TRIBAC: Discovering Interpretable Clusters and Latent Structure in Graphs

Jeffrey Chan, Christopher Leckie, James Bailey and Kotagiri Ramamohanarao
Department of Computing and Information Systems, University of Melbourne, Australia
{*jeffrey.chan, caleckie, baileyj, kotagiri*}*@unimelb.edu.au*

***Abstract*—Graphs are a powerful representation of relational data, such as social and biological networks. Often, these entities form groups and are organised according to a latent structure. However, these groupings and structures are generally unknown and it can be difficult to identify them. Graph clustering is an important type of approach used to discover these vertex groups and the latent structure within graphs. One type of approach for graph clustering is non-negative matrix factorisation However, the formulations of existing factorisation approaches can be overly relaxed and their groupings and results consequently difficult to interpret, may fail to discover the true latent structure and groupings, and converge to extreme solutions. In this paper, we propose a new formulation of the graph clustering problem that results in clusterings that are easy to interpret. Combined with a novel algorithm, the clusterings are also more accurate than state-of-the-art algorithms for both synthetic and real datasets.**

***Keywords*-Graph clustering; blockmodelling; interpretability; non-negative matrix factorisation**

I. Introduction

Graphs are natural representations of relational data. Often, the entities represented by these graphs form groups of similar relationships. Example groupings include communities of similar interests (e.g., in social networks), and can be used to profile users and provide them with targeted marketing. In addition, these networks are typically organised by a latent structure. For example, a graph representing the email communications of a company is commonly organised in a hierarchical manner, reflecting the company's organisational structure.

An important type of analysis to discover these groupings and latent structures is *graph clustering*, which involves grouping the vertices based on the similarity of their connectivity. Two popular approaches for graph clustering are community detection [1] and blockmodelling [2]. Community detection decomposes a graph into a community structure, where vertices from the same communities have many edges between themselves, and vertices of different communities have few edges. Community structure has been found in many graphs [1], but it is only one of many alternatives for grouping vertices and inferring possible graph structure.

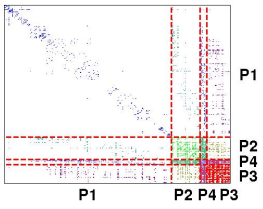
Consider Figure 1, which is an example of a flight routing network. The vertices are airports, and edges model the presence of flights between the two airports. Using a state-of-the-art community finding algorithm [1], it cannot find any structure, as all vertices are placed into a single group. In fact, a reasonable structure of the network consists of four positions (we use the social network analysis nomenclature and refer to a set of vertices as a *position*): hub airports (P_4), large regional airports (P_3), regional airports (P_2) and local airports (P_1). See Figure 1a for the adjacency matrix whose rows and columns are rearranged to illustrate this structure, and the red dotted lines denote the boundaries of positions. This demonstrates that a more general approach to graph clustering is needed.

Blockmodelling is a powerful approach to decomposing graphs [2]. Vertices are in the same position if they have similar patterns of interactions to vertices of other positions. The routing structure of Figure 1 certainly fits this definition, e.g., local airports in P_1 are well connected to hub and regional airports (P_4 and P_3) but not to other local airports in P_1. The inherent structure can be identified by visualising the *image matrix* (Figure 1e), where the positions are the rows and columns and each matrix entry represents the inter-position interactions. The positions and the image matrix clearly summarise the core-periphery structure of the routing in airports, and together form a *blockmodel*. Note that the blockmodelling definition can also discover community structure. Therefore, blockmodelling is a general approach and allows us to discover positions and how they relate to each other, and understand and characterise the underlying structure (e.g., is it a community or core-periphery structure).

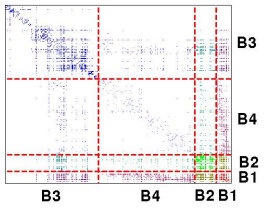
Non-negative matrix factorisation is a powerful technique that approximates a matrix by two low dimensional, non-negative ones [4]. In [5], the authors have shown that the blockmodelling problem can be considered as a non-negative matrix tri-factorisation (three matrix approximation) problem, where the original adjacency matrix is factorised into a position membership matrix (the membership of each vertex to each position, as illustrated in Figure 1c) and an image matrix. While promising results were reported, there are three important, unresolved challenges.

The *first challenge* relates to how blockmodelling is formulated as a factorisation problem. Existing algorithms [6][7][3] focus on constraining the membership and image matrices to be non-negative and do not upper bound their values. This can make the resulting factorisation difficult to interpret. For example, consider the situation were two vertices v_1 and v_2 have memberships of $[1, 0.2, 0.2]$ and $[5, 1, 1]$ to three positions, respectively. v_1 can be considered as mostly affiliated with position 1 (value of 1). But v_2

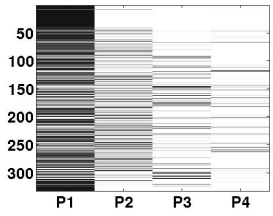
1550-4786/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDM.2014.118



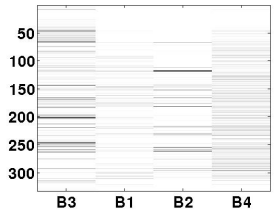
(a) Rearranged adjacency matrix according to TRIBAC.



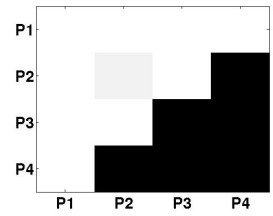
(b) Rearranged adjacency matrix according to BNMTF [3].



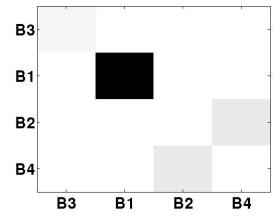
(c) Membership matrix for TRIBAC. Values in 0 to 1.



(d) Membership matrix for BNMTF. Values in 0 to 0.89.



(e) Image matrix for TRIBAC. Values in 0 to 1.



(f) Image matrix for BNMTF. Values in 0 to 123.5.

Figure 1: Airport routing network. In Figures 1a and 1b, the pixels representing the edges of the adjacency matrices are coloured according to the positions of their incident vertices (average coloured if vertices in different positions), and the red dotted lines represent the boundaries of the positions after they are harden/discretised. Darker blocks in Figures 1c to 1f represent larger values.

is also strongly associated with position 1 (value of 5) – is v_2 more strongly affiliated with position 1, or are both v_1 and v_2 more associated with position 1 than the other positions? It is unclear which is the correct interpretation when the memberships only have to be non-negative. In Section III we argue that the memberships of each vertex should sum to 1. With this constraint, the interpretation of the discovered memberships becomes unambiguous, e.g., if v_2 is normalised to 1 then v_1 and v_2 have the same level of affiliation to position 1. In addition, the lack of row sum constraints can result in solutions with vertices that have no membership to any positions (i.e., have no influence on the positions or structure) or their membership values span a large range, both making it even more difficult to interpret and understand the results. For example, consider Figure 1d, which illustrates the membership matrix found by [3]. Rows 10–20 are all zero and the corresponding vertices have no position assignments.

The *second challenge* is that existing work [7][6][3] considers the image matrix as a scaling factor for the membership matrix and ignores its representation as the latent structure. Because of its scaling purpose, the only necessary constraint is non-negativity. This can lead to some image matrices that are difficult to comprehend and interpret. For example, consider the image matrix found by [3] (Figure 1f) with non-zero entries ranging from 4.98 to 123.5. It is difficult to interpret what these entries mean, apart from showing that one entry is dominant. Instead, we argue that each entry should be upper-bounded (e.g., 1 for unweighted graphs). With this restriction, we can interpret the entries of the image matrix as the expected number of edges between vertices of two positions. In addition, because the new formulation has fewer degrees of freedom, this new formulation helps to guide the optimisation to more accurate results. Again consider Figure 1f and 1b. Figure 1b shows that [3] broke up the local airports (P_4) into two positions and failed to identify the highly clustered hub airports (P_1).

The *third challenge* is to develop efficient and accurate blockmodelling algorithms using this new interpretable formulation. State-of-the-art algorithms typically propose post-optimisation normalisation as a potential solution to the membership interpretability challenge. But this is unprincipled and can lead to inferior solutions, which we demonstrate in Section V. Instead, in this paper, we show how to incorporate row sum constraints within the objective, which leads to more accurate and interpretable results.

To address these challenges, we introduce a new non-negative matrix tri-factorisation formulation for graph clustering. In addition, we propose a new algorithm TRIBAC (Non-Negative Matrix **Tri**-Factorisation **B**lockmodelling **a**ssisted by **C**ontraints) that optimises this new formulation and produces interpretable and more accurate memberships and latent structures. In summary, our contributions are:

- We propose a new non-negative matrix tri-factorisation blockmodelling formulation for graph clustering with easier interpretability of the results.
- We propose a novel algorithm TRIBAC that optimises our new formulation and produces more accurate and interpretable clusterings and latent structures than existing methods.

II. Related Work

Non-Negative Matrix Factorisation: Seung et al. [4] were the first to popularise non-negative matrix factorisation in machine learning. Ding et al. [8] extended this to a three factor factorisation and introduce orthogonality constraints on the membership factors. Long et al. [7] and Wang et al. [6] introduced the idea of non-negative matrix tri-factorisation for finding blockmodels for graphs, and produced different multiplicative optimisation approaches.

Zhang et al. [3] introduced a coordinate descent algorithm to find overlapping position blockmodels. Chan et al. [5] proposed a framework of algorithms and objectives to tackle sparse and noisy graphs. All of these approaches do not impose row sum constraints, making it more difficult to interpret the discovered blockmodels.

Blockmodelling: We concentrate on those blockmodelling algorithms that are most similar to our work. In [9], Chan et al. proposed the novel approach of finding blockmodels in evolving graphs using a minimum description length (MDL) coding approach, with more ideal blockmodels resulting in shorter codes. However, the MDL principle breaks down when applied to larger graphs. Airold et al. [10] introduced a mixed membership probabilistic model, where vertices can belong to multiple positions. Edges, the position of vertices and other variables are modelled as random variables. However, the fitting process can be slow.

III. Non-Negative Matrix Factorisation Blockmodelling

In this section, we introduce the key concepts of blockmodelling [2], the notation used and our formulation of blockmodelling as a matrix factorisation problem.

A graph $G(V, E)$ consists of a set of vertices V and a set of edges $E : V \times V$. The edges can be represented by an adjacency matrix $\mathbf{A} \in \mathcal{R}_+^{n \times n}$ whose rows and columns are indexed by V and n is the number of vertices. For unweighted graphs, $\mathbf{A} \in \{0, 1\}^{n \times n}$. For notational convenience, we denote the number of edges by m.

A blockmodel decomposes a graph $\mathbf{A}$ into a set of k vertex *positions* represented by a membership matrix $\mathbf{C}$ (with dimensions n by k) and a lower dimension matrix called the image matrix $\mathbf{M}$ (dimensions k by k). The image matrix represents the position to position interactions and the overall latent structure. The blockmodel decomposition approximates $\mathbf{A}$ as $\mathbf{CMC}^T$. The challenge of blockmodelling is to find a $\mathbf{C}$ and $\mathbf{M}$ which yield a good approximation to $\mathbf{A}$, as well as themselves being interpretable and understandable. Interpretability means the values of $\mathbf{C}$ and $\mathbf{M}$ should fall into a particular range and have meaning. We first describe how existing work approach this and then our proposed solution.

In [6] and [7], the authors defined $\mathbf{C}$ and $\mathbf{M}$ as non-negative. The focus of [6] and [7] is to find the best vertex grouping, where $\mathbf{M}$ is regarded as a scaling matrix for $\mathbf{C}$. In addition, our experience has been that with the only constraints on $\mathbf{M}$ (and $\mathbf{C}$) being non-negativity, there are too many degrees of freedom for the values of $\mathbf{M}$, and it is more difficult to recover the latent structure. Consider the following scenario.

Let $\mathbf{CMC}^T$ be an exact approximation of $\mathbf{A}$, i.e., $\mathbf{A} = \mathbf{CMC}^T$. Consider the equation for each element of $\mathbf{A}$: $\mathbf{A}_{ij} = \sum_x^n \sum_y^n \mathbf{C}_{ix} M_{xy} \mathbf{C}_{jy}$. We wish to analyse what values M_{xy} can take in order to satisfy equality. Let $\mathbf{A}_{ij} = 1$. Assume each vertex has only one non-zero membership (i.e., $\mathbf{C}_{ia} > 0$ and $\forall x \neq a$, then $\mathbf{C}_{ix} = 0$). If $\mathbf{C}_{ia} = \mathbf{C}_{ib} = 1$, then M_{ab} should equal 1. If $\mathbf{C}_{ia} = \mathbf{C}_{ib} = 0.1$, then $M_{ab} = 100$. As it can be seen, the values are arbitrary. This can lead to extreme values (e.g., in our experiments in Section V we witness values of $\mathbf{C}$ in the order of 10^{50}).

Furthermore, as explained in Section I, it is difficult to interpret the values of $\mathbf{M}$ when there are only non-negativity constraints, apart from zero corresponding to no interactions between two vertex groups. In summary, $\mathbf{M}$ and $\mathbf{C}$ need additional constraints in order to obtain interpretable and accurate soft clusterings.

A. Additional Constraints for Interpretable $\mathbf{C}$ and $\mathbf{M}$

For soft clustering, we argue that the membership of each vertex should sum to the same constant value. If we desire a probabilistic interpretation, then that constant should be 1 (i.e., $\sum_k \mathbf{C}_{ik} = 1, \forall v_i \in V$).

$\mathbf{M}$ could take on any value in theory, but we believe that $\mathbf{M}$ should lie in $[0, 1]$ for unweighted graphs. In this case, we can naturally interpret M_{xy} as the expected number of edges between a vertex with full membership (i.e., 1) in position x to a vertex with full membership in position y. Restricting $\mathbf{M}$ to $[0, 1]$ permits us to make this type of interpretation of the discovered $\mathbf{M}$ matrix, which the other unbounded formulations do not.

B. Optimisation Objective

We can estimate the approximate error by a number of different loss functions. In this paper, we use the popular Euclidean loss (sum of squared errors) and the aim is to find a $\mathbf{C}$ and $\mathbf{M}$ that minimises the following error:

$$J(\mathbf{C}, \mathbf{M}) = \min_{\mathbf{C},\mathbf{M}} ||\mathbf{A} - \mathbf{CMC}^T||_F^2 \quad (1)$$

$$\texttt{s.t. } \mathbf{C} \in [0, 1]^{n \times k}, \mathbf{M} \in [0, 1]^{k \times k}$$

$$\sum_k \mathbf{C}_{ik} = 1, 1 \leq i \leq n$$

For weighted graphs, we have $\mathbf{M} \in [0, \max_{ij}(A_{ij})]^{k \times k}$.

IV. TRIBAC

None of the existing measures [5][6][7][3] can solve Equation 1. Hence, in this section, we describe our proposed TRIBAC optimisation algorithm to optimise Equation 1.

Blockmodelling of three or more positions is a NP-Hard problem [5]. Optimisation of $\mathbf{M}$ or $\mathbf{C}$ can be individually convex, but optimisation of both is non-convex. Hence, existing methods and our proposed algorithm TRIBAC alternate between optimising for $\mathbf{M}$ and $\mathbf{C}$ until some convergence criterion is satisfied.

Optimising C: Holding $\mathbf{M}$ constant, we solve for $\mathbf{C}$. We propose a multiplicative rule approach to solve this subproblem.

$$\mathbf{C}_{ij} = \mathbf{C}_{ij} \left(\frac{\Theta^-_{ij} + \sum_b \Theta^+_{ib} \mathbf{C}_{ib}}{\Theta^+_{ij} + \sum_b \Theta^-_{ib} \mathbf{C}_{ib}} \right)^{\frac{1}{4}} \quad (2)$$

where $\mathbf{\Theta}^-_{ij} = \mathbf{A}^T\mathbf{CM} + \mathbf{ACM}^T$ and $\mathbf{\Theta}^+_{ij} = \mathbf{CM}^T\mathbf{C}^T\mathbf{CM} + \mathbf{CMC}^T\mathbf{CM}^T$.

Theorem 1: $J(\mathbf{C})$ is non-increasing under the update rule of Equation 2.

Proof: See supplementary material[1]. ■

Optimising M: To solve $\mathbf{M}$ for Equation 1 requires solving for box constraints on $\mathbf{M}$, which requires computing the active and inactive constraint sets. This is very difficult for the multiplicative approach, hence instead we propose to use a coordinate descent approach, similar to [3][5], but with an additional upper limit constraint on $\mathbf{M}$. Using the unit basis as the conjugate basis, we solve the following problem for the optimal step size ψ, subject to $\mathbf{M} \in [0, 1]$:

$$\min_{\psi} L_{i,j}(\psi) = ||(\mathbf{A} - \mathbf{C}(\mathbf{M} + \psi\mathbf{E}_{i,j})\mathbf{C}^T)||^2_F \quad (3)$$

where $\mathbf{E} \in [0, 1]^{k \times k}$. After expanding the RHS of Equation 3 and taking the derivative w.r.t ψ and equating it to 0 we obtain: $\psi = \frac{\mathtt{Tr}(\mathbf{Y0}^T\mathbf{Y1})}{\mathtt{Tr}(\mathbf{Y1}^T\mathbf{Y1})}$, where $\mathbf{Y0} = \mathbf{A} - \mathbf{CMC}^T$ and $\mathbf{Y1} = \mathbf{CE}_{i,j}\mathbf{C}^T$. We require $0 \leq M_{ij} + \psi \leq 1$. Hence:

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

Theorem 2: TRIBAC is non-increasing with respect to $J(\mathbf{C}, \mathbf{M})$.

Proof: See supplementary material[1]. ■

In summary, there are three existing algorithms, their post-optimisation, normalising variants and TRIBAC for matrix factorisation blockmodelling. We compare the algorithm on their: a) constraints on $\mathbf{C}$; b) constraints on $\mathbf{M}$; and c) constraints on the row sum. In Table I we illustrate each algorithm and what characteristics they possess.

V. Evaluation

In this section, we compare the accuracy of the existing and TRIBAC algorithms to find the true $\mathbf{C}$ and the latent structures. We use both real and synthetic datasets to evaluate our algorithms. The synthetic datasets allow us to control and evaluate how different graph characteristics affect the algorithms. We also use well studied datasets from social network analysis [3] for our evaluation.

A. Datasets and Evaluation Criteria

1) Datasets: We generate our synthetic datasets with the aim of evaluating how noise, sparsity and the latent structure affect the accuracy of the different algorithms. We used the same underlying generation approach to construct the synthetic graphs[1], which all have 100 vertices and 5 positions.

[1] Available at people.unimelb.edu.au/jeffreyc.

Name	Vert. #	Edge #	Directed?
Baboon	14	23	N
Monastery	18	34	Y
Karate	34	78	N
Les Mis'erables	77	254	N
Politic Books	105	441	N
Adj-nouns Adjacencies	112	425	N
College Football	115	613	N
Jazz Musicians	198	2742	N
C. Elegans	297	2359	Y
Airport Routing	332	4252	N
Politic Blogs	1490	19090	N

Table II: Statistics of real graphs tested.

This approach was used in [5] and can be considered as the reverse of the blockmodelling problem. We first generate the memberships ($\mathbf{C}$) and the image matrix ($\mathbf{M}$). Then we generate the graph ($\mathbf{A}$) using $\mathbf{A} = \mathbf{CMC}^T$. We generate $\mathbf{C}$ by drawing from a hyper-geometric distribution, where the probability of a vertex to each position is the position's relative size. To generate $\mathbf{M}$, we need to decide which blocks are dense, which is dependent on the evaluation task. We generated $\mathbf{M}$ such that their dense blocks replicate two common graph structures: community and hierarchy.

To vary the sparsity, we change the densities of the dense blocks in $\mathbf{M}$. To vary the noise, we generate the desired graph structure as a true image matrix, $\mathbf{M}_{act}$. A background image matrix, $\mathbf{M}_{back}$, with the same expected number of edges, is a uniformly random distributed assignment of the edges. We then control the amount of noise in the graph by weighting the contribution of $\mathbf{M}_{back}$ and $\mathbf{M}_{act}$, via $\mathbf{M} = (1 - \lambda)\mathbf{M}_{act} + \lambda\mathbf{M}_{back}$, where $0 \leq \lambda \leq 1$.

We evaluate the algorithms using 11 real networks[2] (see Table II). These are graphs that are commonly used to evaluate social network analysis and blockmodelling algorithms.

2) Evaluation Criteria: To evaluate how well the memberships are recovered, we use the same approach as [6][7] by setting the cluster label of a vertex to the maximum of its row in $\mathbf{C}$ (i.e., $max_k(C_{ik})$), then using hard cluster comparison measures. Following those papers, we used Normalised Mutual Information (NMI) [11].

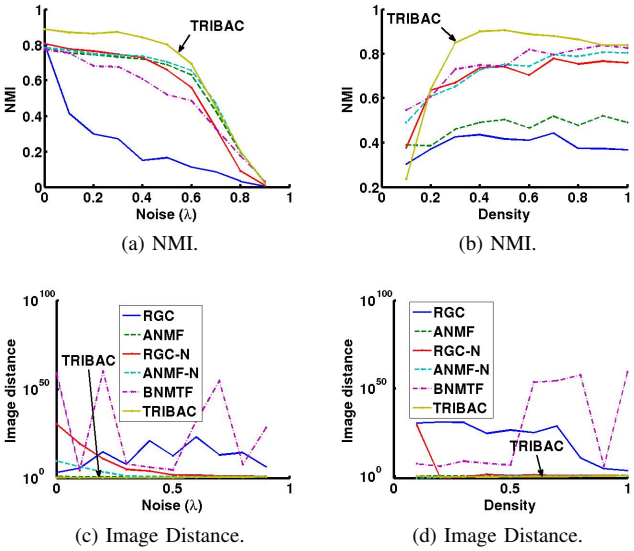
To evaluate how well the latent structure is recovered, we compute the Euclidean distance between the reference and recovered image matrices. Because there is a correspondence issue with position labels, we first find the best permutation matrix $\mathbf{P} \in \{0, 1\}^{k \times k}$ that minimises the distance between the membership matrices: $\mathbf{P} = \arg\max_{\mathbf{P}} d(\mathbf{C}^{(1)}, \mathbf{C}^{(2)}\mathbf{P}^T)$. Then we can compute the distance between the images as $d(\mathbf{M}^{(1)}, \mathbf{M}^{(2)}) = ||\mathbf{M}^{(1)} - \mathbf{PM}^{(2)}||^2_F$.

For the real datasets, there are no reference image matrices to compare against. Hence, we use an encoding measure [9] to evaluate how well the factorisation conforms to ideal clustering structures. It uses a codeword that encodes the graph using the blockmodel structures. If the blockmodel is

[2] Available at http://www-personal.umich.edu/~mejn/netdata/

Name	Position Approach	Image Approach	$\mathbf{C}$ constraint	$\mathbf{M}$ constraint	$\sum_k \mathbf{C}_{ik} = 1$?
RGC [7]	Multiplicative	Multiplicative	$0 \leq \mathbf{C}$	$0 \leq \mathbf{M}$	N
ANMF [6]	Multiplicative	Multiplicative	$0 \leq \mathbf{C}$	$0 \leq \mathbf{M}$	N
BNMTF [3]	Coordinate Descent	Coordinate Descent	$0 \leq \mathbf{C} \leq 1$	$0 \leq \mathbf{M}$	N
RGC-N [7]	Multiplicative, ad-hoc	Multiplicative	$0 \leq \mathbf{C} \leq 1^*$	$0 \leq \mathbf{M}$	Y*
ANMF-N [6]	Multiplicative, ad-hoc	Multiplicative	$0 \leq \mathbf{C} \leq 1^*$	$0 \leq \mathbf{M}$	Y*
TRIBAC	Multiplicative	Coordinate Descent	$0 \leq \mathbf{C} \leq 1$	$0 \leq \mathbf{M} \leq 1$	Y

Table I: Summary of the algorithms and objectives. Names in italic represent algorithms proposed in this paper. [*] Achieved with ad-hoc post-optimisation normalisation.



(a) NMI. (b) NMI.

(c) Image Distance. (d) Image Distance.

Figure 2: NMI and the image distance results for the synthetic datasets. The 1st column are results for the hierarchy structure as we vary the background noise levels. The 2nd column are the results for varying the sparsity on community structured graphs. All the lines are consistently coloured across the plots, and their legends are available in the 2nd row.

an ideal one, than the codeword length is minimised.

We used the default number of iterations for RGC (400), ANMF (400) and BNMTF (100). For a fair comparison, we set the number of iterations for TRIBAC to 100 and for each experimental run, we initialise each of the algorithms with the same $\mathbf{C}$ and $\mathbf{M}$. All implementations are in Matlab 2013b and the experiments ran on a PC with an Intel Core i7-4600U CPU and 12GB of memory.

B. Results: Varying Noise, Structure and Sparsity

Figure 2 shows the results of the algorithms when we vary the background noise levels (1st row) and sparsity (2nd row) of the generated datasets. The results are the average of 50 runs for each of the five graphs generated at each parameter setting. We vary the noise levels from 0 to 0.9 and the sparsity from 0.1 to 1.

Algorithm	Code Len.	$\sum \mathbf{C}$	$\sum \mathbf{M}$	Zero rows	Time (s)
RGC	236497	67.9	70.94	268	15.3
ANMF	240967	250.5	2.457	268	18.7
RGC-N	237321	1224	0.122	268	15.6
ANMF-N	239220	1224	0.116	268	19.0
BNMTF	230667	100.5	19.2	113.7	749
TRIBAC	**217653**	1490	1.002	0	31.6

Table III: Results for the PolBlog dataset. "Zero rows" column is the number of rows in $\mathbf{C}$ that sum to 0.

First, we consider the NMI results when we vary the noise (Figure 2a). TRIBAC is generally more accurate in recovering the generated positions, particularly at lower background noise levels. It is interesting to observe that for RGC, the ad-hoc row normalised results (RGC-N) have much higher accuracy than no normalisation, which further highlights the importance of the right scaling for the factors.

Now consider the image distance (Figure 2c). The figures show that RGC and BNMTF have large scaling issues, with some of their image matrix entries being incredibly large. Even for ANMF and the normalised variants RGC-N and ANMF-N, they can have sizeable image distances from the ground truth over some of the noise levels. Only TRIBAC consistently has low image distance across all structures and noise levels. This again shows the importance of proper scaling for both the image and membership matrices.

Figures 2b and 2d show the NMI and image difference results when the density (sparsity) of the generated graphs are varied. Apart from the most sparse graphs (density of 0.1), TRIBAC has the highest NMI across the other density values. Again, the image differences for RGC, RGC-N and BNMTF are very large, indicating the difficulty with their interpretation and issues with their formulations.

C. Results: Real Dataset Evaluation

Table III shows the results for the PolBlog dataset. We compare the algorithms based on their code lengths, sums of their membership and image matrices, number of zero sum rows and the running times. We also compare the code length of the algorithms across all 11 real datasets in Table IV. All reported results are the average of 100 runs.

First consider Table III. For PolBlog, TRIBAC has the shortest code length (2nd column) and most able to recover useful structure. Next, consider the $\sum \mathbf{C}$ and $\sum \mathbf{M}$ (columns 3 and 4), which provide an indication of the scales of $\mathbf{C}$

Algorithm	Baboon	Monast.	Karate	Les Mis.	Pol. Books	Adj-Nouns	Football	Jazz	C.Elegans	Airport	Pol. Blogs
RGC	140.5	142.9	585.5	2065	3838	4349	3987	14343	15086	21679	2.38×10^5
ANMF	140.4	**122.7**	576.2	2111	3746	4331	4073	12964	15028	22002	2.38×10^5
RGC-N	141.7	148.7	581.0	1987	3760	4350	**3873**	14343	15086	23344	2.38×10^5
ANMF-N	141.6	135.3	577.1	2039	**3710**	4347	3980	**12980**	14956	23534	2.41×10^5
BNMTF	140.3	144.4	578.4	2117	3772	4349	4742	12984	14826	22644	2.29×10^5
TRIBAC	**135.1**	138.4	**555.0**	**1585**	3802	**4091**	4017	13334	**14038**	**16486**	$\mathbf{2.17}\times10^5$

Table IV: Average coding length results (in bits) for the different algorithms and datasets.

and $\mathbf{M}$ and their interpretability. Both RGC and BNMTF have very large image matrix sums, meaning that some of the entries are excessively large and difficult to understand. They also cause the scales for $\mathbf{C}$ (see the $\sum \mathbf{C}$ column) to be relatively small, again making them harder to understand. In contrast, all the row normalised algorithms (RGC-N, ANMF-N and TRIBAC) have similar $\mathbf{C}$ sums that equal the number of vertices and their image sums are also similar. Column 5 shows the number of rows with zero sum. Reconsider Table III, which shows RGC and ANMF and their normalised variants all having 268 vertices (rows) out of 1490 vertices with zero membership to all positions. BNMTF have 113.7 vertices, which is still a large number, while TRIBAC has no zero rows. This demonstrates the importance of the row sum constraints on $\mathbf{C}$ to ensure good and understandable solutions. Column 6 shows the running times of the algorithms. BNMTF has much longer running times than the other algorithms, which can make it impractical to use. All the other algorithms, including TRIBAC, generally have comparable running times.

We now consider the codeword lengths for all 11 datasets (Table IV). We can observe that TRIBAC has the shortest codewords for 7 out of the 11 datasets, demonstrating that TRIBAC is generally more accurate at recovering good blockmodel structure than other algorithms. Furthermore, the row normalised/constrained algorithms (RGC-N, ANMF-N and TRIBAC) have the shortest code lengths in 10 out of 11 datasets, emphasising the importance of the row sum constraint of $\mathbf{C}$ in the discovery of higher quality blockmodels.

VI. Conclusion

In this paper, we have described the important problem of blockmodelling in graph clustering and shown why the current state-of-the-art factorisation methods cannot discover blockmodels accurately in graphs. We proposed a new objective that incorporates additional constraints on the membership and image factors. These constraints impose interpretable semantics onto the factorisation, as well as helping to avoid extreme blockmodels. In addition, we have proposed a novel algorithm TRIBAC, that combines multiplicative and coordinate descent approaches to optimise our new objective. In our evaluation, we showed that TRIBAC can recover generated blockmodels more accurately than existing algorithms as well as produce blockmodels that have the clearest structure, while having comparable running time to the state-of-the-art methods.

For future work, we plan to investigate approaches such as [12] to speed up the bottleneck coordinate descent part of TRIBAC. Another potential direction is to extend TRIBAC to possiblistic clustering but at the same time avoid extreme values sometimes returned by BNMTF.

References

[1] M. Rosvall and C. Bergstrom, "Maps of random walks on complex networks reveal community structure," *PNAS*, vol. 105, pp. 1118–1123, 2008.

[2] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications*. Cambr. Univ. Press, 1994.

[3] Y. Zhang and D. Yeung, "Overlapping Community Detection via Bounded Nonnegative Matrix Tri-Factorization," in *Proceedings of KDD*, 2012, pp. 606–614.

[4] D. Lee and H. Seung, "Algorithms for Non-negative Matrix Factorization," *Proceedings of NIPS*, vol. 13, pp. 556–562, 2001.

[5] J. Chan, W. Liu, 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.

[6] F. Wang, T. Li, X. Wang, S. Zhu, and C. Ding, "Community discovery using nonnegative matrix factorization," *DMKD*, vol. 22, no. 3, pp. 493–521, 2010.

[7] B. Long, Z. Zhang, and P. Yu, "A general framework for relation graph clustering," *KAIS*, vol. 24, no. 3, pp. 393–413, 2009.

[8] C. Ding, T. Li, W. Peng, and H. Park, "Orthogonal Nonnegative Matrix Tri-Factorizations for Clustering," in *Proceedings of KDD*, 2006, pp. 126–135.

[9] J. Chan, W. Liu, C. Leckie, J. Bailey, and R. Kotagiri, "SeqiBloc: Mining Multi-time Spanning Blockmodels in Dynamic Graphs," in *Proceedings of KDD*, 2012, pp. 651–659.

[10] E. Airoldi, D. Blei, S. Fienberg, and E. Xing, "Mixed membership stochastic blockmodels," *JLMR*, vol. 9, pp. 1981–2014, 2008.

[11] M. Meila, "Comparing clusterings - an information based distance," *J. Multi. Anal.*, vol. 98, no. 5, pp. 873–895, 2007.

[12] C. Hsieh and I. Dhillon, "Fast coordinate descent methods with variable selection for non-negative matrix factorization," in *Proceedings of KDD*, 2011, pp. 1064–1073.