Image Constrained Blockmodelling: A Constraint Programming Approach

Mohadeseh Ganji[1], Jeffrey Chan[2], Peter. J. Stuckey[1], James Bailey[1], Christopher Leckie[1], Kotagiri Ramamohanarao [1], Ian Davidson[3]
[1]School of Computing and Information Systems, The University of Melbourne, Australia
[2]School of Computer Science and Software Engineering RMIT University, Australia
[3]Computer Science Department, University of California at Davis, USA
[1] {sghasempour, pstuckey, baileyj, caleckie, kotagiri}@unimelb.edu.au,
[2]jeffrey.chan@rmit.edu.au, [3]davidson@cs.ucdavis.edu

Abstract

Blockmodelling is an important technique for detecting underlying patterns in graphs. However, existing blockmodelling algorithms do not provide the user with any explicit control to specify which patterns might be of interest. Furthermore, existing algorithms focus on finding standard community structures in graphs, and are likely to overlook informative but more complex patterns, such as hierarchical or ring blockmodel structures. In this paper, we propose a generic constraint programming framework for blockmodelling, which allows a user to specify and search for complex blockmodel patterns in graphs. Our proposed framework can be incorporated into existing iterative blockmodelling algorithms, operating as a hybrid optimization scheme that provides high flexibility and expressiveness. We demonstrate the power of our framework for discovering complex patterns, via experiments over a range of synthetic and real data sets.

Keywords: Blockmodelling, Constraint Programming.

1 Introduction

Networks are an important and challenging form of data, and discovering the latent structure of networks can reveal interesting system properties. Network structures have been extensively studied in the context of community detection and there are many algorithms for finding communities in graphs [5, 11]. A common definition of a community is a group of vertices that are densely connected among themselves and sparsely connected to the rest of the network.

However, communities are not the only possible type of latent structure in networks. Indeed, some networks may contain several valid structures. For instance, consider the well studied karate club network [27], which is a social network representing associations between members of a university-based karate club. After a conflict arising between the instructor and the administrative officer, the club was split into two separate communities shown in Figure 1 (left). However, further analyzing the karate club network, one can uncover that there is another dominant pattern in the graph (Figure 1 right), where some club members try to mediate between the separate groups and interact strongly with both the instructor and the administrative officers, as well as having substantial interactions amongst themselves. This alternative structure (which is called a core-periphery structure) is not obvious from a standard community detection viewpoint and is unlikely to be found by community detection algorithms. This motivates a more general approach, in order to uncover such patterns in graphs.

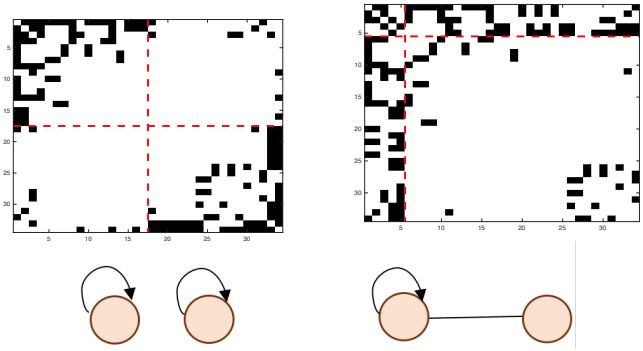


Figure 1: The Karate club network: community structure and its image diagram (left), core-periphery structure and its image diagram (right)

Blockmodelling is a powerful approach that partitions graphs to groups of vertices that play the same role in the graph [26]. Vertices playing the same role (equivalent vertices), have similar connections to other vertices in the network. For example, vertices in a community structure have more connections among themselves and fewer connections between two communities. The core-periphery structure of the karate club network also fits this definition, as the intermediary members have simi-

lar connections. Hence, blockmodelling is a general approach to discover different types of graph structures in addition to communities.

However, existing blockmodelling algorithms are not able to target a specific structure to search for. Existing blockmodelling algorithms would be expected to find the community structure of a graph (e.g karate club) and others may find other structures but it often requires human interpretation to spot the less obvious structure (e.g. the core-periphery in karate club). However, none of the existing blockmodelling algorithms has the flexibility to search for the existence of a specified structure of interest within a graph. Note that by "search", we do not mean simple querying of the graph, but rather identification of communities with an expected pattern of connections between them.

1.1 Motivation for complex blockmodel structure search There are several motivations for discovery of complex blockmodel structures:

Firstly, as the Karate network example illustrates, a graph may contain multiple valid structures and groupings. Similarly, it has been shown that in many real networks where communities are a primary structure within the graph, there exist other interesting patterns that can reveal behavioral characteristics of the system, such as hierarchy structures in biological networks [4] and core-periphery structures in scientific collaboration networks [22].

Secondly, rather than completely unsupervised structure discovery, there may be some pre-existing knowledge available about expected patterns in the graph. Incorporating such background knowledge in the blockmodelling process can improve the performance and quality of the solutions.

Thirdly, consider scenarios where one is interested to evaluate whether a given graph fits a desired functional structure. E.g. in a gene regulatory network it could be reasonable to have a hierarchy structure. If an analyst has a tool that allows him to find the most tree-like structure in the network, then he can evaluate how different that structure is from a perfect hierarchy by counting the number of network edits (e.g. edge additions or deletions) needed. Alternatively for an employee interaction network in a company, measuring deviations in current communication flow from an ideal functional desired pattern may help in diagnosing interaction bottlenecks.

1.2 Challenges, our approach and contributions Whilst one can attempt to design specialised algorithms that can discover specific structures, it is considerably more challenging to formulate a framework that allows discovery of arbitrary structures. We address this gap by proposing a constraint programming based framework for targeted blockmodelling.

Constraint programming (CP) [23] is a constraint solving paradigm for solving combinatorial optimization problems which has provided high performance solutions for difficult industrial scheduling and routing problems, and has been successfully used for constrained data mining problems. The power of constraint programming to express complex logical and mathematical constraints makes it an ideal technology for our targeted blockmodelling framework.

Solving network community detection problems for large size graphs can be slow using standard constraint programming [13]. However, we show that it is possible to model the targeted blockmodelling problem with constraints in a way that allows solutions to be discovered in comparable time to existing blockmodelling algorithms. In our framework, the problem instances solved by the CP solver are of the order of the number of blocks and are independent of the number of network vertices or edges. Since the number of blocks (clusters) is not naturally very big for many applications, scalability for the CP part of the framework is achievable.

The contributions of this paper are as follows:

- We study complex interactions between groups of vertices/blocks and propose a generic and interpretable constraint programming approach to targeted discovery of complex structures in graphs.
- Our proposed method allows incorporation of background and expert knowledge for finding interesting patterns in graphs according to different application requirements and provides the user with flexibility to search for specific patterns and model complex block-level constraints.
- Our proposed CP approach can be integrated with a range of existing blockmodelling algorithms and We demonstrate the flexibility and high accuracy of the CP model using synthetic and real data sets.

2 Background

In this section we formally explain the blockmodelling problem and constraint programming, modelling and solving technology.

2.1 Blockmodelling: Consider a graph $G(V, E)$ where V is a set of vertices and edges are represented by E. The graph can be represented by its adjacency matrix, denoted by A which for each pair of vertices i and j, A_{ij} indicates whether or not an edge exists between i and j (or from i to j in a directed graph).

Blockmodelling aims to decompose the graph into groups of vertices (called positions) with similar relations. This can be represented by a membership matrix $C \in [0,1]^{n\times k}$ and an image matrix $M \in [0,1]^{k\times k}$ where n is the number of graph vertices and k is the number of blocks or positions. While the membership matrix shows the assignment of vertices to blocks, the image matrix summarizes the communications within and between blocks. The inherent graph structure can be identified by visualizing the image matrix. The positions and the image matrix together form a blockmodel. The blockmodel decomposition aims to approximate the graph adjacency matrix, A as CMC^T. Blockmodelling is then the task of finding $M \geqslant 0$ and $C \geqslant 0$ that minimize the approximation error, captured by sum of squared differences (Equation 2.1).

$$min_{M,C}||A - CMC^T||^2 \tag{2.1}$$

In this paper, we focus on structural equivalence [26], according to which, two vertices are in the same position if they have similar sets of out and in neighbours. In terms of the image matrix, this means the densities of the entries are ideally close to 0 or 1.

2.2 Constraint Programming: Constraint programming is a optimization technology which enables high level modelling of complex problems. In CP, the relations between variables are modeled using constraints and a feasible solution is an assignment to all the variables that satisfies the constraints.

Constraint programming solvers find solutions by repeatedly reducing the possible values variables can take using systematic search (in the solution space) and inference (propagating the information contained in one constraint to the related constraints). Constraint solving can benefit from a *global constraint* which is a constraint that captures common substructures of the problem and simplifies the modelling effort. 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 performance than if one decomposes the constraint as the conjunction of several simple logical or mathematical constraints.

3 Related Work

3.1 Blockmodelling: It has been shown that blockmodelling is an NP-hard problem [10]. Therefore, blockmodelling algorithms try to find a good local optimum to this problem. Blockmodelling has been formulated as a nonnegative matrix tri-factorization problem [18, 25, 6]. Zhang et al. [28] proposed a coordinate descent optimization for overlapping blockmodelling. Karrer and Newman [16] proposed a stochastic blockmodelling approach considering heterogeneity in the degrees of vertices. Chan et al. [6] proposed a framework for sparse and noisy graphs. They also proposed objective functions and an incremental approach for optimizing the membership matrix, which only updates the necessary entries of the C matrix each time.

Reichardt et al. [21] used a simulated annealing approach to optimize an objective function based on the difference between the adjacency matrix and its blockmodel approximation.

To the best of our knowledge, none of the existing blockmodelling techniques can automatically find a desired structure in a graph, or provide the flexibility to search for complex application-specific patterns.

In some existing algorithms,[21, 6] one can initialize the image matrix with an arbitrary matrix of a desired structure, but there is no guarantee that these algorithms eventually find the specified structure and even less guarantee that they will find the desired structure that best summarizes the graph.

If one is interested just in a core-periphery structure, some approaches exist to determine which nodes are part of a densely connected core and which are part of a sparsely connected periphery [22]. The core-score of Rombach et al. [22], assigns a score to each node of the graph to represent the extent to which they are qualified as a core based on the notion of network centrality. Then, one can find the core group by setting a core size limit or observing a drop in core score of the nodes ranked by their score. However, these algorithms are designed to just find core-periphery structures and one cannot not use them to find other structures in the network, even communities. Another approach suggested in [22] is to find communities using a community detection algorithm and then assign a core score to each community. However, community and core-periphery structures are not necessarily related to each other, since core vertices might belong to different communities (this is the case in the karate club network). *Although such specialized approaches have been discussed for core-periphery structure, to the best of our knowledge there are no generic framework which can also find other structures, such as hierarchies, rings or stars.*

3.2 Constraint Programming: The application of constraint solving in data mining and machine learning problems is of growing interest. It allows the capture of complex constraints on machine learning problems and the ability to benefit from existing side information and domain knowledge. Incorporating domain knowledge in unsupervised learning tasks have been studied for

itemset and pattern mining [19, 15], clustering [9, 7, 12] and community detection [13, 14]. This can result in more efficient and actionable solutions.

4 Proposed CP-based framework

Blockmodelling algorithms attempt to find good M and C matrices to minimize Equation 2.1. As it is known that blockmodelling of three or more positions is NP-hard [10], most blockmodelling algorithms are incomplete, and do not guarantee to find the best M and C. One common approach is to iteratively solve the optimization problem for C given that M is fixed and then solve the optimization problem for M given that C is fixed, and alternate until the change in membership or image matrices is very small, e.g. the algorithm converges.

For optimizing the membership matrix C, we can use coordinate descent [28] or projected gradient descent [3] approaches. If hard membership is desired, we can use the incremental approach of [6] which only updates the necessary entries of the C matrix and hence is more efficient than recomputing the entire objective value for each single vertex and position. Coordinate descent [28] or projected gradient descent [3] approaches can also be used for optimizing the image matrix M. However, one cannot specify a target structure in these approaches

The focus of this paper is to optimize the image matrix, M, given some constraints on the type of structure we are looking for. Note that our proposed CP based optimization of image matrix is independent of the optimization approach used for the membership matrix. Hence, our approach can be coupled with any of the optimization approaches for membership matrices mentioned earlier.

We can consider the image matrix as a graph whose vertices are blocks and edges are the number (or probability) of shared edges between or within blocks. The image matrix, similar to the adjacency matrix, represents the relations between blocks/positions. Then, the complex structure and relations between blocks can be seen as image graph properties. The problem of (constrained) image matrix optimization is then as follows: given an existing image matrix (based on the current membership values), what is the best pre-specified structure that most accurately summarizes the original graph (A). The structure might simply be connectedness, a ring, a tree or a core-periphery structure.

The large variety of possible image graph structures makes it difficult for common mathematical modelling and optimization approaches to handle these constraints. However, constraint programming is ideal for modelling a wide variety of constraints (mathematical and logical) and defining specialized propagators for capturing complex structures. Indeed there are many specialized propagators for finding complex graph (sub)-structures [1]. Hence it is an ideal technology for solving the constrained image matrix optimization problem.

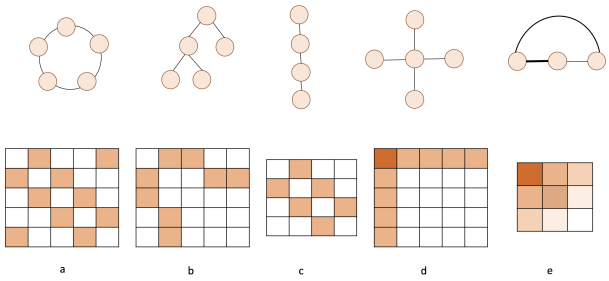


Figure 2: Some examples of image graph structures: a) ring, b) hierarchy, c) stick, d) star, e) core-periphery

The input to the CP model is the image matrix according to the current membership values. The current image matrix, is calculated as $C^{-1}AC^{-1^T}$ where C^{-1} is pseudo inverse of C. Given the current image matrix and the desired structure, the CP model finds the specified structure that best summarizes the graph.

4.1 Modelling complex structures

In the following we explain the modelling of the objective functions and constraints for a variety of graph structures.

Hierarchy (tree, stick): A simple and yet useful structure is where blocks communicate to each other in a chain or stick structure, where one block is related to a second one, and the second block to a third one and so on. This could be a directed communication or an undirected one. Relations between groups of similar vertices in real networks such as gene regulatory networks have been shown to reflect hierarchical and stick patterns [4]. To model this structure in CP, we define a vector variable denoted by x which is an array of k elements representing the order of blocks in the stick. As there is no loop in stick structure, we use the global constraint `alldifferent`(x) to ensure no value is repeated in $x[1], ..., x[n]$ and this is the only constraint needed to capture stick structure. The best stick then can be found by maximizing the objective function 4.2 which sums up the image matrix values along the stick.

$$\text{(4.2)} \qquad maximize \quad \sum_{i=1}^{k-1} M(x[i], x[i+1])$$

To capture a more complex hierarchical structure in the network we use a global tree constraint. Decision variables are *root* (an integer variable), *nodes* (an array of k Boolean variables) and *edges* (an array of $k \times k$ Boolean variables) connecting nodes of the tree. We also need two fixed arrays *from* and *to* which define each edge. In our case since the graph is complete,

they are simply $k \times k$ arrays defining the complete graph, i.e. $from[i] = (i-1) \div k + 1$ and $to[i] = (i-1) \mod k + 1$. Having defined these variables and parameters, we use the global constraint $\texttt{dtree}(k, k*k, from, to, root, node, edge)$ [2] to find a tree of k nodes among the $k*k$ possible edges. A solution will show the *root* of the tree, the nodes in the tree (i where $node[i]$ is true) and the edges ((i,j) where $edge[(i-1)*k+j]$ is true). The objective function 4.3 for the CP model looks for the tree with highest summation of image values among the blocks in the tree structure, such that the edges form a directed tree.

$$maximize \quad \sum_{l=1}^{k \times k} edge[l] \times M(from[l], to[l]) \tag{4.3}$$

Note that other global constraints can find forests (sets of unconnected trees) in the graph (set of trees).

Core-periphery: In core-periphery structure, some nodes belong to a densely connected core and others are part of a sparsely connected periphery. Core nodes should also be reasonably well-connected to peripheral nodes, but the latter are not often well-connected to each other. Star is a basic core-periphery in which the core is connected to all periphery blocks but the periphery blocks are not connected to each other. Then, the problem is to find the core block (integer variable x) that has the highest connection to the other blocks. This can be done simply using the objective function 4.4

$$maximize \quad M(x,x) + \sum_{i=1}^{k} M(x,i) + M(i,x) \tag{4.4}$$

Core-periphery structure can be more complex than the star structure by having multiple layers where the boundary between core and periphery is not clear. In this structure, the connections gradually decreases from the most inner core to the outer periphery layers.

To find complex core-periphery structures, we use an array of size k denoted by x as the variable which ranks the blocks from the innermost core ($x[1]$) to the outermost periphery ($x[k]$). As each block is either a core or periphery we ensure it is not repeated in the decision variable x using the constraint $\texttt{alldifferent}(x)$. Then the objective function can be defined as equation 4.5 in which w_1 and w_2 specify the relative importance of interconnections and connections to others.

(4.5)

$$maximize \quad \sum_{i=1}^{k} \left(\frac{1}{w_1 * i}\right) * [M(x[i], x[i]) + \sum_{j=i+1}^{k} \left(\frac{1}{w_2 * j}\right) * (M(x[i], x[j]) + M(x[j], x[i]))]$$

Ring: A ring/circuit structure occurs when groups of vertices communicate to each other (similar to stick structure) but the last block in the chain, also communicates to the first block. As an example consider the internal communications in a market research company for a project. Once the consultant understands the client's problem, he communicates to a team of questionnaire designers to design appropriate questions to gather data (there might be interactions with analysts to help design the questions too). Once the questionnaires are designed and the data are gathered, then data analysis and modelling are done by analysts and at the end the results are communicated to the consultant again to get feedback or present to the client. So the team of consultants, questionnaire designers and analysts communicate in a ring structure. To model this structure, we define the variable x as an array of size k, for which each block has a successor block.

In order to look for a ring structure including all blocks, one could use the global constraint $\texttt{circuit}(x)$, which constrains the elements of array x to define a circuit where $x[i] = j$ means that j is the successor of i. However, because it may not be always the case that all the blocks participate in the ring structure, we use another global constraint called $\texttt{subcircuit}(x)$ to force the elements of array x to form a sub-circuit where $x[i] = j, i \neq j$ means that j is the successor of i and $x[i] = i$ means that i is not in the circuit. This global constraint makes it possible to find ring substructures where not all blocks are included. For instance, $x = [2,3,1,4]$, means the optimized ring is $1-2-3-1$ and position 4 does not participate in the ring structure.

Finally, the objective function of the CP model is shown in equation 4.6. Unlike typical mathematical optimization models, CP can handle logical constraints. The term $x[i] \neq i$ evaluates to zero if $x[i] = i$ and one otherwise.

$$maximize \sum_{i=1}^{k} (x[i] \neq i) M(i, x[i]) \tag{4.6}$$

Connected: Another property in graphs which can be reflected in block relationships connectivity indicates there is no isolated groups of vertices in the graph and all positions are communicating at least with one other position. This can be modelled using the global constraint $\texttt{connected}(x)$ [8] where x is an array of size k representing the adjacent blocks to each block. The connected constraint then selects a subset of arcs of the graph so that the corresponding graph is symmetric) and connected (i.e., there is a path between any pair of vertices of the graph).

The objective function 4.7 then ensures to find the

best connected subset of the graph simply maximizes the weight of the arcs, where $j \in x[i]$ evaluates to 1 if j is in the set $x[i]$ and 0 otherwise.

$$maximizesum_{i,j\in 1..k}(j \in x[i])M(i,j) \tag{4.7}$$

Transitivity: A graph is transitive if for all blocks i, j, l if i and j are related and j and l are related, then i and l also communicate. Let $x[i, j]$ be a Boolean representing that the edge (i,j) is in the adjacency matrix. We can enforce transitivity by the constraint

$$\forall i, j, l \in 1..k.x[i, j] \wedge x[j, l] \rightarrow x[i, l] \tag{4.8}$$

We can look for the best transitive graph explaining the block model using the objective

$$maximize \quad \sum_{i,j\in 1..k} x[i, j] * M(i, j) \tag{4.9}$$

Arbitrary constraints: Constraint programming can capture any arbitrary constraints on blocks as well. For instance, if a relationship is known or desired for a subset of blocks, one could specify them as additional constraints to the model. E.g. by adding a constraint such as $M(1, 2) = 1$, one can guide or constrain the optimization so that blocks 1 and 2 are related.

4.2 The hybrid framework The image matrix and membership matrices are optimized iteratively to best summarize the graph. The output to the CP based image optimization step should be an updated image matrix that act as an input to the membership optimization part. So far, we have discussed the modelling of some principal complex constraints and structures of the image matrix.

The output to the CP model (e.g. optimized decision variables) can be shown in a mask matrix S which is a $k \times k$ matrix whose elements are zero except for those which are part of the optimized structure. For instance, consider the optimized ring structure (output of the corresponding CP model) as represented by variable $x = [2, 3, 4, 1, 5]$, then it means the optimized ring is $1-2-3-4-1$ and position 5 does not participate in the ring structure. Then the corresponding mask matrix S would be a 5×5 matrix of all zero entries except for $S_{12} = S_{23} = S_{34} = S_{41} = W_i$[1]. Note that if a ring of communities is desired, the main diagonal of the mask matrix also takes nonzero weight W_i for each $i \in 1, ..., k$.

The weight W_i is an additional way to include domain knowledge. If no prior knowledge exists about the relative strength of communications between positions, $W_i = 1$ for all $i \in 1, .., k$. However, in a hierarchy structure for example, one might consider more weight for immediate communications to the root block. This is also natural for a core-periphery structure where the strength of connections decrease towards the outer layers of periphery. In our settings, we considered $W_i = 1$ for all structures, except for core-periphery where the weights are according to the weights in the objective function of the CP model (Equation 4.5).

[1]if an undirected ring is required $S_{21} = S_{32} = S_{43} = S_{14} = W_i$ as well since the image matrix is symmetric

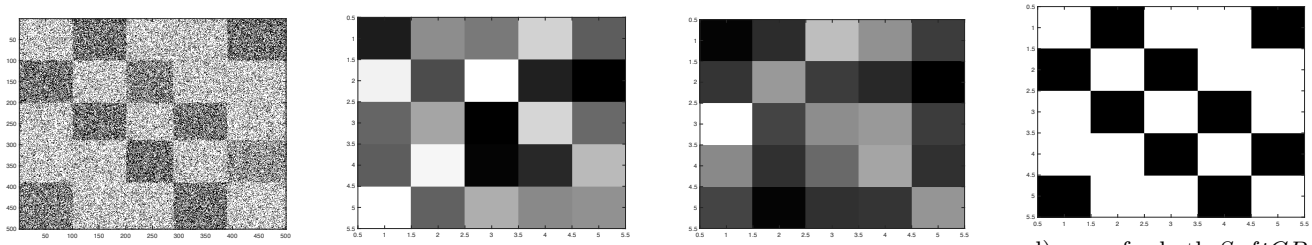
Procedure *SoftCP*(A,k, *Structure*)
1. Random initialization of membership (C) and image (M)
3. **Repeat until** $Obj^{t+1} - Obj^t < \epsilon$
3. $\quad C^{t+1} \leftarrow \quad Membership\_Optimizer(C^t)$
4. $\quad S \leftarrow CP(C^{t+1}, Structure)$
5. $\quad M^{t+1} \leftarrow M^t + \alpha \times S$
6. $\quad M^{t+1} \leftarrow M^t - \beta \times S^c$
7. $\quad M^{t+1} \leftarrow postprocess(M^{t+1})$
8. $\quad t \leftarrow t + 1$
9. **end**

Figure 3: SoftCP algorithm

Pseudo-code of our CP framework with soft updates (called $SoftCP$) is shown in procedure 3. In addition to the adjacency matrix and number of blocks, the desired structure is also given as an input to the algorithm. After initialization of the image and membership matrices, the iterative process of optimizing the membership (image) matrix is performed, fixing the other matrix, until the change in objective value is minimized. Note that different error approximation function (objective function) and membership optimization algorithm ($Membership\_Optimizer$ in line 2) discussed in the literature [6, 21] can be plugged into our algorithm. Given the current membership matrix, the CP model finds the optimum specified structure in the current image matrix. The output of the CP model is then summarized in a mask matrix S (line 4). The optimum structure is then promoted in updating the image matrix using a coefficient α (line 5) and other elements of the image matrix S^c (complement of S) are uniformly discouraged by coefficient β which equals $\beta = \sum_{i,j=1,..,k} \alpha S_{ij} / \sum_{i,j=1,..,k} S^c_{ij}$. Then post-processing is done (line 7) to ensure non-negativity of the image matrix and make it interpretable by forcing $0 \leqslant M \leqslant 1$. Another approach to update the image matrix is to directly impose the optimum structure found by the CP model, S. In this algorithm called $HardCP$, lines 5–7 of $SoftCP$ are replaced by $M^{t+1} \leftarrow S$.

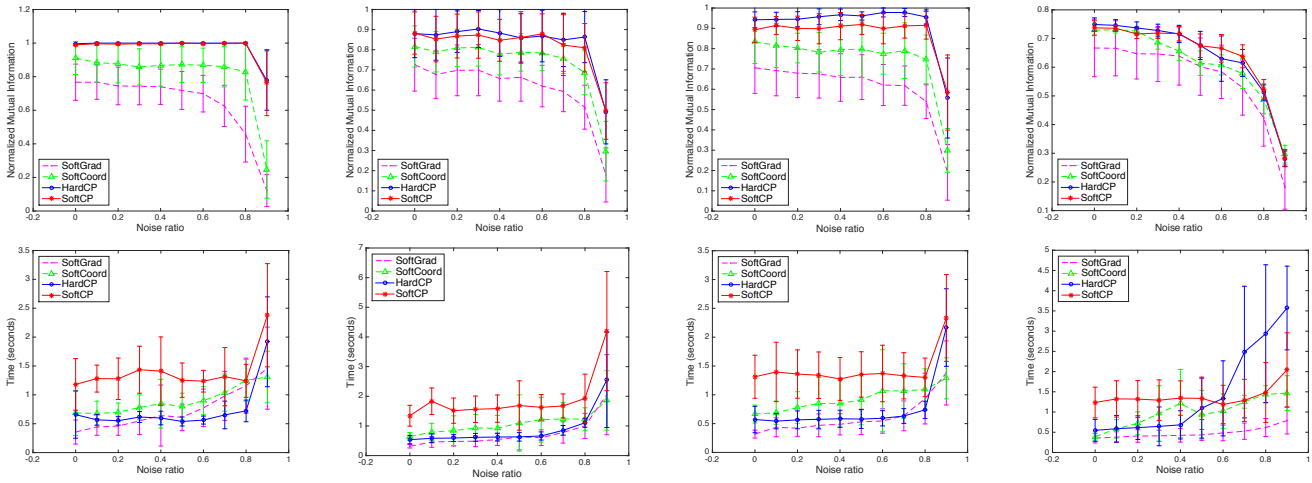
5 Experiments and discussions

5.1 Data sets and set up To evaluate the CP framework and compare to other algorithms on graphs



a) The original data b) *SoftGrad* c) *SoftCoord* d) same for both *SoftCP* and *HardCP*

Figure 4: A network with ring structure and the image matrices found by different methods



a) ring b) hierarchy c) stick d) core-periphery

Figure 5: sensitivity to noise on different graph structures, NMI (first row) and running time (second row).

with different structures, we use a real gene regulatory network and also generate some synthetic datasets according to [6]. Given a random memberships (C) and the image matrix (M), the adjacency matrix (A) is generated using $A = CMC^T$. To generate C, first the block sizes are drawn from a uniform distribution and then the position memberships of vertices are determined by drawing from a hyper-geometric distribution according to which, the probability of each position is relative to its size. Different graph structures such as core-periphery, ring and hierarchy are also replicated in M. Uniform random background noise is added to M, with the noise ratio parameter λ, to control the amount of noise.

All the CP based image optimization are performed using Gecode CP solver [24] in Minizinc 2.0.12 [20] on a Macbook with 8GB RAM, 2.7 GHz Core i5.

We compare our CP-based framework with the best algorithms proposed by Chan et al. [6], hard membership algorithm with coordinate descent (*Softcoord*) and gradient descent (*Softgrad*) image optimization algorithms. These two algorithms have been shown to outperform other algorithms proposed in [6] and also the blockmodelling algorithm of Reichardt et al. [21].

For consistency we use the same membership matrix optimization algorithm (hard incremental membership [6]) and objective function (adjusted Euclidean objective function [6]) for all the algorithms.

To evaluate quality of the discovered blockmodels, we use Normalized Mutual Information (NMI) [17], which is an information theoretic measure to evaluate the quality of solutions against the ground truth.

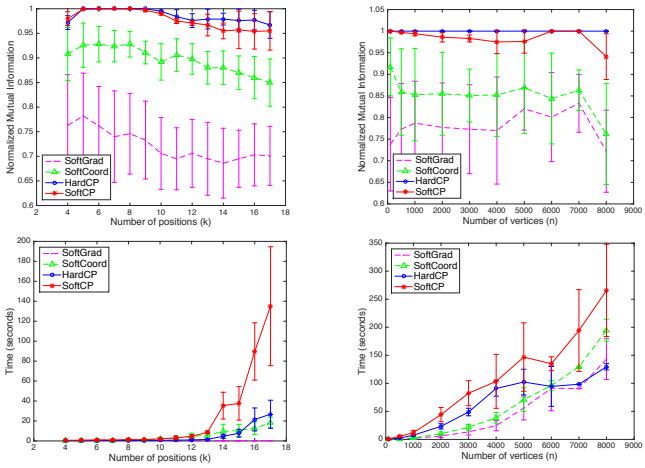
5.2 Discovering complex structures in image matrix To evaluate how well the CP blockmodelling framework can discover complex structures in the image matrix, we generated synthetic data sets, as explained earlier, for which the image matrix is known. We generated data sets of 500 vertices and 5 positions with different structures and perturbed the data with noise (λ=0.7). Then we visualized the final image matrix found using different algorithms. Figure 4 shows the results on a data set with a ring structure where 4-a is the original data represented as a permuted adjacency matrix according to the true positions. As is shown in 4-b and 4-c, the gradient descent and coordinate descent algorithms couldn't retrieve the underlying patterns of the graph in the image matrix while both CP

based algorithms could perfectly reflect the inherent structure of the network as shown in 4-d. Similar results are also observed for other structures including hierarchy, stick and core-periphery (see supplementary materials). The results clearly demonstrate that using the existing methods, one cannot interpret the image matrix, whereas using our CP framework results in clear and insightful encodings of the graph structure.

5.3 Sensitivity to noise In this experiment, we evaluate the performance of the algorithms when background noise increases. We generated datasets with ring, hierarchy, stick and core-periphery structures and increased the noise ratio from 0 to 0.9 and recorded the performance of different algorithms. The results shown in Figure 5 are the average and standard deviation of 10 different data sets and 20 different random initializations for different block structures (200 executions). For having a rigorous comparison, we used the same random initializations for all the methods. In Figure 5, the first and second rows show the NMI and runtime of different algorithms respectively. As shown in the first row of 5, the CP frameworks outperform gradient descent and coordinate descent algorithms in terms of finding the graph structure more accurately.

5.4 Sensitivity to number of blocks and number of vertices As explained in the previous section, the complexity and run time of the CP framework is only related to the image matrix (which is a $k \times k$ matrix) optimization part and not the membership matrix optimization. To empirically show these complexities, we ran two experiments on a graph with a ring structure with $\lambda = 0.2$. In the first experiment, we fixed the network size to 500 and increased the number of blocks and recorded the effect on NMI and runtime of the algorithms. In the second experiment, we fixed the number of blocks to 5 and increased the number of vertices from 100 to 8000.

As the results of these experiments show in Figure 6, when the number of blocks and graph size grows, the proposed CP frameworks maintain their high performance in finding accurate block structures in comparison to the coordinate descent and gradient descent algorithms (shown in first row of Figure 6). Looking at the second row of Figure 6, when the number of blocks/positions increases, the search space for the CP solver increases which makes it slower to find the optimum structure, and the running time increases dramatically for larger k, as expected (Figure 6–a). However, as the graph grows in terms of the number of vertices and edges, the running time of the CP framework grows similarly to the other algorithms (shown in Figure 6–b).



a) Sensitivity to k b) Sensitivity to n

Figure 6: Sensitivity to graph size (n) and number of blocks (k) on ring structure: NMI (first row) and time (second row).

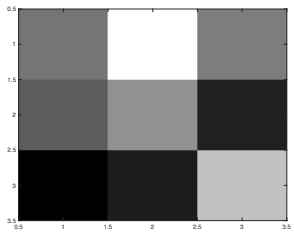
5.5 Complex structure in a real data set In this experiment we evaluate the performance of the CP framework on a real biological data set. The data set is a gene regulatory network with regulator and target interactions for yeast transcriptional regulatory mechanisms [4]. The graph contains 4410 vertices of known regulators and targets. From the domain knowledge we know that targets have strong connections to regulators and to other targets while regulators have sparse interactions among themselves (similar to a star structure). It has been studied that regulators tend to form a hierarchy/stick structure with top, middle and bottom layers. In this experiment, we use the CP based blockmodelling framework to discover the mixed structure of this graph. The CP model is a combination of the models for stick and star with an integer variable c for the star structure and an array of integers x for the stick pattern. The objective function is:

(5.10)
$$M(c,c) + \sum_{i=1}^{k} (M(c,i) + M(i,c)) + \sum_{i=1}^{k} M(x[i], x[i+1])$$

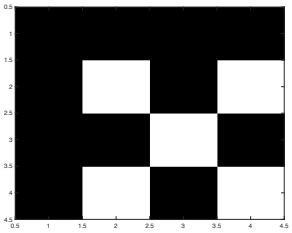
Both CP frameworks perfectly retrieved the mixed structure of this real graph (see Figure 7-b) and find the memberships with an NMI of 77 percent.

6 Conclusion

In this paper we looked at a variety of complex patterns that can be captured in image matrices. We proposed a constraint programming approach to target complex structures in the image matrix and discover interpretable image matrices that clearly reflect the dominant pattern of the graph. The proposed CP framework is not only able to incorporate domain knowledge and



a) $Softgrad$, NMI=0.25



b) $SoftCP$ and $HardCP$, NMI=0.77

Figure 7: Image matrix and NMI retrieved by different algorithms for yeast gene regulatory network

find complex patterns, but can also be coupled with existing blockmodelling techniques with a comparable running time. Exploring other complex structures and exploiting other types of domain knowledge in blockmodelling is an interesting future direction.

References

[1] http://sofdem.github.io/gccat/.

[2] N. Beldiceanu, P. Flener, and X. Lorca. The tree constraint. In *CPAIOR*, pages 64–78. Springer, 2005.

[3] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational statistics & data analysis*, 52(1):155–173, 2007.

[4] N. Bhardwaj, K.-K. Yan, and M. B. Gerstein. Analysis of diverse regulatory networks in a hierarchical context shows consistent tendencies for collaboration in the middle levels. *Proc. of the National Academy of Sciences*, 107(15):6841–6846, 2010.

[5] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *JSTAT*, 2008(10):P10008, 2008.

[6] J. Chan, W. Liu, A. Kan, C. Leckie, J. Bailey, and K. Ramamohanarao. Discovering latent blockmodels in sparse and noisy graphs using non-negative matrix factorisation. In *Proc. of the 22nd ACM international conf. on Information & Knowledge Management*, pages 811–816. ACM, 2013.

[7] I. Davidson, S. Ravi, and L. Shamis. A SAT-based framework for efficient constrained clustering. In *Proc. of SIAM International Conf. on Data Mining*, pages 94–105, 2010.

[8] G. Dooms. *The CP(Graph) Computation Domain in Constraint Programming*. PhD thesis, Catholic University of Louvain, Belgium, 2006.

[9] K.-C. Duong, C. Vrain, et al. Constrained clustering by constraint programming. *Artificial Intelligence*, 2015.

[10] J. Fiala and D. Paulusma. The computational complexity of the role assignment problem. In *ICALP*, pages 817–828. Springer, 2003.

[11] S. Fortunato. Community detection in graphs. *Phys. Reports*, 486(3), 2010.

[12] M. Ganji, J. Bailey, and P. J. Stuckey. Lagrangian constrained clustering. In *Proc. of SIAM International Conf. on Data Mining*, pages 288–296. SIAM, 2016.

[13] M. Ganji, J. Bailey, and P. J. Stuckey. A declarative approach to constrained community detection. In *International Conf. on Principles and Practice of Constraint Programming*, pages 477–494. Springer, 2017.

[14] M. Ganji, J. Bailey, and P. J. Stuckey. Lagrangian constrained community detection. In *To appear in proc. of AAAI Conf. on Artificial Intelligence*. AAAI, 2018.

[15] T. Guns, S. Paramonov, and B. Negrevergne. On declarative modeling of structured pattern mining. *AAAI Workshop on Declarative Learning Based Programming*, 2016.

[16] B. Karrer and M. E. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.

[17] A. D.-G. L. Danon, J. Duch and A. Arenas. Comparing community structure identification. *JSTAT*, 2005.

[18] B. Long, Z. Zhang, and S. Y. Philip. A general framework for relation graph clustering. *Knowledge and information systems*, 24(3):393–413, 2010.

[19] B. Negrevergne and T. Guns. Constraint-based sequence mining using constraint programming. In *CPAIOR*, pages 288–305. 2015.

[20] N. Nethercote, P. Stuckey, R. Becket, S. Brand, G. Duck, and G. Tack. Minizinc: Towards a standard CP modelling language. In *Proc. of the 13th International Conf. on PPCP*, volume 4741 of *LNCS*, pages 529–543, 2007.

[21] J. Reichardt and D. R. White. Role models for complex networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 60(2):217–224, 2007.

[22] P. Rombach, M. A. Porter, J. H. Fowler, and P. J. Mucha. Core-periphery structure in networks (revisited). *SIAM Review*, 59(3):619–646, 2017.

[23] F. Rossi, P. v. Beek, and T. Walsh. *Handbook of CP*. Elsevier, 2006.

[24] C. Schulte et al. Gecode, the generic constraint development environment. 2016.

[25] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22(3):493–521, 2011.

[26] S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.

[27] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.

[28] Y. Zhang and D.-Y. Yeung. Overlapping community detection via bounded nonnegative matrix tri-factorization. In *Proc. of the 18th ACM SIGKDD international conf. on Knowledge discovery and data mining*, pages 606–614. ACM, 2012.