A Declarative Approach to Constrained Community Detection

Mohadeseh Ganji, James Bailey, and Peter J. Stuckey

Department of Computing and Information Systems, University of Melbourne, Australia
sghasempour@student.unimelb.edu.au, {baileyj,pstuckey}@unimelb.edu.au

Abstract. Community detection in the presence of prior information or preferences on solution properties is called semi-supervised or constrained community detection. The task of embedding such existing kinds of knowledge effectively within a community discovery algorithm is challenging. Indeed existing approaches are not flexible enough to incorporate a variety of background information types. This paper provides a framework for semi-supervised community detection based on constraint programming modelling technology for simultaneously modelling different objective functions such as modularity and a comprehensive range of constraint types including community level, instance level, definition based and complex logic constraints. An advantage of the proposed framework is that, using appropriate solvers, optimality can be established for the solutions found. Experiments on real and benchmark data sets show strong performance and flexibility for our proposed framework.

1 Introduction

Community detection is the task of identifying densely connected sub-graphs in networks. Although most research on community detection has focused on unsupervised learning, which only relies on network topology, there is emerging interest in semi-supervised or constrained community detection which benefits from existing side information as well. This can result in more efficient and actionable solutions.

More generally, the increasing flexibility of data mining techniques to deal with complex constraints has attracted increasing attention. The topic of constrained-based mining aims to develop data mining techniques that can handle complex and domain-specified constraints. This has been shown to be possible for some data mining tasks such as pattern and sequence mining, item set mining and constrained clustering using constraint solving technology (e.g. [18, 27, 14]).

There are two main motivations for constrained (semi-supervised) community detection:

Quality solutions: The community detection process can benefit from prior information to improve the quality of solutions. For example, the supervision effect has been studied in the presence of noisy links in the network and it has

been shown that semi-supervised community detection approaches are usually more robust to noise than topology-based approaches [15].

Figure 1 illustrates the effect of supervision on the quality of solutions of two different community detection problems. Figure 1-a is a network of a clique of size 50 connected to two small cliques of size 5. The left figure shows how pure modularity maximization merges the two small cliques to one community while adding a supervision constraint to force the number of communities to be 3 leads to more meaningful communities (right figure). Figure 1-b shows a circle structure of 30 cliques of size 5. The communities found using the modularity criterion (left figure) are unconvincing, since adjacent cliques are grouped into one community. Imposing a size constraint of between 5 to 9 on this community detection problem reveals the intuitively correct communities (right figure).

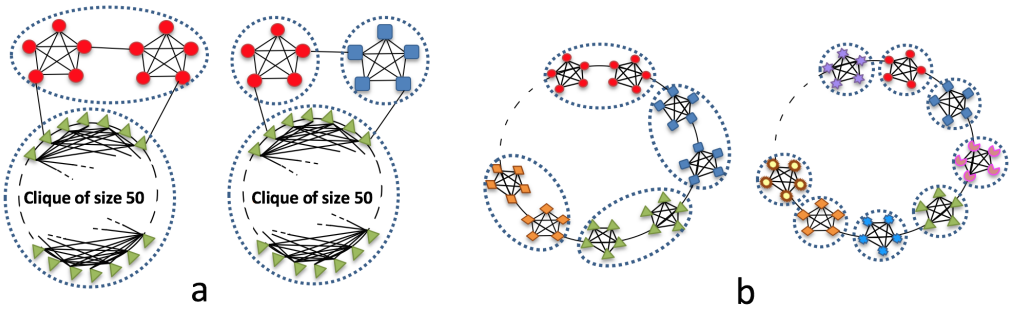

Fig. 1. The effect of adding a) a bound on the number of communities and b) bound on size of communities. Each figure shows partitions before (left) and after (right) applying the constraints.

Complex problem solving: Constrained community detection is the only way to tackle some challenging problems, in which different types of constraints must be satisfied at the same time. Constraints arise by the imposition of user preferences on community properties, or natural complexity of some problems with a variety of requirements to be satisfied simultaneously. As an example, consider finding groupings for a class of students engaged on a project. One may desire to balance the number of female and male students in each group. To make teamwork smoother, we may also require that everyone has several of his Facebook friends or classmates in the group. These are examples of user and ad hoc constraints to help detect communities with desired properties.

Some attempts have been made to adapt existing community detection algorithms to incorporate background knowledge [15, 2]. However, they are limited in the types of prior information that can be used. This lack of flexibility narrows the scope of problems they can tackle. For example, some algorithms can only incorporate pairwise instance level supervision [15, 2] while others are only capable of finding size-constrained communities [11]. When more than one type of supervision exists for a community detection task, *it is not clear how to integrate*

the results of such different algorithms. In other words, none of the existing approaches and algorithms has the flexibility to be able to solve complex problems by incorporating both classic and ad-hoc types of supervision and user-defined constraints at the same time. In this paper, *we propose a generic constraint programming framework for the constrained community detection task with the flexibility to be able to capture a wide variety of possible supervision types.*

Constraint programming (CP) [35] is a paradigm for modelling solving combinatorial optimization problems where relations between variables are represented by constraints. One of the strengths of the CP approach is that the constraints can be arbitrary. CP provides state of the art solutions to many industrial scheduling and routing problems, and has been successfully used for constrained data mining problems [18]. Constraint programming modelling technology has the power to express different sorts of logical and mathematical constraints, which provides flexibility in modelling a variety of classic and ad-hoc user defined constraints for constrained community detection. We can use constraint programming modelling technology without committing to a particular solving methodology. If we use complete solving methods, it has the advantage of being able to prove optimality of a solution. But we can also use incomplete solving methods.

Many community detection problems are of large size and cannot currently be solved to optimality using complete methods in reasonable time. However, finding optimal solutions to small constrained community detection problems can give us better insight about the task itself and the characteristics of optimal communities. In addition, there are some small data problems which are naturally very complex and sufficiently important to justify the resources required to find an optimal solution. When dealing with bigger problems, complete solver technology can often quickly find good feasible solutions in a reasonable time, and continue to improve them given more time. Using incomplete solver technology can often generate better solutions in less time, although we give up the possibility of proving optimality. But in any case there is no competing approach we are aware of for tackling complex constrained community detection problems.

A summary of our contributions in this paper are:

- We examine new types of community level constraints based on community definitions and complex logic constraints to dynamically capture properties of interest during the community detection process.
- We show how we can model constrained community detection problems with modularity maximization or other objective functions using constraint programming models. This allows the expression of instance level, community level, definition based constraints and other ad hoc and complex logic constraints.
- We demonstrate via experiments that the framework, using modern complete solving technology, can effectively solve smaller scale, complex problems to optimality. And it is the only approach to solve very complex real world constrained community detection problems.

2 Background

Given a graph $G = (V, E)$ of vertices V and edges E, many community detection algorithms optimize a criterion such as *modularity* [30] to find the best communities. A *community mapping* x on G is a mapping from V to integers. Two vertices v_1 and v_2 are in the same community if $x(v_1) = x(v_2)$.

The modularity value of a partition is given by equation (1) where W is the modularity matrix which quantifies the deviation of the network from randomness: $W_{ij} = (A_{ij} - \frac{d_i \times d_j}{2|E|})$ where A_{ij} is 1 if $(i, j) \in E$ and 0 otherwise and d_i is the degree of node $i \in V$. The modularity of a partition is the summation of the modularities between pairs of the same community.

$$Q = \frac{1}{2|E|} \sum_{i,j} W_{ij}(x(i) = x(j)) \tag{1}$$

It has been shown that finding a partition with maximum modularity is an NP-hard problem [7]. One of the main heuristics for modularity maximization was proposed by Blondel *et al* [6]. It is a greedy hierarchical algorithm which merges communities in each phase to improve the partition's modularity value and continues till no more improvement is possible. Aloise *et al* [3] introduced a column generation based exact method for finding communities by modularity maximization which can solve small size problems to optimality.

Rather than optimizing a criterion, in other schemes, any partition satisfying some conditions is a solution to the community detection task. Two of such conditional definitions were proposed by Raddichi [34], termed "communities in the strong and weak sense." Let $nbh(v) = \{v' \mid (v, v') \in E\}$ be the neighbours of v in G. Given a community mapping x, the *in-degree of a vertex* v, $in(v)$ is the number of neighbours in the same community, i.e. $in(v) = |nbh(v) \cap \{v' \in V | x(v) = x(v')\}|$. Similarly the *out-degree of a vertex* v, $out(v)$ is the number of neighbours in a different community, i.e. $out(v) = |nbh(v) \cap \{v' \in V | x(v) \neq x(v')\}|$. A sub-graph S is a *strong community* if and only if each vertex in S has more in-degree than out-degree:

$$\forall v \in S \quad in(v) > out(v) \tag{2}$$

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 sum of its external degrees:

$$\sum_{v \in S} in(v) > \sum_{v \in S} out(v) \tag{3}$$

Later on, Hu *et al* [19] introduced a comparative definition of community or semi-strong community. Sub-graph S is a *community in the semi-strong sense* if and only if all its vertices have more neighbours within the community than the maximum number of neighbours in any other community, where m is the number of communities.

$$\forall v \in S \quad in(v) > \max_{\substack{t=1,\ldots,m \\ t \neq x(v)}} |nbh(v) \cap \{v' \in V \mid x(v') = t\}| \tag{4}$$

Similar to the above mentioned criteria, Cafieri *et al* [8] defined another relaxed version of strong community called communities in the almost strong sense and they designed a heuristic algorithm based on a set of rules to find such communities in networks. Sub-graph S is a *community in the almost strong sense* if and only if each of its vertices of degree other than two shares more edges within the sub-graph S than with the rest of the network.

$$\begin{aligned} &\forall v \in S || nbh(v)| \neq 2 \qquad in(v) > out(v) \\ &\forall v \in S || nbh(v)| = 2 \qquad in(v) > 0 \end{aligned} \tag{5}$$

Among different supervision types for constrained community detection, size-constrained community detection has been studied by Ciglan and Nørvåg [11]. They proposed a greedy algorithm based on label propagation to find sized constrained communities based on the semi-strong community definition.

Background knowledge can also be represented as known labels and pairwise constraints which model whether a pair of vertices must lie within the same community (*must-link or ML*) or lie within different communities (*cannot-link or CL*). Allahverdyan *et al* [2] studied the problem of community detection in networks where community assignments for a fraction of vertices are known in advance. They designed a so called *planted bisection graph model* and investigated the effect of such supervision scheme on detectability threshold of communities.

Eaton and Mansbach [15] proposed a spin-glass model for incorporating pairwise constraints in a modularity maximization scheme. Their model penalizes partitions violating the guidance by adding/subtracting a fixed term to modularity value of pairs involved in must-link/cannot-link constraints.

Cafieri *et al* [9] extended the column generation model in [3] for modularity maximization to incorporate cohesion constraints as in general, it is recognized that communities found by modularity maximization do not necessarily satisfy variations of strong community conditions (cohesions). However, their column generation algorithm doesn't incorporate other constraint types.

Although there is a lack of flexibility in encoding different supervision types in constrained community detection approaches, there are some studies in constrained clustering schemes to address the flexibility and makes use of complete solving technology. Babaki *et al* [4] incorporated pairwise constraints in an exact column generation scheme for minimum-sum-of-square constrained clustering with pairwise constraints. Berg et al [5] proposed a MAXSAT approach for constrained correlation clustering. Davidson *et al* [12] proposed a SAT based approach and Duong *et al* [13] used constraint programming to encode several instance and cluster level constraints in clustering problems. However, in spite of the high level similarity of clustering and community detection tasks, their optimization criteria are often different ie. modularity vs sum-of-squared-distances. In addition, many constraint types which can be defined in graphs are not applicable in clustering schemes and vice versa, for example, community definition based constraints, such as strong/weak community constraints are only applicable in network community detection. In addition, capturing such complex community level constraints requires a dual view to the partitioning problem which

makes the modelling of constrained community detection different to constrained clustering.

This paper addresses the existing gap in the literature for constrained community detection to propose a flexible and generic CP based framework to handle a variety of constraint types at the same time.

3 Preliminaries

Constraint programming [35] is an effective and generic paradigm to address and solve constraint satisfaction problems (CSP), or constraint optimisation problems (COP). A CSP $P = (X, D, C)$ consists of a set of variables X, a finite *domain* D for each variable $x \in X$ that defines the possible values that it can take, and a set of *constraints* C. A COP is a CSP together with an objective function f which maps a solution θ on X to an objective value $f(\theta)$. The aim is then to find a solution that maximizes (or w.l.o.g. minimizes) the objective function.

The strength of constraint programming arises from the ability to combine arbitrary different constraints in the same model. This naturally gives rise to very expressive modelling. Traditional complete CP solvers are able to tackle this heterogeneous constraint solving problem by using propagators to infer information from individual constraints, communicating through shared variable domains.

Because of this approach the community has developed many *global constraints* which define important combinatorial substructures that reoccur in many problems, and algorithms to propagate them. An example is the constraint `alldifferent`$([x_1, ..., x_n])$ which requires all the variables $x_1, ..., x_n$ to be pairwise distinct. Global constraints have a custom propagator able to exploit the semantics of constraints. This leads to more efficient solving than if one would decompose that constraint as the conjunction of several simple logical or mathematical constraints.

The existence of global constraints further enriches the modelling capabilities for CP, as we try to understand discrete optimization problems as a combination of combinatorial substructures. It is this rich modelling approach that will allow us to express constrained community detection problems succinctly.

We will make use of a few global constraints in capturing community detection problems.

The `global_cardinality_low_up` global constraints is a generalization of `alldifferent` constraint. The `global_cardinality_low_up` $([x_1, \ldots, x_n]$, $[d_1, \ldots, d_m]$,$[l_1, \ldots, l_m]$,$[u_1, \ldots, u_m])$ requires each value d_j is assigned to at least l_j and at most u_j of the variables $x_1, \ldots x_n$ for each $1 \leq j \leq m$.

The `value_precede_chain`$([d_1, \ldots, d_m], [x_1, \ldots, x_n])$ requires d_i precedes d_{i+1} in $[x_1, \ldots, x_n]$ for each $1 \leq i \leq m-1$. This global constraint is very useful in value symmetry breaking of CP models and avoiding multiple symmetric representations of the same solution.

4 Constraint Based Community Detection

We now show how we can model constrained community detection problems using constraint programming. We study four main categories of constraints including instance level, community level, definition based and complex logic constraints which have not been simultaneously applied for constrained community detection before.

The input to our CP model is the network's number of vertices (n) and, if needed, the modularity matrix (or some variations, e.g. the generalized modularity matrix [16]) denoted by W which is used for building the objective function. We also assume a maximum number of communities parameter m, which by default can be n, and a description of the adjacency relation either as an adjacency matrix, A, or the neighbourhood function nbh where $nbh(v)$ are the vertices adjacent to v. Without loss of generality, we assume vertices are indexed from 1 to n and we refer to them by their index.

4.1 Decision Variables:

The critical decisions of the problem are for each vertex which community it belongs to. A one dimensional array x represents the communities to which each (index) vertex belongs. The length of x is equal to the number of vertices n. The domain of each $x[i], 1 \leq \imath \leq n$ is $1..m$ where m is the maximum number of communities possible. While using $|x|$ variables is the most natural way to model the problem (it directly encodes the community mapping) and enables us to use efficient global constraints, it has limitations in modelling some of the community level constraints.

There is a dual viewpoint for the problem. For each community, describe which vertices are contained in that community. We denote this by an array of sets of vertices S indexed from 1 to m. Since we have multiple viewpoints concerning the same decisions, we need to connect them via channeling constraints [10], as follows:

$$\forall j \in 1..m, \quad \forall i \in V \quad i \in S[j] \Leftrightarrow x[i] = j \tag{6}$$

Note that if the dual viewpoint is not needed to express any constraints, then the S variables and the channelling constraints (6) can be omitted.

4.2 Constraints on representation:

In the solution represented by an array x, more than one representation exists for a unique solution. e.g. $x = [1, 1, 2, 2, 2]$ and $x = [2, 2, 1, 1, 1]$. This is called *value symmetry* [33] which can dramatically effect the ability of complete solvers to find solutions, and in particular to prove optimality of solutions. For any solution with k communities there are $k!$ symmetric solutions by permuting the community numbers. To avoid these situations we use the global constraint `value_precede_chain`$([i|i \in 1..m], x)$ which ensures that no community i can

have a vertex j unless all communities $1...i-1$ have at least one lower numbered vertex (less than j) as a member. This constraint enforces a unique community numbering for any particular partition. It can be viewed as a lexicographic ordering constraint on the assignment of vertices to communities. This value symmetry removal is essential for efficiency of the complete solution methods. Note that the addition of the symmetry breaking constraint is typically counter productive for incomplete solving methods, since it creates an artificial constraint that they must satisfy. It is omitted (rewritten away by preprocessing) when we use incomplete solving methods.

4.3 Objective:

The modularity objective is defined as follows:

$$OBJ = \sum_{i,j \in V} (x[i] = x[j]) * W[i,j]$$

One of the advantages of the CP system is the ability to encode logical expressions. The expression $(x[i] = x[j])$ will evaluate to 1 if the expression is true or 0 otherwise.

Although the primary objective function studied in this paper is modularity maximization, the CP framework can encode a variety of other complex arbitrary objective functions. For example, minimizing the differences in sizes of the communities is encoded as minimizing the variable OBJ :

$$OBJ = \max_{t \in 1..m} |S[t]| - \min_{t \in 1..m} |S[t]|$$

4.4 Modelling instance level supervision:

Instance level supervision is usually represented by pairwise must-link and cannot-link constraints which is given to the model by a set of pairs of indices denoted by ML (CL). Similar to [14], the pairwise constraints for constrained community detection then can be encoded as follows:

$$\begin{array}{ll} \forall (m_1, m_2) \in ML, & x[m_1] = x[m_2] \\ \forall (c_1, c_2) \in CL, & x[c_1] \neq x[c_2] \end{array}$$

4.5 Modelling community level supervision:

The CP framework enables incorporating a vast range of community level constraint types. For instance:

- Maximum number of communities: This is implicit in the representation given by the integer m.

– Minimum number of communities: We can enforce that the first l communities are non-empty by simply

$$\forall i \in 1..l, \qquad |S[i]| > 1$$

We can do the same without using the dual viewpoint using the global cardinality constraint

$$\texttt{global\_cardinality\_low\_up}(x, [i|i \in 1..l], [1|i \in 1..l], [n|i \in 1..l])$$

– Minimum and maximum community size:

$$\forall i \in 1..m, \quad (|S[i]| \geq minsize) \wedge (|S[i]| \leq maxsize)$$

Again we can avoid the dual viewpoint using global cardinality constraints. We can also set just a minimum or maximum by using a trivial bound for the other end of the range (0, or n).

– Minimum community size and unknown number of communities: For this combination of constraints we need to make use of the dual viewpoint.

$$\forall i \in 1..m, \quad |S[i]| \geq minsize \vee |S[i]| = 0$$

– Minimum separation between communities: this is defined based on the maximum number of edges between communities which can be set to be less than a predefined threshold T as follows.

$$\forall l, l' \in 1..m \text{ where } l < l', \qquad \sum_{i \in S[l], j \in S[l']} A[i,j] < T$$

– Distribution of different tags in communities. This is an example of user preferences in networks with known tags. For example, consider a scientific collaboration network in which each vertex has a tag: student (S), faculty (F) or research staff (R). The university is interested in the collaboration communities where the ratio of students to faculty is less than p in each group.

$$\forall l \in 1..m, \quad \sum_{i \in S[l]} (i \in S) < p \times \sum_{i \in S[l]} (i \in F)$$

4.6 Modelling definition based constraints

We can encode various community definitions as constraints to the CP model. In this case, the CP model will find the partitions satisfying the local community definitions with the maximum possible modularity (objective) value. This flexibility enables us to benefit from both categories of definitions. Below we model some community definition constraints for our CP framework (recall the definitions from Section 2).

– Communities in the strong sense (equation 2):

$$\forall i \in V, \sum_{j \in nbh(i)} (x[i] = x[j]) > |nbh(i)|/2$$

– Communities in the weak sense (equation 3):

$$\forall t \in 1..m, \quad \sum_{i,j \in S[t]} A[i,j] > \sum_{i \in S[t], j \in nbh(i)} 1 - (j \in S[t])$$

– Communities in the semi-strong sense (equation 4):

$$\forall i \in V, \forall t \in 1..m, \quad t \neq x[i] \quad \rightarrow \sum_{j \in nbh(i)} (x[i] = x[j]) > \sum_{j \in nbh(i)} (j \in S[t])$$

– Communities in the almost-strong sense (equation 5):

$$\begin{aligned}
&\forall i \in V, |nbh(i)| > 2 \quad \rightarrow \quad \sum_{j \in nbh(i)} (x[i] = x[j]) > \sum_{j \in nbh(i)} (x[i] \neq x[j]) \\
&\forall i \in V, |nbh(i)| = 2 \quad \rightarrow \quad \sum_{j \in nbh[i]} (x[i] = x[j]) > 0 \\
&\forall i \in V, |nbh(i)| = 1 \quad \rightarrow \quad \forall j \in nbh(i).x[i] = x[j]
\end{aligned}$$

Note that we treat the case of $|nbh(i)| = 1$ specially since we can enforce that the unique neighbour is in the same community.

The CP framework can provide further flexibility. Using implication one can require a proportion of the network to follow a community definition based on other constraints.

4.7 Modelling complex logic constraints

Unlike any other existing approaches for semi-supervised community detection, CP modelling technology can enable encoding complex logic supervision such as conjunction, disjunctions, negation and implication of any instance level, community level and definition based constraints. For example, the constraint "when instance i belongs to a community, the size of that community should be bounded by α and β" can be modelled as follows.

$$\forall t \in 1..m, \quad (i \in S[t]) \rightarrow (|S[t]| < \beta) \wedge (|S[t]| > \alpha)$$

The above constraint is just an illustration that any logic constraint can be captured by a CP modelling framework. In practice, complex logic constraints may arise in different applications in real world problems. For example, in power grid networks, complex requirements may have to be imposed to implement strategies for network reliability and improving the network behaviour in cascading events [32].

5 Experimental Results

In this section, we present experiments on real and benchmark data sets to evaluate the performance and flexibility of the proposed framework. The constrained community detection problems are written in Minizinc 2.0.12 [28]. All the experiments are performed using the Gecode [37] or OSCAR CBLS [31] CP solver with a timeout of one hour on a Macbook with 8GB RAM and 2.7 GHz Intel Core i5.

The main questions we address in this section are:
Q1: How the solving use CP modelling compares with other constrained community detection methods?
Q2: Can the quality of the solutions be improved by adding community level supervision to instance level constraints?
Q3: Can the modelling framework enable us to simultaneously encode different constraint types on a complex real problem?
Q4: How scalable can approaches based on constraint programming modelling be?

5.1 Comparison to other methods

The proposed modelling framework can encode a variety of objectives as well as classic and arbitrarily complex instance and community level constraints at the same time while there is no other approach in the literature with such ability for constrained community detection. However, to address the question Q1 and compare the proposed modelling framework to an state-of-the-art algorithm in constrained community detection, we limit the supervision type to only ML and CL pairwise constraints and set the objective to modularity maximization. There exist some approximate approaches for incorporating pairwise constraints in modularity optimization scheme [24, 39, 15]. Here we compare the proposed framework with the spin-glass model [15] because it is based on modularity and it is shown to perform better than some other approaches [15]. To implement this approach, we set the parameters according to [15] and used the GenLouvain algorithm [20] for optimizing the spin-glass model.

Since we have the ground truth of the data sets, for evaluating quality of partitioning, we use the Normalized Mutual Information (NMI) measure (equation 7) proposed by Danon *et al* [22].

$$I_{norm}(A,B) = \frac{-2\sum_{i=1}^{CA}\sum_{j=1}^{CB} N_{ij}\log(N_{ij}N/N_{i.}N_{.j})}{\sum_{i=1}^{CA} N_{i.}\log(N_{i.}/N) + \sum_{j=1}^{CB} N_{.j}\log(N_{.j}/N)} \tag{7}$$

In equation (7), A represents the real communities and B represents the detected communities while CA and CB are the number of communities in A and B respectively. In this formula, N is the confusion matrix with rows representing the original communities and columns representing the detected communities. The value of N_{ij} is the number of common vertices that are in the

original community i but found in community j. The sum over the ith row is denoted by $N_{i.}$ and the sum over the jth column is denoted by $N_{.j}$

For each data set listed in Table 1, in the second column section we give the size, number of constraints, the ground truth number of communities k and maximum number of communities use in the CP model m. For each data set, we generated 5 different sets of random constraints (equally divided to ML and CL) based on the ground truth. To have more rigorous comparison, we executed the spin-glass model 50 times on each data set and reported the NMI correspond to the best solution (the highest number of constraints satisfied and then highest modularity score) in the third column section of Table 1. The average NMI and runtime of the complete solver Gecode on the model are reported in the fourth column of Table 1. In the model, the maximum number of communities m are set according to the solution of the corresponding unconstrained modularity maximization problem. The P-value corresponding to Friedman statistical test is reported in the last line of Table 1 . The null hypothesis of this test is two algorithms have no significant difference in their performance. This hypothesis is rejected based on the very small p-values, indicating that Gecode applied to the model statistically significantly outperforms the spin-glass method in solution quality.

Using Gecode on the CP model finds high quality solutions while it is often very fast as well. In addition, Gecode can prove optimality in reasonable time for solutions of smaller problems. For bigger problems sizes such as Political blogs, Gecode could not prove optimality within one hour. Still Gecode could find a better solution than the spin-glass solution in 17 seconds and it kept improving the solutions so that it achieved a much higher quality solution than the spin-glass method within the timeout. This shows the ability of the CP modelling framework to produce promising solutions even in big problem instances in reasonable time where proving optimality is not possible.

Table 1. Comparison to other methods and effect of adding community level constraints

					Spin-glass(Pairwise)		CP(Pairwise)		CP(Pairwise+community-level)		
Data	n	#const	k	m	NMI	time	NMI	time	NMI	time	supervision type
Sampson [36]	25	24	2	4	0.82	<1	0.88	<1	0.96	<1	weak
Strike [26]	24	24	3	5	0.72	<1	0.78	<1	0.81	<1	# of community
Zachary [38]	34	34	2	4	0.85	<1	0.87	<1	0.9	<1	weak
Mexican [17]	35	34	2	4	0.32	<1	0.45	60	0.53	35	weak
Dolphin [23]	62	124	2	5	0.81	<1	0.95	<1	0.98	<1	almost strong
Adjacent Words [29]	112	224	2	4	0.05	<1	0.52	73	0.8	1.7	cardinality
Political books[21]	105	210	3	5	0.67	<1	0.94	15	0.95	2.5	cardinality
Political blogs[1]	1490	2980	2	4	0.59	1.2	0.88	3600(17)	0.97	142(14)	# of community
p-value					0.0047		baseline		0.0047		

5.2 Effect of community level supervision

To address the second question, we add community level constraints to pairwise ML and CL constraints for the data sets of Table 1 agreeing with their ground

truth. The results of adding community level constraints including communities in weak and almost-strong sense (equations 3 and 5), number of communities and size (cardinality) constraints are shown in the last column section of Table 1. For the cardinality constraint, we generated 10 sets of random samples from the ground truth and set the minimum and maximum size of the communities according to the observed number of samples from each community.

Comparing the last two column sections of Table 1 shows that adding community level supervision can lead the community detection process to more accurate solutions, while often enhancing the runtime of the optimization algorithm. For instance, in the Political Blogs dataset, adding the number of community constraints leads to a significant improvement in runtime and solution quality comparing to considering just pairwise constraints. The significancy of the results are verified using Friedman statistical test and the very low p-value.

5.3 Case study

To address question Q3, we consider contacts and friendship relations between students in a high school in Marseilles, France, in December 2013. Students were asked to record their contacts with other students in a diary and also list their friends at school. The Facebook network is available for a subset of these students. Gender, student ID and the teaching class of each student is also reported. We consider the network of 81 students whose Facebook and friendship information is known [25].

We consider a hypothetical scenario in which the School Principal wants to group the students based on their Facebook communications network, while at the same time, desiring certain properties to hold based on students' friendships and contacts information. For example, students who had more than 1 hour contact during the data collection period, must be assigned to the same community and students who declared friendship and had contact during the data collection period also must be in the same community. These kinds of properties can be captured by a set of must-link constraints. 50 must-link constraint were extracted based on the above mentioned requirements on individual student assignments. For balancing group populations, suppose the School Principal wants each community to have 15-25 students and having 4 groups of students is desired. The School Principal also wants to balance the gender distribution in groups by requiring the difference in number of male and female students to be less than or equal to 5. To make students feel more comfortable, it is also required that each student has at least two other students from their original class for the new group they are assigned into.

There is no existing community detection algorithm capable of automatically incorporating these complex instance and community level constraints at the same time. For example, the spin-glass model [15] can only incorporate must-link and cannot-link constraints. The best solution found by the spin-glass model from 1000 executions is shown in first row section of Table 2 satisfying all the must-link constraints. The size column shows the cardinality of each community

Table 2. Community profiling of the case study

methods	size	gender	class distribution
spin glass	39	F=29, M=10	4,7,25,2,1
	38	F=15, M=23	3,1,20,5,9
	4	F=3, M=1	4
CP just ML	40	F=23, M=17	3,2,12,18,5
	30	F=20, M=10	1,5,13,3,6,1,1
	11	F=4, M=7	2,1,8
CP final	25	F=12, M=13	3,17,5
	25	F=15, M=10	4,4,14,3
	15	F=10, M=5	3,3,9
	16	F=10, M=6	3,5,5,3

and the next column shows the number of females and males in each community respectively. The class distribution column shows the number of classmates based on original class labels for each of the new communities. Based on this information, it is clear that the solution found by the spin-glass model violates all the other constraints.

Real world problems such as this example often include a large number of varying requirements on communities which often cannot be satisfied by existing approaches. However, our modelling framework can easily deal with various complex constraints at the same time. Within the timeout of one hour, a solution of the model just considering must-link constraints (to compare with the spin-glass solution), using the complete solver Gecode, and a full solution also incorporating size, number of community, gender and class distribution constraints are shown in the second and third rows of Table 2. As shown in Table 2, the solution found using Gecode satisfies all the constraints required by the School Principal.

5.4 Scalability

One of the advantages of using a solver-independent modelling framework is that we do not commit to a particular solving technology. While complete solving methods are effective on constrained supervision problems which are not too large, by their nature they do not scale as well as incomplete methods. We can use an incomplete solver to tackle the same model. Indeed since we use MiniZinc [28] we can send the same model to both solvers.[1] Incomplete solvers are typically more scalable, but may struggle to satisfy all the constraints of the problems.

We consider solving the problem using the Oscar CBLS solver [31]. On the smaller examples of Table 1 Gecode is uniformly better than Oscar, in both time to solve and NMI of resulting solution, but as the size of the graphs grow Oscar becomes quicker to find solutions.

[1] Note that we also tried running MIP solvers on the models, but they were non-competitive, which is unsurprising since the linear relaxation of these problems is very weak.

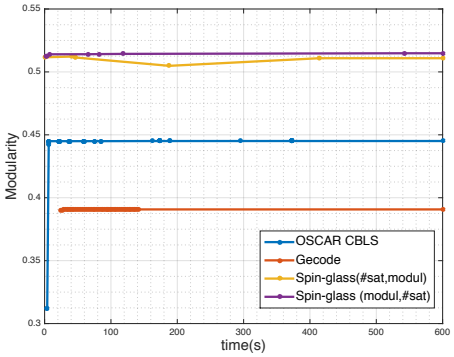

Fig. 2. Modularity over time for Political Blogs data set using Gecode, Oscar and spin-glass methods

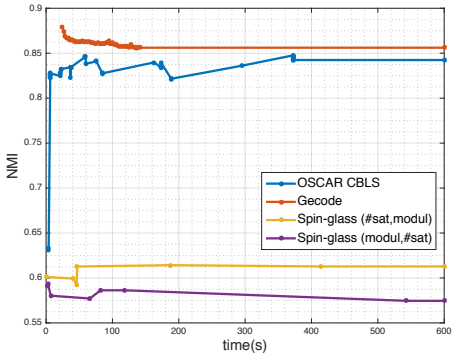

Fig. 3. NMI over time for Political Blogs data set using Gecode, Oscar and spin-glass methods

Figure 2 shows the best modularity value of the solution over time using Gecode, Oscar and the spin-glass on the Political Blogs example. The spin-glass method is repeatedly run keeping either the lexicographic best solution (modularity, number of constraints satisfied) or the lexicographic best solution in the other order. Clearly the spin-glass concentrates on modularity maximization, and never satisfies all constraints (never finds a feasible solution), and reaches much higher modularity values.

Figure 3 shows the plot of the same solutions, but here ranked on NMI value. Clearly the spin glass method never achieves a good NMI, since building solutions that satisfy more constraints is preferable for NMI. Oscar quickly gets a good solution to the problem with high NMI and gradually improves. Interesting Gecode actually finds the best solution in terms of NMI as its first solution, then gradually degrades as it optimizes the modularity objective. The first solution found by Gecode requires 24 seconds, while the first solution found by Oscar requires only 3.5 seconds.

6 Conclusion

The challenging problem of constrained community detection with a variety of constraint types and objective functions has been explored in this paper. We proposed a generic framework based on constraint programming modelling approach, which enables including a variety of instance and community level, definition based and complex logic supervision types as constraints. Our models are able to prove optimality of the solutions, when using complete solving methods, and in our experiments we have shown it can work with real networks and complex problems.

An obvious direction for future work is to consider specialized propagators for community definitions. For examples, a global propagator constraining that all communities are strong, may well be able to reason about community information stronger than the vertex by vertex definition.

References

1. Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proc. of Link discovery*, pages 36–43. ACM, 2005.
2. Armen E Allahverdyan, Greg Ver Steeg, and Aram Galstyan. Community detection with and without prior information. *Europhysics Letter*, 90(1), 2010.
3. Daniel Aloise, Sonia Cafieri, Gilles Caporossi, Pierre Hansen, Sylvain Perron, and Leo Liberti. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4), 2010.
4. Behrouz Babaki, Tias Guns, and Siegfried Nijssen. Constrained clustering using column generation. In *CPAIOR*, pages 438–454. Springer, 2014.
5. Jeremias Berg and Matti Järvisalo. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artificial Intelligence*, 2015.
6. Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *JSTAT*, 2008(10):P10008, 2008.
7. Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Gorke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. On modularity clustering. *IEEE Tr. Know. Disc. Eng*, 20(2):172–188, 2008.
8. Sonia Cafieri, Gilles Caporossi, Pierre Hansen, Sylvain Perron, and Alberto Costa. Finding communities in networks in the strong and almost-strong sense. *Physical Review E*, 85(4):046113, 2012.
9. Sonia Cafieri, Alberto Costa, and Pierre Hansen. Adding cohesion constraints to models for modularity maximization in networks. *Journal of Complex Networks*, 3(3):388–410, 2015.
10. C.W. Choi, J.H.M. Lee, and P.J. Stuckey. Removing propagation redundant constraints in redundant modeling. *ACM Tr. on Comp. Logic*, 8(4), 2007.
11. Marek Ciglan and Kjetil Nørvåg. Fast detection of size-constrained communities in large networks. In *WISE*, pages 91–104. Springer, 2010.
12. Ian Davidson, SS Ravi, and Leonid Shamis. A sat-based framework for efficient constrained clustering. In *SIAM Data Mining*, pages 94–105, 2010.
13. Khanh-Chuong Duong, Christel Vrain, et al. A declarative framework for constrained clustering. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 419–434. Springer, 2013.
14. Khanh-Chuong Duong, Christel Vrain, et al. Constrained clustering by constraint programming. *Artificial Intelligence*, 2015.
15. Eric Eaton and Rachael Mansbach. A spin-glass model for semi-supervised community detection. In *AAAI*. Citeseer, 2012.
16. Mohadeseh Ganji, Abbas Seifi, Hosein Alizadeh, James Bailey, and Peter J Stuckey. Generalized modularity for community detection. In *ECML-PKDD*, pages 655–670. Springer, 2015.
17. Jorge Gil-Mendieta and Samuel Schmidt. The political network in mexico. *Social Networks*, 18(4), 1996.
18. Tias Guns, Anton Dries, Siegfried Nijssen, Guido Tack, and Luc De Raedt. Miningzinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, 244:6 – 29, 2017.
19. Yanqing Hu, Hongbin Chen, Peng Zhang, Menghui Li, Zengru Di, and Ying Fan. Comparative definition of community and corresponding identifying algorithm. *Physical Review E*, 78(2):026121, 2008.

20. Inderjit S. Jutla, Lucas G. S. Jeub, and Peter J. Much. A generalized louvain method for community detection implemented in matlab. *http://netwiki.amath.unc.edu/GenLouvain*, 2012.
21. V. Krebs. *www.orgnet.com/*.
22. Albert Daz-Guilera Leon Danon and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *JSTAT*, page P11010, 2006.
23. David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.
24. Xiaoke Ma, Lin Gao, Xuerong Yong, and Lidong Fu. Semi-supervised clustering algorithm for community structure detection in complex networks. *Physica A: Statistical Mechanics and its Applications*, 389(1):187–197, 2010.
25. Rossana Mastrandrea, Julie Fournet, and Alain Barrat. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one*, 10(9):e0136497, 2015.
26. Judd H Michael. Labor dispute reconciliation in a forest products manufacturing facility. *Forest products journal*, 47(11/12):41, 1997.
27. Benjamin Negrevergne and Tias Guns. Constraint-based sequence mining using constraint programming. In *CPAIOR*, pages 288–305. 2015.
28. N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, and G. Tack. Minizinc: Towards a standard CP modelling language. In *Proc. of PPCP*, volume 4741, pages 529–543, 2007.
29. Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 74(3):036104, 2006.
30. Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2), 2004.
31. OscaR Team. OscaR: Scala in OR, 2012. Available from `https://bitbucket.org/oscarlib/oscar`.
32. Sakshi Pahwa, Amelia Hodges, Caterina Scoglio, and Sean Wood. Topological analysis of the power grid and mitigation strategies against cascading failures. In *Systems Conference, 2010 4th Annual IEEE*, pages 272–276. IEEE, 2010.
33. Jean-Francois Puget. Symmetry breaking revisited. In *Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, volume 2470 of *LNCS*, pages 446–461. Springer, 2002.
34. Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proc. of the Nat. Acad. of Sci.*, 101(9):2658–2663, 2004.
35. Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of CP*. Elsevier, 2006.
36. Samuel Franklin Sampson. A novitiate in a period of change: An experimental and case study of social relationships. *Cornell University*, 1968.
37. Christian Schulte et al. Gecode, 2016. http://www.gecode.org/.
38. Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.
39. Zhong-Yuan Zhang. Community structure detection in complex networks with partial background information. *EPL (Europhysics Letters)*, 101(4):48005, 2013.