Alghamdi and Greene *Applied Network Science* (2019) 4:63
https://doi.org/10.1007/s41109-019-0175-7

Applied Network Science

RESEARCH Open Access

Active semi-supervised overlapping community finding with pairwise constraints

Elham Alghamdi* and Derek Greene

*Correspondence:
elham.alghamdi@ucdconnect.ie
University College Dublin, Dublin, Ireland

Abstract

Algorithms for finding communities in complex networks are generally unsupervised, relying solely on the structure of the network. However, these methods can often fail to uncover meaningful groupings that reflect the underlying communities in the data, particularly when they are highly overlapping. One way to improve these algorithms is by incorporating human expertise or background knowledge in the form of pairwise constraints to direct the community detection process. In this work, we explore the potential of semi-supervised strategies to improve algorithms for finding overlapping communities in networks. We propose a method, based on label propagation, for finding communities using pairwise constraints. Furthermore, we introduce a new strategy, inspired by active learning, for intelligent constraint selection, which is designed to minimize the level of human annotation required. Extensive evaluations on synthetic and real-world datasets demonstrate the potential of this strategy for effectively uncovering meaningful overlapping community structures, using a limited amount of supervision.

Introduction

In many real-world applications involving machine learning, the tasks do not neatly correspond to the standard distinction between supervised and unsupervised learning. Rather, a limited degree of background knowledge or human annotation would be available to guide the process. In the area of network analysis, tasks such as community detection can potentially benefit from the introduction of "lightweight" supervision originating from domain experts or crowdsourced annotations. Usually, this knowledge is encoded as constraints, indicating that a pair of nodes in the network should always be assigned to the same community or should never be assigned to the same community. For instance, we might be interested in grouping users on a social media platform such as Twitter, based primarily on their follower connections, in order to discover communities of individuals with shared ideologies. In order to improve our ability to achieve this, and go beyond simply looking at connections, we could present pairs of user profiles to a human annotator (referred to as the "oracle") and ask whether those two users should be assigned to the same community or different communities. By harnessing this kind of external knowledge, we can potentially uncover communities of users, which would otherwise be difficult to identify using methods that are solely unsupervised.

Similar scope exists in other areas where community finding has previously been applied and relevant background knowledge exists. In the context of biological network

analysis, researchers have frequently attempted to discover communities corresponding to complexes in protein interaction networks (Jonsson et al. 2006). Here, constraints might be derived from external knowledge bases, such as gene ontologies, in order to support the more accurate identification of biologically-meaningful groups. In the case of other inter-disciplinary applications, such as the extraction of communities from literary character networks (Grayson et al. 2016), constraints can provide a means of incorporating "human in the loop" domain expertise into the community finding process.

Initial work in community detection focused on the development of algorithms to produce disjoint groups (Blondel et al. 2008). However, in many real-world networks we observe pervasive overlap, where nodes belong to many highly-overlapping groups (Ahn et al. 2010). More recently, overlapping community finding algorithms have been developed for application to these networks (Ahn et al. 2010; Lee et al. 2010), but the work has focused only on the unsupervised case. In contrast, work on semi-supervised community finding continues to focus on cases where communities are strictly required to be disjoint (Li et al. 2014).

The aim of this study is to explore the potential of semi-supervised strategies, in particular those employing pairwise constraints, to improve algorithms for finding overlapping communities in social networks. To the best of our knowledge, these constraints have not been previously considered in the context of overlapping communities in the literature. To address this deficit, we introduce two main contributions. The First contribution, we describe a semi-supervised method for overlapping community finding based on a label propagation strategy, which has previously been applied in a purely unsupervised context (Xie et al. 2011). The proposed method, referred to as Pairwise Constrained SLPA (PC-SLPA), involves a speaker-listener information propagation process (Alghamdi and Greene 2018). To encode external supervision, we use pairwise constraints to influence the community finding process.

Since the choice of constraints in semi-supervised learning has been shown to be highly important (Leng et al. 2013), we propose a new strategy, integrated into PC-SLPA, for selecting constraint pairs for which the oracle should be queried. This strategy is specifically designed for the case where communities overlap in a network. However, in most networks only a small percentage of the nodes would provide truly informative pairwise constraints from the perspective of community finding. Thus, we introduce both semi-supervised learning and active learning to propose our second contribution: a novel active semi-supervised algorithm based on PC-SLPA with an active pairwise constraint selection component for selecting informative constraints with limited annotation budget (AC-SLPA). The key challenge is to minimize the level of supervision required, while maintaining or improving our ability to find meaningful community structure. Based on extensive experiments, involving both synthetic and real networks, the results show that the introduction of a relatively small number of constraints can substantially improve our ability to correctly uncover the underlying communities in the data.

The remainder of this paper is structured as follows. "Related work" section provides a summary of relevant work pertaining to semi-supervised learning, active-learning, and community finding. In "Methods" section we describe the proposed methods for community finding, together with an appropriate constraint selection strategy, which incorporates a novel active pairwise constraint selection component. In "Evaluation" section we perform a benchmark evaluation of these techniques on a range of synthetic

and real networks. Finally, "Conclusion" section presents suggestions for extending this work in new directions.

Related work

To provide context for our work, in this section we will describe relevant existing work on community detection in "Community finding" section, semi-supervised learning in "Semi-Supervised learning in community finding" section, and active learning in community detection in "Active learning" section.

Community finding

Finding non-overlapping communities. Algorithms in this context can be broadly grouped into three types: *(1) Hierarchical algorithms* which construct a tree of communities based on the network topology. These can be one of two types: divisive algorithms (Girvan and Newman 2002) or agglomerative algorithms (Clauset et al. 2004). *(2) Modularity-based algorithms* which optimize a well-known modularity objective function to uncover communities in a network (Newman 2006). *(3) Other algorithms* which include those based on label propagation approaches (Xie et al. 2011), spectral methods that make use of the eigenvectors of a graph's adjacency matrix, and methods based on statistical modeling (Fortunato 2010).

Finding overlapping communities. Existing algorithms in this context can be classified into four main categories:

(1) Clique percolation methods: Many algorithms for detecting overlapping communities use cliques based on the assumption that the internal connections of a community are likely to form cliques due to their high density. One of the most popular techniques has been in this category the Clique Percolation Method (CPM) proposed in Palla et al. (2005). This method first finds all maximal cliques in a network, and then constructs a clique-clique overlap matrix using the algorithm from (Everett and Borgatti 1998). The CFinder (Adamcsek et al. 2006) algorithm is a widely-used implementation of the CPM strategy.

(2) Local expansion methods: This category of algorithms detects communities by starting with initial seeds (i.e., single nodes or small groups of nodes), and then expanding them into communities by adding nodes that maximize a given quality function. OSLOM (Lancichinetti et al. 2011) is an example of such an algorithm, which expands communities based on a fitness function measuring the statistical significance of communities with respect to random variations. Another example is MOSES (McDaid and Hurley 2010), which is based on a statistical model and uses an objective function similar to that employed by many greedy optimization techniques. Other algorithms, which follow this type of expansion strategy include LFM (Lancichinetti et al. 2009), GCE (Lee et al. 2010), and EAGLE (Shen et al. 2009).

(3) Link clustering algorithms: This category of algorithms detects communities by splitting the network edges rather than the nodes, typically by using a line graph (Amelio and Pizzuti 2014).

(4) Label propagation algorithms: These attempt to group each node into a community based on its neighboring nodes' affinities. COPRA (Gregory 2010) was the first algorithm which followed this approach. More specifically, COPRA allows a node label

to have more than one community identifier. However, one challenge with COPRA is the selection of an appropriate value for the input parameter v,the maximum number of communities, which effectively controls community overlap. This limitation was overcome by BMLPA (Wu et al. 2012), which introduced a new balanced update strategy that does not limit the number of communities to which a node can be assigned.

Another algorithm based on label propagation is SLPA (Xie et al. 2011), where every node is associated with a corresponding memory to store labels received from other nodes, in terms of their frequency of occurrence. During the process of updating nodes' labels, SLPA takes into account past information that has been observed about each label to make an update decision. This algorithm implements the principle of speaker-listener based information propagation process. Each node can take the role of either a *listener* or a *speaker*, and the roles are switched based on a node's state – i.e. whether a node is providing information or consuming it. In the listener state, a node accepts labels from its neighbors, based on certain rules. In the speaker state, the node chooses a label from its own memory according to certain rules and sends it to neighboring listener nodes. Initially each node is assigned to its own unique label. Then an iterative evaluation stage is repeatedly applied, as follows:

1. Randomly select one node as a listener.
2. Each neighbor of the listener randomly chooses a label from its own memory with a probability proportional to the frequency of occurrence of this label, and sends the label to the listener.
3. The listener chooses the most popular label among the received labels, and then adds it to its own memory.

As a stopping criterion, SLPA stops when reaching a predefined maximum number of iterations, T. Overall, SLPA produces relatively stable outputs when T is more than 20, regardless of network size or structure. A subsequent post-processing stage converts each node?s memory into a probability distribution of labels. If the probability of the frequency of a certain label is less than a predefined threshold $r \in [0, 1]$, the label is removed from a node?s memory. After this thresholding step, all nodes which have the same label are grouped into one community. Nodes that have more than one label, naturally belong to multiple communities. The main feature of this algorithm is the accumulative knowledge of previously-seen labels, which is not present in other label propagation algorithms.

Other methods: A number of overlapping community detection algorithms have been proposed, which cannot be neatly classified into any of the categories above. For instance, CONGA (Gregory 2007) and CONGO (Gregory 2008) are both based on the concept of *betweenness*, which refers to the number of shortest paths that pass through a node in the network. An alternative strategy for finding communities is non-negative matrix factorization (NMF) (Lee and Seung 1999), a general unsupervised learning strategy for performing factorization-based dimensionality reduction and clustering, simultaneously. Recently NMF has been applied to the problem of community detection, in order to factorize the affinity matrix representation of a network, as implemented in algorithms such as BIGCLAM (Yang and Leskovec 2013), BNMTF (Zhang and Yeung 2012), and NMFGR (Liu et al. 2016).

Semi-Supervised learning in community finding

Several types of prior knowledge have been used in semi-supervised strategies to guide the community detection process. The most widely-used approach has been to employ pairwise constraints, either *must-link* or *cannot-link*, which indicate that either two nodes must be in the same community or must be in different communities. This strategy has been implemented via several algorithms, including modularity-based methods (Li et al. 2014), spectral partitioning methods (Habashi et al. 2016; Zhang 2013; Zhang et al. 2013), a spin-glass model (Eaton and Mansbach 2012), and matrix factorization methods (Shi et al. 2015; Zhang 2013). Such approaches have often provided significantly better results on benchmark data, when compared to standard unsupervised algorithms. However, all of these algorithms have been designed to only find non-overlapping communities.

Other authors have used different kinds of prior knowledge to provide limited supervision for community detection. Ciglan et al. (2010) developed an algorithm for finding communities with *size constraints*, where the upper limit size of communities is given as a user-specified input. This algorithm is based on standard label propagation methods for finding disjoint communities. In Wu et al. (2016) an optimization algorithm based on *density constraints* was proposed. This algorithm constructs an initial skeleton of the community structure by maximizing a new criterion function, which incorporates constraints to only find communities with intra-cluster densities above a given threshold. The remaining nodes are subsequently classified with respect to this skeleton. Other algorithms have used *node labels* as prior knowledge to improve the performance of community detection, using an approach which resembles traditional training data in classification (Leng et al. 2013; Liu et al. 2014; Wang et al. 2015). Liu et al. (2015) developed a method that uses a semi-supervised label propagation algorithm based on node labels and negative information, where a node is deemed not to belong to a specific community.

The majority of algorithms in this area aim at detecting disjoint groups of nodes, whereas many real-world networks naturally contain overlapping community structure (Adamcsek et al. 2006). To the best of our knowledge, very little work has been done in the context of finding overlapping communities from a semi-supervised perspective. Dreier et al. (2014) performed some initial work in the area, using supervision in the forming of seeding. Specifically, a small set of seed nodes was selected, whose affinities to a community was provided as prior knowledge in order to infer the rest of the nodes' affinities in the network. In contrast, for our study, we focus on the problem of semi-supervised community detection based on the use of pairwise constraints, since they have proven to be effective in a range of learning contexts (Basu et al. 2004; Greene and Cunningham 2007).

Active learning

Active learning for classification. *Active learning* refers to a sub-field of supervised machine learning, where the goal is to build models that achieve high classification accuracy, while using as little labeled training data as possible. In the standard supervised learning scenario, an algorithm is trained on a predefined batch of labeled data. Since there is no interaction with a human during the training process, this is referred as *passive learning*. In contrast, in an active learning scenario, the model carefully selects the training instances from which the model learns in an incremental manner. This is done by interactively making queries to an "oracle" for labels during the training process. Queries

are usually structured so as to minimize the number of requests that must be made to the oracle, thus reducing the "budget" required for manual work and avoiding the risk of feeding the algorithm with redundant or uninformative information (Krishnakumar 2007). A variety of different active learning strategies have been proposed. Most involve either *stream-based sampling*, which selects instances for labeling from a continuous stream of unlabeled data, or *pool-based sampling*, which selects instances for labeling from a fixed collection of unlabeled data. The key component of active learning is the *query selection strategy*, which determines which instances should be labeled by the oracle. Query strategies can be organized into two main approaches (Prince 2004). The first approach refers to model-based strategies, which generate a model using the current labeled instances, and assess the informativeness of the unlabeled instances based on the derived information from this model. The second general approach for query selection involves a model-free strategy, which evaluates the informativeness of a given instance based on the features of the entire data set. An example method of this latter approach is density-weighted selection, which chooses the instances that the model is most uncertain about, while also ensuring that the instances are representative of the underlying distribution of the data (Donmez et al. 2007).

Active learning for community detection. Beyond the task of classification, researchers have also applied the general concept of active learning to other machine learning tasks, such as semi-supervised clustering (Basu et al. 2004; Greene and Cunningham 2007). This has subsequently motivated the idea of extending active learning to the field of community detection. However, only limited research has been done on this problem until now, with researchers focusing solely on the problem of finding disjoint communities. Leng et al. (2013) did the first work on active learning in community detection. Their proposed method aims to find a small number of "core nodes" that can cover as many underlying communities in a given network as possible, which are subsequently passed to the oracle for annotation. This method uses a model-free-inspired selection strategy, where the nodes are chosen solely based on their local density and representativeness of the communities in a given network. Another model-free-inspired approach was proposed by Cheng et al. (2014), which uses pairwise constraints in order to provide supervision. Specifically, a fitness score is calculated for each node in the network and nodes with values larger than a given threshold are added to a target set. This resulting target set is then partitioned into clusters, and for each cluster the maximal degree nodes along with the boundary nodes are chosen as pairs to query the oracle for labeling as must-link or cannot-link. Yang et al. (2015) also proposed an active learning-style method, which combines pairwise constraints with NMF community detection. Firstly, NMF is applied on the basic network topology. Then an entropy measure is applied to the results, which actively identifies the most uncertain edges for which the endpoint nodes should be labeled by the oracle. A final stage involves removing certain inter-community edges based on the label information.

Methods

In "Related work" section we observed that the majority of work in the semi-supervised learning for community detection literature has focused on the problem of finding disjoint groups, to the exclusion of the task of finding overlapping groups. Therefore, we

have sought to address this gap in the literature by proposing two novel overlapping community finding algorithms which use pairwise constraints to encode external supervision, and suitable for application to arbitrary undirected, unweighted networks. Both algorithms adopt a label propagation approach for community finding, due to the conceptual simplicity of this approach and also due to the linear time complexity. Therefore, we have used SLPA (Speaker-listener Label Propagation Algorithm) (Xie et al. 2011), which is a representative example of label propagation approach. The main feature of SLPA algorithm is the accumulative knowledge of previously-seen labels, which is not present in other label propagation algorithms. Our first proposed algorithm, referred to as Pairwise Constrained SLPA (PC-SLPA), is described in "Semi-Supervised overlapping community finding" section.

Since the choice of constraints in semi-supervised learning has been shown to be highly important (Leng et al. 2013), in "Active semi-supervised overlapping community finding" section we propose a further strategy for selecting informative constraint pairs to pass to the oracle for annotation. Since this strategy is inspired by previous work from active learning, we refer to it as Active Semi-supervised SLPA (AC-SLPA). The goal of this approach is to select informative constraints while using a limited annotation budget.

Pairwise constraints for overlapping communities

Before describing the proposed methods, we firstly discuss the issue of selecting appropriate pairwise constraints for networks containing overlapping communities.

Given a network that contains a set of nodes V, semi-supervised pairwise constraints typically take two possible forms:

1. A *must-link constraint* specifies that two nodes must be in the same community. Let C_{ML} be the must-link constraint set: $\forall\ v_i, v_j \in V$ where $i \neq j$, $(v_i, v_j) \in C_{ML}$ indicates that two nodes v_i and v_j must be assigned to the same community.
2. A *cannot-link constraint* specifies that two nodes must be in different communities. Let C_{CL} be the cannot-link constraint set: $\forall\ v_i, v_j \in V$ where $i \neq j$, $(v_i, v_j) \in C_{CL}$ indicates that v_i and v_j must be assigned to separate communities.

These constraints are provided by the oracle, typically an individual expert or a committee of annotators. The simplest approach for selecting pairwise constraints to present to the oracle is to naïvely select a pair of nodes (v_i, v_j) at random, and query the oracle about whether the pair share a must-link or cannot-link relationship. This process is typically repeated until some supervision budget is exhausted.

In non-overlapping community finding, must-link constraints have a *transitive property*, such that a third must-link relationship can be inferred from two other associated must-link constraint pairs. So, if $(v_i, v_j) \in C_{ML}$, and $(v_i, v_k) \in C_{ML}$, then we can also infer that $(v_j, v_k) \in C_{ML}$ (see Fig. 1a).

However, incorporating constraints into the context of overlapping communities is more challenging. This is because the transitive property does not hold here (see the second example in Fig. 1). Specifically, if $(v_i, v_j) \in C_{ML}$, and $(v_i, v_k) \in C_{ML}$, there are two possible scenarios for the pair (v_j, v_k). It can be the case that either $(v_j, v_k) \in C_{ML}$ or $(v_j, v_k) \in C_{CL}$. This is because an overlapping node v_j can have a must-link constraint with both v_i and v_k, yet these two nodes could belong to two different communities. However, it is also possible that all three nodes are in fact in the same community. Unless we explicitly

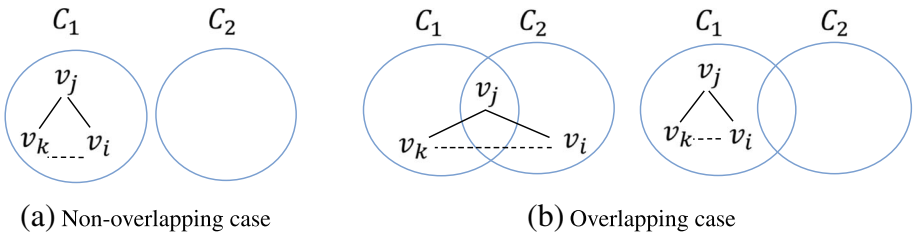

Fig. 1 In the non-overlapping case (**a**), the transitive property allows us to infer a third must-link constraint from two existing must-link constraints. However, this does not automatically apply in the overlapping case (**b**), where two possible situations exist, as shown

inform the algorithm about whether a must-link or cannot-link constraint exists for the pair (v_j, v_k), the algorithm cannot reliably distinguish between the two alternative cases.

If the network has highly-overlapping communities (i.e. each node typically belongs to many communities), then this problematic situation will occur more frequently. Therefore, if we attempt to incorporate pairwise constraints into overlapping community finding without taking this situation into account, the quality of the resulting communities can potentially suffer, even with more constraints are added. Next we introduce a strategy to resolve this issue.

Semi-Supervised overlapping community finding

Here we describe a semi-supervised label propagation procedure for finding overlapping communities, which consists of two distinct phases:

1. Select and pre-process constraints, to resolve the problem of lack of the transitive property for must-link constraints.
2. Apply label propagation-based community finding, in a manner that takes into account the information provided by the selected constraints.

Phase 1: Selecting and pre-processing constraints. After selecting an initial set of pairwise constraints by querying an oracle, we can view the set of pairwise constraints as a new graph, where an edge exists between two nodes from the original network if they share a pairwise constraint (either must-link or cannot-link). Then we look for all possible *forbidden triads* among the nodes involved in the must-link set. Given three nodes A, B, C, a *forbidden triad* (sometimes referred to as an *open triad*) occurs when A is connected to B and C, but no edge exists between B and C. In our pre-processing step, we look for such cases — i.e. where we do not know whether a must-link or cannot-link exists between a pair of nodes B and C. To control the size of the constraints set, we greedily expand it until we reach a pre-defined maximum size. The complete constraint selection strategy can be summarized as follows (see also Fig. 2):

1. Select a small random set of both must-link and cannot-link constraints.
2. Find all possible forbidden triads in the must-link set, to identify pairs to query the oracle about their relationship.
3. For each resulting pair, if their relationship is must-link, then add the pair to the must-link set. Otherwise, add the pair to the cannot-link set.
4. Repeat all steps until the maximum number of selected constraints is reached.

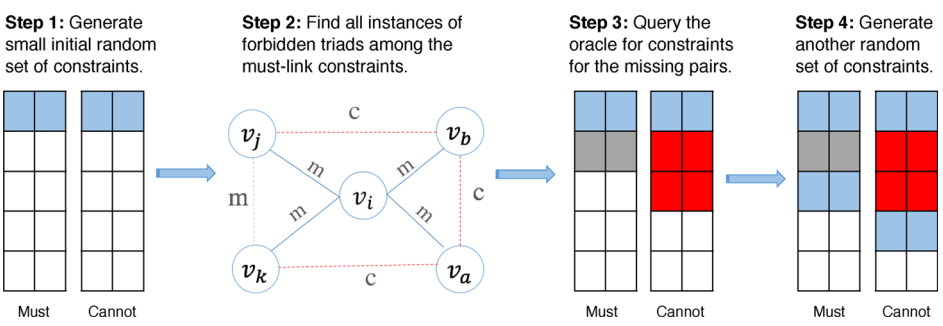

Fig. 2 An illustration of all steps in the overlapping constraint selection process

At the end of this process, the pairwise constraints are ready to be supplied to the community detection algorithm, which we describe next.

Phase 2: Pairwise Constrained SLPA. Given a threshold $r \in [0, 1]$ and a predefined maximum number of iterations T, we incorporate the selected pairwise constraints as follows (see also Fig. 3 for an illustration):

1. In the *initialization step*:
   a. Assign a unique label to each node in the network.
   b. For each pair of nodes that share a must-link relationship, the two nodes exchange labels (i.e. update each node's memory with the other node's label).
2. The *evaluation step* broadly follows a similar process as unsupervised SLPA (see "Community finding" section). However, we account for the pairwise constraints as follows:

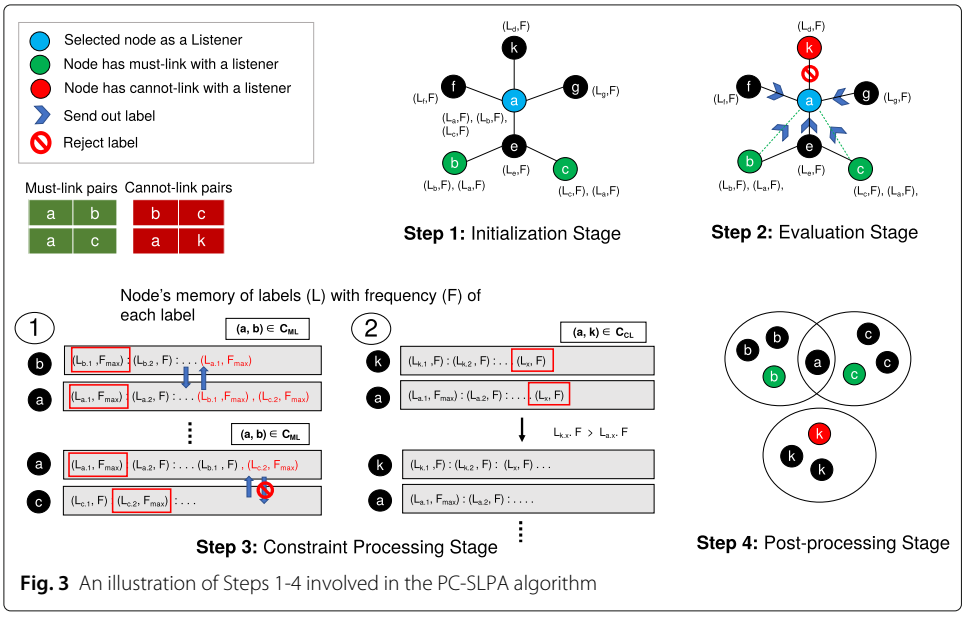

Fig. 3 An illustration of Steps 1-4 involved in the PC-SLPA algorithm

   a. Randomly select one node as a listener, and identify the set of speakers (i.e. the neighbors of the listener).
   b. Augment the set of speakers by adding all nodes that have must-link relationship with the listener and removing all nodes that have cannot link relationship with the listener. Then each speaker sends out a label according to the rule defined in standard SLPA. Removing cannot link nodes from speakers set avoids grouping together pairs of nodes having a cannot-link relationship since all nodes that hold the same label will be grouped together as a community at the end of the process.
   c. Each listener accepts a label according to the rule defined in standard SLPA.

3. The *constraint processing step* considers both sets of pairwise constraints:

   a. For each must-link pair, compare the memories of the two nodes to ensure they both share the same highest occurrence frequency label. If they do not, both nodes exchange their most frequently-occurring labels with each other under the condition that each node does not have a cannot link relationship with any nodes assigned to that label. If this condition is not met, then the exchange does not occur for that label.
   b. For each cannot-link pair, compare the memories of both nodes. If both nodes have a common label, remove this label from the node that has the lowest label occurrence frequency.

4. In the *post-processing step*, convert each node's memory into a probability distribution of labels. If the node's probability of a certain label is less that a threshold $r \in [0, 1]$, the label is removed from the node's memory. Then all nodes having the same label are grouped into one community. Nodes that have more than one label, correspond to overlapping cases, which belong to multiple communities.

Active semi-supervised overlapping community finding

This section describes a novel active learning-inspired approach for a semi-supervised community detection designed to improve the algorithm's performance, while simultaneously reducing the annotation effort required on the side of the oracle (i.e., reducing the annotation budget). The main idea is to develop an iterative uncertainty-based method for selecting informative pairwise constraints for PC-SLPA, as presented in the last section.

Node pair selection

The labeling of pairs of nodes as constraints can potentially be a time-consuming process, especially for large complex networks. Therefore, to reduce this effort, we design our method to select nodes to be labeled from an *important node pair set*, rather than sampling from the entire network. We propose a method for selecting pairs of important nodes which the algorithm is most *uncertain* (i.e. we are unsure about the node's membership of a given community). We define *uncertain nodes* as: 1) nodes that are located at the boundaries between two or more communities; 2) overlapping nodes that have been previously assigned to more than one community.

For each uncertain node i, we retrieve the set of communities to which each of its neighbors has been assigned. Then for each retrieved community, we extract the highest internal degree node j, and add the pair of nodes (i, j) to the important node pairs set. We select high degree nodes j from a community on the basis that such nodes are likely to be highly representative of that community. The final step, we sort the set in ascending order of degree of the *uncertain nodes*. Usually, high degree nodes in the boundary have more dense connections within their own communities, rather than with other communities. In contrast, lower degree nodes in the boundary tend to have an almost equal level of connectivity between communities, and thus are more likely to be misclassified during community detection. Therefore lower degree nodes have a higher priority to be selected for querying. By presenting the pair (i, j) to the oracle, we should be able to accurately determine whether i belongs to the same community as j (i.e, either a must-link or a cannot-link constraint). This approach addresses the misclassification of boundary nodes as overlapping nodes and vice versa, especially for relatively low degree nodes. Thus, the selection process can contribute significantly to generating informative pairwise constraints. This process is summarized in Algorithm 1.

Algorithm 1: Node Pair Selection Method

```
input  : A network G; a set of communities C;
output: A list of pairs to be labeled ImportantPairs
N_Overlap= extract all overlapping nodes from C;
N_Boundary= extract all boundary nodes from C;
N_uncertain=N_Overlap+N_Boundary;
ImportantPairs = {};
for i ∈ N_uncertain do
    N(i)=get all the neighboring nodes of the node i;
    for n ∈ N(i) do
        Comm(n) = all communities containing node n;
        for each community S ∈ Comm(n) do
            Extract node j with the highest internal degree in community S;
            Add (i, j) to ImportantPairs list with the degree value of node i ;
        end
    end
end
Rank ImportantPairs by their associated degree values.
```

Active semi-Supervised sLPA (AC-SLPA)

Now we discuss the integration of the proposed constraint selection method described above with the PC-SLPA algorithm to produce a novel active semi-supervised SLPA algorithm (AC-SLPA). This approach is divided into three phases, which are executed iteratively. Note that we run the unsupervised SLPA algorithm as an initialization step to generate a starting set of communities, which then allows us to select the constraints. The phases of the proposed algorithm are as follows: In the first phase, we run the method from "Node pair selection" section on the current communities to select a set of informative constraint pairs (i.e. the important node pairs set). In the second phase, this set is passed to the oracle for annotation. In the third phase, we run PC-SLPA with the resulting pairwise constraints to find communities, which will be used as input for the next run of the first phase. These three phases are repeated until either we have exhausted a

predefined maximum annotation budget, or the annotator is satisfied with the current set of communities. The complete process is summarized in Algorithm 2.

Algorithm 2: Active Semi-Supervised SLPA (AC-SLPA)

input : A network G
output: Set of communities
Initialization: Apply unsupervised SLPA to generate set of initial communities
while *budget available* **do**
  Phase 1: Apply Node Pair Selection method
  Input(Set of communities)
    Model selects the best to query
  Output(List of important pairs to be labeled)
  Phase 2: Generate the pairwise constraints
  Input(List of important pairs to be labeled)
    Generate pairwise constraints from oracle
  Output(pairwise constraints)
  Phase 3: Apply PC-SLPA algorithm
  Input(pairwise constraints)
    Running PC-SLPA algorithm
  Output(Set of communities)
end

Time complexity

In this subsection, we detail the time complexity of each phase of the algorithms described in this paper. The notation used to define running time is as follows:

n	number of nodes in the network
K	average degree of the nodes in the network
T	maximum number of iterations specified by the user
PC	total number of pairwise constraints
M	number of must-link pairs
C	number of cannot-link pairs

PC-SLPA: The first phase, which involves selecting and pre-processing constraints, requires time $O(PC)$. For the second phase, the initialization stage contains two iterative steps. The first loop initializes nodes, which takes $O(n)$, and the second loop iterate through M. Since on average $M < n$, the entire stage requires $O(n)$. The complexity of both the evaluation and post-processing stages is $O(Tn)$, as per the original SLPA algorithm (Xie et al. 2011). As for the final constraint processing stage, it involves loops for processing the two constraint sets, requiring time $O(PC)$. The overall time complexity is $O(Tn + PC)$.

AC-SLPA: The first phase of the algorithm consists of three nested loops. The first loop processes the uncertain nodes (i.e. the overlapping and boundary nodes), the second loop processes the neighbor set of each uncertain node, and the third loop processes the communities to which each uncertain node belongs. In the worst case, all nodes of the network are overlapping nodes, therefore the complexity of the first loop is $O(n)$. The second loop requires $O(K)$ on average. Given that the minimum size of a community here is two nodes, in the worst case an overlapping node belongs to all communities, thus the third loop takes $O(n/2)$, which reduces to $O(n)$. Ultimately, the time complexity of this first phase is $O(n^2)$. As described above, the subsequent execution of PC-SLPA requires $O(Tn + PC)$.

Evaluation

In this section, the performance of both proposed methods (PC-SLPA and AC-SLPA) is evaluated via experiments performed on two groups of synthetic benchmark networks and real-world networks, both containing overlapping communities. As we observed in "Related work" section, no work has been conducted in the literature regarding pairwise constrained algorithms for finding overlapping communities. Therefore, for the sake of comparison, the results of the proposed algorithms (PC-SLPA and AC-SLPA) are compared with the corresponding unsupervised version and a group of popular unsupervised community finding algorithms. Our objectives here are: (1) determine the extent to which introducing varying levels of constraints can improve the performance of overlapping community finding through PC-SLPA; (2) evaluate the effectiveness of the selected constraints by our proposed selection component (Node Pair Selection) integrated in AC-SLPA compared to the constraints selected at random in PC-SLPA.

Experimental setup

Data. Firstly, we evaluate on synthetic data created using the widely-used LFR generator (Lancichinetti et al. 2008), which can produce networks with properties similar to real-world networks, with overlapping ground truth communities. The selection of network parameters shown in Table 1 is based on those used to evaluate the original algorithm SLPA (Xie et al. 2011) and other works in the literature. We generate two different groups of synthetic networks with different sizes and degree of overlapping communities, each containing small and large communities and mixing parameter μ varies from 0.1 to 0.3. Small communities have 10–50 nodes, while large communities have 20–100 nodes. Each group consists of 64 networks with different combinations of the parameter O_m, which controls the number of communities per node, and O_n, which controls the number of overlapping nodes. For the first network in each set, all nodes belong to two communities ($O_m = 2$). For each successive network, this parameter value is incremented by 1 until $O_m = 8$ is reached. As for the parameter O_n is set to 10% and 50% of the total number of nodes, indicating low and high overlapping degree respectively.

Secondly, we consider three real-world networks which have previously been used in the community finding literature (Leskovec and Krevl 2015): 1) a co-purchasing network from Amazon.com; 2) a friendship network from YouTube; 3) a scientific collaboration network from DBLP. These networks contain annotated ground truth overlapping communities. For each network, we include only the 5,000 largest such communities, as per (Yang and Leskovec 2015). We then perform filtering as per (Harenberg et al. 2014) – the remaining communities are ranked based on their internal densities and the bottom quartile is discarded, along with any duplicate communities.

Table 1 Lists parameters used for the generation of the LFR synthetic networks

Parameter	*Description*	*Value*	Parameter	*Description*	*Value*
N	Number of nodes	1000-5000	t_1	Degree exponent	2
k	Average degree	10	t_2	Community exponent	1
K_{max}	Max degree	50	μ	Mixing parameter	0.1-0.3
C_{min}	Min community size	10/20	O_n	Num of overlapping nodes	10%/50%
C_{max}	Max community size	50/100	O_m	Communities per node	1-8

Finally, as an additional step, we eliminated extremely small communities. For the Amazon and YouTube networks, communities of size < 5 nodes are removed, while for the DBLP network communities with < 10 nodes are removed. Details of the resulting networks are listed in Table 2.

Evaluation metrics and baselines. The results of PC-SLPA and AC-SLPA are compared with outputs of the following popular unsupervised overlapping community detection algorithms: SLPA(Xie et al. 2011), OSLOM (Lancichinetti et al. 2011), BIGCLAM (Yang and Leskovec 2013), MOSES (McDaid and Hurley 2010), and COPRA (Adamcsek et al. 2006). For OSLOM and MOSES, we use the default parameters recommended by the original authors. For COPRA, we use the settings recommended in Xie et al. (2011). For BIGCLAM, the number of communities to detect is selected automatically. However, it is necessary to provide a range of values from which to select. Thus, we provide the algorithm with broad ranges based on the number of communities in the ground truth for each data set. To evaluate the performance of these algorithms relative to the ground truth, we use the overlapping form of Normalized Mutual Information (NMI), proposed in Lancichinetti et al. (Lancichinetti et al. 2009). For this measure, a value close to 1 indicates a high level of agreement with the ground truth communities, while a value close to 0 indicates that the communities generated by an algorithm are no better than random.

Part of our objective in this work is to develop techniques that can accurately identify overlapping communities. Therefore, to quantify an algorithm's ability to correctly identify overlapping nodes, we used the overlapping F-measure (FS) (Xie et al. 2011; Xie et al. 2013). This is calculated as the harmonic mean of precision and recall, in the same way as the corresponding measure commonly used in information retrieval. In this context, *precision* is defined as the number of "true" overlapping nodes detected, divided by the total number of overlapping nodes identified by the algorithm. *Recall* is defined here as the number of overlapping nodes detected correctly, divided by the total number of "true" overlapping nodes. A higher value for the F-measure here indicates more accurate detection of nodes which belong to multiple communities. Since SLPA and COPRA are non-deterministic, we average the NMI and F-measure scores over 10 runs.

Experimental methodology. We conducted three experiments in our evaluation. The first experiment aims to assess the performance of the unsupervised algorithms, which provides a baseline for evaluating the performance of our proposed methods. For SLPA, PC-SLPA, and AC-SLPA, we use the default parameters values $T = 100$ and $r \in [0, 1]$, as suggested in Lee et al. (2010). The second experiment evaluates the performance of PC-SLPA with increasing numbers of constraints, from 1% to 5% of the total number of possible pairs in each network. Since the initial pairwise constraints are selected at random, we repeat the semi-supervised

Table 2 Summarizes details of the real-world networks

Real-world Networks	*Amazon*	*YouTube*	*DBLP*
#Nodes - # Edges - #Communities	7411 - 21214 - 876	6426 - 23226 - 1058	7233 - 33045 - 613
Max community size	27	31	38
Min community size	5	5	10
Max communities per node	4	11	8
#Overlapping nodes	1394(18%)	865(13%)	214 (3.3%)

process for 10 runs and average the resulting scores. The third experiment evaluates the performance of AC-SLPA with limited numbers of constraints not exceeding 1% of the total number of possible pairs in each network. Finally, we compare the results obtained from AC-SLPA with PC-SLPA, then compare them with the baseline algorithms. The full set of experiments and associated parameters are summarized in Table 3.

Results and discussion

Firstly, we compare the accuracy of the proposed algorithms, PC-SLPA and AC-SLPA, to the unsupervised version of these algorithms, SLPA. Secondly, we compare the accuracy of AC-SLPA using limited number of pairwise constraints selected by the proposed active learning-inspired approach Node Pair Selection, to PC-SLPA, where the pairwise constraints are selected at random. Finally, we compare the accuracy of all algorithms according to their average ranks.

Synthetic networks. When evaluating on LFR-generated networks, different factors can affect algorithm's performance, such as the mixing parameters, and the size of both networks and embedded communities. The larger the value of μ, the poorer the communities were detected by the algorithms, due to the weaker intra-community connectivity. As we see from Figs. 4 and 5, the performance of SLPA drops as μ increases from 0.1 to 0.3. However, AC-SLPA and PC-SLPA show more stability with higher values of μ. For instance, in the case of small network of big communities with $\mu = 0.1$, the NMI score of SLPA is 0.94 at $O_m = 2$ and drops to 0.82 with $\mu = 0.3$. In contrast, AC-SLPA and PC-SLPA show more moderate decrease in accuracy as the value of μ increases to 0.3 with the same network, (PC-SLPA: NMI=0.88 at $\mu = 1$ to NMI=0.80 at $\mu = 3$, AC-SLPA: NMI=0.998 at $\mu = 1$ to NMI=0.99 at $\mu = 3$).

Another factor that affects the algorithmic performance is the degree of overlapping communities in terms of O_m and O_n. As the values of O_m increases from 2 to 8, AC-SLPA and PC-SLPA demonstrate better NMI values than SLPA across all the networks. As for O_n, we can see that as the fraction of overlapping nodes increases from 10% to 50%, AC-SLPA and PC-SLPA show a medium decrease compared to SLPA specifically with large networks. For instance, in large and small community networks at $\mu = 0.1$, the NMI value of SLPA falls from 0.91 to 0.52 with $O_m = 2$. On the other hand, the NMI values of AC-SLPA and PC-SLPA decreases from 0.94 to 0.63 and from 0.98 to 0.90 respectively, where AC-SLPA shows better stability than PC-SLPA. As for the size of the networks, both the

Table 3 Parameters used for all three experiments (*E1*, *E2*, *E3*)

	E1	*E2*	*E3*
Algorithms	Baseline algorithms: SLPA, OSLOM, MOSES, COPRA, and BIGCLAM	PC-SLPA	AC-SLPA
PW selection	None	Random	Node Pair Selection
% of constraints	0%	1% — 5%	0.5% — 1%
Evaluation metrics	NMI and F-score		
Networks	Synthetic and real-world networks		
Type of experiment	Deterministic, except SLPA and COPRA	Non-deterministic	Non-deterministic

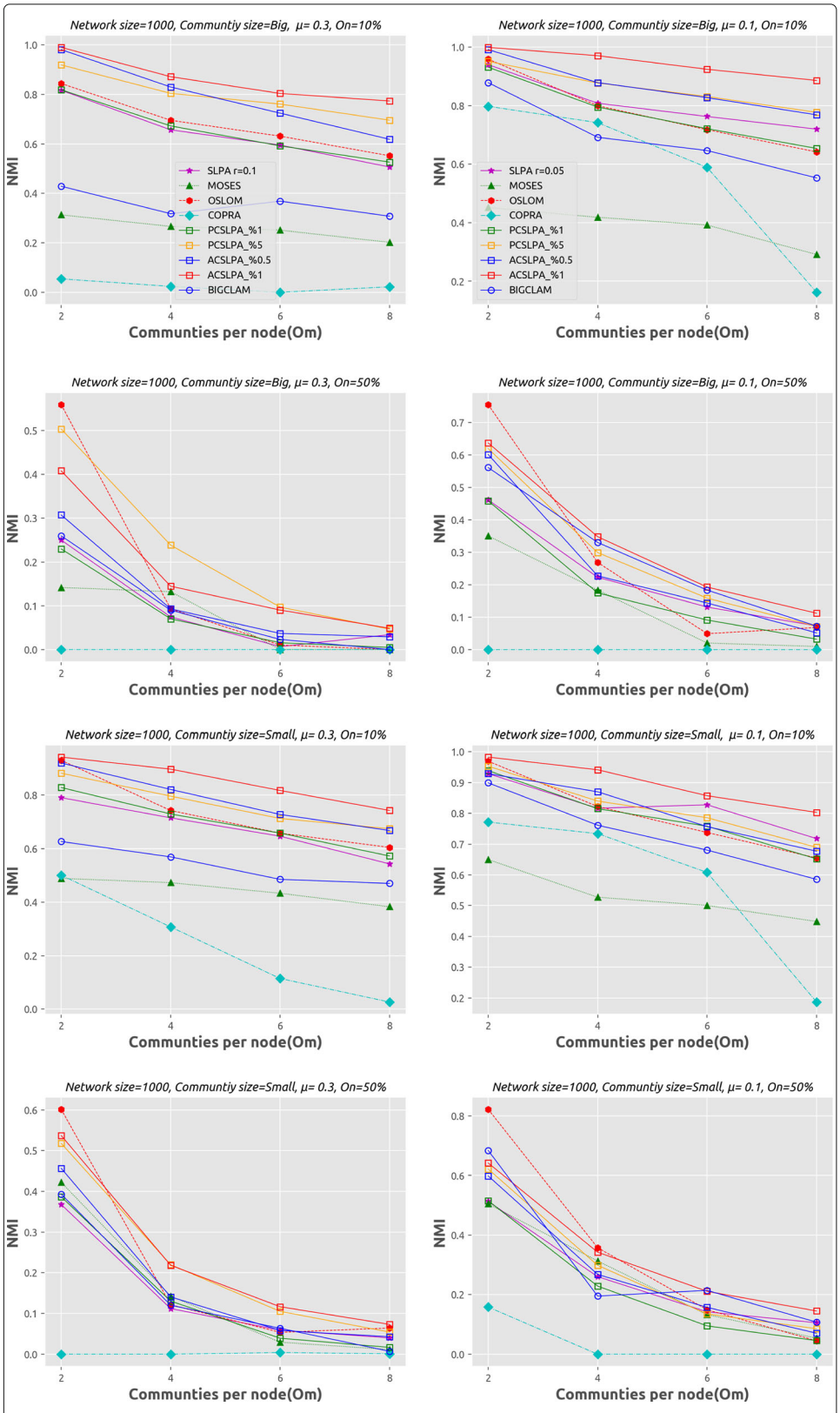

Fig. 4 Summary of the performance of all algorithms on small synthetic networks, containing both small and large communities, where the mixing parameter μ varies from 0.1 to 0.3. NMI values are plotted against the number of communities per node (O_m), with 4 networks in each plot

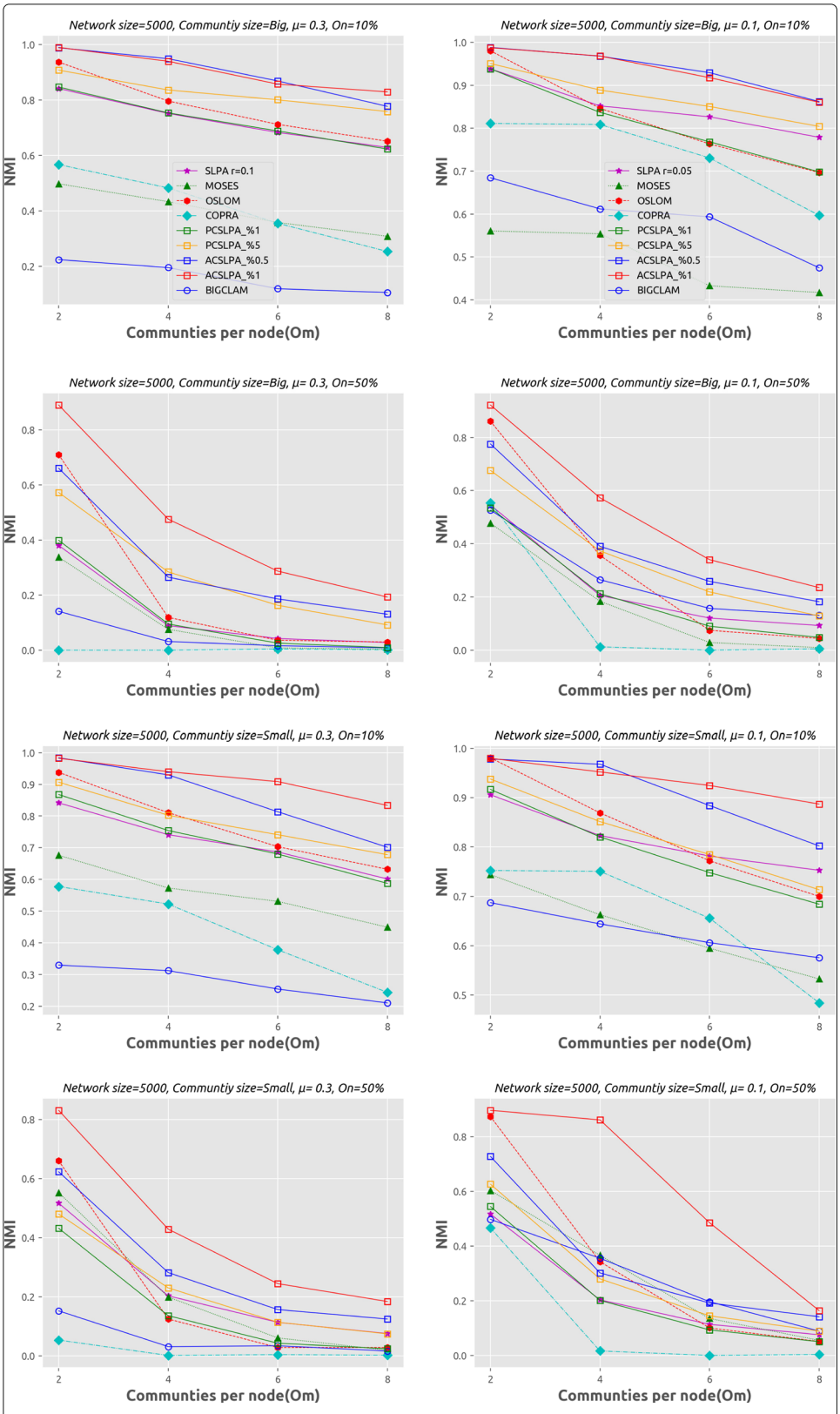

Fig. 5 Summary of the performance of all algorithms on large synthetic networks, containing both small and large communities, where the mixing parameter μ varies from 0.1 to 0.3. NMI values are plotted against the number of communities per node (O_m), with 4 networks in each plot

standard SLPA and the proposed algorithms show a better performance when the network increases from 1,000 to 5,000 nodes, with PC-SLPA achieving the best performance on the networks with larger communities, while AC-SLPA shows consistent performance with both large and small communities.

Overall, PC-SLPA achieves consistently higher NMI scores than the standard SLPA algorithm, except with PC-SLPA with 1%. Here, PC-SLPA attains lower NMI values than SLPA, until the number of constraints increases towards 5%. This is due to the random selection of a small set consisting of largely non-informative constraints, which do not aid in the detection of the ground truth communities. In such a situation, PC-SLPA may not achieve an improvement compared to the standard SLPA algorithm. Nevertheless, as PC-SLPA shows a consistent pattern of improvement correlated with the number of constrains this shows the potential of using semi-supervised strategies in particular pairwise constraints in the context of overlapping communities which was the main motivation for this study to propose AC-SLPA that implement active learning-inspired approach for selecting informative constraints with small annotation budget. As a result, AC-SLPA significantly outperforms both PC-SLPA algorithm and the standard SLPA algorithm on most of the networks, particularly when using a limited annotation budget.

When comparing the proposed algorithms to the baseline algorithms, we observe that BiGCLAM, COPRA and MOSES show lower performance than PC-SLPA and AC-SLPA on all synthetic networks, except for BIGCLAM which shows almost equivalent performance in limited cases as illustrated in Figs. 4 and 5. When we look at the level of *coverage* provided by the algorithms (i.e. the percentage of nodes in the network assigned to at least one community), we notice that many nodes are not assigned to any communities by MOSES, which may partly explain the low NMI values. For instance, for networks at $Om = 6$ and $Om = 8$ with $On = 50\%$, the percentage of coverage is 46% and 46.6% respectively. As for OSLOM, it shows a slightly better performance than PC-SLPA and AC-SLPA for networks with a low level of community overlap. However, as the number of communities per node increases, both PC-SLPA and AC-SLPA start to outperform all of the baseline algorithms, indicating that it is effective in highly-overlapping contexts.

In terms of identifying overlapping nodes, Fig. 6 shows the algorithms' accuracy in identifying the true overlapping nodes using the overlapping F-measure. We can observe that, in most cases, the AC-SLPA and PC-SLPA outperform the standard SLPA. On the other hand, it is worth noticing that in most networks as O_m increases, the F-measure values of PC-SLPA increase, the same pattern of SLPA, while the F-measure values of AC-SLPA decrease, however still AC-SLPA provides higher values than PC-SLPA in most cases.

Table 4, summarizes the average ranks of NMI and FS scores of all algorithms on synthetic networks with 10% and 50% of overlapping nodes respectively. Each table entry shows the average ranks (lower values are better) of an algorithm (on the columns) over each overlapping category and evaluation score (on the rows). The best ranks are shown in boldface. As we can see, AC-SLPA with 1% pairwise constraints is the top-ranked algorithm in all overlapping categories and evaluations scores. The next best alternatives are AC-SLPA and PC-SLPA with 0.5% and 5% pairwise constraints respectively, both algorithms show similar ranks in both evaluations scores.

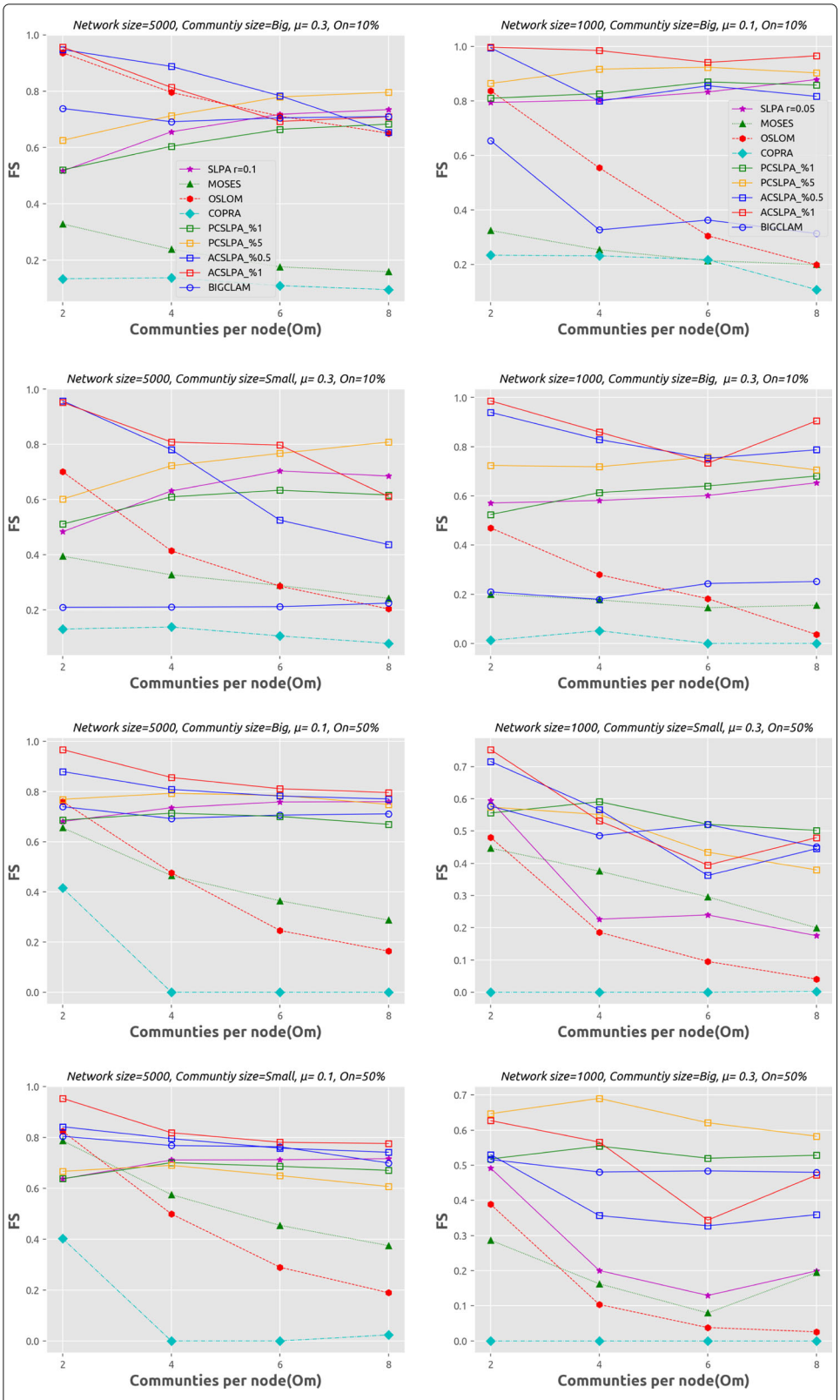

Fig. 6 Summary of the performance of all algorithms on synthetic networks, containing both small and large communities, where the mixing parameter μ varies from 0.1 to 0.3. F-measure values are plotted against the number of communities per node (O_m), with 4 networks in each plot

Table 4 Average ranks over all the networks in their corresponding overlapping categories and evaluation scores

Evaluation metrics	% Overlapping nodes	ACSLPA%1	ACSLPA%0.5	PCSLPA%5	PCSLPA%1	OSLOM	MOSES	COPRA	BIGCLAM	SLPA
NMI	10%	**1.25**	2.25	3.12	5.28	4.09	8.19	8.00	7.81	5.00
	50%	**1.34**	3.56	3.28	6.47	4.45	6.44	8.78	5.30	5.38
FS	10%	**2.12**	3.06	2.31	4.53	6.22	7.50	8.97	6.34	3.94
	50%	**1.56**	3.06	3.59	4.47	7.19	6.91	9.00	4.34	4.88

Lower ranks are better, Best ranks shown in boldface

To understand the effectiveness of the proposed method over the different synthetic networks with various degrees of node overlap, statistical significance tests were performed. Following the recommendations of (García et al. 2010) and the statistical experimental methodology in Pakrashi and Mac Namee (2019), a post-hoc Friedman Aligned Rank test with the Finner p-value adjustment was performed on the network results for each node overlap level and evaluation metric. The rank plots (with a significance level of $\alpha = 0.05$) of the tests are provided in Fig. 7. Each diagram shows the average ranks of the algorithms in the corresponding node overlap level. The algorithms not connected with the horizontal bars in each plot in Fig. 7 indicate that they are significantly different based on a significance level of $\alpha = 0.05$, whereas for the algorithms connected with the horizontal bars, the null hypothesis could not be rejected. The detailed p-values and the win/lose/tie counts for each algorithm pairs are provided in Appendix A.

The statistical results indicate that AC-SLPA with 1% pairwise constraints was significantly better than all the baseline algorithms on both evaluations metrics. AC-SLPA and PC-SLPA with 0.5% and 5% pairwise constraints respectively were not found to be statistically different. Although both algorithms attained better average ranks than OSLOM for NMI and better average ranks than SLPA and BIGCLAM for the F-measure, the tests did not identify a statistically significant difference.

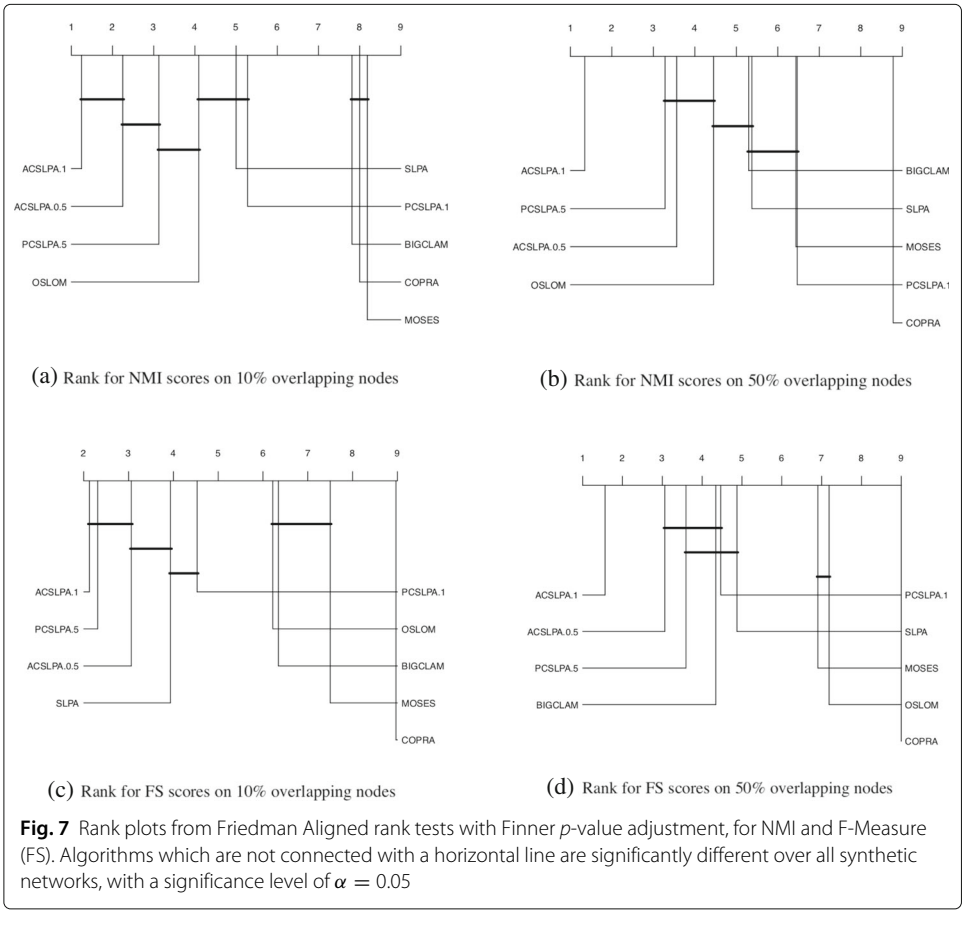

Fig. 7 Rank plots from Friedman Aligned rank tests with Finner p-value adjustment, for NMI and F-Measure (FS). Algorithms which are not connected with a horizontal line are significantly different over all synthetic networks, with a significance level of $\alpha = 0.05$

Real-world networks. Next we discuss our experiments on three real-world networks. We compare the NMI performance of our proposed semi-supervised method with increasing numbers of pairwise constraints, relative to the benchmark algorithms. For the non-deterministic algorithms, 10 runs were executed and NMI scores were averaged. Tables 5 and 6 summarize the corresponding performance of all algorithms. The results show that AC-SLPA significantly outperforms PC-SLPA algorithm, the standard SLPA algorithm, and the benchmark algorithms on the three real networks, using only small percentages of constraints. Table 5 lists the NMI and FS scores for PC-SLPA and AC-SLPA, and also reports the actual effective annotation budget of pairwise constraints used for each algorithm, beyond which no further constraints were added. In all cases we observe that the level of constraints required by AC-SLPA is far less than the predefined maximum budget ($1\% - 5\%$), while still yielding high accuracy.

When comparing the performance of the proposed methods to the benchmark algorithms, we see that PC-SLPA achieves high NMI scores (> 0.9) on the Amazon and DBLP networks. However, PC-SLPA shows moderate performance on the YouTube network, which may be due to the poor separation between the ground truth clusters in this network. In fact, the addition of $< 4\%$ of constraints does not yield an improvement over the unsupervised approach. The effect of high inter-community overlap is far more pronounced in the cases of the OSLOM, MOSES, and COPRA algorithms. Overall, PC-SLPA outperforms the four alternative algorithms in most cases on these networks, with a consistent increases as the number of constraints is increased from 1% to 5%. We would expect this trend to continue as more constraints are added, although it may be impractical to generate larger numbers of constraints in real-world scenarios. On the other hand, AC-SLPA achieved significantly higher performance relative to the benchmark algorithms, particularly on the YouTube network, with a limited annotation budget not exceeding 1% of all possible pairwise constraints.

In general, our results suggest that PC-SLPA is an appropriate choice for community finding in undirected, unweighted networks, in applications where external knowledge or human annotation is available. For purely unsupervised cases, our results indicate that OSLOM remains a good alternative.

Conclusion

In this study, we have explored the potential of semi-supervised strategies to improve existing unsupervised algorithms for finding overlapping communities in complex networks. Our primary contributions are three-fold: 1) we explored the nuances around the selection of constraints, which are specific to contexts where the communities

Table 5 NMI and F-Measure (FS) scores of benchmark algorithms on three real-world networks

	NMI					FS				
	OSLOM	MOSES	COPRA	BIGCLAM	SLPA	OSLOM	MOSES	COPRA	BIGCLAM	SLPA
Amazon	0.9668	0.9084	0.96228	0.0227	0.9565	0.0029	0.1143	0.0210	0.3228	0.01553
YouTube	0.4490	0.4209	0.1907	0.3385	0.6257	0.1192	0.3556	0.0512	0.3614	0.1036
DBLP	0.8485	0.7707	0.9136	0.8729	0.8982	0.2534	0.2462	0.1203	0.1647	0.0496

Table 6 Comparison of AC-SLPA and PC-SLPA on three real-world networks

		Amazon				YouTube				DBLP			
		PC-SLPA		AC-SLPA		PC-SLPA		AC-SLPA		PC-SLPA		AC-SLPA	
% annotation budget	Predefined max %	1%	5%	0.5%	1%	1%	5%	0.5%	1%	1%	5%	0.5%	1%
	Actual used %	1%	5%	0.013%	0.014%	1%	5%	0.5%	0.78%	1%	5%	0.024%	0.025%
NMI score		0.961	0.972	0.988	0.989	0.600	0.643	0.883	0.934	0.904	0.933	0.984	0.985
FS score		0.0404	0.0643	0.0095	0.0100	0.1335	0.1941	0.4504	0.6755	0.0888	0.1923	0.5050	0.5199

in the data naturally overlap; 2) we proposed a new semi-supervised algorithm (PC-SLPA) for detecting overlapping communities, based on the use of a label propagation strategy that is informed by the addition of external information encoded as pairwise constraints selected at random; 3) we introduced a new semi-supervised approach, inspired by active learning, which uses an active pairwise constraint selection component to limit the level of annotation required for effective community finding (AC-SLPA). Based on extensive experiments, the results of PC-SLPA show that overlapping community finding algorithms with constraints can considerably outperform their unconstrained counterparts on both synthetic and real-world networks. As one might expect, their performance improves with increasing the number of pairwise constraints.

In general, the results show the potential of using semi-supervised strategies for finding overlapping communities. Furthermore, the results for AC-SLPA demonstrate that the active pairwise constraint selection component can significantly improve the effectiveness of community detection, while using a small annotation budget. This was demonstrated on both synthetic and real-world networks. In terms of practical applications, one consideration is around the time required for an oracle to generate constraint pairs. If constraints are being generated from a source of background knowledge, such as an existing ontology, there may be little overhead. However, if constraints are being provided via human annotation, the effort and cognitive load will vary depending on the domain and the complexity of the task from a human perspective. In some cases, this annotation effort can be crowdsourced (Tang and Lease 2011). In either case, applying an active approach to constraint selection still allows us to minimize the number of constraints required, thereby reducing the overall human effort.

One other practical consideration is the extent to which we can rely on the judgment of the oracle to provide reliable constraint information. While this issue has been considered to some extent in the context of semi-supervised clustering (Basu et al. 2008), relatively little work has been performed in this area in the context of community detection. In future work we aim to explore the impact of noisy, potentially-incorrect constraints upon the performance of semi-supervised community finding algorithms, and to investigate how we might mitigate against such cases in real-world network analysis tasks.

Appendix A. Complete results of statistical tests

Table 7 Result of post-hoc Friedman Aligned Rank test with Finner *p*-value adjustment for the multiple algorithm performance for different networks with different percentage of overlapping nodes and different scores

(a) NMI scores on 10% overlapping nodes

	OSLOM	MOSES	COPRA	BIGCLAM	SLPA	PCSLPA%1	PCSLPA%5	ACSLPA%0.5	ACSLPA%1
OSLOM		32/0/0	32/0/0	32/0/0	22/10/0	26/6/0	9/23/0	3/29/0	1/31/0
MOSES	0.0000 ***		16/16/0	10/22/0	0/32/0	0/32/0	0/32/0	0/32/0	0/32/0
COPRA	0.0000 ***	0.7934		16/16/0	0/32/0	0/32/0	0/32/0	0/32/0	0/32/0
BIGCLAM	0.0000 ***	0.6158	0.7934		0/32/0	0/32/0	0/32/0	0/32/0	0/32/0
SLPA	0.2121	0.0000 ***	0.0000 ***	0.0001 ***		17/15/0	3/29/0	2/30/0	0/32/0
PCSLPA%1	0.1052	0.0000 ***	0.0001 ***	0.0003 ***	0.7020		0/32/0	2/30/0	0/32/0
PCSLPA%5	0.1854	0.0000 ***	0.0000 ***	0.0000 ***	0.0089 ***	0.0025 ***		8/24/0	0/32/0
ACSLPA%0.5	0.0094 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0001 ***	0.0000 ***	0.2234		7/25/0
ACSLPA%1	0.0001 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0089 ***	0.1757	

(b) NMI scores on 50% overlapping nodes

	OSLOM	MOSES	COPRA	BIGCLAM	SLPA	PCSLPA%1	PCSLPA%0.5	ACSLPA%0.5	ACSLPA%1
OSLOM		23/9/0	31/1/0	18/13/1	19/13/0	23/9/0	12/20/0	14/18/0	5/27/0
MOSES	0.0061 ***		30/2/0	14/18/0	8/24/0	14/18/0	3/29/0	4/28/0	0/32/0
COPRA	0.0000 ***	0.0014 ***		2/30/0	1/31/0	1/31/0	0/32/0	0/32/0	0/32/0
BIGCLAM	0.2351	0.1213	0.0000 ***		18/14/0	18/14/0	9/23/0	10/22/0	2/30/0
SLPA	0.2037	0.1430	0.0000 ***	0.9152		24/8/0	5/27/0	5/27/0	0/32/0
PCSLPA%1	0.0058 ***	0.9636	0.0015 ***	0.1184	0.1349		0/32/0	1/31/0	0/32/0
PCSLPA%5	0.1184	0.0000 ***	0.0000 ***	0.0058 ***	0.0042 ***	0.0000 ***		16/16/0	4/28/0
ACSLPA%0.5	0.2147	0.0001 ***	0.0000 ***	0.0162 **	0.0121 **	0.0001 ***	0.7020		0/32/0
ACSLPA%1	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0073 ***	0.0024 ***	

(c) F-Scores on 10% overlapping nodes

	OSLOM	MOSES	COPRA	BIGCLAM	SLPA	PCSLPA%1	PCSLPA%0.5	ACSLPA%0.5	ACSLPA%1
OSLOM		23/9/0	32/0/0	15/17/0	7/25/0	7/25/0	5/27/0	0/32/0	0/32/0

Table 7 Result of post-hoc Friedman Aligned Rank test with Finner *p*-value adjustment for the multiple algorithm performance for different networks with different percentage of overlapping nodes and different scores (*Continued*)

MOSES	0.0755 *		31/1/0	8/24/0	0/32/0	0/32/0	0/32/0	0/32/0	0/32/0
COPRA	0.0001 ***	0.0423 **		0/32/0	0/32/0	0/32/0	0/32/0	0/32/0	0/32/0
BIGCLAM	0.8551	0.1085	0.0003 ***		2/30/0	4/28/0	1/31/0	3/29/0	2/30/0
SLPA	0.0015 ***	0.0000 ***	0.0000 ***	0.0008 ***		23/9/0	1/31/0	11/21/0	8/24/0
PCSLPA%1	0.0197 **	0.0000 ***	0.0000 ***	0.0127 **	0.4032		0/32/0	12/20/0	5/27/0
PCSLPA%5	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0243 **	0.0020 ***		17/15/0	12/20/0
ACSLPA%0.5	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.2234	0.0423 **	0.2941		9/23/0
ACSLPA%1	0.0000 ***	0.0000 ***	0.0000 ***	0.0000 ***	0.0127 **	0.0008 ***	0.7934	0.0196	
				(d) F-Scores on 50% overlapping nodes					
	OSLOM	MOSES	COPRA	BIGCLAM	SLPA	PCSLPA%1	PCSLPA%5	ACSLPA%0.5	ACSLPA%1
OSLOM		9/23/0	32/0/0	4/28/0	5/27/0	5/27/0	3/29/0	0/32/0	0/32/0
MOSES	0.6915		32/0/0	0/32/0	7/25/0	3/29/0	2/30/0	0/32/0	0/32/0
COPRA	0.0121 **	0.0038 ***		0/32/0	0/32/0	0/32/0	0/32/0	0/32/0	0/32/0
BIGCLAM	0.0001 ***	0.0004 ***	0.0000 ***		16/16/0	17/15/0	13/19/0	8/24/0	3/29/0
SLPA	0.0013 ***	0.0049 ***	0.0000 ***	0.4768		18/14/0	9/23/0	5/27/0	0/32/0
PCSLPA%1	2.0000 ***	0.0007 ***	0.0000 ***	0.8551	0.5736		11/21/0	11/21/0	6/26/0
PCSLPA%5	0.0000 ***	0.0000 ***	0.0000 ***	0.3098	0.0781 *	0.2363		11/21/0	8/24/0
ACSLPA%0.5	0.0000 ***	0.0000 ***	0.0000 ***	0.0781 *	0.0121 **	0.0529 *	0.4768		1/31/0
ACSLPA%1	0.0000 ***	0.0000 ***	0.0000 ***	0.0001 ***	0.0000 ***	0.0001 ***	0.0049 ***	0.0392 **	

Upper diagonal: win/lose/tie. Lower diagonal: Friedman test *p*-values. *: $\alpha = 0.1$, **: $\alpha = 0.05$ and ***: $\alpha = 0.01$

Abbreviations
AC-SLPA: Active semi-supervised SLPA; NMI: Normalized mutual information; PC-SLPA: Pairwise Constrained SLPA; PC: Pairwise constrained; SLPA: Speaker-listener label propagation algorithm

Acknowledgements
Not applicable.

Authors' contributions
EA implemented the code, performed and analyzed the experiments. EA and DG wrote, read, and approved the final manuscript.

Funding
This research was supported by Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289, and by The Ministry of Higher Education in Saudi Arabia.

Availability of data and materials
The datasets generated during and analyzed during the current study are available in the GitHub repository: https://github.com/elhamalghamdiUCD/Semi-Supervised-SLPA.

Competing interests
The authors declare that they have no competing interests.

Received: 4 March 2019 Accepted: 22 July 2019
Published online: 23 August 2019

References
Adamcsek B, Palla G, Farkas IJ, Derényi I, Vicsek T (2006) Cfinder: locating cliques and overlapping modules in biological networks. Bioinformatics 22(8):1021–1023
Ahn YY, Bagrow JP, Lehmann S (2010) Link communities reveal multiscale complexity in networks. Nature 466(7307):761–764
Alghamdi E, Greene D (2018) Semi-supervised overlapping community finding based on label propagation with pairwise constraints. In: Proc. 7th International Conference on Complex Networks and Their Applications. Springer, Cham
Amelio A, Pizzuti C (2014) Overlapping community discovery methods: a survey. In: Social Networks: Analysis and Case Studies. Springer, Vienna. pp 105–125
Basu S, Bilenko M, Mooney RJ (2004) A probabilistic framework for semi-supervised clustering. In: Proc. 10th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining. CRC Press, USA. pp 59–68
Basu S, Davidson I, Wagstaff K (2008) Constrained clustering: Advances in algorithms, theory, and applications. ACM, New York
Blondel V, Guillaume J, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech 2008(10):10008
Cheng J, Leng M, Li L, Zhou H, Chen X (2014) Active semi-supervised community detection based on must-link and cannot-link constraints. PloS ONE 9(10):e110088
Ciglan M, Nørvåg K (2010) Fast detection of size-constrained communities in large networks. In: International Conference on Web Information Systems Engineering. Springer, Berlin. pp 91–104
Clauset A, Newman ME, Moore C (2004) Finding community structure in very large networks. Phys Rev E 70(6):066111
Donmez P, Carbonell JG, Bennett PN (2007) Dual strategy active learning. In: European Conference on Machine Learning. Springer, Berlin. pp 116–127
Dreier J, Kuinke P, Przybylski R, Reidl F, Rossmanith P, Sikdar S (2014) Overlapping communities in social networks. arXiv preprint arXiv:1412.4973
Eaton E, Mansbach R (2012) A spin-glass model for semi-supervised community detection. In: Proc. AAAI'12. AAI Press, USA. pp 900–906
Everett MG, Borgatti SP (1998) Analyzing clique overlap. Connections 21(1):49–61
Fortunato S (2010) Community detection in graphs. Phys Rep 486(3-5):75–174
García S, Fernández A, Luengo J, Herrera F (2010) Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Inf Sci 180(10):2044–2064. http://www.sciencedirect.com/science/article/pii/S0020025509005404, special Issue on Intelligent Distributed Information Systems
Girvan M, Newman ME (2002) Community structure in social and biological networks. PNAS 99(12):7821–7826
Grayson S, Wade K, Meaney G, Rothwell J, Mulvany M, Greene D (2016) Discovering structure in social networks of 19th century fiction. In: Proceedings of the 8th ACM Conference on Web Science. ACM, New York. pp 325–326
Greene D, Cunningham P (2007) Constraint selection by committee: An ensemble approach to identifying informative constraints for semi-supervised clustering. In: Proc. 18th European Conference on Machine Learning (ECML'07). Springer, Berlin. pp 140–151
Gregory S (2007) An algorithm to find overlapping community structure in networks. In: European Conference on Principles of Data Mining and Knowledge Discovery. Springer, Berlin. pp 91–102
Gregory, S (2008) A fast algorithm to find overlapping communities in networks. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer, Berlin. pp 408–423
Gregory S (2010) Finding overlapping communities in networks by label propagation. New J Phys 12(10):103018
Habashi S, Ghanem NM, Ismail MA (2016) Enhanced community detection in social networks using active spectral clustering. In: Proc. 31st Annual ACM Symposium on Applied Computing. ACM, New York. pp 1178–1181

Harenberg S, Bello G, Gjeltema L, Ranshous S, Harlalka J, Seay R, Padmanabhan K, Samatova N (2014) Community detection in large-scale networks: a survey and empirical evaluation. Wiley Interdiscip Rev Comput Stat 6(6):426–439
Jonsson PF, Cavanna T, Zicha D, Bates PA (2006) Cluster analysis of networks generated through homology: automatic identification of important protein communities involved in cancer metastasis. BMC Bioinformatics 7(1):2
Krishnakumar A (2007) Active learning literature survey. Tech. rep., Technical Report, University of California, Santa Cruz
Lancichinetti A, Radicchi F, Ramasco J, Fortunato S, Ben-Jacob E (2011) Finding statistically significant communities in networks. PLoS ONE 6(4):e18961
Lancichinetti A, Fortunato S, Kertész J (2009) Detecting the overlapping and hierarchical community structure in complex networks. New J Phys 11(3):033015
Lancichinetti A, Fortunato S, Radicchi F (2008) Benchmark graphs for testing community detection algorithms. Phys Rev E 78(4):046110
Lee C, Reid F, McDaid A, Hurley N (2010) Detecting highly overlapping community structure by greedy clique expansion. In: Workshop on Social Network Mining and Analysis
Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401:788–91
Leng M, Yao Y, Cheng J, Lv W, Chen X (2013) Active semi-supervised community detection algorithm with label propagation. In: International Conference on Database Systems for Advanced Applications. Springer, Berlin. pp 324–338
Leskovec J, Krevl A (2015) SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data
Li L, Du M, Liu G, Hu X, Wu G (2014) Extremal optimization-based semi-supervised algorithm with conflict pairwise constraints for community detection. In: Proc. ASONAM'14. IEEE, USA. pp 180–187
Liu D, Duan D, Sui S, Song G (2015) Effective semi-supervised community detection using negative information. Math Probl Eng 2015:2015
Liu D, Liu X, Wang W, Bai H (2014) Semi-supervised community detection based on discrete potential theory. Phys A Stat Mech Appl 416:173–182
Liu X, Wei YM, Wang J, Wang WJ, He DX, Song ZJ (2016) Community detection enhancement using non-negative matrix factorization with graph regularization. Int J Mod Phys B 30(20):1650130
McDaid A, Hurley N (2010) Detecting highly overlapping communities with model-based overlapping seed expansion. In: Proc. ASONAM'10. IEEE, USA. pp 112–119
Newman ME (2006) Modularity and community structure in networks. Proc. national academy of sciences 103(23):8577–8582
Pakrashi A, Mac Namee B (2019) Kalman filter-based heuristic ensemble (kfhe): A new perspective on multi-class ensemble classification using kalman filters. Inf Sci 485:456–485
Palla G, Derényi I, Farkas I, Vicsek T (2005) Uncovering the overlapping community structure of complex networks in nature and society. Nature 435(7043):814
Prince M (2004) Does active learning work? a review of the research. J Eng Educ 93(3):223–231
Shen H, Cheng X, Cai K, Hu MB (2009) Detect overlapping and hierarchical community structure in networks. Phys A Stat Mech Appl 388(8):1706–1712
Shi X, Lu H, He Y, He S (2015) Community detection in social network with pairwisely constrained symmetric non-negative matrix factorization. In: Proc. ASONAM'15. IEEE, USA. pp 541–546
Tang W, Lease M (2011) Semi-supervised consensus labeling for crowdsourcing. In: SIGIR 2011 workshop on crowdsourcing for information retrieval (CIR). pp 1–6
Wang Z, Wang W, Xue G, Jiao P, Li X (2015) Semi-supervised community detection framework based on non-negative factorization using individual labels. In: International conference in swarm intelligence. Springer, Cham. pp 349–359
Wu J, Wang F, Xiang P (2016) Automatic network clustering via density-constrained optimization with grouping operator. Appl Soft Comput 38:606–616
Wu ZH, Lin YF, Gregory S, Wan HY, Tian SF (2012) Balanced multi-label propagation for overlapping community detection in social networks. J Comput Sci Technol 27(3):468–479
Xie J, Kelley S, Szymanski BK (2013) Overlapping community detection in networks: The state-of-the-art and comparative study. ACM Comput Surv 45(4):43
Xie J, Szymanski BK, Liu X (2011) SLPA: Uncovering overlapping communities in social networks via a speaker-listener interaction dynamic process. In: Proc. IEEE 11th International Conference on Data Mining Workshops. IEEE, USA. pp 344–349
Yang J, Leskovec J (2013) 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, New York. pp 587–596
Yang J, Leskovec J (2015) Defining and evaluating network communities based on ground-truth. Knowl Inf Syst 42(1):181–213
Yang L, Jin D, Wang X, Cao X (2015) Active link selection for efficient semi-supervised community detection. Sci Rep 5:9039
Zhang Y, Yeung DY (2012) Overlapping community detection via bounded nonnegative matrix tri-factorization. In: Proc. 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, New York. pp 606–614
Zhang ZY (2013) Community structure detection in complex networks with partial background information. EPL Europhys Lett 101(4):48005
Zhang ZY, Sun KD, Wang SQ (2013) Enhanced community structure detection in complex networks with partial background information. Sci Rep 3:3241

Publisher's Note