Noname manuscript No.
(will be inserted by the editor)

Semi-Supervised Louvain: Fast and Scalable Community Detection Under Semi-Supervision in Large Networks

Emir Demirović · Mohadeseh Ganji · Peter J. Stuckey · Tias Guns · Jeffrey Chan · James Bailey · Kotagiri Ramamohanarao · Christopher Leckie

Received: date / Accepted: date

Abstract Detecting groups of strongly interconnected nodes is a critical task in the analysis of complex networks. Many real-world applications, when converted into a graph representation, exhibit such *community* structures, and there exist fast and scalable algorithms to identify the underlying communities. Apart from the graph structure, additional information might be available to guide the search for relevant communities. In *semi-supervised community detection* the information is used to extract constraints (e.g., must-link and cannot-link constraints) that are taken into account by community detection algorithms. However, existing algorithms are not as scalable as unconstrained community detection algorithms. In this work, we address this gap by proposing a fast and scalable semi-supervised community detection algorithm that outperforms previous semi-supervised methods in terms of speed, quality, scalability, and the number of constraints satisfied. Experiments on both real-world and artificial datasets show that the scalability improvements are several orders of magnitude, allowing semi-supervision to be applied to networks with millions of nodes, and enable its application to a broader set of problems than was previously possible.

Keywords Community Detection · Semi-Supervision · Louvain Algorithm · Cannot Link Constraints · Must Link Constraints · Large Networks

Emir Demirović* (corresponding author), Mohadeseh Ganji, Peter J. Stuckey, James Bailey, Kotagiri Ramamohanarao, Christopher Leckie
{emir.demirovic; mohadeseh.ganji; pstuckey; baileyj; kotagiri; caleckie}@unimelb.edu.au
University of Melbourne, Australia

Tias Guns
tias.guns@vub.be
Vrije Universiteit Brussel, Belgium

Jeffrey Chan
jeffrey.chan@rmit.edu.au
RMIT University, Australia

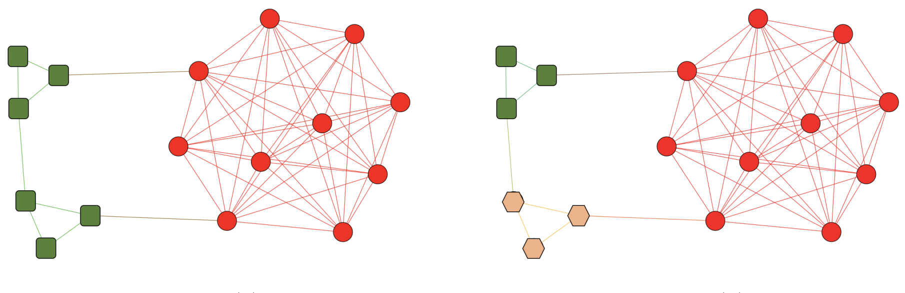
1 Introduction

Community detection aims to identify densely interconnected groups of nodes in a network. This is a vital operation in the structural analysis of complex graphs, including the study of social and biological networks [11, 16], phone networks [6], and road networks.

In the scientific literature, a number of definitions and algorithms have been proposed, tailored to the needs of specific applications. Communities are (loosely) defined as sets of nodes that are more densely connected among themselves than to nodes outside of their community. A variety of metrics [14, 21, 24] are available to evaluate the quality of the identified groups. One of the most widely-used measures is *modularity*, which evaluates the difference between the assigned communities and those in a randomized version of the network. Modularity has been widely studied in the context of community detection. Despite its known limitations, such as resolution limit [12], modularity has been one of the most popular community detection measures as according to [11]: "Modularity has the unique privilege of being at the same time a global criterion to define a community, a quality function and the key ingredient of the most popular method of graph clustering." Since modularity's introduction, several exact and approximate algorithms have been proposed for modularity based community detection among which, the Louvain algorithm [6] is the leading approach to computing communities with maximum modularity. It has been shown that the modularity based algorithm of Louvain is among the highest performing community detection methods [19] in terms of modularity maximization, speed, and scalability.

In addition, community detection may be performed in an *unsupervised* or *semi-supervised* manner. The former receives a graph as input and produces a label for each node representing the community it belongs to. The solution is based solely on a predefined objective function that is assumed to be appropriate for the application at hand. As a mean of incorporating domain-specific knowledge, semi-supervised community detection additionally includes a set of constraints as part of the input. The constraints provide information about the problem that are not present in the raw structure of the graph, derived from such sources as resource limitations, laws of physics, and domain knowledge. Taking such constraints into account can lead to solutions that are more suited to the specifics of the application in question.

Example 1 Consider the graph shown in Figure 1(a). There are three cliques of size ten, three, and three, that are weakly linked making up the graph. The modularity criterion is maximized by regarding the large clique as one community and merging the two cliques of size three. Thus, unsupervised community detection algorithms based on modularity maximization find two communities as shown in Figure 1(a). By using semi-supervision techniques, it is possible to prevent the merge of the two small cliques, which should form two separate communities as there is only one edge out of the nine possible between the two sets of vertices. This is accomplished by adding a single



(a) (b)

Fig. 1: An example showing the weakness of modularity for defining communities. (a) The communities that maximize modularity with no constraints, and (b) the desired communities, which can be recovered by add a cannot-link constraint.

cannot-link constraint between any two nodes of the two cliques of size three, which forbids the two nodes to be placed in the same community. Hence, a *reasonable* community structure of the graph is recovered. □

Previous approaches for semi-supervision are based on altering the graph [36], metric [10], or using Lagragian multipliers [13] to include the external information. There are two main limitations for current techniques. First, constraint satisfaction is not guaranteed. Therefore, there is an inherent issue with these techniques as they do not take advantage of the available information. As the constraints encode valuable knowledge about the problem that cannot be seen merely based on the graph, it is important to respect these constraints to obtain the desired result. Second, only graphs with a few thousand nodes can be dealt with effectively. In contrast, unsupervised community detection is able to efficiently address large graphs involving millions of nodes. This considerably limits the applicability of semi-supervised community detection to real-world problems, which can be have billions of nodes, e.g., the Facebook friendship graph. Thus, for a particular problem, even though valuable information might be available, it may not be possible to exploit it to obtain a more informative result. All of the mentioned issues are addressed in this paper.

We propose an alternative approach and introduce a novel technique for semi-supervised community detection. Our method, which is based on the Louvain algorithm [6], computes community assignments with progressively higher modularity values while respecting the given constraints. There are several notable contributions of our algorithm. First, the scalability matches unsupervised community detection algorithms, allowing semi-supervision to be applied to graphs with millions of nodes. Second, the most commonly used constraints in semi-supervision, *must-links* and *cannot-links*, are fully accounted for: they are satisfied at any given point during the algorithm. Thus, side information about the problem is appropriately utilized to provide high quality solutions. Third, the computation time does not differ significantly when compared to the unsupervised case, i.e., the time difference is within a small constant. As a result, semi-supervised community detection can now be applied to a significantly broader set of problems. We conduct extensive experimentation on important benchmarks to analyze the efficiency of the proposed algorithm and compare to existing approaches.

The rest of the paper is organized as follows. In the next section we provide the necessary preliminaries for community detection, modularity, semi-supervision, and the Louvain algorithm [6] as an important component of our method. We then proceed to describe our novel techniques in Section 3. In Section 4, we present detailed experimental results, where we study our proposed method and compare with previous approaches on a diverse set of benchmarks. Related approaches are given in Section 5. We conclude in the last section.

2 Preliminaries

Community Detection. Given an undirected graph $G = (V, E)$ made up of a set of vertices V and a set of edges (unordered pairs of vertices) E, community detection is the task of partitioning the graph such that vertices in a community communicate densely with members of the same community and have sparse connections to the rest of the network. The partition can be shown by a mapping c for each vertex showing to which community it belongs.

We make use of weighted graphs in the intermediate stages of community algorithms. A weighted graph $W = (V, E, w)$ consists of a graph (V, E) and a weight function w mapping each edge to a non-negative weight (natural numbers will suffice for our purposes). We let $k_i, i \in V$ be the (weighted) degree of vertex i, given by $k_i = \sum_{ij \in E} w(ij)$. Furthermore, we denote the number of vertices in the graph as $n = |V|$ and with m the sum of the weights of edges in the graph (for the unweighted case, $m = |E|$).

Modularity. One of the most well-known measures of community detection is modularity [30]. It has been widely used in physics, statistics, computer science and complex network research community. The underlying assumption of modularity is that randomized graphs (where edges are distributed randomly among vertices), do not show community structure. Hence, modularity compares the graph structure to expected structure in the corresponding randomized graph to quantify how community-like the graph structure is.

Given a weighted undirected network $W = (V, E, w)$, and a community mapping c, the modularity of the partition defined by the mapping is given by:

$$Q = \frac{1}{2m} \sum_{i,j} [w(ij) - \frac{k_i k_j}{2m}] \delta(c(i), c(j)). \tag{1}$$

The term $w(ij) - \frac{k_i k_j}{2m}$ between pairs of vertices i and j is the modularity value for the pair. Modularity quantifies the difference between the weight of the edge connecting each pair of vertices and the expected weight of such edges in a randomized network, also known as the null model. The coefficient $\frac{1}{2m}$ normalizes modularity to the interval [-1, 1].

The modularity of a partition is the summation of the modularities between each pair of vertices that lie in the same community. In Equation 1, $\delta(c(i), c(j))$, the Kronecker delta function performs this task by limiting the summation to be just over vertex pairs of the same community. The Kronecker function has the value 1 if its arguments are equal and 0 otherwise.

Rather than defining modularity for each pair of vertices, modularity can also be defined based on edges inside and between communities. Let $S_{\ell k}$ be the sum of (weighted) edges in the network that connect vertices in community ℓ to vertices in community k, $S_{\ell k} = \sum_{ij \in E, c(i)=\ell, c(j)=k} w(ij)$. Let $S_{E\ell}$ be the sum of (weighted) edges that connect to community ℓ, $S_{E\ell} = \sum_{ij \in E, c(i)=\ell} w(ij)$. Then the modularity value of a partition is the summation of modularity values of its K communities as:

$$Q = \sum_{\ell=1}^{K} \left(\frac{S_{\ell\ell}}{2m} - (\frac{S_{E\ell}}{2m})^2 \right). \tag{2}$$

The modularity value of each community is the fraction of edges that fall within that community (denoted by $S_{ll}/2m$), minus the expected value of the same quantity in a randomized network, where $((S_{El}/2m)^2)$ is the fraction of ends of (weighted) edges that are connected to vertices in community l.

Semi-supervision. Background information is often represented in the form of pairwise relations between vertices. *Must-link constraints* require that the pairs belong to the same community. We assume must-link constraints are given as a set of unordered pairs ML, where $ij \in ML \to c(i) = c(j)$. Similarly, *cannot-link constraints* require that the pairs belong to different communities, they are represented as a set of unordered pairs CL, where $ij \in CL \to c(i) \neq c(j)$. Constraints can be imposed on two nodes regardless of whether there is an edge connecting them.

Our focus in this work is on situations where satisfying all the constraints are required. Hence, the success of our algorithm should be judged by how many constraints it satisfies, in addition to the quality of the resulting partition. This implicit assumption is suitable for situations where the constraints are true representation of the ground truth. Although this is not always the case in complex problems, we provide three examples to illustrate cases where this assumption holds. First, the constraints may describe a known physical or functional characteristic of a system, where the user has complete confidence about the community membership for some nodes, e.g., a pair of unreachable objects in a road network can be represented as cannot-link. Second, the constraints may be obtained as a result of a procedure that discovers the ground truth (cluster membership) for a small sample of instances, for example, an expensive/invasive medical test like a biopsy. Third, in some applications such as image segmentation, users can easily generate a large number of supervision constraints with a high confidence (pairs of pixels that do/do not belong to the same object). Note that for the cases where the constraints only reflect a belief about the ground truth, we would still expect the resulting communities obtained with semi-supervision to be closer to the ground truth than when compared to a pure unsupervised approach.

The Louvain algorithm. A fast and scalable two-phase local search algorithm for modularity maximisation has been given in [6], known in the community as the *Louvain* algorithm.

Let $c(i)$ be the community identifier for vertex i. At the start of the algorithm, each vertex is placed in its own community ($c(i) = i, i \in V$) and the weight function simply weights each edge as a unit $w(e) = 1, e \in E$.

In the first phase we consider the effect on modularity of moving vertex i to the community of each of its neighbours $j, ij \in E$. We choose the community (including its own current community $c(i)$) that leads to the maximum increase in modularity, and move i to that community.

We can calculate the change in modularity for moving vertex i to community $l = c(j)$ as

$$\Delta Q_{il} = \left(\frac{S_{ll} + d_{il}}{2m} + \left(\frac{S_{El} + d_i}{2m} \right)^2 - \left(\frac{S_{ll}}{2m} - \left(\frac{S_{El}}{2m} \right)^2 - \left(\frac{d_i}{2m} \right)^2 \right) \right),$$

where d_{il} is the sum of weighted edges from vertex i to community l, $d_{il} = \sum_{ij \in E, c(j) = l} w(ij)$. This process is repeatedly applied to all vertices, until no further vertices move.

The second phase of the Louvain algorithm is to merge all vertices in a community into a single vertex. Given a community mapping c, we create new nodes $V' = \{c(j) | j \in V\}$ and new weighted edges. $E' = \{c(i)c(j) | ij \in E\}$ where $w'(ab) = \sum_{ij \in E, c(i) = a, c(j) = b} w(ij), ab \in E'$. Note that the merged graph may contain self loops.

We then repeat Phase 1 of the algorithm. The entire algorithm halts when the entire Phase 1 makes no changes to the community mapping.

```
L(V,E)
    $w := \{e \mapsto 1 \mid e \in E\}$              % initial weights are 1
    $c := \{v \mapsto v \mid v \in V\}$              % assign singleton communities
    while($true$)
        $q := modularity(V, E, w, c)$          % record current modularity
        $c :=$ phase1$(V, E, w)$
        if $modularity(V, E, w, c) = q$
            break                             % no improvement, finish
        $(V, E, w) :=$ phase2$(V, E, w, c)$
        $c := \{v \mapsto v \mid v \in V\}$          % singleton communities after merge
    return $c$

phase1(V,E,w)
    repeat
        $c_s := c$                            % record current community
        forall($i \in V$)                     % foreach node
            $l := \arg\max_{l \in \{c(j) \mid ij \in E \cup \{ii\}\}} \Delta Q_{ic(j)}$
            $c(i) := l$                         % move it to maximize modularity
    until $c = c_s$                    % until no change is made
    return $c$

phase2($V, E, W, c$)
    $V' := \{c(j) \mid j \in V\}$
    $E' := \{c(i)c(j) \mid ij \in E\}$
    $w := \{ab \mapsto \sum_{ij \in C, c(i)=a, c(j)=b} w(ij)\}$
    return $(V', E', w')$
```

Example 2 Consider the graph shown in Figure 1. The Louvain algorithm starts with each vertex in its own community. After the first iteration through the nodes, the resulting communities are shown in Figure 1(b) except that the northwestern node in the largest community is assigned to the small community to the left. Note that since each iteration is greedy it makes poor decisions for some vertices. But in the next iteration, these decisions are fixed, and Phase 1 ends with the communities as shown in Figure 1(b). These communities are merged into single super-nodes. In the next iteration Louvain merges the two super-nodes for the small communities into one community, arriving at the communities as shown in Figure 1(a), achieving the higher modularity for the example. Clearly modularity maximization is not giving us the obvious result.

3 Novel Semi-Supervised Community Detection Algorithm

Our semi-supervised community detection method is based on the previously described Louvain algorithm [6]. We consider two types of constraints: *cannot-links* and *must-links*. In our setting, the set of cannot and must links are provided in the input. The constraints represent external information about the problem that should be accounted for in addition to the graph structure. Their main purpose is to guide the community detection algorithm towards a high quality solution for the application at hand. Note that such a solution might have a lower modularity value than the optimum, but is likely to lead to a solution close to a reasonable underlying community structure. The main idea is to perform moves in the Louvain algorithm while maintaining constraint satisfaction. Therefore, there are two main components: merging nodes to respect must-link constraints and preventing infeasible moves to account for cannot-link constraints.

Merging for Must-Link Constraints. Must-link constraints are handled by merging must-link constrained nodes into the same clusters. We start with an initial community assignment that reflects the must-link constraints i.e. $c(i) = \min\{j \mid ij \in tc(ML)\}$, where $tc(ML)$ is the transitive closure of the must-link relation. Afterwards, Phase 2 of Louvain is applied to merge these communities into super-nodes, which will not be separated in further iterations of the algorithm. Hence, the must-link constraints are guaranteed to be satisfied at any point of the algorithm.

Preventing Infeasible Moves for Cannot-Link Constraints. We forbid moves that would lead to cannot-link violations. This is achieved as follows. When we consider moving a node i into community l, we check if there exists $ij \in CL$ where $c(j) = l$. If this is the case, we do not consider the move in computing the *arg max* in Phase 1. By storing the cannot-links attached to each node, this is done efficiently.

After merging nodes into super-nodes in Phase 2, the cannot-links must be updated to reflect the constraints considering the new super-nodes. We compute the modified cannot-link set as follows: $CL' = \{c(i)c(j) \mid ij \in CL\}$.

Example 3 Consider again the graph in Figure 1 with the addition of a single cannot-link constraint between the two adjacent nodes in the two smaller communities. The EL algorithm proceeds like the Louvain algorithm as described in Example 2 for the first Phase 1 and Phase 2. In the next round, the cannot-link constraint, which is satisfied, prevents the merging of the two super-nodes representing the two small communities. Hence the final answer is as shown in Figure 1(b). $\square$

We note that we experimented with several variants of the algorithm, such as delaying must-link node merging and/or ignoring constraints and later performing *repairs* on the solution. However, we did not observe significant differences in performance when compared to the approach described in this paper despite its simplicity.

Computational Complexity. The computational complexity is similar as for the Louvain algorithm, differing only in a small constant. The precise runtime of Louvain is not known analytically since the number of iterations in Phase 1 and *passes* (Phase 1 + Phase 2) are not known apriori. According to [6], in practice the algorithm behaves as $n(log(n))$. Similar reasoning holds for our Extended Louvain algorithm. However, we can show analytically that the number of operations in a single iteration and Phase 2 in our algorithm is similar to standard Louvain.

Let n be the number of nodes, m the average node degree, and p the average number of cannot links for a node. A single iteration of Phase 1 iterates through each node and computes its best move. The best move for a given node can be determined in $O(mlog(m))$ operations. In our algorithm, in addition to computing the move, we need to filter out forbidden moves due to cannot-links, which can be done at an additional $O(mp)$ operations per node. Hence, a single iteration requires $O(n(mlog(m) + mp))$. In Phase 2, each node is used to update the edges of its corresponding super-node. In our algorithm, cannot-links must be updated as well, which incurs an additional $O(mp)$ expense per node. Therefore, the total complexity for one execution of Phase 2 is $O(n(m+mp))$. However, it is expected that n will be significantly larger than m and p ($n \gg m$ and $n \gg p$), reducing the complexity expressions to $O(n)$ for both algorithms. Hence, the difference in computation time when compared to standard Louvain is only a small constant. Note that the performance of the algorithm is largely dominated by the first Phase 1 and 2 execution. Afterwards, the nodes are merged into super-nodes, which reduces the number of nodes substantially.

4 Experiments

We compare with four existing approaches from the literature: Spinglass [10], SC [36], SNMF-SS [26], and LagCCD [13]. See Section 5 for more details.

We use a variety of artificial and real-world datasets to assess the effectiveness of our approach. For each benchmark, we are given *a priori* a target community structure. Solutions are measured with respect to the provided ground truth using Normalized Mutual Information (NMI) [22]. It computes a similarity value between 0 and 1, where higher values are desired. Note that when given a network in a real-life scenario, the ground truth is not known in advance but it is something the algorithm is attempting to discover. Therefore, in our experimental setting, we are measuring how successful the methods are in retrieving metadata groups (community assignments) given the constraints. In addition, we aim to test the scalability of our technique.

Constraints are generated in the following way: two random nodes from the same (different) community form a must-link (cannot-link) constraint. Unless explicitly specified, we maintain an equal balance between must- and cannot-link constraints.

We note that the NMI metric is commonly used for evaluating semi-supervised community detection (e.g., [8]) and NMI has a long history as a tool for assessing dependence between discrete random variables such as clusterings or communities [1]. Its normalized nature (between 0 and 1) enables easy interpretation and its foundation in mutual information links it to the log likelihood ratio test commonly used in statistics for comparing the goodness of fit of two statistical models.

4.1 Benchmarks and Computational Environment

The used datasets can be divided into three groups: standard real-world instances, Lancichinetti-Fortunato-Radicchi (LFR) [20] graphs, and two large real-world networks from the Stanford Network Analysis Platform (SNAP) datasets with ground truth [23].

Standard Real-World Instances. These benchmarks have been collected throughout the years by researchers and are commonly used as benchmarks for comparison between semi-supervised methods. Their origin and graph characteristics are given in Table 1.

Lancichinetti-Fortunato-Radicchi (LFR) Graphs. These are synthetic datasets designed to benchmark (semi-supervised) community detection algorithms. They take into account the heterogeneity of node degree distributions and community sizes, aiming to produce realistic test instances. The following parameters were used: the average node degree was set to 15, mixing parameter to

Table 1: Characteristics of the Standard Real-World Instances

Name	#nodes	#edges	#communities
Sampson T1T5 [32]	25	69	2
Strike [28]	24	38	3
Zakhary's Karate club [35]	34	78	2
Mexican [15]	35	117	2
Dolphin [25]	62	159	2
Political Books [18]	105	441	3
Word adjacencies [29]	112	425	2
Political blogs [2]	1,490	9,545	2

Table 2: Characteristics of the SNAP datasets used.

Name	#nodes	#edges	#disjoint communities
com-DBLP	317,080	1,049,866	13,122
com-Amazon	334,863	925,872	28,220

0.75, minimum and maximum community size set to 50 and 100 (respectively), and the exponents set to their default values (1 and 2). In our preliminary experimentation, varying the parameters influenced the resulting NMIs, but the relative performance of the algorithms did not change.

Stanford Network Analysis Platform (SNAP) Datasets with Ground Truth[23]. We used two large real-world networks whose characteristics are given in Table 2. The datasets provide overlapping ground truths. We converted them into disjoint versions by sequentially assigning each vertex to exactly one of its or its neighbors' overlapping communities, giving preference towards groups containing a greater number of neighboring nodes.

All tests were performed on an Intel Core i7-3612 CPU @ 2.10GHz with 8 GB of RAM and each instance was given a single core.

4.2 Results

We label our method as Extended Louvain (EL). The experimentation is divided into three phases, with each phase addressing a set of distinct research questions. These are explained in the following.

Comparison with existing algorithms from the literature. We evaluate the obtained solutions with respect to existing approaches. The summary is given in Table 3. Our approach provides competitive results when compared to LagCCD. Friedman statistical tests with 95% confidence levels revealed that EL statistically outperforms the remaining algorithms, obtaining P-values of order 10^{-5} in pairwise evaluations. We attribute EL's success to the strict constraint-management policy in addition to maximizing modularity. This is a crucial difference with the other approaches. Constraints lead the algorithm towards solutions closer to the underlying community structures. Our method takes full advantage of the provided constraints and hence achieves higher NMI values. This is emphasized as the number of constraints increases. Similar behavior, although to a lesser extent, can be seen in LagCCD. It manages to keep the constraint violations low and indeed provides competitive NMIs. Therefore, we conclude that respecting constraints plays an important role in obtaining high NMIs.

Evaluation of scalability and the impact of different ratios of must- and cannot-links. We compare our approach with LagCCD on large LFR instances in Table 4. We perform a direct comparison with LagCCD as it was able to achieve good solutions in terms of constraint violations and solution quality compared to the other semi-supervised approaches.

On each benchmark, our algorithm provided a better solution and achieved a consistent performance in terms of NMI and execution time. This is further confirmed by achieving P-values of order 10^{-2} in Friedman statistical tests. In addition, we may observe the expected linear scaling in time with the instance size. Furthermore, given that LagCCD requires a quadratic amount of memory with respect to the number of nodes, it was unable to handle the graphs with 10^5 or more nodes. Our approach requires $O(|E|)$ memory and therefore were not subject to this limitation.

The ratios of must-links is an important factor for NMI. Must-link constraints contribute towards higher NMI values, as each must-link constraint provides valuable information about the problem structure. In contrast, cannot-links have a higher chance of being uninformative: they might state that two already distant nodes in the graph are in different communities.

Analyze applicability to large real-world graphs. Results for the two large real-world networks are given in Table 5. The conclusions drawn from previous experimentation hold on these benchmarks as well. Clearly the overhead of our method over Louvain even for these very large graphs is not large.

Benchmark Name	# Constraints	L	#viols	EL
com-amazon.ungraph	167430	0.61 (3.00)	13159	**0.67 (3.00)**
	334862	0.61 (4.00)	26201.50	**0.70 (4.00)**
com-dblp.ungraph	158540	0.40 (4.00)	55236	**0.49 (4.20)**
	317080	0.40 (4.40)	109865	**0.65 (4.20)**

Table 5: Results for two large real-world networks.

5 Related Work

Community Detection. Communities can be defined as groups of vertices that are densely connected to each other but have sparse connections to the rest of the network. However, there is no universally agreed definition for communities. In some schemes of community detection, conditions are imposed on partitions and any partition satisfying those conditions is a solution to the community detection problem. Two examples of such conditions are introduced by Radicchi [31] named communities in weak and strong senses.

Sub-graph S is a *strong community* if and only if each vertex in S has more neighbors in the same community than the neighbors from the rest of

Benchmark Name	# Constraints	L	#viols	EL	LagCCD [13]		#viols	Spinglass [10]	#viols	Spectral [36]	#viols	NMF [26]	#viols
Polblogs	744	0.38	185.80	0.47	0.36	(26.66)	0.74	0.35	43.26	0.32	61.48	**0.53**	371.38
	1490	0.38	355.38	0.62	0.47	(29.40)	5.46	0.43	86.31	0.34	135.54	**0.67**	743.44
SampsonT1T5	12	0.66	1.86	**0.65**	0.61	(<1)	0.00	0.61	0.30	0.61	0.80	0.53	2.26
	24	0.66	4.80	0.84	**0.91**	(<1)	0.08	0.82	0.70	0.61	3.72	0.50	6.46
	50	0.66	8.00	**1.00**	**1.00**	(<1)	0.00	0.86	1.82	0.61	6.06	0.51	11
Strike	12	0.59	1.88	0.71	**0.73**	(<1)	0.00	0.70	0.50	0.62	1.30	0.64	2.40
	24	0.59	4.20	0.68	**0.75**	(<1)	0.00	0.71	1.42	0.63	1.80	0.66	3.36
	48	0.59	10.60	**0.90**	**0.90**	(<1)	0.00	0.80	1.78	0.71	3.96	0.65	8.50
UndirectPolbooks	52	0.55	10.00	0.64	**0.66**	(26.66)	0.00	0.55	8.36	0.49	12.58	0.43	18.32
	104	0.54	21.14	**0.77**	0.76	(29.40)	0.20	0.59	14.88	0.49	22.70	0.43	34.24
	210	0.55	42.60	**0.94**	0.93	(32.98)	0.18	0.63	26.14	0.49	49.3	0.43	68.42
Zakhary2k	16	0.59	3.12	**0.73**	0.72	(<1)	0.00	0.71	0.46	0.69	2.80	0.68	3.20
	34	0.59	9.20	**0.88**	**0.88**	(<1)	0.00	0.88	0.24	0.68	4.36	0.69	8.88
	68	0.59	17.60	**0.97**	**0.97**	(<1)	0.00	0.93	0.60	0.65	10.86	0.68	15.56
Adjourn	56	0.00	26.60	**0.02**	**0.02**	(2.36)	0.04	0.01	14.36	0.01	16.98	0.01	27.66
	112	0.00	54.00	**0.08**	0.06	(2.44)	1.20	0.01	26.98	0.02	35.84	0.01	56.18
	224	0.01	113.60	**0.49**	0.32	(2.35)	8.88	0.02	60.14	0.02	71.70	0.01	111.92
Dolphin2	30	0.51	8.20	**0.67**	0.65	(<1)	0.00	0.58	4.98	0.49	6.78	0.49	9.02
	62	0.51	17.60	**0.72**	0.71	(<1)	0.00	0.62	8.10	0.51	12.52	0.48	18.32
	124	0.51	32.71	0.95	**0.96**	(<1)	0.00	0.76	9.36	0.60	17.86	0.49	32.02
Mexican	16	0.26	5.43	**0.33**	0.31	(<1)	0.00	0.28	3.56	0.22	5.04	0.29	5.28
	34	0.26	13.40	0.42	**0.43**	(<1)	0.00	0.28	7.40	0.24	10.9	0.28	12.02
	70	0.26	25.60	0.76	**0.79**	(<1)	0.00	0.45	10.22	0.29	18.18	0.27	26.04

Table 3: Comparison of EL (our method) with existing approaches on the standard real-world datasets. Average NMI achieved displayed, with time in brackets for LagCCD (the remaining methods were all under a second for the complete benchmark suite). Best results for a benchmark are given in bold. The number of constraint violations shown for algorithms that do not guarantee constraint satisfaction.

n	#constraints	0% ml - 100% cl		25% ml - 75% cl		50% ml - 50% cl		75% ml - 25% cl		100% ml - 0% cl	
		EL	LagCCD	EL	LagCCD	EL	LagCCD	EL	LagCCD	EL	LagCCD
10^3	n	**0.07 (0.04)**	0.07 (14.29)	**0.18 (0.08)**	0.16 (12.49)	**0.52 (0.04)**	0.36 (10.31)	**0.80 (0.12)**	0.66 (9.25)	**0.87 (0.04)**	0.78 (8.03)
	2n	**0.08 (0.04)**	0.08 (15.41)	**0.54 (0.00)**	0.39 (11.36)	**0.91 (0.08)**	0.82 (8.76)	**0.97 (0.00)**	0.95 (10.31)	**0.96 (0.00)**	0.96 (8.18)
10^4	n	**0.11 (0.48)**	0.11 (1066.22)	**0.69 (0.24)**	0.44 (537.66)	**0.85 (0.32)**	0.64 (419.82)	**0.85 (0.32)**	0.73 (327.3)	**0.84 (0.24)**	0.76 (314.31)
	2n	**0.12 (0.44)**	0.07 (1112.48)	**0.89 (0.40)**	0.03 (51.32)	**0.90 (0.20)**	0.31 (10.31)	**0.89 (0.28)**	0.00 (11.52)	**0.84 (0.32)**	0.01 (5.18)
10^5	n	**0.24 (5.24)**	—	**0.74 (3.84)**	—	**0.79 (2.88)**	—	**0.78 (2.76)**	—	**0.76 (2.68)**	—
	2n	**0.29 (5.56)**	—	**0.82 (3.28)**	—	**0.82 (3.04)**	—	**0.80 (2.88)**	—	**0.76 (2.68)**	—
10^6	n	**0.34 (157.76)**	—	**0.63 (64.44)**	—	**0.67 (40.40)**	—	**0.65 (37.68)**	—	**0.63 (39.72)**	—
	2n	**0.40 (171.24)**	—	**0.72 (46.24)**	—	**0.70 (43.32)**	—	**0.68 (48.00)**	—	**0.62 (36.00)**	—

Table 4: Analyzing scalability and impact of must- and cannot-links for on LFR graphs. Comparison of EL (our method) and LagCDD. The columns indicate the ratio of each constraint type. Total of 25 instances per row. Average NMI achieved displayed with time in brackets. Constraint violations by LagCDD not shown for simplicity. Out of memory denoted by —.

the network. A sub-graph S is a *community in the weak sense* if and only if the sum of the internal degrees of the community is larger than the number of edges connecting the community to the rest of the network.

A range of community detection algorithms are based on modularity maximization. As explained in Section 2, modularity compares the network connections with a random network called the null model and quantifies the deviations. However, Brandes et al. [7] showed that finding a partition of the network with maximum modularity is an NP-hard problem.

Medus et al. [27] proposed a simulated annealing algorithm and Duch et al. [9] proposed an extremal optimization for modularity maximization.

There are some attempts to mathematically model the modularity maximization problem by integer programming [3] and mixed integer convex quadratic programming [34].

Among exact methods of modularity maximization, the one introduced by Aloise et al. [5], which is a column generation model, can find communities of optimal modularity value for problems of up to 512 vertices.

To overcome the scalability barrier, heuristic methods have been developed for modularity maximization among which the Louvain algorithm [6], described in Section 2 and used in this paper, is one of the most prominent. GenLouvain [17] is an implementation of the Louvain algorithm directly working with the modularity matrix as an input.

Semi-Supervised Community Detection. There has been an increasing interest in semi-supervised community detection where background domain-knowledge is incorporated in the community detection process. Such information is mainly represented in the forms of known labels or pairwise constraints. A label propagation algorithm, proposed by Silva et al. [33], starts by assigning each vertex to its own community (as in Louvain) and iteratively merges non-labeled communities with the labeled ones in a way that aims to maximize the modularity score. This process is continued until all communities are labeled. Allahverdyan et al. [4] investigated the effect of known-label supervision on the detectability threshold of communities in sparse graphs. It has been shown that communities are not detected properly when inter-cluster connections exceed a certain threshold.

The approach in [36] directly applies the information from the constraints to modify the adjacency matrix by connecting and disconnecting edges between must-link and cannot-link pairs in the original graph. The modified adjacency matrix can then be used in a spectral clustering or non-negative matrix factorization (NMF) algorithm. Xiaoke Ma et al. [26] proposed a NMF algorithm called SNMF-SS to incorporate must-link and cannot-link constraints. They define a violation cost for unsatisfied pairs and modify the similarity matrix K to $\overline{K} = K - \alpha W_{ML} + \beta W_{CL}$ where parameters α and β are the relevant importance of the two constraint types. These parameters are set such that $\overline{K}$ remains nonnegative.

In spite of the popularity of modularity for unsupervised community detection, there are few approaches in the literature that use a modularity-based measure in a semi-supervised community detection scheme. Among the existing approaches, Eaton et al. [10] proposed a measure based on a variation of Potts spin-glass model from statistical mechanics to handle pairwise constraints. A constraint violation cost is added to penalize both must-link and cannot-link violations. The spin-glass approach can then be seen as the optimization of a modified modularity matrix.

Ganji et al. [13] proposed a modularity-based Lagrangian framework for semi-supervised community detection that systematically increases the penalties on violated must-link and cannot-link constraint violations with the aim to satisfy the constraints eventually. In contrast to other methods, the Lagrangian framework aims to satisfy all of the constraints, although constraint satisfaction is not guaranteed. Hence, to the best of our knowledge, none of the existing semi-supervised community detection approaches can fully satisfy the constraints and benefit from each and every constraint.

6 Conclusion

We provided a new semi-supervision community detections algorithm based on the Louvain algorithm. Our algorithm is fast, scalable, and outperforms previous semi-supervised community detection algorithms on most benchmarks. The proposed method is at the level of unsupervised community detection algorithms in terms of scalability and speed. Hence, semi-supervised community detection can now be applied to a larger class of problems with millions of nodes. Increasing the accuracy of our method, possibly through combining with other existing approaches and incorporating, in addition to must- and cannot-links, other types of constraints, are directions for future work.

References

1.
2. Adamic, L.A., Glance, N.: The political blogosphere and the 2004 us election: divided they blog. In: Proc. of Link discovery, pp. 36–43. ACM (2005)
3. Agarwal, G., Kempe, D.: Modularity-maximizing graph communities via mathematical programming. The European Physical Journal B **66**(3), 409–418 (2008)
4. Allahverdyan, A.E., Ver Steeg, G., Galstyan, A.: Community detection with and without prior information. Europhysics Letter **90**(1), 18,002 (2010)
5. Aloise, D., Cafieri, S., Caporossi, G., Hansen, P., Perron, S., Liberti, L.: Column generation algorithms for exact modularity maximization in networks. Physical Review E **82**(4), 046,112 (2010)
6. Blondel, V.D., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment **2008**(10), P10,008 (2008)
7. Brandes, U., Delling, D., Gaertler, M., Gorke, R., Hoefer, M., Nikoloski, Z., Wagner, D.: On modularity clustering. IEEE Transactions on Knowledge and Data Engineering **20**(2), 172–188 (2008)
8. Davidson, I., Basu, S.: A survey of clustering with instance level constraints. ACM Transactions on Knowledge Discovery from Data **1**, 1–41 (2007)
9. Duch, J., Arenas, A.: Community detection in complex networks using extremal optimization. Physical review E **72**(2), 027,104 (2005)

10. Eaton, E., Mansbach, R.: A spin-glass model for semi-supervised community detection. In: 26th AAAI Conference on Artificial Intelligence, pp. 900–906 (2012)
11. Fortunato, S.: Community detection in graphs. Physics Reports **486**(3), 75–174 (2010)
12. Fortunato, S., Barthelemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Sciences **104**(1), 36–41 (2007)
13. Ganji, M., Bailey, J., Stuckey, P.J.: Lagrangian constrained community detection. In: Proceedings of the AAAI conference on Artificial Intelligence (AAAI 2018)
14. Ganji, M., Seifi, A., Alizadeh, H., Bailey, J., Stuckey, P.J.: Generalized modularity for community detection. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 655–670. Springer (2015)
15. Gil-Mendieta, J., Schmidt, S.: The political network in mexico. Social Networks **18**(4) (1996)
16. Girvan, M., Newman, M.E.: Community structure in social and biological networks. Proceedings of the national academy of sciences **99**(12), 7821–7826 (2002)
17. Inderjit S. Jutla, L.G.S.J., Mucha, P.J.: (A generalized Louvain method for community detection, http://netwiki.amath.unc.edu/GenLouvain)
18. Krebs, V.: www.orgnet.com/
19. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Physical review E **80**(5), 056,117 (2009)
20. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Physical Review E **78**(4) (2008)
21. Leicht, E., Holme, P., Newman, M.E.: Vertex similarity in networks. Physical Review E **73**(2), 026,120 (2006)
22. Leon Danon, A.D.G., Arenas, A.: The effect of size heterogeneity on community identification in complex networks. Journal of Statistical Mechanics: Theory and Experiment p. P11010 (2006)
23. Leskovec, J., Krevl, A.: SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data` (2014)
24. Li, K., Pang, Y.: A unified community detection algorithm in complex network. Neurocomputing **130**, 36–43 (2014)
25. Lusseau, D., Schneider, K., Boisseau, O.J., Haase, P., Slooten, E., Dawson, S.M.: The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. Behavioral Ecology and Sociobiology **54**(4), 396–405 (2003)
26. Ma, X., Gao, L., Yong, X., Fu, L.: Semi-supervised clustering algorithm for community structure detection in complex networks. Physica A **389**(1), 187–197 (2010)
27. Medus, A., Acuña, G., Dorso, C.: Detection of community structures in networks via global optimization. Physica A: Statistical Mechanics and its Applications **358**(2), 593–604 (2005)
28. Michael, J.H.: Labor dispute reconciliation in a forest products manufacturing facility. Forest products journal **47**(11/12), 41 (1997)
29. Newman, M.E.: Finding community structure in networks using the eigenvectors of matrices. Physical review E **74**(3), 036,104 (2006)
30. Newman, M.E., Girvan, M.: Finding and evaluating community structure in networks. Physical review E **69**(2), 026,113 (2004)
31. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. Proceedings of the National Academy of Sciences **101**(9), 2658–2663 (2004)
32. Sampson, S.F.: A novitiate in a period of change: An experimental and case study of social relationships. Cornell University (1968)
33. Silva, T.C., Zhao, L.: Semi-supervised learning guided by the modularity measure in complex networks. Neurocomputing **78**(1), 30–37 (2012)
34. Xu, G., Tsoka, S., Papageorgiou, L.G.: Finding community structures in complex networks using mixed integer optimisation. The European Physical Journal B-Condensed Matter and Complex Systems **60**(2), 231–239 (2007)
35. Zachary, W.W.: An information flow model for conflict and fission in small groups. Journal of anthropological research pp. 452–473 (1977)
36. Zhang, Z.Y.: Community structure detection in complex networks with partial background information. Europhysics Letters **101**(4), 48,005 (2013)