Semi-Supervised Blockmodelling with Pairwise Guidance

Mohadeseh Ganji[1], Jeffrey Chan[2], Peter. J. Stuckey[1], James Bailey[1], Christopher Leckie[1], Kotagiri Ramamohanarao[1], Laurence Park[3]

[1]School of Computing and Information Systems, University of Melbourne, Australia
[2]School of Computer Science and Software Engineering RMIT University, Australia
[3]Computer Science Department, Western Sydney University, Australia
[1] {mohadeseh.ganji, pstuckey, baileyj, caleckie, kotagiri}@unimelb.edu.au,
[2]jeffrey.chan@rmit.edu.au, [3]l.park@westernsydney.edu.au

Abstract. Blockmodelling is an important technique for detecting underlying patterns in graphs. Existing blockmodelling algorithms are unsupervised and cannot take advantage of the existing information that might be available about objects that are known to be similar. This background information can help finding complex patterns, such as hierarchical or ring blockmodel structures, which are difficult for traditional blockmodelling algorithms to detect. In this paper, we propose a new semi-supervised framework for blockmodelling, which allows background information to be incorporated in the form of pairwise membership information. Our proposed framework is based on the use of Lagrange multipliers and can be incorporated into existing iterative blockmodelling algorithms, enabling them to find complex blockmodel patterns in graphs. We demonstrate the utility of our framework for discovering complex patterns, via experiments over a range of synthetic and real data sets.
Keywords: Blockmodelling, Pairwise information, Lagrange multipliers.

1 Introduction

Understanding the latent structure beneath real world complex interactions allows us to gain a deeper insight into the underlying reason and purpose of these interactions. Representing these interactions as a graph allows us to discover these informative structures. Community structure has been extensively studied in the context of graph mining and many algorithms have been introduced to locate communities in graphs [2, 12]. Commonly, communities are defined as a group of vertices that are densely connected among themselves but have sparse connections to the rest of the graph.

However, graphs may contain other inherent and latent structures as well. For example, consider a network of interactions in a question answering forum where members ask or answer questions. In such forums often there is a group of novices who ask many questions but rarely reply to questions, and a group of experts who answer questions but may not ask many questions. If you consider the graph

representing such a question answering relationship, community detection fails to find the underlying groups as it mixes all the members (novices and experts) together due to the many edges between them. Hence, a more general approach is needed that not only detects communities but is also able to reveal deeper patterns in the network.

Blockmodelling is a powerful approach that partitions graphs into groups of equivalent vertices (also called blocks or positions) that play a similar role in the graph [22]. Equivalent vertices have connections to similar vertices that may or may not be in the same group. For example, vertices in a community structure mainly link to their own community, while for the question answering forum example, the expert group (answerers) have similar connections (e.g., they are mainly connected to novices), and vice versa. Hence, blockmodelling is a general approach to discover deeper graph structures, not only communities. In blockmodelling, the inherent structure of the graph can be identified by visualizing the interactions within and between blocks, which is captured in the so-called image matrix. Given k, the number of blocks, the image matrix is a $k \times k$ non-negative real-valued matrix whose elements represent the probability of interaction between and within the blocks. For instance, the image matrix of a graph with a community structure, would have high values along the main diagonal showing the highly probability of edges appearing inside the community and low off-diagonal values showing the sparse connections between communities.

Blockmodelling so far has been studied as an unsupervised task that just relys on the network topology to discover latent patterns and structures, ignoring any potential existing background information about the latent groups.

However, rather than relying completely on unsupervised structure discovery, there may be pre-existing knowledge available about expected patterns in the graph. For example, in the question answering forum, a subset of participants may be known as experts, or some indicators such as the number of upvotes for forum participants may highlight participants as experts. As another example, side-information may be derived using an expensive/invasive medical test that discovers ground truth (block membership) for a small sample of objects. Background information is typically represented as known labels (such as "expert" or "novice") or pairwise information, which can be seen as constraints that show whether two vertices should be in same group (*must-link* constraint) or they should belong to different groups (*cannot-link* constraint). An example of pairwise side-information is a must-link between two sets of proteins with known same functionality in a protein-protein interaction network.

Incorporating such background knowledge in the blockmodelling process can improve the performance and result in a more accurate role discovery of vertices. In addition, it can enable existing blockmodelling algorithms to discover complex structures that they would not be able to find otherwise. For instance, consider an almost bipartite graph shown in Figure 1, similar to the question answering forum example in which the two groups of askers and answerers (left and right groups) communicate densely to each other but the interactions within the groups are sparse. However, as it is shown in part (b) of Figure 1, the unsuper-

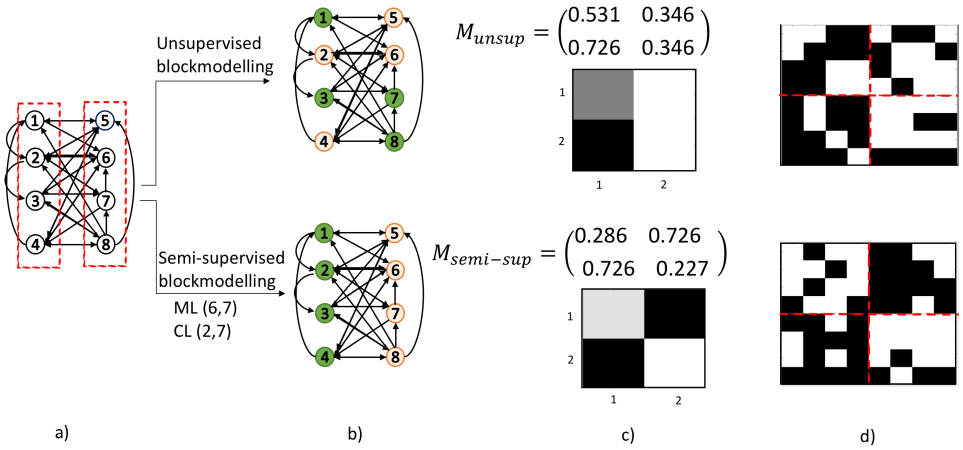


Fig. 1: An example of comparision between unsupervised blockmodeling (top row) and our semi-supervised blockmodelling (bottom row, ML = must-link, CL = cannot-link) on an almost-bipartite graph. a) The original almost-bipartite graph and the ground truth. b) Resulting membership assignments, represented by colors. c) Resulting image matrices and their visualization. d) The reordered adjacency matrices with dash lines representing the block borders.

vised blockmodelling algorithm (top-row) cannot find the true block assignments and mixes the two groups together while the bottom row of the figure shows that by just incorporating one must-link and one cannot-link supervision, our proposed semi-supervised blockmodelling is able to correctly group the vertices. Parts (c) and (d) in Figure 1 compare how accurately the structure of the graph is captured in the image matrix and the reordered adjacency matrix respectively. As explained earlier, one would expect low diagonal values and high anti-diagonal values in the image matrix of an almost bipartite graph. This has been accurately captured by our semi-supervised method where the higher probabilities in anti-diagonal elements represent the bipartite nature of the graph while the image matrix from the unsupervised blockmodelling does not reflect such a pattern. Similarly, no bipartite pattern is captured in the reordered adjacency matrix of the unsupervised blockmodelling method. However, the higher density of links in the top-right and bottom-left sections of the reordered adjacency matrix of our semi-supervised method illustrates the bipartite characteristic of the graph.

Incorporating side-information and background knowledge has been shown to improve the results for other machine learning tasks. For instance, the effect of must-link and cannot-link constraints has been studied on clustering [8, 9] and community detection [14, 10, 15], and it has been shown that they can improve the quality of the solutions and robustness to noise [10, 14, 13, 8].

However, to the best of our knowledge, none of the existing blockmodelling algorithms offers the flexibility to incorporate side information in the form of pairwise constraints. Hence, the side information is ignored by these methods even if the constraints are required or highly desirable to be satisfied.

Klein et al. [18] showed that partitioning in the presence of cannot-link constraints is NP-complete because it can be represented as a reduction from the Graph K- Colorability problem (K-Color). We extended this NP-Completeness proof for blockmodelling in Appendix A. It has been shown that the CL-feasibility problem is NP-complete even when the number of constraints is linear in the number of points [7]. Note that applying an unsupervised partitioning and then fixing the constraint violations afterwards (by flipping the $0/1$ assignments in the output result) is also shown to be NP-complete [7]. In addition, in the case of blockmodelling, this does not naturally reflect the constraints in the structure captured by the image matrix, causing inconsistencies in the results.

In this paper, we propose a semi-supervised blockmodelling framework based on the method of Lagrange multipliers, which can be coupled with several state of the art algorithms in blockmodelling to benefit from pairwise supervision in the form of must-link and cannot-link constraints. The method of Lagrange multipliers is a powerful constrained optimization technique, which has performed very well in difficult NP-hard graph coloring problems and has been used successfully for constrained community detection and clustering problems [13, 15]. In this paper we focus on nonnegative matrix tri-factorization based blockmodelling [19, 21, 5] to discover the membership assignments and graph structure at the same time. Our Lagrange multipliers method encourages satisfaction of the supervision constraints throughout the blockmodelling task by introducing and increasing the penalty for violated constraints from one iteration to the next. The main contributions of this paper are as follows:

- We present a flexible method of blockmodelling that allows background and expert knowledge to be incorporated in the form of pairwise constraints, which results in improved ability to find complex patterns in graphs;
- We demonstrate the high accuracy and noise resistance of our framework using synthetic and real-life data sets.

2 Background

Consider a graph $G(V, E)$ where V is a set of vertices and E is a set of edges. The graph can be represented by its adjacency matrix, A, where for each pair 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 equivalent vertices, which are called positions. Relations between positions are captured by an image matrix whose entries show the probability of communication (edges) between positions. The positions and the image matrix together form a blockmodel. Blockmodelling has been modeled as a nonnegative matrix tri-factorization problem [19, 21, 5] in which positions are represented by a membership matrix $C \in [0, 1]^{n \times k}$ and $M \in [0, 1]^{k \times k}$ represents the image matrix where n is the number of graph vertices and k is the number of blocks or positions. The membership matrix shows the assignment of vertices to blocks and the image matrix summarizes the communications within and between positions. The 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 minimizes a pre-defined approximation error function, for instance, the sum of squared differences (Equation (1)) where $||.||_F$ is the Frobenius norm.

$$\arg\min_{M,C} ||A - CMC^T||_F^2 \tag{1}$$

It is known that blockmodelling of three or more positions is NP-hard [11]. Hence, most blockmodelling algorithms try to find locally optimal 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. The iteration of these alternating steps continues until the algorithm converges. The optimization is done using update rules. In optimizing/updating the membership (image) matrix, the entries can be forced to accept binary or real values, which are called hard or soft membership (image), respectively. It is also possible to turn soft membership to hard membership assignments by assigning each vertex i to the position k with maximum C_{ik} value.

There are several types of equivalence for blockmodelling. In this paper, we focus on structural equivalence [22], according to which, two vertices are in the same position if they have similar sets of in and out neighbours. For instance, if two students have the same supervisor, the students are structurally equivalent. According to structural equivalence, the elements of the image matrix have densities ideally close to 0 or 1.

3 Related Work

It has been shown that even unsupervised blockmodelling is an NP-hard problem [11]. Therefore, blockmodelling algorithms try to find a good solution which has a (locally) minimal approximation error. Blockmodelling has been formulated as a nonnegative matrix tri-factorization problem [19, 21, 5]. Zhang et al. [23] introduced a coordinate descent optimization approach for overlapping blockmodelling. Karrer and Newman [17] considered the heterogeneity in vertex degrees and proposed a stochastic blockmodelling approach. Chan et al. [5] proposed a framework focusing on sparse and noisy graphs. They also introduced objective functions and proposed an incremental approach for optimizing the membership matrix, which only updates the necessary entries of the C matrix each time. The same authors later [4] introduced a soft membership formulation that ensured the vertex to cluster memberships sum to 1, and showed that this made a significant difference to the discovered blockmodels. Reichardt et al. [20] used a simulated annealing method to optimize an objective function based on the difference between the adjacency matrix and its blockmodel approximation. However, none of the existing blockmodelling techniques, to the best of our knowledge, can incorporate pairwise background information to find complex structures in graphs.

In some existing algorithms [20, 5], an instance of the desired structure can be applied as an initialization of the image matrix. In another approach [3], new

forms of equivalence were introduced and a blockmodel could consist of different block types (each corresponding to a different type of equivalence). These blockmodels can incorporate supervision in the form of desired block types. But for all these algorithms, there is no guarantee that they will eventually find the specified structure. In addition, none of the algorithms can incorporate pairwise instance level constraints. Recently, constraint programming has been coupled with a non-negative matrix tri-factorization method to force some structural patterns in image-constrained blockmodelling [16]. While, in contrast to the earlier approaches, the constraint programming framework can incorporate block(image)-level supervision in blockmodelling, it is not able to incorporate pairwise instance-level information, e.g., must-links and cannot-links. Hence, in this paper we close this gap in the literature by proposing semi-supervised blockmodelling incorporating pairwise instance-level constraints. Our method also has the advantage that it can be coupled with the existing approaches above to enable them to benefit from instance-level information.

The effect of must-link and cannot-link constraints has been studied for clustering tasks using a SAT formulation [8], constraint programming [9] and Lagrange multiplier methods [13]. Community detection has also been shown to benefit from pairwise supervision [10, 15]. Ganji et al. [14] used constraint programming to model several constraint types for community detection, including community size, distribution and pairwise instance level constraints. However, scalability is an issue for this method. In addition, even by incorporating constraints, none of the existing semi-supervised community detection and clustering methods can find other structural patterns in the graph such as the groups in the question answering forum example.

To the best of our knowledge, incorporating pairwise background knowledge has not been studied for blockmodelling and none of the existing methods are able to incorporate pairwise constraints. This paper addresses this gap and proposes a semi-supervised framework for blockmodeling based on non-negative matrix tri-factorization and Lagrange multipliers.

4 Proposed semi-supervised framework

In this section we elaborate on our proposed framework and provide more details on modeling and optimization of our semi-supervised blockmodelling method.

4.1 Method for modelling semi-supervised blockmodelling using Lagrange multipliers

Pairwise information for blockmodelling is given in two ways. A *must-link* (ML) constraint between two vertices (i, j) requires that the two vertices are mapped to the same position, i.e., they are known to take the same role in the graph. A *cannot-link* (CL) constraint between two vertices (i, j) requires that they are not mapped to the same position, i.e., they are known to take different roles in the

graph. Let ML be the set of must-link constraints and CL the set of cannot-link constraints.

We enforce these constraints using the Lagrange multipliers method which requires mapping them to a penalty term for each violation and adding the penalties to the original objective function to guide the optimization algorithm towards satisfying the constraints.

In order to define the penalty terms, we introduce matrices Q_{ML} and Q_{CL} which represent the cost coefficients for each pairwise constraint. Q_{ML} and Q_{CL} are $n \times n$ non-negative real valued matrices quantifying the cost of violating each of the must-link and cannot-link constraints respectively. If no must-link and cannot-link are imposed on the pair (i, j), then the corresponding element in Q_{ML} and Q_{CL} is equal to zero, meaning assignments of the pair (i, j) to same or different blocks will not have any supervision violation cost. If the pair (i, j) is involved in a must-link constraint ($ML(i, j)$ or $ML(j, i)$), then $Q_{ML}(i, j)$ and $Q_{ML}(j, i)$ are equal to the corresponding Lagrange multiplier for that constraint, denoted by $\lambda_{[i,j]}$. Similarly $Q_{CL}(i, j) = Q_{CL}(j, i) = \mu_{[i,j]}$ if and only if there exist a cannot-link constraint between i and j. Note that λ and μ are vectors (of Lagrange multipliers corresponding to each constraint type) whose lengths are equal to the number of must-link and cannot-link constraints respectively; the notation $[i, j]$ is only for indexing these vectors and the order is not important.

Given the cost matrix Q_{ML} and a partition represented by C, the total cost associated with must-link constraint violation can be obtained using $\frac{1}{2}(\mathbf{1} - C) \otimes (Q_{ML} \times C)$ where $\mathbf{1}$ is the matrix of the same size as C whose elements are equal to 1 and $\otimes$ denotes element-wise multiplication (and then sum). $R = (Q_{ML} \times C)$ is a $n \times k$ matrix which has a straightforward interpretation. Considering the binary membership case, R_{ik} represents the (must-link) cost of not assigning vertex i to position k. Hence, the total cost for all vertices can be captured by element-wise multiplication of $1/2(\mathbf{1} - C)$ to R. The constant coefficient is because each violated constraint has been penalized twice in our matrix representation. Note that one could prevent this by defining the Q_{ML} as an upper triangular matrix. However, R would no longer have the explained interpretation.

Similarly, given the cost matrix Q_{CL} and a partition represented by C, the total cost associated with cannot-link constraint violation can be obtained using $\frac{1}{2}C \otimes (Q_{CL} \times C)$. The interpretation of $R' = (Q_{CL} \times C)$ in this case is R'_{ik} represents the (cannot-link) cost of assigning vertex i to position k.

These penalty functions for must-link and cannot-link constraint violations naturally generalize to the soft membership version of the problem, where C may not have a unique non-zero (1-valued) entry for each row.

The Lagrangian objective function shown in equation (2) can be calculated by the addition of the penalty terms corresponding to must-link and cannot-link constraint violations to the original blockmodelling objective of equation (1). The problem of constrained blockmodelling is then equivalent to finding a good M and C that minimize the Lagrangian function of equation (2).

$$L(M, C, Q_{ML}, Q_{CL}) = ||A - CMC^T||_F^2 + \frac{1}{2}(\mathbf{1} - C) \otimes (Q_{ML} \times C) + \frac{1}{2} C \otimes (Q_{CL} \times C) \tag{2}$$

Example: We illustrate the calculations of the violation penalty terms in equation (2) using the following small example. Note that we consider a hard (or binary) membership in this example. However, our method naturally works for soft memberships as well.
Let $n = 3$ and in our first scenario, suppose we have two must-link constraints, $ML(1, 2)$ and $ML(1, 3)$. Given the following membership matrix C, we aim to calculate the ML penalty term as follows:

$$Q_{ML} = \begin{bmatrix} 0 & \lambda_{[1,2]} & \lambda_{[1,3]} \\ \lambda_{[1,2]} & 0 & 0 \\ \lambda_{[1,3]} & 0 & 0 \end{bmatrix} \qquad C = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad R = Q_{ML} \times C = \begin{bmatrix} \lambda_{[1,2]} & \lambda_{[1,3]} \\ \lambda_{[1,2]} & 0 \\ \lambda_{[1,3]} & 0 \end{bmatrix} \tag{3}$$

And finally, the must-link penalty value for the partition C is equal to $1/2(\mathbf{1} - C) \otimes (Q_{ML} \times C) = \lambda_{[1,3]}$. This occurs because $ML(1, 3)$ is the only violation within partition C.

In the second scenario, consider the same membership matrix C but this time suppose we only have the cannot-link constraints $CL(1, 3)$.

$$Q_{CL} = \begin{bmatrix} 0 & 0 & \mu_{[1,3]} \\ 0 & 0 & 0 \\ \mu_{[1,3]} & 0 & 0 \end{bmatrix} \quad R' = Q_{CL} \times C = \begin{bmatrix} 0 & \mu_{[1,3]} \\ 0 & 0 \\ \mu_{[1,3]} & 0 \end{bmatrix} \quad \frac{1}{2} C \otimes (Q_{CL} \times C) = 0 \tag{4}$$

We find that the cannot-link penalty is zero since no cannot-link constraints were violated in C.

4.2 Optimization procedure

In this section we elaborate upon the optimization procedure to minimize the Lagrangian function of equation (2). Recall that blockmodelling algorithms attempt to find good M and C matrices that satisfy Equation (1). Coordinate descent [23] and projected gradient descent [1] are the two main approaches for optimizing image and membership matrices in a soft manner. If hard membership is desired, the incremental approach of [5] can be used, 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.

For our semi-supervised blockmedelling approach, we use a similar iterative approach where we iteratively optimize for C given fixed M, and M given fixed C, to minimize the Lagrangian objective function. In the optimization of the image matrix, M, we can directly use the existing unsupervised coordinate descent [23] or projected gradient descent [1] or other image optimization approaches. This is because the image matrix, M, does not participate in any of the penalty

terms in our Lagrangian objective function. Hence, the M matrix is only responsible in minimizing the first part of the Lagrangian function, which is the original unsupervised blockmodelling objective function.

However, in optimizing the membership matrix C, we adapt the hard membership algorithm of [5] to update the C matrix in order to minimize the Lagrangian function (2). After each iterative optimization for both M and C is done, our algorithm updates (increases) the Lagrange multipliers λ and μ corresponding to the violated must-link and cannot-link constraints based on the update rule in equations (5) and (6). This is the main mechanism of the Lagrange multipliers method to encourage satisfying the constraints from one iteration to the next. The corresponding cost matrices Q_{ML} and Q_{CL} are then updated based on the updated Lagrange multipliers λ and μ.

$$\lambda_{i,j} = \lambda_{i,j} + \alpha(1 - (C_{i.} \times C_{j.})) \qquad \forall (i,j) \in ML \tag{5}$$

$$\mu_{i,j} = \mu_{i,j} + \alpha(C_{i.} \times C_{j.}) \qquad \forall (i,j) \in CL \tag{6}$$

In the update shown in equations (5) and (6), α is the learning rate determining how fast each penalty increases if a constraint remains violated from one iteration to the next and $C_{i.}$ refers to the ith row of the membership matrix C.

Pseudo-code of our proposed semi-supervised blockmodelling framework is shown in Figure 2. As shown in Figure 2, after random initialization of the image and membership matrices, the Lagrange multipliers are initialized with the value 1 and the corresponding cost matrices are built. The algorithm then iterates until the improvement on the objective value $L(M, C, Q_{ML}, Q_{CL})$ is smaller than a threshold (ϵ), which indicates convergence of the algorithm. In each iteration, the image and membership matrices are updated according to a given optimization approach. Note that, in optimizing the M matrix assuming C is fixed, the Lagrange penalty terms for must-link and cannot-link constraints are constant values because they are only dependant to the membership matrix C, which is fixed throughout the image optimization. Hence, the optimization of the image matrix (the *Image_ Optimizer* in Figure 2) can be done using existing unsupervised approaches such as projected gradient descent [1] or coordinate descent [23] and the objective function of equation (1).

In optimizing the membership matrix, we use the incremental hard membership algorithm of Chan et al. [5], where each vertex is assigned to a position such that the objective value is locally minimized, and it iterates until no vertex changes position. Chan et al. suggested some precomputed elements to calculate the change in the objective value by flipping a vertex to a different position rather than recomputing the whole objective. We also follow a similar idea for calculating the change in the penalty values, rather than checking all the constraints. Hence, in membership optimization, a vertex is flipped to another position only if this results in a drop in the $L(M, C, Q_{ML}, Q_{CL})$ value, which means a drop in the original approximation error and/or a decrease in the constraint violation penalty.

After optimizing the membership matrix, the Lagrange multipliers are updated (lines 9 and 10). If a constraint remains violated, its corresponding La-

Procedure *SemiSupBlock*(A,k, ML,CL)
▷ Initializing image, membership, Lagrange multipliers and violation cost matrices:
1. Random initialization of membership (C^0) and image (M^0)
2. $\lambda^0_{[i,j]} = 1 \quad \forall (i,j) \in ML$
3. $\mu^0_{[i,j]} = 1 \quad \forall (i,j) \in CL$
4. Initialize Q^0_{ML} and Q^0_{CL} based on λ^0, μ^0, ML and CL.
5. $t \leftarrow 1$
▷ Update image, membership, Lagrange multipliers and violation cost matrices until convergence:
6. **Repeat until** $L^t(M^t, C^t, Q^t_{ML}, Q^t_{CL}) - L^{t-1}(M^{t-1}, C^{t-1}, Q^{t-1}_{ML}, Q^{t-1}_{CL}) < \epsilon$
7. $\quad M^t \leftarrow \quad Image\_Optimizer(M^{t-1})$
8. $\quad C^t \leftarrow \quad Membership\_Optimizer(C^{t-1}, Q^{t-1}_{ML}, Q^{t-1}_{CL})$
9. $\quad \lambda^t_{[i,j]} = \lambda^{t-1}_{[i,j]} + \alpha(1 - (C^{t-1}_{i.} \times C^{t-1}_{j.})) \quad \forall (i,j) \in ML$
10. $\quad \mu^t_{i,j} = \mu^{t-1}_{i,j} + \alpha(C^{t-1}_{i.} \times C^{t-1}_{j.}) \quad \forall (i,j) \in CL$
11. $\quad$ Update Q^t_{ML} and Q^t_{CL} based on λ^t, μ^t, ML and CL
12. $\quad t \leftarrow t + 1$
13. **end**

Fig. 2: Semi-supervised blockmodelling algorithm

grange multiplier is increased by a factor $\alpha > 1$, while it remains the same as in the previous iteration if the constraint is satisfied. This increase in the Lagrange multipliers imposes a higher penalty on violating the same constraint in the next iteration, driving the algorithm to satisfy that constraint eventually.

5 Experiments and Discussion

In this section we evaluate our proposed semi-supervised blockmodelling method and compare it with other state of the art blockmodelling algorithms.

We compare with the best algorithms proposed by Chan et al. [5]: hard incremental membership algorithm with coordinate descent (*Coord*) and gradient descent (*Grad*) image optimization algorithms. These two algorithms have been shown to outperform other algorithms proposed in [5] and also the blockmodelling algorithm of Reichardt et al. [20].

Our semi-supervised framework has the advantage that it can be integrated with different unsupervised blockmodelings. Our two semi-supervised blockmodelling algorithms in this section are semi-supervised gradient descent based (called *S-Grad*) and semi-supervised coordinate descent based (called *S-Coord*) blockmodelling algorithms. The learning rate α in our experiments is set to 1.5. We observed that our proposed methods are mostly robust to the choice of α.

To evaluate the quality and accuracy of the discovered blockmodels, we use Normalized Mutual Information (NMI) [6], which is an information theoretic measure to evaluate the quality of each solution in comparison to the ground truth. All the experiments in this section are performed on a 8GB RAM, 2.7 GHz Core i5 Mac. The source code and data sets used in our experiments are available at http://people.eng.unimelb.edu.au/mganji.

5.1 Experiments on real benchmarks

In this experiment, we compare the performance of the unsupervised and our semi-supervised blockmodelling algorithms on a set of real benchmark data sets that are commonly used for finding community structure. We randomly generated pairwise constraints (equally divided into must-link and cannot-link constraints) from the ground truth. To generate the must-link (cannot-link) constraints, we pick a vertex randomly and pair it with another randomly selected vertex of the same (different) block based on the ground truth (the true labels).

The real data sets used in this experiment are benchmarks from Mark Newman's homepage[1] and Pajek repository[2]. The statistics of the data sets and constraints are shown in Table 1 where n, k and $\#const$ refer to the number of vertices, blocks and pairwise constraints respectively.

The results of this experiment are shown in Table 2 where the average quality of the solutions (NMI) and number of violated constraints are reported. As shown in this table, our semi-supervised blockmodelling techniques (*S-Grad* and *S-Coord*) improve the performance of both baseline methods significantly. Note that the Lagrange methods are not guaranteed to satisfy every constraint, since they are simply treated as penalties. However, there are very few constraints left unsatisfied compared to the original methods that ignore them. A pairwise Friedman statistical test is performed between each pair of the unsupervised method and our corresponding semi-supervised method. The hypothesis of the statistical tests is that the ranking of the two sets of average results are not different. The consistently small p-values indicate that the difference is highly unlikely due to random chance which confirms the statistically significant effect of incorporating side-information into blockmodelling.

Table 1: Real data sets and number of generated constraints

Data	Karate	Strike	Sampson	Mexican	Dolphin	Polbooks	Adjword	Polblogs
n	34	24	25	35	62	105	112	1490
k	2	3	2	2	2	3	2	2
#Const	34	24	24	34	62	104	112	1490

5.2 Experiments on synthetic benchmarks

In this experiment, we evaluate the performance of our method on different graph structures other than community structures. For this purpose, we generated synthetic datasets with ring, star, hierarchy, chain, bipartite and core-periphery structures according to the method described in [5]. Given 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

[1] http://www-personal.umich.edu/ mejn/
[2] http://vlado.fmf.uni-lj.si/pub/networks/pajek/

Table 2: Solution quality (NMI, higher is better) and number of violated constraints on real benchmarks

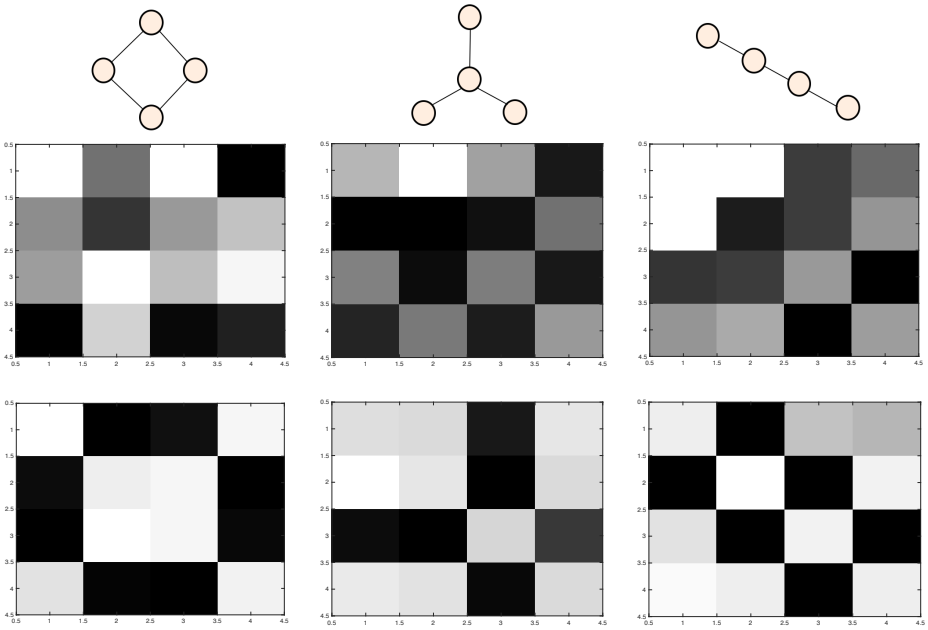
Data	NMI				Violation Count			
	Grad	S-Grad	Coord	S-Coord	Grad	S-Grad	Coord	S-Coord
Karate	0.08	**0.54**	0.09	**0.61**	7.91	**0.00**	8.35	**0.00**
Strike	0.29	**0.40**	0.32	**0.57**	4.45	**0.00**	4.21	**0.00**
Sampson	0.12	**0.54**	0.11	**0.77**	5.51	**0.00**	5.08	**0.01**
Mexican	0.07	**0.58**	0.07	**0.66**	6.95	**0.03**	7.09	**0.00**
Dolphin	0.07	**0.55**	0.06	**0.81**	13.73	**0.00**	13.72	**0.00**
Political books	0.16	**0.39**	0.26	**0.68**	21.86	**0.01**	18.80	**0.00**
Adjacent Word	0.06	**0.59**	0.06	**0.71**	29.33	**0.00**	28.78	**0.00**
Political blogs	0.01	**0.19**	0.01	**0.17**	365.63	**73.91**	358.06	**82.04**
P-Value	0.0047		0.0047		0.0047		0.0047	

Table 3: Solution quality (NMI, higher is better) and number of violations on synthetic data

Data	k	NMI				Violation Count			
		Grad	S-Grad	Coord	S-Coord	Grad	S-Grad	Coord	S-Coord
ring	5	0.76	**0.82**	0.94	**1.00**	26.88	**5.34**	7.05	**0.17**
star	5	0.44	**0.61**	0.46	**0.68**	63.45	**10.48**	62.46	**5.24**
chain	5	0.67	**0.79**	0.88	**0.99**	37.17	**5.83**	13.43	**0.68**
hierarchy	5	0.68	**0.76**	0.86	**0.91**	35.67	**6.76**	15.26	**1.72**
bipartite	2	0.53	**0.91**	0.80	**1.00**	23.32	**0.08**	10.38	**0.00**
core-periphery	3	0.58	**0.85**	0.65	**0.95**	24.01	**0.18**	20.24	**0.07**
P-Value		0.014		0.014		0.014		0.014	

distribution and then the position memberships of vertices are determined by drawing from a multivariate hyper-geometric distribution according to which the probability of each position is relative to its size. Different graph structures such as ring and star are also replicated in M. Uniform random background noise is also added to M, with an specific noise ratio in each experiment.

In this experiment, for each graph structure, we generated 10 different graphs of 100 vertices and perturbed the structure with 20 percent uniformly distributed noise. We generated three sets of constraints (n must-link and n cannot-link constraints) based on the ground truth of each of the generated graphs. The structures and number of positions, k, as well as the average results based on NMI and the number of constraint violations are shown in Table 3. The results show a substantial improvement in accuracy of the semi-supervised methods over the unsupervised ones, and a substantial reduction in the number of violated constraints.



a) ring b) hierarchy c) chain

Fig. 3: Effect of supervision on latent block discovery for different structures. Ground truth Image diagram (top row), discovered image matrix by unsupervised (middle row) and proposed semi-supervised (bottom row) methods.

5.3 Effect of supervision on latent structure discovery

So far we investigated the effect of adding supervision constraints (and the performance of our semi-supervised blockmodelling) on finding accurate vertex assignments to positions. Apart from that, in blockmodelling, the latent structure of the graph is discovered by the image matrix. In this section we evaluate the effect of adding supervision constraints, and the performance of our semi-supervised framework, on structure discovery of the graph.

We generated synthetic data sets of different structures containing 4 blocks and 40 percent background noise, according to the procedure described in section 3. Figure 3 shows the visualized ground truth block interactions using image diagrams and also visualized image matrices found by the coordinate descent blockmodelling method and our corresponding semi-supervised *S-Coord* algorithm (incorporating n must-link and n cannot-link constraints) on a sample graph of different structures[3]. As shown in Figure 3, the inherent structure of the graphs are not clear in the image matrices found by the unsupervised method. However, the latent structure is clearly captured in the image matrices found by our semi-supervised framework including the ring structure $(1-2-4-3-1)$, star structure $(1-3, 2-3, 4-3)$ and chain structure $(1-2-3-4)$.

[3] We observed similar images from other samples in our experiments as well.

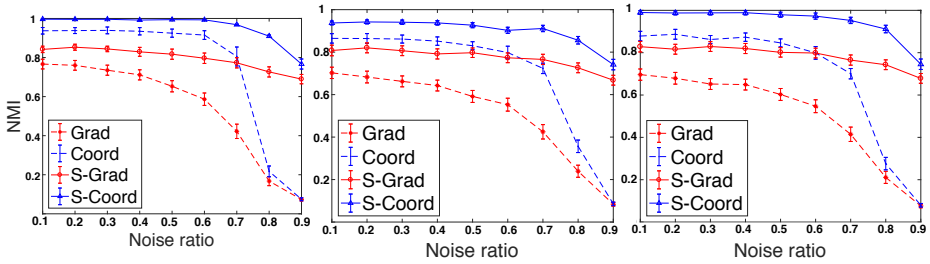


Fig. 4: Sensitivity to noise on ring (left), hierarchy (middle), and chain structure (right)

5.4 Sensitivity to noise

In this experiment, we evaluate the performance as background noise increases. We generated data sets of ring, hierarchy and chain structures with 100 vertices and 5 blocks. We increased the background noise ratio for each data set from 0 to 0.9 and recorded the performance of different algorithms. The results shown in Figure 4 are the sample mean and 95% confidence interval for the mean of 10 different data sets, 3 different constraint sets (100 must-link and 100 cannot-link constraints) for each data set and 20 different initializations (600 executions). The results shown in Figure 4 demonstrate that the addition of background information to blockmodelling improves the noise resistance of the baseline methods. We can see that when the noise ratio increases, the performance of (*Coord*) and (*Grad*) drops significantly in comparison to the corresponding proposed semi-supervised (*S-Coord* and *S-Grad*) versions of the algorithms, respectively.

5.5 Sensitivity to the number of constraints

In this experiment we evaluate to what degree the supervision constraints affect the quality of blockmodel solutions and runtime of the algorithms. We generated data sets with ring, hierarchy and chain structures containing 100 vertices and 5 blocks. We increased the number of supervision constraints, derived from the ground truth labels, from zero to ten percent of the total number of possible pairwise constraints[4]. The results are shown in Figure 5. Clearly, our semi-supervised algorithms (*S-Coord* and *S-Grad*) improve the solution quality significantly when more and more supervision information is available, whereas the unsupervised algorithms ignore the information.

Incorporating the pairwise information, however, increases the runtime of the algorithms so that according to Figure 5, in the worst case, *S-Coord* and *S-Grad* are around 2 to 4.5 times slower than their corresponding unsupervised algorithms. This increased runtime is partly due to the cost of updating the Lagrange multipliers in each iteration but mainly, depending on the constraints, the possibility of requiring more iterations to converge. However, more pairwise constraints do not always increase the runtime. After some amount (around 4%),

[4] The total amount of pairwise information for a graph of size n is $n \times (n-1)/2$.

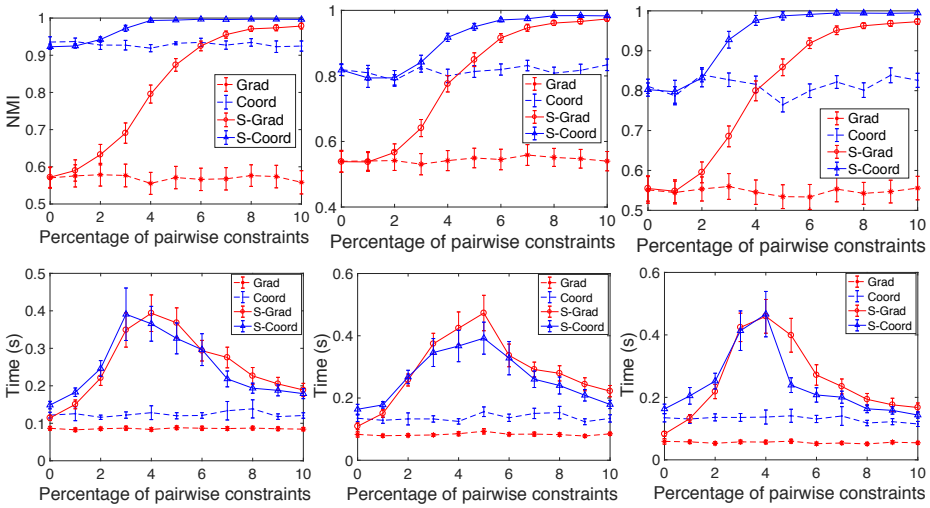


Fig. 5: Sensitivity to the amount of constraints: NMI (top row, higher is better) and runtime (bottom row) for ring (left), hierarchy (middle), and chain (right) structures

adding more constraints decreases the runtime because the stronger information requires fewer iterations to converge.

6 Conclusion

In this paper we proposed a semi-supervised blockmodelling framework that is able to incorporate background knowledge to better find the latent structure and position assignments in complex networks. Our framework is based on the method of Lagrange multipliers and can be coupled with existing iterative optimzation approaches for blockmodeling. It has been shown in our experiments on real and synthetic data sets that our framework improves the quality of the solution and noise resistance of the blockmodelling algorithms. An interesting direction for future research is to exploit other types of domain knowledge in semi-supervised blockmodelling and also further scaling it.

References

1. 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.
2. V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.
3. J. Chan, S. Lam, and C. Hayes. Generalised blockmodelling of social and relational networks using evolutionary computing. *Social Network Analysis and Mining*, 4(1):155, 2014.

4. J. Chan, C. Leckie, J. Bailey, and K. Ramamohanarao. Tribac: Discovering interpretable clusters and latent structures in graphs. In *Proc. of the 15th IEEE International Conf. on Data Mining*, pages 737–742, 2015.
5. 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.
6. L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas. Comparing community structure identification. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(09):P09008, 2005.
7. I. Davidson and S. Ravi. Intractability and clustering with constraints. In *Proc. of the 24th International Conf. on Machine Learning*, pages 201–208. ACM, 2007.
8. 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.
9. K.-C. Duong, C. Vrain, et al. Constrained clustering by constraint programming. *Artificial Intelligence*, 244:70–94, 2017.
10. E. Eaton and R. Mansbach. A spin-glass model for semi-supervised community detection. In *AAAI*, pages 900–906, 2012.
11. J. Fiala and D. Paulusma. The computational complexity of the role assignment problem. In *International Colloquium on Automata, Languages, and Programming*, pages 817–828. Springer, 2003.
12. S. Fortunato. Community detection in graphs. *Phys. Reports*, 486(3), 2010.
13. 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.
14. 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.
15. M. Ganji, J. Bailey, and P. J. Stuckey. Lagrangian constrained community detection. In *To appear in proc. of AAAI*, 2018.
16. M. Ganji, J. Chan, P. J. Stuckey, J. Bailey, C. Leckie, K. Ramamohanarao, and I. Davidson. Image constrained blockmodelling: A constraint programming approach. In *To appear in Proc. of SIAM International Conf. on Data Mining*, 2018.
17. B. Karrer and M. E. Newman. Stochastic blockmodels and community structure in networks. *Physical Review E*, 83(1):016107, 2011.
18. D. Klein, S. D. Kamvar, and C. D. Manning. From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering. Technical report, Stanford, 2002.
19. B. Long, Z. Zhang, and S. Y. Philip. A general framework for relation graph clustering. *Knowledge and information systems*, 24(3):393–413, 2010.
20. 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.
21. 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.
22. S. Wasserman and K. Faust. *Social network analysis: Methods and applications*, volume 8. Cambridge university press, 1994.
23. 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.