Improving Single and Multi-View Blockmodelling by Algebraic Simplification

Rishabh Ramteke
Indian Institute of Technology
Bombay, India
rishabhramtekegsr@gmail.com

Peter J. Stuckey
Monash University
Melbourne, Australia
peter.stuckey@monash.edu

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

Kotagiri Ramamohanarao
University of Melbourne
Melbourne, Australia
kotagir@unimelb.edu.au

James Bailey
University of Melbourne
Melbourne, Australia
baileyj@unimelb.edu.au

Christopher Leckie
University of Melbourne
Melbourne, Australia
caleckie@unimelb.edu.au

Emir Demirović
University of Melbourne
Melbourne, Australia
emir.demirovic@unimelb.edu.au

***Abstract*—Blockmodelling is an important technique in social network analysis for discovering the latent structures and groupings in graphs. State-of-the-art approaches approximate the graph using matrix factorisation, which can discover both the latent graph structures and vertex groupings. However, factorisation is a one-way approximation, in that it only approximates the graph with a lossy model that removes the background noise. Traditional Blockmodelling methods rely on an alternating 2-step optimization that involves iteratively updating the matrix representing membership while fixing the matrix representing the graph's underlying structure, and then updating the structure matrix while keeping the membership matrix fixed. We propose a single step optimization method, which uses algebraic simplification to directly update the lower dimensional, latent structure representation. This helps improve both the convergence and accuracy of blockmodelling. We also show that this approach can solve multi-view blockmodelling problems, involving multiple graphs over the same vertices. We use real datasets to show that our approach has much higher accuracy and comparable running times to competing approaches.**

***Index Terms*—Blockmodelling, Multi-view, Algebraic simplification**

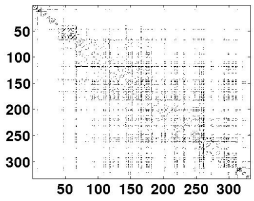
INTRODUCTION

Finding the inherent groups in graphs is an important problem in social network analysis and has applications in many domains, such as identifying communities in social networks [1], functional modules in protein-protein interaction networks [2] and analysing the effect of community structure on the diffusion of ideas in social networks [3]. This involves assigning each vertex in the graph to one or more inherent groups, where each vertex can represent a person or a protein for example.

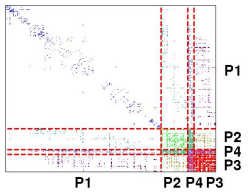
How vertices are assigned to their respective groups depends on what it means for two vertices to belong to the same group. The most popular definition is the *community* one. Vertices belong to the same community/group if they have many connections among themselves and few connections to other communities [4]. Although this definition has been useful in numerous applications [1], there are other alternative types of inherent graph structures. Consider the example shown in Figure 1, which shows a blockmodel decomposition of the flight routing network between airports. Each vertex represents an airport and an edge between them represents one or more flights between the airports. Figure 1a shows the ungrouped adjacency matrix of the graph. Figure 1b shows a partition of the vertices into groups or *positions*, delimited by the red dotted lines, and the rows and columns are rearranged so vertices in the same group are adjacent. We label the positions as P1-P4. The decomposition shows the P4 vertices are highly connected among themselves and to vertices in other positions. This grouping found by the blockmodel algorithm corresponds to highly-connected, international hub airports. P3 vertices are highly connected among themselves and P4 vertices, and represent the large national hub airports. P2 and P1 have higher connectivity to the core (P4) and fewer connections among themselves. The overall structure is core-periphery, which can be summarised by the combination of (1) the image matrix of Figure 1d, which shows the aggregate connectivity between positions, and (2) the membership matrix of Figure 1c, which shows the membership of each vertex (airport) to each position. A decomposition based on the community definition is unable to find the structure in Figure 1, as it cannot capture the off-diagonal blocks for the rearranged adjacency matrix (e.g., Figure 1b). In a community model, airports within each group are mainly connected among themselves, which is clearly not the case for how airports are organised. Therefore a more general approach to finding the inherent graph structure is needed.

An alternative and more general approach is *blockmodelling* from social network analysis. Blockmodelling decomposes graphs based on different grouping definitions, such as *structural equivalence*, where vertices are in the same position if they have similar patterns of interactions to vertices of other positions. The core-periphery structure of the example shown in Figure 1 fits this definition. The inherent structure

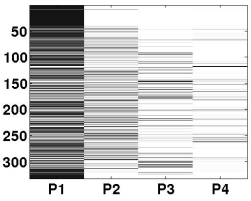
This research was supported (partially or fully) by the Australian Government through the Australian Research Council's Discovery Projects funding scheme (project DP170103174).



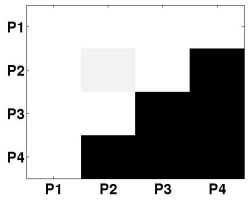
(a) Adjacency matrix of the airport routing network.



(b) Rearranged adjacency matrix according to a 4 position blockmodel.



(c) Membership matrix. Rows are the vertices, and columns are the positions. Values range from 0 to 1.



(d) Image matrix. Rows and columns are the positions. Values range from 0 to 1.

Fig. 1: Airport routing network, represented by the adjacency matrix in Figure 1a. In Figure 1b, the pixels representing the edges of the adjacency matrices are coloured according to their position assignments and the red dotted lines represent the boundaries of the positions after they are discretised. Edges whose incident vertices belong to two different positions are coloured with the average of the two interacting positions. Darker blocks in the image matrix of Figure 1c represent larger values.

is revealed by the image diagram (Figure 1d), and this can include hierarchical, community and many other important structures. Blockmodels summarise the structure of a graph succinctly, allow us to explore the membership of the vertices and how they relate to each other, permit us to interpret the underlying structure (e.g., it is a core-periphery), and facilitate discovery of important roles (e.g., the Core role is highly connected to all other roles and are key hubs for flights).

Two broad classes of approaches for blockmodelling have been developed. The first class, Bayesian Blockmodelling, takes a probabilistic approach using graphical models [5]. Challenges for this type of approach are how to efficiently perform inference and the need to correctly model the underlying structure.

The second class of approach formulates the blockmodelling task as a matrix decomposition problem, targeting the discovery of latent factors [6]. An advantage of this type of approach is the ability to leverage the large body of related work on efficient matrix factorisation. Existing work in blockmodelling using matrix factorisation takes as input the adjacency matrix of the graph, and then decomposes this matrix into factors that correspond to elements of the blockmodel. The standard formulation of matrix decomposition is solved by iteratively updating the membership matrix while holding the image matrix fixed, and then updating the image matrix while holding the membership matrix fixed. We show that we can construct a closed form solution for updating the image matrix. We then show that using this direct approach to computing membership leads to performance and accuracy improvements for blockmodelling.

A further challenge we address in our formulation is the incorporation of multi-view aspects. Multi-view refers to the case whereby a given set of vertices can be used to form different types of graphs or "views". Each such view corresponds to a different type of relationship. For example, a social network may be constructed where vertices correspond to tweets. One view for this network consists of edges indicating the existence of reply relationships between pairs of tweets. Another view for this network consists of edges that indicate the existence of a given threshold of textual similarity between pairs of tweets.

In this paper, we propose a new method that:

- improves accuracy of the decomposition by using only a single-step update method, and
- supports multi-view blockmodelling, improving the accuracy of the decomposition by utilising more sources of information when they are available.

BLOCKMODELLING

A directed graph $G(V, E)$ consists of a set of n vertices V and a set of edges E, $E \subseteq V \times V$. It can be represented by an $n \times n$ adjacency matrix A where $A_{ij} = 1$ iff there is a directed edge from v_i to v_j.

A *blockmodel* is a form of dimension reduction that decomposes a graph represented by an adjacency matrix into a set of k vertex partitions (called positions), represented by a membership matrix $C \in [0,1]^{n \times k}$ where $c_{ij} = 1$ means that vertex i is assigned to position j, and an image matrix $M \in [0,1]^{n \times k}$, where m_{ij} describes the likelihood of an edge between a vertex from position i to a vertex of another position j. Each entry in M is called a block. The blockmodel decomposition approximates A as $CMC^\top$. The objective of blockmodelling is, given A and number of positions k, to find the most accurate blockmodel decomposition, i.e.,

$$\underset{C,M}{min} f = \|A - CMC^\top\|_F^2$$

where $\|.\|_F^2$ is the Frobenius norm (squared).

Blockmodelling is NP-Hard [7], hence blockmodelling methods only seek good local optima.

[6] introduced an improvement to the objective to account for the imbalanced nature of edges to non-edges (no edges) in A, which we also use. Typically a graph is sparse, hence non-edges dominate. If we just approximate A without adjusting for this, the edges are less likely to be approximated well. Hence a weighting scheme was introduced.

$$\min_{C,M} \left(\|(A - CMC^\top) \circ (A - R)\|_F^2\right) \tag{1}$$

where $R \in [0,1]^{n \times n}$, $R_{i,j} = \frac{m}{n^2}$, $\forall i, j \in V$ and $\circ$ represents the element-wise multiplication operator. When a graph is sparse, $\frac{m}{n^2}$ is small, and the weighting factor $(A - R)$ will be large when $A_{ij} = 1$ and small when $A_{ij} = 0$, thus achieving the reweighting.

There are two versions of blockmodelling. In *hard* blockmodelling, each vertex can only belong to one position. Recall that each row of C indicates the membership of a vertex to each of the k different positions. Hence for hard blockmodelling, for each row i of C, if vertex v_i belongs to position j, then $C_{ij} = 1$ and other entries in that row are zero, i.e., $C_{ix} = 0, 1 \leq x \leq k, x \neq j$.

In *soft* blockmodelling, vertices can belong to multiple positions. This allows greater flexibility and ability to model scenarios such as people (represented by vertices) having multiple groups such as friends, family, work and hobbies. Thus, soft blockmodelling can model overlapping memberships which gives better results, but at the expense of more degrees of freedom, and potentially memberships where a vertex belongs to all positions. In this paper we examine both hard and soft blockmodelling.

Existing Blockmodelling approaches

Traditional matrix decomposition approaches to blockmodelling solve for M and C by starting from some random initial solution, and updating C while holding M fixed, and then updating M while holding C fixed.

Algorithm 1: Blockmodelling matrix decomposition algorithm, given $n \times n$ adjacency matrix A and number of positions k, find the best blockmodel C and M

```
M ← k × k matrix with random values in [0,1]
C ← n × k matrix with random values in [0,1]
while method has not converged do
    C ← updateC(C, M, A)
    M ← updateM(C, M, A)
return (C, M)
```

In hard blockmodelling, the update function for C, $updateC$, is based on solving a discrete constrained optimisation problem [6]. In soft blockmodelling $updateC$ is usually based on gradient descent. Similarly, the update function for M, $updateM$, can be based on multiplicative update rules [6] or coordinate descent [8].

The update functions need to ensure that the values in M matrices continue to lie in the range [0,1]. For hard blockmodelling the values in C should be in $\{0, 1\}$.

updateC: We explain the approach that [6] used and we also adopt. When solving hard blockmodelling, we are solving a discrete, NP-hard problem. Updating C involves assigning memberships and assessing the goodness of the change. To reduce the computation needed, [6]. introduced an incremental approach to evaluating Equation 1.

Let the current membership and image matrices at iteration t be denoted by $C^{(t)}$ and $M^{(t)}$. Without loss of generality, consider that the vertex v_i be reassigned from its current position g to its new position h. We can write these atomic reassignment operations as matrices of dimensions $n \times k$. Let $\Delta^-_{ig} = -1$ for entry (i, g) and 0 elsewhere. Let $\Delta^+_{ih} = +1$ for entry (i, h) and 0 elsewhere. This reassignment and its effect on C can be written as $C^{(t+1)} = C^{(t)} - \Delta^-_{ig} + \Delta^+_{ih}$. When we perform the reassignment, we evaluate the new value for the objective $\|(A - C^{(t+1)}M^{(t)}(C^{(t+1)})^\top) \circ (A - R)\|^2_F$. To simply the notation, let $D^{(t)} = \|(A - C^{(t)}M(C^{(t)})^\top) \circ (A - R)\|^2_F$ (we drop the superscript for M to simply the notation, as it is not updated here). The relationship between objectives $D^{(t+1)}$ and $D^{(t)}$ can rewritten as:

$$D^{(t+1)} = D^{(t)} + S^{(t)}\Lambda^T + \Lambda U^{(t)} - \Lambda M \Lambda^T$$

with $S^{(t)} = C^{(t)}M$, $U^{(t)} = M(C^{(t)})^\top$ and $\Lambda = \Delta^-_{ig} - \Delta^+_{ih}$.

As most of the values in $D^{(t+1)}$ do not change due to the reassignment, we can isolate the changes and the hence computation to evaluate the goodness of it. We can compute $D^{(t+1)}$ by first assuming $D^{(t+1)} = D^{(t)}$ and then:

$$D^{(t+1)}_{*,i} = S^{(t)}_{*,g} - S(t)_{*,h} \tag{2}$$

$$D^{(t+1)}_{i,*} = U^{(t)}_{g,*} - U^{(t)}_{h,*} \tag{3}$$

$$D^{t+1}_{i,i} = M_{g,g} - M_{h,g} - M_{g,h} + M_{h,h} \tag{4}$$

Then for vertex v_i, the position that results in a minimal sum of the $D^{(t+1)}$ terms in Equations 2 to 4 is its best position assignment. Computing Equations 2 to 4 has a complexity of $O(k)$ and since we can precompute $D^{(t)}$, $S^{(t)}$ and $U^{(t)}$, the complexity of position evaluation is also $O(k)$. $S^{(t+1)}$ and $U^{(t+1)}$ can be similarly updated using rules similar to Equations 2 and 3.

updateM: For updating M and ensuring M lies within $[0, 1]$, [8] used a coordinate descent approach, where each entry of M is optimised one by one. Let E_{ij} be a $k \times k$ matrix with all entries 0, apart from row i and column j, which has value of 1 (this is the conjugate basis). Let ψ be a step size. They update M_{ij} by $M_{ij} = M_{ij} + \psi E_{ij}$. To find the appropriate ψ for each M_{ij}, the following objective was solved: $\min_\psi L_{ij}(\psi) = \|A - C(M + \psi E_{ij})C^T\|^2_F$.

Taking the derivative and solving for ψ, resulted in the following update rule:

$$\psi = \begin{cases} \min(\psi, 1 - M_{ij}) & \text{if } \psi \geq 0 \\ \min(\psi, -M_{ij}) & \text{if } \psi < 0 \end{cases}$$

For Soft Blockmodelling, [8] updated the membership and image matrix using gradient descent approach where each entry of C is optimized one by one. The idea was to compute the gradient of C and M, then use a line search to find the appropriate step size (ψ) that results in the minimal objective value. To ensure that both M are C are non-negative, they were projected to the non-negative quadrant.

ASBLOCK : IMPROVED BLOCKMODEL SOLVING

We show that by algebraic manipulation of the blockmodel objective we can generate a closed form solution for computing the image matrix M given a membership matrix C.

Assuming a perfect blockmodel decomposition we have that $A = CMC^\top$. Let $N = C^T C$. Then we have that

$$\begin{array}{rrcl} & A & = & CMC^\top \\ \Leftrightarrow & AC & = & CMC^\top C \\ \Leftrightarrow & AC & = & CMN \\ \Leftrightarrow & C^\top AC & = & C^\top CMN \\ \Leftrightarrow & C^\top AC & = & NMN \\ \Leftrightarrow & M & = & N^{-1}C^\top ACN^{-1} \end{array}$$

So we have a closed form formula for determining the image matrix M given the membership matrix C.

Assuming that C is a hard membership matrix (i.e., each row i has exactly one non-zero entry $C_{ij} = 1$ indicating vertex i should be placed in position j), then we have that $N = C^\top C$ is a diagonal matrix of size $k \times k$ where each diagonal entry $N_{jj} = \sum_{i=1}^{n} C_{ij}$ records the sum of column j of the C matrix (which for hard membership equals the number of vertices assigned to position j). When we have hard partitions, then N is a diagonal matrix, hence we have the following equivalence since multiplication by a diagonal matrix (and its inverse) is associative and commutative:

$$M = N^{-2}C^\top AC$$

Updating M: In previous work [8], gradient or coordinate descent was used to update M. But as the previous analysis showed, we can optimise for M directly.

Rather than updating M by gradient or coordinate descent we can simply use this equation to compute the "best" M given A and C. We define $updateM(A, M, C) = N^{-2}C^\top AC$ for hard blockmodelling where N is the diagonal $k \times k$ matrix with $N_{jj} = \sum_{i=1}^{n} C_{ij}, 1 \leq j \leq k$. In the case of hard blockmodelling with a perfect decomposition this update will drive us to the optimal solution for M immediately. For soft blockmodelling $updateM(A, M, C) = N^{-1}C^\top ACN^{-1}$ where since $N = C^\top C$, its inverse is $N^{-1} = \frac{C^\top C}{\sum_i^n (C^\top C)_{ii}}$. When substituted into $updateM$, N^{-1} becomes a normalisation factor and M is guaranteed to lie between $[0, 1]$.

Updating C: We adopt the same approach as [6] for updating our hard membership blockmodels.

Multiple View Blockmodelling

Standard blockmodelling treats all edges between vertices as equivalent, and simply tries to find common structure within the graph using this view of the edges. But in many cases the graph will have different kinds of edges, and we know that they represent different relationships between vertices: for example in a protein-protein interaction (PPI) network their are activation edges and inhibition edges. Each plays a very different role in interaction. If we treat all interactions as the same during blockmodelling then we would be unable take advantage of this information.

[9] were the first to consider the idea of clustering vertices based on multiple graphs among the edges. They show that by considering a tri-matrix factorization approach, it is simple to break the problem into finding a common position matrix C and a separate image matrix M_i for each view of the graph.

The *multiple view blockmodelling problem* finds a *single* position matrix C and p image matrices $M_1, \ldots, M_p$ that minimize the objective function :

$$\begin{array}{rl} \min\limits_{C, M_1, \ldots, M_p} & (\|A_1 - CM_1C^\top\|_F^2 \\ & +(\|A_2 - CM_2C^\top\|_F^2 \\ & +(\|A_3 - CM_3C^\top\|_F^2 \\ & \vdots \\ & +(\|A_p - CM_pC^\top\|_F^2 \end{array}$$

This defines a multiple view blockmodel.

For example, for a PPI network we construct the adjacency matrix A_a of activation edges, and the adjacency matrix A_i of inhibitory edges, in order to determine the position matrix that takes into account both kinds of information separately.

Extending Algorithm 1 to handle the multiple view case is straightforward. The procedure $updateC$ simply uses the larger objective function described above. We replace line 5 by a loop updating each image matrix $M_i \leftarrow updateM(C, M_i, A_i)$.

In the case that we use our direct computation of the image matrices M_i the $updateM$ function is simply $updateM(C, M_i, A_i) = N^{-2}C^\top A_i C$ for hard blockmodelling and $updateM(C, M_i, A_i) = N^{-1}C^\top A_i CN^{-1}$ for soft blockmodelling. Since all M_i are independent of each other we can calculate each M_i separately and in parallel, which saves computation time.

Evaluation

In this section, we evaluate our approach and compare them against the baseline algorithms. We first evaluate the single view formulation and its improvement in recovering latent structures using real world datasets. We then evaluate the multi-view formulation and demonstrate how it improves on single view formulations and multi-view community detection. We use normalized mutual information (NMI) as the principle measure of the accuracy of the approach, as is standard for work on blockmodelling and community detection [10]. All algorithms are implemented in Matlab 2019a.[1] Experiments were performed on an Intel i7 8^{th} gen laptop with 1.8 GHz CPU and 8 GB RAM. The time limit for each run was 3hrs.

Single View Evaluation

We compare our improved blockmodelling approach versus a number of existing blockmodelling algorithms on several standard datasets used in the literature. Statistics of the datasets are shown in Figure I.

In previous work [6], several baselines in community detection and matrix factorisation based clustering were used. Some of these algorithms perform either hard (**-H**) blockmodelling, soft (**-S**) or have both a hard and soft version (we evaluate against both variants). These algorithms include: **RGC**

[1]To avoid violating the double blind policy, we will release the code and provide a link in paper after the reviewing process.

Name	Vert. #	Edge #	Part. #
Baboon	14	23	2
Karate	34	78	2
Politic Books	105	441	3
College Football	115	613	12
Politic Blogs	1490	19090	2

TABLE I: Statistics of single view real world datasets.

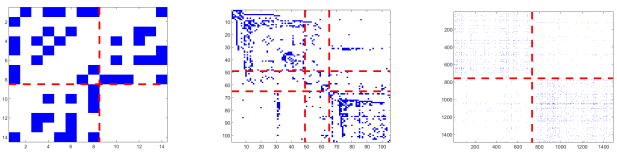
Name	Vert. #	Views #	Part. #
3sources	169	3	6
100leaves	1600	3	100
NewsGroup	500	3	5
WebKB	203	3	4
CiteSeer	3312	2	6
CoRA	1662	2	3

TABLE II: Statistics of multi view real datasets.

Relational Graph Clustering [11], **Reic** Role Models [12], **ANMF** Asymmetric Nonnegative Matrix Factorization [13], **BNMTF** Bounded Non-negative Matrix Tri-factorization [14]. [6] introduced several variants of their algorithm, and generally found the following two variants outperform the other hard and soft variants, hence we restrict our testing to these: **Grad-H-Ad** optimises an objective that penalties M that deviate from [0,1]. It uses the same position assignment approach as ASBlock and coordinate descent to optimise M. **Grad-H-AdCn** is similar to Grad-H-Ad, but it optimises an objective that additionally encourages M to be 0 or 1. It uses similar optimisation approaches as Grad-H-Ad. For other baseline algorithms like **DANMF** Deep Autoencoder-like Nonnegative Matrix Factorization [15] and **SBM** stochastic block model for modular networks [16], we use the default parameters as proposed in their original papers.

We also ran two different initialisations and update of C in our algorithm. The first is a random membership assignment of C (i.e., $\{0,1\}$ assignment) which uses hard blockmodelling update – we call this **Hard ASBlock** . The second is for each row of C, we assign values in $[0, 1]$ and each row adds up to 1 (i.e., membership adds up to 1) and this uses soft blockmodelling update – we call this **Soft ASBlock** .

We ran each algorithm 30 times with random initial matrices and report the mean performance. The evaluation results for real datasets are shown in Table III. As can be seen, both the hard and soft initialisation variants of our proposed approach have much better NMI performance than all the baselines. Our hard initialisation approach is bettered by SBM twice.



(a) Baboon (b) Pol. books (c) Pol. blogs

Fig. 2: Blockmodel of single view datasets on applying ASBlock

Our soft initialisation approach is the best among all. As the main difference between both our approaches and Grad-H-Ad and Grad-H-AdCn is essentially the direct computation of the image matrix M, this highlights the value of this treatment of the image matrix.

It is interesting that the results for the two different initialisations and updates lead to large variations in the results. We conjecture that the hard initialisation and update for some datasets has insufficient freedom to explore and escape its initialisation. The soft initialisation and update has more freedom to move, and hence reach a globally better solution.

The time requirements for our methods are modest, as they are faster in general than the Grad-H-Ad methods that they are based on, showing that the direct computation of M_i improves the convergence. They are typically twice as fast as the DANMF method. The fastest methods in the table, RGC-S and ANMF-S, are highly inaccurate.

To illustrate the benefit of the image decomposition, we produced the adjusted adjacency matrices of three datasets (Figure 2). The ideal types of structures have been discovered - Baboon is similar to a core-periphery network, while political blogs and books are community structure. Note that for political books, it is known that there are generally two communities of books (Democrats and Republicans, the two dominant parties in United States), but also a small number of books that sit between the two dominant communities (the small community in the middle with links to the two other communities). This demonstrates that our novel image matrix update rule can find accurate and interpretable blockmodels to explain the underlying networks.

Multi View Evaluation

We evaluate our multi-view blockmodelling approach on six real-world datasets, summarised in Figure II. [2]

We compare our ASBlock method against the baselines **RMKMC** [18] Robust Multi-View K-Means Clustering, **SNMTF** [19] Simplicial Nonnegative Matrix Tri-Factorization, **SC-ML** [20] Spectral clustering on Multi-Layer graphs, **MCGL** [21] Multiview clustering with graph learning, **ASMV** [22] Adaptive structure-based multi view clustering and **GraphFuse** [23]. In each case we used the default parameters proposed in the original papers.

Table IV contains results of our method and the baselines on the benchmark multi-view datasets. Since some of these algorithms including ours depend on the initializations, without loss of generality, we run all the methods 30 times using random initializations and report the mean performance. This table shows that our approach soft ASBlock is highly competitive, providing the best results for all of the six benchmarks, and hard ASBlock is clearly the second best approach.

[2]3Sources available from http://mlg.ucd.ie/datasets/3sources.html, 100leaves from the UCI repository, NewsGroup and WebKB from http://lig-membres.imag.fr/grimal/data.html. CiteSeer and CoRA from [17] are kindly made available by Vladimir Gligorijevic.

Algorithms\Datasets	Baboon		Karate		Pol Books		football		Pol Blogs	
	NMI	time	NMI	time	NMI	time	NMI	time	NMI	time
RGC-H	0.5295	0.062	0.0064	0.18	0.0131	0.54	0.2618	3.34	0.0002	922.40
Reic-H	0.0588	17.73	0.0380	34.67	0.0238	135.70	0.2352	2342.80	—	—
RGC-S	0.0026	0.02	0.0023	0.02	0.0014	0.04	0.2481	0.07	0.0010	3.89
ANMF-S	0.0669	0.05	0.0123	0.02	0.0035	0.04	0.2453	0.06	0.0013	5.34
BNMTF-S	0.0175	2.70	0.0129	7.04	0.0171	24.95	0.3141	173.90	—	—
DANMF	0.2453	0.29	0.1678	0.95	0.4241	0.37	0.4229	4.05	0.1663	751.62
SBM*	0.5295	0.21	0.4006	1.73	0.4674	1.12	0.2798	7.2	0.0931	1092.31
Grad-H-Ad	0.2376	0.14	0.0955	0.14	0.0810	0.48	0.0668	4.00	0.0001	480.40
Grad-H-AdCn	0.1637	0.19	0.1115	0.14	0.1704	0.43	0.1075	2.11	0.0010	610.30
Hard ASBlock	**0.6945**	0.082	0.2896	0.29	0.4304	0.53	0.5805	3.38	0.2552	364.85
Soft ASBlock	**0.6945**	0.061	**0.4016**	0.093	**0.4939**	0.177	**0.6312**	1.34	**0.2716**	147.85

TABLE III: Computing **NMI** and **time** (secs) for various real datasets. A — indicates the method timed out after 3 hours. * indicates that the algorithm was run in C++ (all others were run in Matlab)

Algorithms\Datasets	3sources	100leaves	NewsGroup	WebKB	CiteSeer	CoRA	AVG
RMKMC	0.3248	0.0215	0.3198	0.2701	0.2679	0.2241	0.2380
SNMTF	0.0492	0.6578	0.0156	0.0383	0.2618	0.3032	0.2210
SC-ML	0.2728	0.7269	0.0942	0.2237	0.3208	0.4047	0.3405
MCGL	0.1162	0.8852	0.1143	0.0812	0.0793	0.1135	0.2316
ASMV	0.0958	0.9132	0.0732	0.2709	0.1202	0.1431	0.2694
GraphFuse	0.2632	0.8622	0.3422	0.3271	0.1618	0.2049	0.3602
hard ASBlock	0.3672	0.9087	0.5866	0.4231	0.3153	0.4638	0.5108
soft ASBlock	**0.4128**	**0.9257**	**0.5941**	**0.4409**	**0.3428**	**0.4682**	**0.5308**

TABLE IV: Average performance (NMI) over 30 executions for multi view real datasets.

Related Work

Single view blockmodelling: : [11] and [13] introduced the idea of non-negative matrix tri-factorisation for finding blockmodels for graphs, and produced different multiplicative optimisation approaches. [14] introduced a coordinate descent algorithm to find overlapping position blockmodels. [6] [8] proposed a framework of algorithms and objectives to tackle sparse and noisy graphs. [24] proposed an information theoretic approach of finding blockmodels in evolving graphs.

[25] introduced a mixed membership probabilistic model, where vertices can belong to multiple positions. [26] proposed a generative probabilistic model that takes the difference in the degree distributions of vertices into account. However, their formulation is specifically targeting heavy tailed graphs, while our model is general and can easily incorporate different types of graphs. [12] proposed a null model formulation that sums the difference between the adjacency matrix and the blockmodel approximation of it. A simulated annealing approach was proposed to optimise this. For the factorisation based methods, our evaluation showed our novel image matrix update approach had superior accuracy.

Multi-view community detection:: [9] devised a similar approach to ours, using tri-factorisation to decompose each graph view, but their approach does not impose any constraints on their matrices apart from symmetry for their corresponding M matrix, which means their approach is unable to find general blockmodel decompositions such as the asymmetric structures of a stick or ring. [27] devised a spectral clustering approach, but similar to [9], spectral clustering can only find communities and not other blockmodel structures. [28] proposed a self-weighting multi-view clustering algorithm across multiple graphs, that learns the importance weighting for each view/graph.

Conclusion

We have presented a novel framework for blockmodelling in social networks. Our framework replaces iterative computation of the image matrix by a closed form solution. This leads to faster convergence and more accurate results. We also show that this framework provides a straightforward formulation for the case of multi-view social networks that involve different types of edges between vertices in the social network. Our empirical results on a range of benchmark datasets demonstrate the advantages of our approach in terms of the accuracy of the resulting blockmodel, as well as the computational efficiency of our approach, in comparison to a wide range of state-of-the-art decomposition approaches.

References

[1] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002. [Online]. Available: https://www.pnas.org/content/99/12/7821

[2] J. Lee, S. P. Gross, and J. Lee, "Improved network community structure improves function prediction," *Nature Scientific Reports*, vol. 3, 2013.

[3] S. Lin, Q. Hu, G. Wang, and P. S. Yu, "Understanding community effects on information diffusion," in *Proceedings of PAKDD*, 2015, pp. 82–95.

[4] A. Lacichinetti and S. Fortunato, "Community detection algorithms: a comparative analysis," *Phys. Rev. E*, vol. 80, no. 5, pp. 56–117, 2009.

[5] T. Snijders and K. Nowicki, "Estimation and prediction for stochastic blockmodels for graphs with latent block structure," *Journal of Classification*, vol. 14, no. 1, pp. 75–100, 1997.

[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 *Proceedings of CIKM*, 2013, pp. 811–816.

[7] J. Fiala and D. Paulusma, "The computational complexity of the role assignment problem," in *Automata, Languages and Programming*. Springer Berlin Heidelberg, 2003, pp. 817–828.

[8] J. Chan, C. Leckie, J. Bailey, and K. Ramamohanarao, "Tribac: Discovering interpretable clusters and latent structures in graphs," in *Proceedings of ICDM*, Dec 2014, pp. 737–742.

[9] W. Tang, Z. Lu, and I. S. Dhillon, "Clustering with multiple graphs," in *Proceedings of ICDM*, 2009, pp. 1016–1021. [Online]. Available: https://doi.org/10.1109/ICDM.2009.125

[10] L. Danon, A. Díaz-Guilera, and A. Arenas, "The effect of size heterogeneity on community identification in complex networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2006, no. 11, pp. P11 010–P11 010, nov 2006.

[11] B. Long, Z. Zhang, and P. S. Yu, "A general framework for relation graph clustering," *Knowledge and Information Systems*, vol. 24, no. 3, pp. 393–413, Sep 2010. [Online]. Available: https://doi.org/10.1007/s10115-009-0255-6

[12] J. Reichardt and D. R. White, "Role models for complex networks," *The European Physical Journal B*, vol. 60, no. 2, pp. 217–224, Nov 2007. [Online]. Available: https://doi.org/10.1140/epjb/e2007-00340-y

[13] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *Data Mining and Knowledge Discovery*, vol. 22, no. 3, pp. 493–521, May 2011. [Online]. Available: https://doi.org/10.1007/s10618-010-0181-y

[14] Y. Zhang and D.-Y. Yeung, "Overlapping community detection via bounded nonnegative matrix tri-factorization," in *Proceedings of KDD*, 2012, pp. 606–614. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339629

[15] F. Ye, C. Chen, and Z. Zheng, "Deep autoencoder-like nonnegative matrix factorization for community detection," in *Proceedings of CIKM*, 2018, pp. 1393–1402. [Online]. Available: http://doi.acm.org/10.1145/3269206.3271697

[16] A. Decelle, F. Krzakala, C. Moore, and L. Zdeborová, "Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications," *Phys. Rev. E*, vol. 84, p. 066106, Dec 2011. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.84.066106

[17] V. Gligorijević, Y. Panagakis, and S. Zafeiriou, "Fusion and community detection in multi-layer graphs," in *Proceedigns of ICPR*, Dec 2016, pp. 1327–1332.

[18] X. Cai, F. Nie, and H. Huang, "Multi-view k-means clustering on big data," in *Proceedings of IJCAI*, 2013, pp. 2598–2604. [Online]. Available: http://dl.acm.org/citation.cfm?id=2540128.2540503

[19] D.-K. Nguyen, Q. Tran-Dinh, and T.-B. Ho, "Simplicial nonnegative matrix tri-factorization: Fast guaranteed parallel algorithm," in *Proceedings of NIPs*, 2016, pp. 117–125.

[20] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov, "Clustering on multi-layer graphs via subspace analysis on grassmann manifolds," in *Proceedings of GlobalSIP*, Dec 2013, pp. 993–996.

[21] K. Zhan, C. Zhang, J. Guan, and J. Wang, "Graph learning for multiview clustering," *IEEE Transactions on Cybernetics*, vol. 48, no. 10, pp. 2887–2895, Oct 2018.

[22] K. Zhan, X. Chang, J. Guan, L. Chen, Z. Ma, and Y. Yang, "Adaptive structure discovery for multimedia analysis using multiple features," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1826–1834, May 2019.

[23] E. E. Papalexakis, L. Akoglu, and D. Ience, "Do more views of a graph help? community detection and clustering in multi-graphs," in *Proceedings of FUSION*, July 2013, pp. 899–905.

[24] J. Chan, W. Liu, C. Leckie, J. Bailey, and K. Ramamohanarao, "Seqibloc: Mining multi-time spanning blockmodels in dynamic graphs," in *Proceedings of KDD*, 2012, pp. 651–659.

[25] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, "Mixed membership stochastic blockmodels," *J. Mach. Learn. Res.*, vol. 9, pp. 1981–2014, Jun. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1390681.1442798

[26] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, p. 016107, Jan 2011. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevE.83.016107

[27] X. Dong, P. Frossard, P. Vandergheynst, and N. Nefedov, "Clustering with multi-layer graphs: A spectral perspective," *IEEE Transactions on Signal Processing*, vol. 60, no. 11, pp. 5820–5831, Nov 2012.

[28] F. Nie, J. Li, and X. Li, "Self-weighted multiview clustering with multiple graphs," in *Proceedings of IJCAI*, ser. IJCAI'17, 2017, pp. 2564–2570.