $S(v_j)$. This derives the working principle for FOCS.

Let $N(v_j)$ be the set of neighbors of a node $v_j \in V$. Or,

$$N(v_j) = \{v_k | (v_j, v_k) \in E\}. \quad (1)$$

Now, let $N_i(v_j)$ be the within community neighborhood of node v_j defined for community $S_i \in S(v_j)$ as follows:

$$N_i(v_j) = \{v_k | (v_j, v_k) \in E \wedge v_k \in S_i\}. \quad (2)$$

FOCS defines *connectedness* of a node with respect to its community as the ratio of the size of its within community neighborhood to the size of the community minus 1. An individual, thus, is considered to be well connected within its community if it has connections to most of the nodes in the community (apart from itself). The *community connectedness* score $\tilde{\zeta}_j^i$, thus, assigned to each node v_j in each community $S_i \in S$ is,

$$\tilde{\zeta}_j^i = \frac{|N_i(v_j)|}{|S_i| - 1}. \quad (3)$$

Further, to ensure that a node in any community has at least K neighbors within the community [33], Equation 3 has been modified to define *community connectedness* score ζ_j^i as follows:

$$\zeta_j^i = \frac{|N_i(v_j)| - K + 1}{|S_i| - K}, \text{if } |N_i(v_j)| > K, \text{ and, } 0, \text{ otherwise} \quad (4)$$

Reasonably, if K is assigned a very large value, small but dense communities will be missed out. On the other hand, a very small value for K allows discovery of sparser large communities and insignificant small communities. It is found that the algorithm is not sensitive to low values of K and performs consistently well over networks of varying sizes with $K = 2$. Figure 2 can be referred for variation in statistics of detected communities when FOCS is applied on Amazon network, with increasing values of K, and OVL (discussed later in Section 3.3.4) set to 0.6. It can be observed that a community structure ceases to exist in extremely sparse graph resulting from setting $K > 5$. In general $K = 2$ is a reasonable choice for community detection unless networks being analyzed are highly dense, in which case K can be set larger.

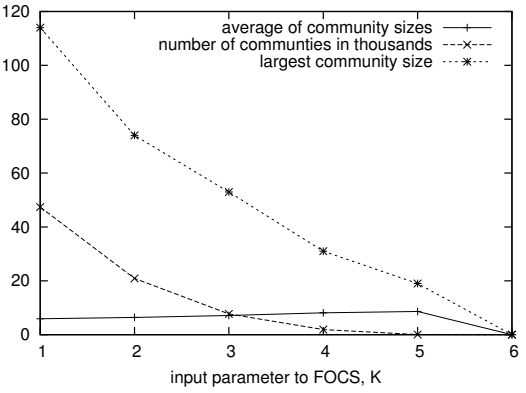


Fig. 2: Change in community statistics when input parameter to FOCS, K is varied (with OVL set to 0.6), simulated on Amazon network [34].

The algorithm also defines *neighborhood connectedness* score ξ_j^i for a node v_j with respect to its community S_i as the ratio of the size of its within community neighborhood to the size of its (overall) neighborhood.

$$\xi_j^i = |N_i(v_j)|/|N(v_j)| \quad (5)$$

This score emphasizes on the fraction of neighborhood of node v_j that is present within the community S_i. It must be noted that *community connectedness* score decides the belongingness of a node to its community, whereas the *neighborhood connectedness* score only defines the interest of a node in joining a new community. Table 1 describes the status of a node in its community on the basis of both the scores.

3.3 The Algorithm

The driving principle for FOCS is that communities are initiated by individuals, and influenced by their neighbors and neighboring communities. A node attracts its neighboring individuals to be a part of its community. Those that find enough connectivity may choose to stay. The communities then expand further as the process is iterated by the newly added members.

3.3.1 Initial Communities

Initially every node v_i, $\forall i \in \{1, 2, .., |V|\}$, that has at least K neighbors, builds a community S_i with its neighbors. The number of communities thus is equal

to the number of nodes with degree greater than K. In this way each node becomes a part of the communities initiated by itself and by its neighbors as well, allowing overlap between the communities at the initiation. This approach further helps a node participating in multiple communities to selectively stay in more than one community based on high *connectedness* scores (and leave the rest), simultaneously.

Let the initial community structure be denoted as S^0. Further, let $Added_i = \{v_k | v_k \in N(v_i) \wedge v_k \in S_i\}$, $\forall S_i \in S^0$ be defined and referred to as the set of *peripheral* nodes of S_i, initially. The algorithm henceforth iterates over two phases: *leave* phase and *expand* phase. Let each iteration comprising these two phases be referred to as a *stage*. Also, let the community structure obtained after a certain *stage* l be denoted as S^l. It is important to note that it is always the *peripheral* nodes for any community that either leave or expand in the *stages* following.

3.3.2 Leave phase

In this phase a node leaves some of its communities when it finds itself not *sufficiently connected* in those. Every node v_j is assigned two scores ζ_j^i and ξ_j^i as defined in Equations 4 and 5 respectively. As a result, we obtain a list of *community connectedness* scores $< \zeta^i >_j = \{\zeta_j^i | v_j \in S_i \wedge S_i \in S^l\}$ and *neighborhood connectedness* scores $< \xi^i >_j = \{\xi_j^i | v_j \in S_i \wedge S_i \in S^l\}$ for each node v_j participating in any community in the community structure S^l. A node is considered to be *sufficiently connected* in its community if it has a *community connectedness* score greater than a certain cut-off score. This cut-off here is referred to as *stay cut-off*, and is computed from the list of its *community connectedness* scores. In this paper, a method similar to the first step in bucket sort has been used for fast determination of *stay cut-off*.

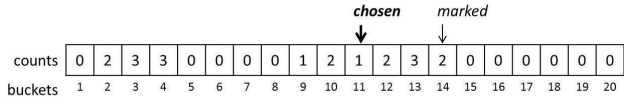


Fig. 3: An illustrative example showing the selection of bucket for given distribution of counts of scores. The rightmost bucket with count greater than 0 is bucket 14 *marked* with an arrow. Next, in the scan towards left, bucket 12 has count as low as that of marked bucket but the bucket to its left has a lower count. So, moving to the next, bucket 11 has count lower than that of 14 and the bucket to its left, i.e., bucket 10 has count greater than this bucket. So bucket 11 is the one *chosen*.

For each node $v_j \in V$, the entire range of scores, which lie between 0 and 1 by definition, is divided into $\max(20, N(v_j))$ number of buckets of equal sizes. The initial count of number of scores that fall in each bucket is set to 0. The count for a bucket is incremented when a score in the list $< \zeta^i >_j$ falls within its range. Once done, the rightmost bucket having count greater than 0 is marked. From there, the bucket list is scanned towards left until, either we have found a bucket that has a count lesser than or equal to that of marked bucket and the count of the bucket to its left is greater than or equal to that of the current one, or we have reached the leftmost bucket. Figure 3 illustrates with an example the marked and the chosen bucket. The lower bound of this bucket is chosen as the *stay cut-off* $\zeta_j^{cut-off}$ for v_j.

The proposed cut-off selection method has been chosen after observing the score distributions. It helps in selecting communities with *near-best* connectivities, unlike those resulting from other simpler alternatives such as mean, median or percentage threshold as cut-off.

Now, for all communities $S_i \in S^l$, a *peripheral* node $v_k \in Added_i$ leaves S_i if its *community connectedness* score ζ_k^i is lower than its *stay cut-off* $\zeta_k^{cut-off}$. Removal of only *peripheral* nodes ensures that nodes that form the core of a community are never eliminated. However, any community with less than k nodes remaining is considered insignificant and is eliminated. Computation of scores and removal of *peripheral* nodes is performed recursively for the entire community structure till no node leaves any community.

3.3.3 Expand phase

After leave phase, the idea of extending a community to its neighboring nodes is pursued. So, in each community S_i *peripheral* nodes $Added_i$ include each neighboring node v_j, if the following conditions hold:

- the node is not already included,
- the node has high interest in joining this community.

High interest in joining a new community is depicted via a high *neighborhood connectedness* score, ξ_j^i. It is ensured that a node has high *neighborhood connectedness* score when the score is greater than its *join cut-off*, $\xi_j^{cut-off}$. *join cut-off* $\xi_j^{cut-off}$ is computed from the list of *neighborhood connectedness* scores $< \xi^i >_j$ in a way similar to *stay cut-off*.

When a community expands, most of its nodes become less *connected*. This is because an existing node is able to connect to very few of the the newly included nodes. Consequently, the maximum of the *community connectedness* scores decreases for all nodes. On the other hand, the number of edges (friends for each node) in that community increases. It is comprehended that a node has interest in joining a new community if it has at least as many connections (friends) in the one concerned as it had in the communities in previous stage. At this point, *neighborhood connectedness* score helps in the decision of a node to participate in a new community and prevents expansion of the already discovered communities into sparser subgraphs.

Algorithm 1 Fast Overlapped Community Search

Input: $G = (V, E)$: input graph, K: minimum connections for a node within a community, OVL: maximum allowed overlap between communities
Output: $S = \{S_i | S_i \subseteq V$ and S_i is a community$\}$
Auxiliary Variables: $n = |V|$, $N(v)$ = neighbors of node v, $Added_i$ = Nodes added to community S_i in last round

procedure PREFERREDCOMMUNITIES(G, K, OVL)
  $S = \emptyset$
  InitializeCommunities(G, K, S)
  $expand \leftarrow 1$
  while $expand$ **do**
    $leave \leftarrow 1$
    while $leave$ **do**
      $leave \leftarrow 0$
      LeaveCommunities(S, K, OVL, $leave$)
    end while
    $expand \leftarrow 0$
    ExpandCommunities(S, $expand$)
  end while
  return S
end procedure

function INITIALIZECOMMUNITIES(G, K, S) /* Community of each node in initialized by the node $v \in V$ and its neighbors $N(v)$ if $|N(v)| \geq K$ */
  for each $i \in \{1, 2, .., n\}$ **do**
    if $|N(v_i)| \geq K$ **then**
      $S_i = \{v_i\} \bigcup N(v_i)$
      $Added_i \leftarrow N(v_i)$
      $S = S \bigcup S_i$
    else
      $S_i = NULL$, $Added_i = NULL$
    end if
  end for
end function

function LEAVECOMMUNITIES(S, K, OVL, $leave$) /* In each community S_i node $v \in S_i$ leaves S_i if its *community connectedness* score is less than *stay cut-off*. Updated communities of size less than K are deleted */
  Eliminate near-duplicate community S_i if $\exists u_j \in S_i$, $j \neq i$ such that $\psi(S_i, S_j) > OVL$ (see Equation 6), $\forall S_i \in S$
  Compute *community connectedness* scores $< \zeta^i >_j$ and *neighborhood connectedness* scores $< \xi^i >_j$ (see Equations 4 and 5 respectively), $\forall v_j \in S_i, \forall S_i \in S$
  Compute *stay cut-off* $\zeta_i^{cut-off}$ (refer text 3.3.2), $\forall v_i \in V$
  for each community $S_i \in S$ **do**
    $S_i = S_i - \{v_k\}$ if $\zeta_k^i < \zeta_k^{cut-off}$, $\forall v_k \in Added_i$
    if S_i updated in previous step **then**
      if $|S_i| \leq K$ **then**
        $S = S - \{S_i\}$ /* Community S_i is deleted */
      else
        $leave \leftarrow 1$
      end if
    end if
  end for
end function

function EXPANDCOMMUNITIES(S, $expand$) /* For each community S_i, each adjacent $u \in N(v)$ of each node $v \in Added_i$ is included in S_i if u is not in S_i and it builds up a *neighborhood connectedness* score greater than its *join cut-off* */
  Compute *join cut-off* $\xi_j^{cut-off}$ (refer text 3.3.3), $\forall v_j \in V$
  for each community $S_i \in S$ **do**
    $Nowadded_i = \emptyset$
    for each $u_k \in N(v_j)$, $\forall v_j \in Added_i$ **do** /* For each node added to S_i in the last round */
      if $\xi_k^i > \xi_k^{cut-off}$ and $u_k \notin S_i$ **then**
        $S_i = S_i \bigcup \{u_k\}$
        $Nowadded_i = Nowadded_i \bigcup \{u_k\}$
      end if
    end for
    $Added_i = Nowadded_i$
    if $|Added_i| \geq 1$ **then**
      $expand \leftarrow 1$
    end if
  end for
end function

After this, $Added_i$ is set to contain the newly added nodes of S_i (or ϕ if no new node added), for either removal or expansion in the next *stage*. Expansion of only *peripheral* nodes, on one hand, allows for re-inclusion of removed nodes, and on the other hand takes care that nodes which do not fit in the community are not repetitively added and later removed during *leave* phase. It also helps FOCS converge.

After *expand* phase, the community structure so obtained at this *stage* is passed as input to the *leave* phase of the next *stage*. At a certain *stage*, all the *peripheral* nodes of some particular communities are removed during the *leave* phase. These communities can not expand further in later *stages*. However, such a community still contributes to the list of *connectedness* scores maintained for its nodes. FOCS stops when, in a *stage*, there are no *peripheral* nodes remaining in all existing communities.

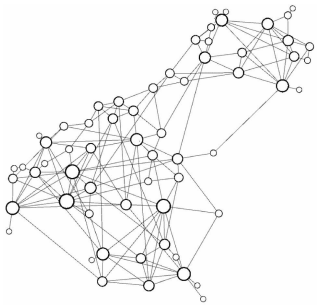
3.3.4 Duplication Removal

Overlapped community detection algorithms allow almost all nodes in a network to be a part of multiple communities. This is because each initial community is allowed to expand to include nodes irrespective of their existence in other communities. Thus, at each phase certain communities may grow to become *near-to-duplicate* communities. Such *near-to-duplicate* community pairs (C, C') are identified via the similarity measure defined as follows [36]:

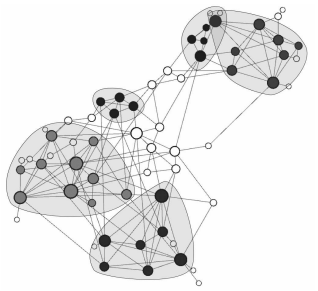
$$\psi(C, C') = \frac{|C \bigcap C'|}{min(|C|, |C'|)} \tag{6}$$

Duplication removal is performed during each stage, before passing communities to *leave* phase and after every iteration within it. Duplication removal is essential from two viewpoints: (i) this prevents the score distribution from being undesirably skewed, (ii) with a number of *near-to-duplicate* communities removed, the computation time is also reduced.

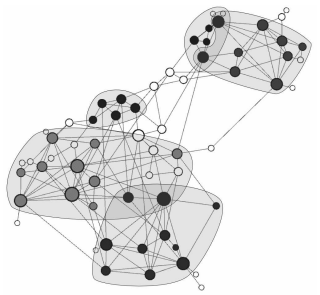
Duplication removal, in FOCS, takes a parameter



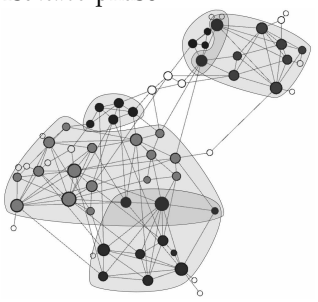
(a) Dolphin network reprsented as graph in force-directed layout using *Gephi* [35] tool



(b) Communities found after 1st *leave* phase



(c) Communities found after 1st *expand* phase



(d) After *expand* phase in *stage* 4, no node added: final communities obtained

Fig. 4: Fast Overlapped Community Search (FOCS) applied to the Dolphin network. Circles and lines represent nodes and edges of the network respectively. Each translucently shaded and bordered region enclosing nodes represents a community. Nodes that are not color-filled do not belong to any community. These communities evolved with OVL set to 0.5 and $K = 2$.

OVL as input. OVL sets a threshold for the maximum overlap allowed between two communities, before they can be identified as *near-duplicates*. The smaller of the two communities S_i and S_j is deleted when similarity measure $\psi(S_i, S_j)$ crosses this threshold. An $OVL = 1$ implies elimination of a duplicate community when it is exactly identical to another. Reasonably, OVL must be set to ≥ 0.5 and less than 1 for early identification of duplicates in community structure. We have taken $OVL = 0.6$ in our work. However, we have experimented with different values of OVL and observed stability in output in qualitative terms when set in the range 0.5 to 0.7. Figure 5 can be referred for change in community statistics with variation in OVL, when FOCS is simulated on Amazon network [34].

Given an underlying static graph $G = (V, E)$, the proposed algorithm, Fast Overlapped Community Search (FOCS)is stated in Algorithm 1. Figure 4 shows how the detected communities evolve over stages with the execution of FOCS on the dolphin network [37].

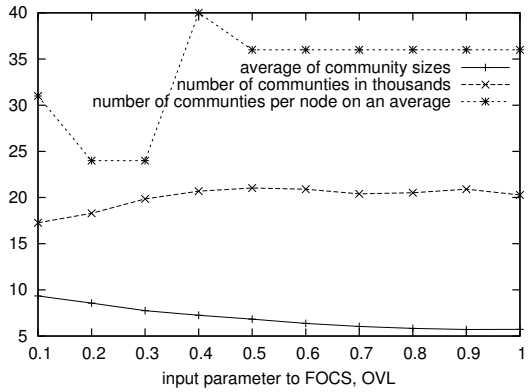


Fig. 5: Change in community statistics when input parameter to FOCS, OVL is varied, simulated on Amazon network [34].

3.4 Complexity Analysis

For a given undirected, unweighted graph $G(V, E)$, let $n = |V|$ be the number of nodes and let $m = |E|$ be the number of edges. During the initialization of communities, the entire adjacency list is scanned once so that a node forms a community if it has more than K neighbors. A scan of the adjacency list requires time $O(n+m)$. However, in most cases the network is connected and the required time is in $O(m)$. Henceforth, the initial communities consecutively shrink and expand through removal or expansion of *peripheral* nodes in the community, respectively. It must be noted that, nodes that are *peripheral* in current stage will not be so in the next stage.

Let l be the average number of communities per node (refer to Theorem A.1 for derivation of upper bound of l). Then, a total of nl *leaves* or *expansions* will take place in the algorithm. The leave phase requires computation of scores ζ_j^i and ξ_j^i for node v_j in community $S_i \in S$, where S is the community structure. Each such computation involves comparison of the community members of S_i against the adjacency list corresponding to v_j to get the total number of adjacents of v_j within the community. Considering the average community length to be l with n number of communities, such computation takes $O(l)$ time. So nl number of computations take $O(nl^2)$ time. Computation of the cut-off scores $\zeta_j^{cut-off}$ and $\xi_j^{cut-off}$ for each node v_j takes time $n * c$ units of time, where c is a constant indicating the number of buckets. So, this takes $O(n)$ time. The elimination of nodes after the computation of scores in the leave phase requires a complete scan of the community structure which is achieved in $O(nl)$ time. Thus, total time taken over all leave phases is in $O(nl^2 + n + nl) = O(nl^2)$.

In the expansion phase, for each *peripheral* node in community S_i, its adjacency list is scanned against S_i to ensure that an adjacent node to be included does not already exist in S_i. This takes $O(nl^2)$ time. Further, for each such adjacent node, computation of

TABLE 2: Comparison of time complexity for various existing methods with FOCS

Algorithm	Time Complexity
CFinder [11]	$O(exp(n))$
Game [28]	$O(m^2)$
MOSES [18]	$O(en^2)$
LFM [9]	$O(n^2)$
OSLOM [38]	$O(n^2)$
COPRA [23]	$O(vmlog(vm/n))$
LinkComm [16]	$O(nK_{max}^2)$
DEMON [25]	$O(nK_{max}^{3-\alpha})$
BigClam [19]	$O(cn+m)$
GCE [10]	$O(mh)$
SLPA [21]	$O(m)$
FOCS	$O(n+m) \simeq O(m) \because m > n$

m = No. of edges in the network
n = No. of nodes in the network
K_{max} = Maximum degree for any node in the network
α : Network has power law degree distribution with $p_k = k^{-\alpha}$
c = Sublinear term in k, the number of communities (exact relation is not clearly stated in [19])
h = No. of cliques
e = No. of edges to be expanded
v = Maximum number of communities a node can participate in

ξ_j^i requires comparison of adjacent list of this adjacent node against S_i, summing up to $O(nl^3)$ time for all expansion phases combined. Now, duplication removal involves comparison of each community S_i against other communities of the *peripheral* nodes. So, for each of the n communities, there are l peripheral nodes (over all stages), each having $l-1$ other communities to be compared with, and comparison takes $O(l)$ time, making it to $O(nl^3)$ time for duplication removal.

Thus, FOCS takes $O(m + nl^2 + nl^3 + nl^3) = O(m + nl^3)$ time across initialization, leave phase, expansion phase and duplication removal. From Theorem A.1, Appendix A it is clear that l takes a constant value irrespective of the size of the network. Thereby, time complexity for FOCS is in $O(n+m)$. Apart from being linearly scalable like some other overlapped community detection algorithms, the proposed algorithm has faster implementation owing to its simplicity and the flags maintained that keep from recomputing scores for nodes and/or communities that do not undergo any change across phases and stages.

Table 2 shows a comparison of the time complexity for various existing methods that has been used for comparison with FOCS.

4 Empirical Results

The performance of a community detection algorithm can be evaluated both on real and simulated networks. However, the simulated networks do not capture some of the important characteristics associated to community structure in real networks.

We may discuss about the LFR benchmark graphs in this regard [9]. LFR benchmark graphs are a family of artificially simulated graphs which allow communities to overlap. These graphs demonstrate some important features of real networks such as the scale-free property and the power law distribution of community sizes. However, LFR assigns for each node, an equal number of its neighbors to different clusters such that sums of the number of neighbors in different communities for the nodes closely follow the degree distribution. This is not the case in real networks (as one can see in Figure 6, the sums are higher than degrees in several cases). Moreover, the standard deviation of the number of neighbors in different communities for a particular node roughly increases with its degree and correspondingly increasing number of communities it is participating in (see Figure 7). Thus, the fraction of neighbors participating in different communities of a node will be far from equal, unlike the case of LFR graphs where they are close to equal. Further, in case of LFR graphs a node is assigned to either a single community or an equal number of multiple communities, which makes them all the more unrealistic because the distribution of number of communities per node in real networks follow a heavy tailed power law distribution [14]. For all these reasons we evaluate the performance of FOCS over the real networks only.

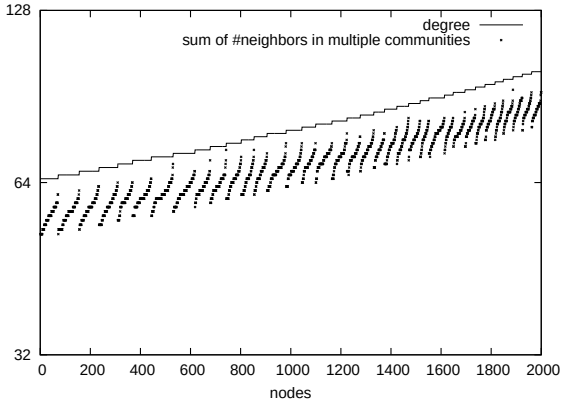
It is argued that modularity tends to produce larger communities, and imposes a limit to resolution [39]. Again, it has been shown that modularity follows the same pattern over different classes of networks [40], thus unable to follow the divergent community structures in different real networks. Therefore, the normalized mutual information (NMI) between the detected and the ground-truth communities is used for the purpose of performance evaluation [9].

To test the performance of FOCS on large-scale real networks, FOCS algorithm is evaluated on 7 real networks of large size. All the networks are undirected and unweighted. Find below short descriptions for all considered networks.

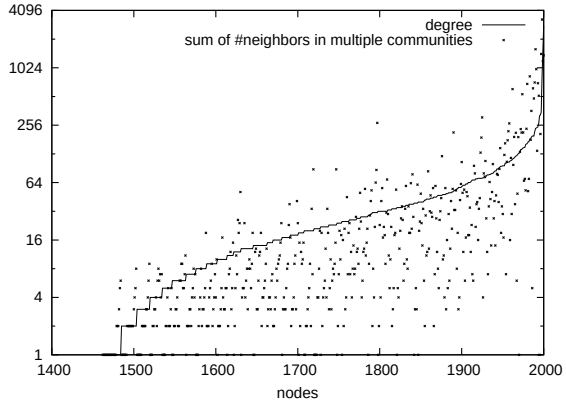
Amazon It is the undirected network of Amazon product co-purchasing. Here, the product categories are hierarchically nested and thus the corresponding network inherently organizes into overlapping community structure. The products in the same ground-truth community share a common function [34].

DBLP It is the scientific collaboration network of DBLP computer science, where two authors are connected if they have published at least one paper together. Here, publication venues i.e., journals and conferences is used as proxies for ground-truth research communities. In such network members are related to each other pertaining to areas of research, and thus highly overlapping community structure is natural to be observed [34].

YouTube It is a video-sharing web site. Users can form friendship with each other and thus YouTube also depicts a social network. Ground-truth communities considered are groups explicitly formed by users [34].

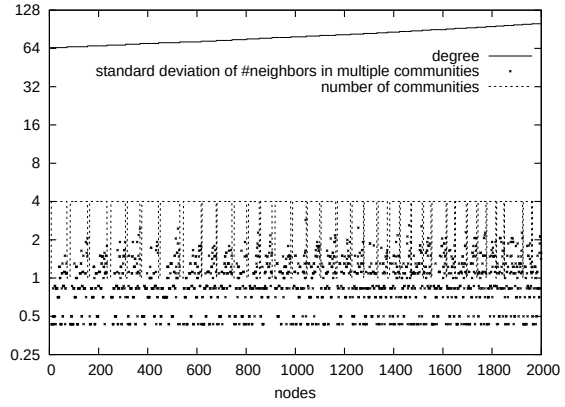


(a) Nodes of artificially simulated LFM graphs

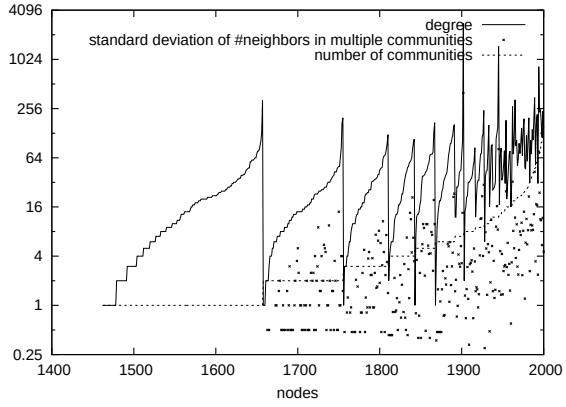


(b) Sampled nodes of LiveJournal network

Fig. 6: Distribution of sum of number of neighbors in multiple communities for nodes of corresponding networks



(a) Nodes of artificially simulated LFM graphs



(b) Sampled nodes of LiveJournal network

Fig. 7: Distribution of standard deviation of number of neighbors in multiple communities for nodes of corresponding networks

LiveJournal It is a free on-line blogging community where users declare friendship with each other. Ground-truth communities are groups explicitly created by users based on common interest topics, affiliations, and geographical regions. Other users in the network then join some of these communities. Communities belong to one of the categories: culture, entertainment, expression, fandom, life/style, life/support, gaming, sports, student life and technology [34].

Orkut It is a free on-line social network where users form friendship with each other. Ground-truth communities are defined on a basis similar to that of LiveJournal [34].

Yeast PPIN The yeast interaction network is collected and combined from 3 different sources –Y2H-Union containing 2930 interactions [41], 2770 interactions from [42], and only the positive examples i.e., top 58 interactions from [43]. Redundancies and self-loops are removed, resulting in a network of 2705 interactions among 1966 proteins. The set of protein complexes considered as true community set is CYC2008 collected from [44]. From the complexes in CYC2008 the proteins that are not in the interaction dataset are removed, followed by elimination of complexes containing 2 or less protein subunits. Following the filtration process 137 out of 408 original complexes remain.

Human PPIN The human protein interactome is the PCDq dataset collected from Results of computational analysis section [45]. It provides for both the interaction network and the complexes (both experimentally verified and computationally predicted using DBClus). The complete interaction network with 32,198 interactions among 9,268 proteins is used. The human protein complex dataset contains 1,078 complexes constituting of 3,759 proteins. Among these, only complexes that belong to either category I or category II are considered. These are the complexes with high number of proteins experimentally verified (for details check [45]). Further complexes of size 2 or less are filtered out, resulting in a total of 1221

TABLE 3: Dataset Statistics. D: average degree, D_{max}: maximum degree, C: number of communities, S: average community size, S_{max}: maximum community size, M: number of communities per node on an average, M_{max}: maximum number of communities any node partcipated in, F: fraction of nodes that participated in atleast one community, F_{2+}: fraction of nodes that participated in more than one community. K denotes a thousand and M denotes a million.

Networks	#Nodes	#Edges	D	D_{max}	C	S	S_{max}	M	M_{max}	F	F_{2+}
Amazon	0.3M	0.9M	5.53	549	151K	19.38	53.5K	8.74	170	0.94	0.91
DBLP	0.3M	1M	6.62	343	13.5K	53.41	7.6K	2.27	124	0.82	0.35
YouTube	1.1M	3M	5.27	28.8K	8.4K	13.5	3K	0.1	173	0.04	0.02
LiveJournal	4M	34.7M	17.35	14.8K	0.3M	22.31	0.18M	1.6	579	0.27	0.18
Orkut	3M	117.2M	76.3	33.3K	6.3M	14.16	9.1K	29	2.6K	0.75	0.70
Yeast PPIN	1.9K	2.7K	3.95	90	137	5.38	21	0.28	5	0.23	0.03
Human PPIN	9.3K	32.2K	6.95	342	1078	4.5	32	0.52	21	0.41	0.06

complexes formed of 4,325 proteins.

The size of the networks ranges from hundreds of thousands to millions of nodes and a hundred of millions of edges. The number of ground-truth communities, community sizes and average node membership for the communities too range over a large scale. Table 3 provides the specifics.

Protein complexes are coherent group of proteins that bind at same time and place, to perform a particular function. A single protein is known to often bind with a multiple set of proteins at different time and location for different functions, thereby resulting in overlapped complexes in the protein interactome. The protein interactomes available till date though incomplete are expected to closely follow the complete interactome structure. The interactomes collected are the most complete available.

Table 4 reports the execution time taken by the various algorithms on the considered networks. Table 5 summaries the results with each cell representing the NMI between the detected and the ground-truth communities.

FOCS is compared with seven widely used overlapped community detetction algorithms namely Greedy Clique Expansion [10], MOSES [18], OSLOM [38], COPRA [23], SLPA [21], Link Communities [16] and BigClam [19]. Greedy clique expansion (GCE) expands cliques greedily to include edges such that within community edge density is improved. MOSES employs stochastic block model based community detection. OSLOM finds communities based on the difference between modularity of a candidate community and that of the same set of nodes in a randomly generated network. In COPRA each node updates its belonging coefficient and decides on its set of community labels by averaging that of its neighbors in synchronous fashion. SLPA propagates community labels between nodes such that a listener node receives and saves the most probable label among those sent by its neighbors where each neighboring node sends a label with probability proportional to its occurrence frequency in memory over multiple iterations. Link Communities (LinkComm), on the other hand, performs agglomerative hierarchical clustering where similarity between nodes is a function of the commonalities in their respective neighborhoods. BigClam employs non-negative matrix factorization method along with block stochastic gradient descent to optimize the model likelihood of explaining the links in network based on communities the nodes participate.

The original implementations have been used for each of the listed algorithms. Further, they have been executed having their parameters set to the default values, except for GCE, where minimum cluster size is changed to 3 instead of 4. Additionally for LinkComm, SLPA, and COPRA, the communities of size 2 or less are filtered out. COPRA also requires one to set the maximum number of communities a node can participate as an input parameter, v which was tested for values starting from 2, increasing by 1 each time until the results became worse. The results reported are those with v set to values that yielded community structure with close match to the number of ground-truth communities for different networks. Similarly, results from SLPA depends heavily on the probability threshold parameter, r which was tested for $r \in [0.01, 0.5]$ and chosen such that the number of output communities was close to that reported for the corresponding ground-truth communities. Tables 4 and 5 report results for COPRA with v set to 9, 4, 3, 2, and 2 for Amazon, DBLP, YouTube, Yeast PPIN, and Human PPIN respectively. Results reported for SLPA are those with r set to 0.01, 0.05, 0.01, 0.5, and 0.05 for these datasets respectively. For BigClam, either number of communities, or a range of number of communities to be tested is required as input. We tested with number of communities equal to that appearing in ground truth communities for the networks, as well as for a range encompassing outputs from other algorithms. The number of communities which yielded the best result was noted, and the time shown in Table 4 is for simulation with these exact number of communities as input parameter. Blanks

TABLE 4: Comparison of time taken in detection of communities by FOCS and by seven of the existing algorithms. The blanks in the table denote that the method was allowed to run for 4 hours before any result was generated, after which it was terminated. h, m, and s denote hour(s), minute(s) and second(s) respectively. K denotes a thousand and M denotes a million.

	#Communities/Time Taken							
Networks	MOSES	GCE	OSLOM	COPRA	SLPA	LinkComm	BigClam	FOCS
Amazon	30.2K/160s	25.9K/10s	18.7K/711s	8.4K/1183s	30.5K/456s	61.5K/14s	151K/1.18h	20.9K/2s
DBLP	46.4K/273s	22.6K/16s	22.2K/21m	14.9K/180s	22.2K/578s	78.4K/34s	39.6K/33m	24.2K/2s
YouTube	8K/1.9h	-	-	12K/238s	39.9K/104m	5.1K/1.5h	8K/1.4h	7K/52s
LiveJournal	-	-	-	-	-	-	-	0.2M/312s
Orkut	-	-	-	-	-	-	-	0.2M/48m
Yeast PPIN	76/18s	92/0s	74/3s	86/0s	243/1s	159/0s	137/1s	32/0s
Human PPIN	106/30s	284/4s	206/60s	26/1s	337/3s	436/1s	1078/57s	114/0s

TABLE 5: Comparison of NMI between ground-truth communities and communities detected by FOCS and by seven of the existing algorithms. The blanks in the table denote that the method was allowed to run for 4 hours before any result was generated, after which it was terminated.

	NMI							
Networks	MOSES	GCE	OSLOM	COPRA	SLPA	LinkComm	BigClam	FOCS
Amazon	0.2239	0.2164	0.1851	0.2076	0.1208	0.2558	0.2421	0.2075
DBLP	0.153	0.1374	0.1276	0.1484	0.1191	0.2112	0.1448	0.2135
YouTube	0.0127	-	-	0.0150	0.0025	0.0161	0.0008	0.0225
LiveJournal	-	-	-	-	-	-	-	0.0307
Orkut	-	-	-	-	-	-	-	0.0611
Yeast PPIN	0.1064	0.1322	0.0481	0.1236	0.0502	0.1148	0.004	0.1284
Human PPIN	0.0793	0.0481	0.0744	0.2510	0.1305	0.1106	0.0328	0.2471

in Tables 4 and 5 show that GCE and OSLOM could not produce results for dataset YouTube within four hours time, while none of the methods except FOCS scaled to datasets LiveJournal and Orkut in the same time. COPRA and SLPA, however, faced memory limitations much earlier for datasets LiveJournal and Orkut. The performance of the other algorithms including LFM [9], DEMON [25], and game-theoretic [28] are eliminated from comparison as they could not produce results even after four hours of execution for any of the social network datasets. The game-theoretic algorithm in contradiction to its claim does not converge.

The results depict significant gain in terms of execution time as compared to the other algorithms. Interestingly, it does not come at the cost of performance. For all networks except Amazon and PPIN networks, FOCS outperforms the other methods. Communities serving as ground-truth for Amazon have very high overlap (about 91% nodes participate in two or more communities as can be seen in Table 3). Thus, NMI values for Amazon mostly conform with methods that yield very high number of overlapping communities. LinkComm, though efficient in detecting most of the communities correctly does not scale well with increasing network sizes. BigClam performs well with input for number of communities set equal to that in ground-truth communities except in the case of DBLP network. It performs competitively only for the case of Amazon dataset, but requires large amount of time. COPRA produces strong results for the PPIN networks which mostly have disjoint communities, with very few nodes participating in overlaps between communities. Figure 8 shows the runtime of FOCS versus the number of edges, m, as compared to all the seven existing algorithms considered. The figure depicts run time for five of the social network datasets considered, with inline figure depicting run-

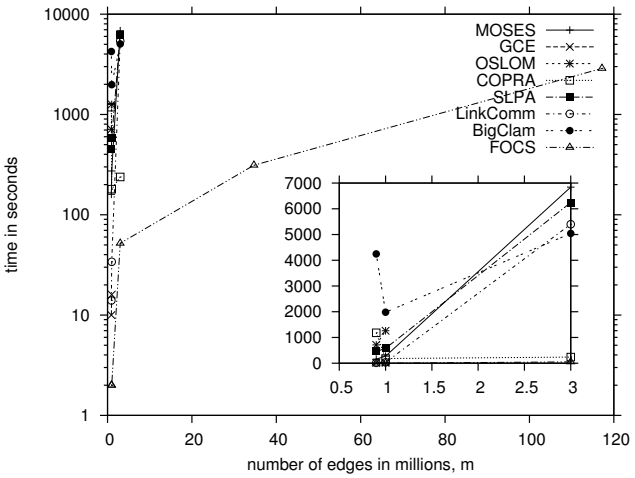


Fig. 8: Runtime of FOCS compared to seven of the existing algorithms for five of the social network datasets [34] with increasing number of edges, m.

time for Amazon, DBLP and YouTube only. None of the algorithms except FOCS scales well to the two largest social network datasets considered, within the given time and memory constraints.

5 Conclusion

Social networks are complex and large. FOCS (Fast Overlapped Community Search) explores communities rapidly by selecting only those where all nodes are locally well connected. The *community connectedness* and *neighborhood connectedness* scores, which are computed for each node throughout the algorithm reflect real world community properties. These make the algorithm applicable to real networks of varying sizes. Users are free to set the maximum allowed overlap between any two communities, and the minimum number of neighbors that a node should have, to determine its membership in any community.

One of the limitations of FOCS is that the maximum number of communities that can be detected by this method is equal to the number of nodes in a network. Whereas, in social networks, as can be seen in Orkut [14], the number of communities can in fact be double the number of nodes. This happens when a node is allowed to create multiple communities. We try to address this issue in our future work. Further, we would like to extend the method to work with weighted and/or directed networks, dynamic networks, etc.

Appendix A

Theorem A.1. *Given OVL as the maximum allowed overlap between any two communities for a network, the average number of communities per node l is maximally bounded by $\frac{1}{1-OVL}$.*

Proof: We are given an undirected, unweighted network represented by graph $G(V, E)$ with V is the set of vertices and E is the set of edges. Let $n = |V|$. In order to find the upper bound for l, one needs to assign maximum number of communities to each of the nodes in the network. As per the proposed algorithm FOCS, a maximum of n communities can be formed. Consequently, each node $v \in V$ may belong to all n communities, resulting in 100% overlap (or, similarity as defined in Equation 6) between any two communities. We need to assign maximum number of communities per node such that the overlap between any pair of communities is constrained by the given overlap threshold OVL. First, we prove by induction that if each node belongs to $l = n - s$ communities on average, where $s \in N$, set of natural numbers, the minimum achievable maximum of overlap between all pairs of communities is, say $O_{max_{min}} = \frac{n-s-1}{n-s}$. So, the induction statement is
$P(s) : l(s) = n - s \Rightarrow O_{max_{min}}(s) = \frac{n-s-1}{n-s}$.

Basis: $P(1)$ holds
When $s = 1$, $l(1) = n - 1$, i.e., on an average each node belongs to $n - 1$ communities. So, we need to remove in total n nodes combined from n communities to achieve $l(1)$ (from the scenario where each node belonged to all n communities). In order for minimum achievable maximum of overlap between all pairs of communities, a unique node is removed from each of the n communities. Thus, each community now has $n - 1$ nodes and there are exactly $n - 2$ nodes in the overlapped region between any pair of communities (since for each of the pair, both the nodes removed belonged to overlapped region). This results in an overlap of $\frac{n-2}{n-1}$. On the other hand, $O_{max_{min}}(1) = \frac{n-1-1}{n-1} = \frac{n-2}{n-1}$. It must be noted that in the current situation, for a community pair, if the same node belonging to the overlapped region is removed from both communities, the overlap for this pair remains 100% which is higher than $O_{max_{min}}(1)$. And, if more than 1 nodes are removed from the same community, it again results in a 100% overlap between this community and the community with no node removed. Hence, $P(1)$ holds.

Inductive Step: $P(k)$ holds $\Rightarrow P(k+1)$ holds
We assume that $P(k)$ holds meaning that when average number of communities per node, $l(k) = n-k$, the minimum achievable maximum of overlap between all pairs of communities is $O_{max_{min}}(k) = \frac{n-k-1}{n-k}$ with each community containing $n-k$ nodes and $n-k-1$ nodes belonging to overlapped region between every pair.

For $s = k+1$, $l(k+1) = n-(k+1) = n-k-1$, i.e., each node belongs to $n-k-1$ communities on average. Now, for condition of maximum achievable minimum overlap between all community pairs, a unique node is removed from each of the n communities (from the scenario where $P(k)$ is true) such that for any community pair, one of the nodes removed belonged to overlapped region while the other did not. Thus, number of nodes in each community becomes $n-k-1$, as a node reduces, and number of nodes in the overlapped region between the pair is reduced by exactly one. This results in an overlap of $\frac{n-k-1-1}{n-k-1} = \frac{n-k-2}{n-k-1}$. On the other hand, $O_{max_{min}}(k+1) = \frac{n-(k+1)-1}{n-(k+1)} = \frac{n-k-2}{n-k-1}$, thereby showing that $P(k+1)$ holds.

Thus, $P(s)$ holds for all natural number s. Now, given maximum allowed overlap OVL for a network, we want to find out the maximum value for average number of communities per node, $l(s)$, which is when $OVL = O_{max_{min}}(s)$. This ensures that the community structure as a whole exists with no community pair having an overlap greater than $O_{max_{min}}(s)$. So, we have

$$\begin{aligned} OVL &= O_{max_{min}}(s) \\ &= \frac{n-s-1}{n-s} \\ &= 1 - \frac{1}{n-s} \\ or, \quad s &= n - \frac{1}{1-OVL} \end{aligned} \tag{7}$$

Now, the average number of communities per node, $l(s)$ in this case is given by $n - s$. So, we have

$$\begin{aligned} l(s) &= n - s \\ &= n - (n - \frac{1}{1-OVL}) \\ &= \frac{1}{1-OVL} \end{aligned} \tag{8}$$

Hence, the theorem follows. □

REFERENCES

[1] K. Dasgupta, R. Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. A. Nanavati, and A. Joshi, "Social ties and their relevance to churn in mobile telecom networks," in *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*. ACM, 2008, pp. 668–677.

[2] J. Xu and H. Chen, "Criminal network analysis and visualization," *Communications of the ACM*, vol. 48, no. 6, pp. 100–107, 2005.

[3] S. Fortunato, "Community detection in graphs," *Physics Reports*, vol. 486, no. 3, pp. 75–174, 2010.

[4] M. Plantié and M. Crampes, "Survey on social community detection," in *Social Media Retrieval*. Springer, 2013, pp. 65–85.

[5] J. Xie, S. Kelley, and B. K. Szymanski, "Overlapping community detection in networks: The state-of-the-art and comparative study," *ACM Computing Surveys (CSUR)*, vol. 45, no. 4, p. 43, 2013.

[6] S. E. Schaeffer, "Graph clustering," *Computer Science Review*, vol. 1, no. 1, pp. 27–64, 2007.

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

[8] 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, p. P10008, 2008.

[9] A. Lancichinetti, S. Fortunato, and J. Kertész, "Detecting the overlapping and hierarchical community structure in complex networks," *New Journal of Physics*, vol. 11, no. 3, p. 033015, 2009.

[10] C. Lee, F. Reid, A. McDaid, and N. Hurley, "Detecting highly overlapping community structure by greedy clique expansion," *ArXiv e-prints*, feb 2010.

[11] 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, 2005.

[12] T. Evans, "Clique graphs and overlapping communities," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no. 12, p. P12037, 2010.

[13] N. Mishra, R. Schreiber, I. Stanton, and R. E. Tarjan, "Finding strongly knit clusters in social networks," *Internet Mathematics*, vol. 5, no. 1-2, pp. 155–174, 2008.

[14] J. Yang and J. Leskovec, "Structure and overlaps of communities in networks," *CoRR*, vol. abs/1205.6228, 2012.

[15] S. L. Feld, "The focused organization of social ties," *American journal of sociology*, pp. 1015–1035, 1981.

[16] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann, "Link communities reveal multiscale complexity in networks," *Nature*, vol. 466, no. 7307, pp. 761–764, 2010.

[17] T. Evans and R. Lambiotte, "Line graphs, link partitions, and overlapping communities," *Physical Review E*, vol. 80, no. 1, p. 016105, 2009.

[18] A. McDaid and N. Hurley, "Detecting highly overlapping communities with model-based overlapping seed expansion," in *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE, 2010, pp. 112–119.

[19] J. Yang and J. Leskovec, "Overlapping community detection at scale: a nonnegative matrix factorization approach," in *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013, pp. 587–596.

[20] J. Xie and B. K. Szymanski, "Community detection using a neighborhood strength driven label propagation algorithm," in *Network Science Workshop (NSW), 2011 IEEE*. IEEE, 2011, pp. 188–195.

[21] ——, "Towards linear time overlapping community detection in social networks," in *Advances in Knowledge Discovery and Data Mining*. Springer, 2012, pp. 25–36.

[22] ——, "Labelrank: A stabilized label propagation algorithm for community detection in networks," in *Network Science Workshop (NSW), 2013 IEEE 2nd*. IEEE, 2013, pp. 138–143.

[23] S. Gregory, "Finding overlapping communities in networks by label propagation," *New Journal of Physics*, vol. 12, no. 10, p. 103018, 2010.

[24] 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, p. 036106, 2007.

[25] M. Coscia, G. Rossetti, F. Giannotti, and D. Pedreschi, "Demon: a local-first discovery method for overlapping communities," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 615–623.

[26] M. Magdon-Ismail and J. Purnell, "Ssde-cluster: Fast overlapping clustering of networks using sampled spectral distance embedding and gmms," in *Privacy, security, risk and trust (passat), 2011 ieee third international conference on and 2011 ieee third international conference on social computing (socialcom)*. IEEE, 2011, pp. 756–759.

[27] S. Tsironis, M. Sozio, M. Vazirgiannis, and L.-E. Poltechnique, "Accurate spectral clustering for community detection in mapreduce."

[28] W. Chen, Z. Liu, X. Sun, and Y. Wang, "A game-theoretic framework to identify overlapping communities in social networks," *Data Mining and Knowledge Discovery*, vol. 21, no. 2, pp. 224–240, 2010.

[29] H. Alvari, S. Hashemi, and A. Hamzeh, "Discovering overlapping communities in social networks: A novel game-theoretic approach," *AI Communications*, vol. 26, no. 2, pp. 161–177, 2013.

[30] R. Narayanam and Y. Narahari, "A game theory inspired, decentralized, local information based algorithm for community detection in social graphs," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 1072–1075.

[31] J. Baumes, M. Goldberg, and M. Magdon-Ismail, "Efficient identification of overlapping communities," in *Intelligence and Security Informatics*. Springer, 2005, pp. 27–36.

[32] F. Bonchi, A. Gionis, and A. Ukkonen, "Overlapping correlation clustering," in *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 2011, pp. 51–60.

[33] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.

[34] J. Yang and J. Leskovec, "Defining and evaluating network communities based on ground-truth," in *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*. ACM, 2012, p. 3.

[35] M. Bastian, S. Heymann, M. Jacomy *et al.*, "Gephi: an open source software for exploring and manipulating networks." *ICWSM*, vol. 8, pp. 361–362, 2009.

[36] J. Baumes, M. K. Goldberg, M. S. Krishnamoorthy, M. Magdon-Ismail, and N. Preston, "Finding communities by clustering a graph into overlapping subgraphs." *IADIS AC*, vol. 5, pp. 97–104, 2005.

[37] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, "The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations," *Behavioral Ecology and Sociobiology*, vol. 54, no. 4, pp. 396–405, 2003.

[38] A. Lancichinetti, F. Radicchi, J. J. Ramasco, and S. Fortunato, "Finding statistically significant communities in networks," *PloS one*, vol. 6, no. 4, p. e18961, 2011.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2015.2445775, IEEE Transactions on Knowledge and Data Engineering

[39] S. Fortunato and M. Barthelemy, "Resolution limit in community detection," *Proceedings of the National Academy of Sciences*, vol. 104, no. 1, pp. 36–41, 2007.
[40] J. Leskovec, K. J. Lang, and M. Mahoney, "Empirical comparison of algorithms for network community detection," in *Proceedings of the 19th international conference on World wide web*. ACM, 2010, pp. 631–640.
[41] H. Yu, P. Braun, M. A. Yıldırım, I. Lemmens, K. Venkatesan, J. Sahalie, T. Hirozane-Kishikawa, F. Gebreab, N. Li, N. Simonis *et al.*, "High-quality binary protein interaction map of the yeast interactome network," *Science*, vol. 322, no. 5898, pp. 104–110, 2008.
[42] K. Tarassov, V. Messier, C. R. Landry, S. Radinovic, M. M. S. Molina, I. Shames, Y. Malitskaya, J. Vogel, H. Bussey, and S. W. Michnick, "An in vivo map of the yeast protein interactome," *Science*, vol. 320, no. 5882, pp. 1465–1470, 2008.
[43] J. P. Miller, R. S. Lo, A. Ben-Hur, C. Desmarais, I. Stagljar, W. S. Noble, and S. Fields, "Large-scale identification of yeast integral membrane protein interactions," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 102, no. 34, pp. 12 123–12 128, 2005.
[44] S. Pu, J. Wong, B. Turner, E. Cho, and S. J. Wodak, "Up-to-date catalogues of yeast protein complexes," *Nucleic acids research*, vol. 37, no. 3, pp. 825–831, 2009.
[45] S. Kikugawa, K. Nishikata, K. Murakami, Y. Sato, M. Suzuki, M. Altaf-Ul-Amin, S. Kanaya, and T. Imanishi, "Pcdq: human protein complex database with quality index which summarizes different levels of evidences of protein complexes predicted from h-invitational protein-protein interactions integrative dataset," *BMC systems biology*, vol. 6, no. Suppl 2, p. S7, 2012.

Sanghamitra Bandyopadhyay did her Ph. D. in Computer Science from ISI. She is currently a Professor at the Indian Statistical Institute, Kolkata, India. She has authored/co-authored more than 250 technical articles and published five authored and edited books. Her research interests include computational biology and bioinformatics, soft and evolutionary computation, pattern recognition and data mining. She is a Fellow of NASI and INAE, India and recipient of several prestigious awards including the Humboldt Fellowship from Germany, ICTP Senior Associate, Trieste, Italy and the Shanti Swarup Bhatnagar Prize in Engineering Science.

Garisha Chowdhary did her B. Tech and M. Tech in computer science from Biju Pattnaik Univeristy of Technology and Jadavpur University respectively. Currently she is a senior research fellow in Machine Intelligence Unit of Indian Statistical Institute, Kolkata, India. Her research interest includes machine learning and complex network analysis.

Debarka Sengupta did his B. Tech and Ph. D. in computer science and engineering from West Bengal University of Technology and Jadavpur University respectively. He worked in Machine Intelligence Unit of Indian Statistical Institute as a research fellow during March, 2009- March, 2013. Currently he is a postdoctoral fellow in Computational and Systems Biology group of Genome Institute of Singapore. His research interest includes computational biology, functional genomics and machine learning.