Scalable Spectral Clustering with Group Fairness Constraints

Ji Wang
Univ. of California, Davis
jiiwang@ucdavis.edu

Ding Lu
University of Kentucky
Ding.Lu@uky.edu

Ian Davidson
Univ. of California, Davis
indavidson@ucdavis.edu

Zhaojun Bai
Univ. of California, Davis
zbai@ucdavis.edu

Abstract

There are synergies of research interests and industrial efforts in modeling fairness and correcting algorithmic bias in machine learning. In this paper, we present a scalable algorithm for spectral clustering (SC) with group fairness constraints. Group fairness is also known as statistical parity where in each cluster, each protected group is represented with the same proportion as in the entirety. While FairSC algorithm (Kleindessner et al., 2019) is able to find the fairer clustering, it is compromised by high computational costs due to the algorithm's kernels of computing nullspaces and the square roots of dense matrices explicitly. We present a new formulation of the underlying spectral computation of FairSC by incorporating nullspace projection and Hotelling's deflation such that the resulting algorithm, called s-FairSC, only involves the sparse matrix-vector products and is able to fully exploit the sparsity of the fair SC model. The experimental results on the modified stochastic block model demonstrate that while it is comparable with FairSC in recovering fair clustering, s-FairSC is $12\times$ faster than FairSC for moderate model sizes. s-FairSC is further demonstrated to be scalable in the sense that the computational costs of s-FairSC only increase marginally compared to the SC without fairness constraints.

1 INTRODUCTION

Machine learning (ML) is widely used to automate decisions in areas such as targeting of advertising, issuing of credit cards, and admission of students. While powerful, ML is vulnerable to biases encoded in the raw data or brought by underlying algorithms against certain groups or individuals, thus resulting in *unfair* decisions (Hardt et al., 2016; Chouldechova and Roth, 2018). Examples of algorithmic unfairness in real life are documented in Flores et al. (2016); Pethig and Kroenung (2022). In the context of algorithmic decision-making, *fairness* commonly refers to the prohibition of any favoritism toward certain groups or individuals based on their natural or acquired characteristics. Such characteristics are also known as sensitive attributes, for instance, gender, ethnicity, sexual orientation, and age group (Mehrabi et al., 2021). The rising stake and growing societal impact of ML algorithms have motivated the study of fairness in academia and industry. Various efforts have been attempted at modeling fairness and correcting algorithmic biases in both supervised and unsupervised ML, see e.g., Dwork et al. (2012); Chierichetti et al. (2017); Samadi et al. (2018); Agarwal et al. (2019); Aghaei et al. (2019); Amini et al. (2019); Zhang et al. (2019); Davidson and Ravi (2020).

There are a wide range of studies in fair ML depending on the choice of algorithms and fairness definitions. In this paper, we focus on spectral clustering (SC) with group fairness constraints. The notion of group fairness is an idea of statistical parity by Feldman et al. (2015); Zemel et al. (2013). It ensures that overall proportion of members in a group receiving positive (negative) consideration is identical to the proportion of the population as a whole. In Kleindessner et al. (2019), a mathematical model is proposed to incorporate the group fairness into the SC framework, FairSC for short. For synthetic networks, FairSC is shown to recover ground-truth clustering with high probability. For real-life datasets, FairSC identifies a fairer clustering compared to SC without fairness constraints. Unfortunately, FairSC can only handle moderate model sizes due to the computational costs in the computations of orthonormal bases of large nullspaces and the square roots of dense matrices. FairSC is not scalable.

In this paper, we present a new formulation of spectral computation of FairSC by incorporating nullspace projection and Hotelling's deflation. The resulting algorithm is named Scalable FairSC, or s-FairSC. In s-FairSC, all computational kernels only involve the sparse matrix-vector multi-

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

plications and therefore are capable of fully exploiting the sparsity of the fair SC model. A comparison of s-FairSC with FairSC on the modified stochastic block model exhibits 12x speed up for moderate model sizes. Meanwhile, s-FairSC is comparable with FairSC in recovering fair clustering. The s-FairSC is further demonstrated to be scalable in the sense that it only has a marginal increase in computational costs compared to the SC without fairness constraints.

The remainder of the paper is organized as follows. Section 2 reviews the basics of spectral clustering and group fairness. Section 3 first recaps FairSC and then derives s-FairSC. Section 4 starts with the descriptions of experimental datasets and then demonstrates the improvements of s-FairSC in computational efficiency and scalability while maintaining the same accuracy as FairSC. Concluding remarks are in Section 5.

2 SC AND FAIR SC

2.1 Clustering and fair clustering

Given a set of data, the goal of clustering is to partition the set into subsets such that data in the same subset is more similar to each other than in those of the other subsets. Mathematically, let $\mathcal{G}(V, W)$ denote a weighted and undirected graph with a set of vertices (data) $V = \{v_1, v_2, \ldots, v_n\}$ and a *weighted adjacency matrix* $W = (w_{ij}) \in \mathbb{R}^{n\times n}$. The matrix W encodes the edge information. We assume $w_{ij} \geq 0$ and $w_{ii} = 0$. If $w_{ij} > 0$, then (v_i, v_j) is an edge with weight w_{ij}. We denote by $d_i = \sum_{j=1}^{n} w_{ij}$ the *degree* of a vertex v_i and $D = \mathrm{diag}(d_1, d_2, \ldots, d_n)$, the *degree matrix* of $\mathcal{G}(V, W)$. For simplicity, we assume there is no isolated vertex, and consequently, D is positive definite.

The task of clustering is to partition V into k disjoint subsets (*clusters*):

$$V = C_1 \cup \cdots \cup C_k, \tag{2.1}$$

such that the total weights within each subset are large and between two different subsets are small. The clustering (2.1) can be encoded in a *clustering indicator matrix* $H = (h_{i\ell}) \in \mathbb{R}^{n\times k}$, where for $i = 1, \ldots, n$ and $\ell = 1, \ldots, k$,

$$h_{i\ell} := \begin{cases} 1, & \text{if } v_i \in C_\ell, \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

Now let us consider how to enforce *group fairness* in clustering. The *groups* refer to a partition of the collected data V (e.g., based on sensitive attributes such as gender and race). We denote the groups with h non-empty subsets:

$$V = V_1 \cup \cdots \cup V_h, \tag{2.3}$$

where $V_i \cap V_j = \emptyset$ for $i \neq j$. Groups can be encoded in a *group indicator matrix* $G = (g_{is}) \in \mathbb{R}^{n\times h}$, where for $i = 1, \ldots, n$ and $s = 1, \ldots, h$

$$g_{is} := \begin{cases} 1, & \text{if } v_i \in V_s, \\ 0, & \text{otherwise.} \end{cases} \tag{2.4}$$

The *group fairness* for clustering refers to the case that objects from all groups are presented proportionately in each cluster, also known as statistical parity. The following definition is due to Kleindessner et al. (2019), which extends the notion of group fairness by Chierichetti et al. (2017).

Definition 2.1. A clustering (2.1) is *group fair* with respect to a group partition (2.3) if in each cluster the objects from each group are presented proportionately as in the original dataset. That is, for $s = 1, 2, \cdots, h$ and $\ell = 1, 2, \cdots, k$,

$$\frac{|V_s \cap C_\ell|}{|C_\ell|} = \frac{|V_s|}{|V|}, \tag{2.5}$$

where $|V|$ denotes the number of vertices in V.

The fairness condition (2.5) can be represented compactly using the matrices H in (2.2) and G in (2.4). To do so, let us first introduce matrices

$$M := G^T H \quad \text{and} \quad Z := (G^T \mathbf{1}_n) \cdot (H^T \mathbf{1}_n)^T, \tag{2.6}$$

where $\mathbf{1}_n$ is a length-n column vector with all elements equal to 1. Then the entries of M and Z are $m_{s\ell} = |V_s \cap C_\ell|$ and $z_{s\ell} = |V_s| \cdot |C_\ell|$ for $s = 1, 2, \ldots, h$ and $\ell = 1, 2, \ldots, k$. Consequently, the fairness condition (2.5) is equivalent to

$$n \cdot M = Z, \tag{2.7}$$

where $n = |V|$. By (2.6), equation (2.7) holds if and only if

$$F_0^T H = 0, \tag{2.8}$$

where $F_0 := G - \mathbf{1}_n z^T \in \mathbb{R}^{n\times h}$ and $z := (G^T \mathbf{1}_n)/n \in \mathbb{R}^h$. Observe that according to the definition of G in (2.4), the entries of the vector z satisfy $z_i = |V_i|/n$, for $i = 1, 2, \ldots, h$.

The following lemma shows that it is sufficient to use the first $h-1$ columns of F_0 in the constraint (2.8). The idea of using the first $h-1$ columns of F_0 is from Kleindessner et al. (2019). Extended from this idea, we justify the choice of $h-1$ through the rank of F_0 and prove that $h-1$ is indeed the least number of columns necessary.

Lemma 2.1. *Let H be the clustering indicator matrix in* (2.2)*, and G be the group indicator matrix as in* (2.4)*. Let $F_0 \in \mathbb{R}^{n\times h}$ be as defined in* (2.8) *and $F := F_0(:, 1 : h-1) \in \mathbb{R}^{n\times(h-1)}$ consists of the first $h-1$ columns[1] of F_0. Then,*

[1]By changing the order of groups $\{V_s\}$, the result also holds for F with arbitrary $(h-1)$ columns of F_0.

(i) $\operatorname{rank}(F_0) = \operatorname{rank}(F) = h - 1$.

(ii) *The clustering* (2.1) *is group fair with respect to* (2.3) *if and only if*

$$F^T H = 0. \tag{2.9}$$

Proof. See Appendix A.1. □

2.2 SC and fair SC

SC. The objective function of a normalized cut (NCut) (Shi and Malik, 2000; Ng et al., 2001) is

$$\text{NCut}(C_1, \cdots, C_k) := \sum_{\ell=1}^{k} \frac{\text{Cut}(C_\ell, V \backslash C_\ell)}{\text{vol}(C_\ell)}, \tag{2.10}$$

where

$$\text{Cut}(C_\ell, V\backslash C_\ell) = \sum_{\substack{v_i \in C_\ell \\ v_j \in V\backslash C_\ell}} w_{ij}, \ \text{vol}(C_\ell) = \sum_{v_i \in C_\ell} d_i.$$

The NCut function calculates the scaled total weights of *between-cluster* edges, and measures the similarities between the clusters: a smaller NCut value implies better clustering. The scaling in (2.10) by $\text{vol}(C_\ell)$ takes into account the size of the cluster to avoid outliers. Hence, the goal is to minimize NCut.

The NCut function (2.10) admits a nice expression using the clustering indicator matrix H. Let us first scale the clustering indicator matrix H in (2.2) to

$$H \leftarrow H\widehat{D}^{-1}, \tag{2.11}$$

where $\widehat{D} = \operatorname{diag}(\sqrt{\text{vol}(C_1)}, \cdots, \sqrt{\text{vol}(C_k)})$. We call the new H the *scaled indicator matrix*. For convenience, we use the same notation for both scaled and unscaled cluster indicator matrices. Then, the NCut function (2.10) is recast to the following matrix trace:

$$\text{NCut}(C_1, \cdots, C_k) = \operatorname{Tr}(H^T L H), \tag{2.12}$$

where $L = D - W$ is the *Laplacian* of $\mathcal{G}(V, W)$. Note that under the assumption of connectivity of $\mathcal{G}(V, W)$, L is semi-positive definite and has exactly one zero eigenvalue. By (2.12), the NCut minimization is equivalent to the trace minimization problem

$$\min \operatorname{Tr}(H^T L H) \quad \text{s.t.} \quad H \text{ is of the form (2.11).} \tag{2.13}$$

Solving problem (2.13) directly is NP-hard (Wagner and Wagner, 1993). In practice, the following relaxed version of the problem (2.13) is solved:

$$\min_{H \in \mathbb{R}^{n\times k}} \operatorname{Tr}(H^T L H) \quad \text{s.t.} \quad H^T D H = I_k. \tag{2.14}$$

Once an optimal solution H of (2.14) is obtained, a discrete solution of (2.13) can be obtained by a properly chosen criterion. Subsequently, the k-means algorithm (for the rows of H) is applied for clustering, although other techniques are also available; see, e.g., Bach and Jordan (2003); Lang (2005).

Problem (2.14) is a classical trace minimization problem initially studied in Fan (1949). The following theorem can be found in (Horn and Johnson, 2012, p. 248).

Theorem 2.1. *For a symmetric matrix $A \in \mathbb{R}^{n\times n}$,*

$$\min_{X^T X = I_k} \operatorname{Tr}(X^T A X) = \operatorname{Tr}(X_*^T A X_*) = \sum_{i=1}^{k} \lambda_i,$$

where $\lambda_1 \le \cdots \le \lambda_k$ are the k smallest eigenvalues of A, and columns of $X_* \in \mathbb{R}^{n\times k}$ are the corresponding eigenvectors.

By Theorem 2.1, we can reformulate problem (2.14) to the standard trace minimization problem by a change of variables $X = D^{1/2}H$:

$$\min_{X \in \mathbb{R}^{n\times k}} \operatorname{Tr}(X^T L_{\text{n}} X) \quad \text{s.t.} \quad X^T X = I_k, \tag{2.15}$$

where $L_{\text{n}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, which is known as the *normalized Laplacian*, and then compute the eigenvectors X corresponding to the k smallest eigenvalues of L_{n}. The solution of the problem (2.14) is recovered by $H = D^{-1/2}X$. The SC algorithm is summarized in Algorithm 1.

Algorithm 1 SC (Spectral Clustering)

Input: weighted adjacency matrix $W \in \mathbb{R}^{n\times n}$; degree matrix $D \in \mathbb{R}^{n\times n}$; $k \in \mathbb{N}$

Output: a clustering of indices $1 : n$ into k clusters

1: compute the Laplacian matrix $L = D - W$;
2: compute the normalized Laplacian $L_{\text{n}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$;
3: compute the k smallest eigenvalues of L_{n} and the corresponding eigenvectors $X \in \mathbb{R}^{n\times k}$;
4: apply k-means clustering to the rows of $H = D^{-\frac{1}{2}}X$.

The SC (Shi and Malik, 2000; Ng et al., 2001) is a highly successful clustering algorithm, and widely used in areas of data exploration, such as image segmentation (Tung et al., 2010), speech separation (Bach and Jordan, 2006) among others. The SC algorithm is efficient and scalable because it can fully take the advantage of sparsity of the SC model and use the state-of-the-art scalable sparse eigensolvers (Bai et al., 2000).

Fair SC. The group fairness constraints can be elegantly incorporated into the SC by simply adding the constraint (2.9) to the trace minimization problem (2.14), which leads to

$$\min_{H \in \mathbb{R}^{n\times k}} \operatorname{Tr}(H^T L H) \ \text{ s.t. } H^T D H = I_k \text{ and } F^T H = 0, \tag{2.16}$$

where $L \in \mathbb{R}^{n\times n}$ is the graph Laplacian, $F \in \mathbb{R}^{n\times(h-1)}$ is from (2.9), and $F^T H = 0$ is the original (2.9) right-multiplied with $\widehat{D}^{-1}$ due to the scaling (2.11) of H.

The idea of enforcing group fairness in spectral clustering using the optimization (2.16) was proposed in Kleindessner et al. (2019). We will show that the additional linear constraints in (2.16) will introduce only marginal extra costs than solving the SC (2.14).

3 ALGORITHMS

In this section, we consider numerical algorithms for solving the constrained trace minimization problem (2.16). We first review the FairSC algorithm proposed in Kleindessner et al. (2019) and then address the scalability issue of the FairSC.

3.1 FairSC algorithm

A nullspace-based algorithm for solving the problem (2.16) proposed in Kleindessner et al. (2019) is as follows. Since the columns of H live in the nullspace of F^T, we can write

$$H = ZY$$

for some $Y \in \mathbb{R}^{(n-h+1)\times k}$, where $Z \in \mathbb{R}^{n\times(n-h+1)}$ is an orthonormal basis matrix of null(F^T). Consequently, the optimization problem (2.16) is equivalent to the following trace optimization without linear constraints:

$$\min_{Y\in\mathbb{R}^{(n-h+1)\times k}} \operatorname{Tr}\left(Y^T[Z^T L Z]Y\right) \text{ s.t. } Y^T[Z^T D Z]Y = I_k. \tag{3.1}$$

We can further transform the problem (3.1) to the standard trace minimization (2.14) by another change of variables

$$X = QY \quad \text{with} \quad Q = (Z^T D Z)^{1/2},$$

which leads to

$$\min_{X\in\mathbb{R}^{(n-h+1)\times k}} \operatorname{Tr}\left(X^T M X\right) \quad \text{s.t.} \quad X^T X = I_k, \tag{3.2}$$

where $M = Q^{-1}Z^T L Z Q^{-1}$. Observe that M is positive semi-definite of size $n-h+1$. According to Theorem 2.1, problem (3.2) is solved by linear eigenvalue problem $Mx = \lambda x$. The optimal solution $X = [x_1, \ldots, x_k]$ consists of the eigenvectors corresponding to the k smallest eigenvalue of M. Finally, $H = ZQ^{-1}X$ is the solution of the fair SC minimization (2.16).

We summarize the aforementioned algorithm for the group-fair spectral clustering in Algorithm 2, called FairSC. FairSC requires two major computational kernels. The first one is the nullspace of F^T in step 2. This can be done by the SVD of $F = U\Sigma V^T$, where $U \in \mathbb{R}^{n\times n}$ and $V \in \mathbb{R}^{(h-1)\times(h-1)}$ are orthogonal, and $\Sigma \in \mathbb{R}^{n\times(h-1)}$ is diagonal. According to Lemma 2.1, F has a full column rank $h-1$. Therefore $U(:, h:n)$ is an orthonormal basis of the nullspace of F^T. We can also use QR decomposition $F = UR$, where $U \in \mathbb{R}^{n\times n}$ is orthogonal and $R \in \mathbb{R}^{n\times(h-1)}$ is upper triangular. We can then set $Z = U(:, h:n)$. For both SVD and QR, the computation complexity is about $\mathcal{O}(n(h-1)^2)$; see, e.g., Golub and Van Loan (1996). The second kernel is the matrix square root of size $n-h+1$ in step 3. This can be done by the blocked Schur algorithm (Higham and Al-Mohy, 2010; Deadman et al., 2012). The computation complexity is $\mathcal{O}((n-h+1)^3)$. For matrices of large sizes, both kernels involving large dense matrices are computationally expensive due to memory space and data communication costs. Consequently, FairSC is only suitable for small to medium size fair SC models; see numerical results in Section 4.

Algorithm 2 FairSC

Input: weighted adjacency matrix $W \in \mathbb{R}^{n\times n}$; degree matrix $D \in \mathbb{R}^{n\times n}$; group-membership matrix $F \in \mathbb{R}^{n\times(h-1)}$; $k \in \mathbb{N}$

Output: a clustering of indices $1:n$ into k clusters

1: compute the Laplacian matrix $L = D - W$;
2: compute an orthonormal basis Z of the nullspace of F^T;
3: compute the matrix square root $Q = (Z^T D Z)^{1/2}$;
4: compute $M = Q^{-1}Z^T L Z Q^{-1}$;
5: compute the k smallest eigenvalues of M and the corresponding eigenvectors $X \in \mathbb{R}^{n\times k}$;
6: apply k-means clustering to the rows of $H = ZQ^{-1}X$.

3.2 First variant of FairSC

As the first variant of FairSC, we can avoid computing the square root of a dense matrix by reordering the changes of variables used in FairSC. Let us begin with a change of variables

$$X = D^{\frac{1}{2}}H$$

and turn the optimization (2.16) to

$$\min_{X\in\mathbb{R}^{n\times k}} \operatorname{Tr}\left(X^T L_{\mathrm{n}} X\right) \quad \text{s.t.} \quad X^T X = I_k \text{ and } C^T X = 0, \tag{3.3}$$

where $L_{\mathrm{n}} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$ is the normalized Laplacian, and $C = D^{-\frac{1}{2}}F$. Recall that the degree matrix D is diagonal, so generating L_{n} and C requires only row and column scaling. Next, we remove the linear constraints $C^T X = 0$ in (3.3) using the nullspace basis of C^T. Specifically, since the columns of X live in the nullspace of C^T we can parameterize

$$X = VY \quad \text{for some } Y \in \mathbb{R}^{(n-h+1)\times k},$$

where $V \in \mathbb{R}^{n\times(n-h+1)}$ is an orthonormal basis matrix of the nullspace of C^T. Then the optimization (3.3) is equiv-

alent to the standard trace minimization

$$\min_{Y\in\mathbb{R}^{(n-h+1)\times k}} \operatorname{Tr}\left(Y^T L_{\mathrm{n}}^{\mathrm{v}} Y\right) \quad \text{s.t.} \quad Y^T Y = I_k, \quad (3.4)$$

where $L_{\mathrm{n}}^{\mathrm{v}} = V^T L_n V \in \mathbb{R}^{(n-h+1)\times(n-h+1)}$. Consequently, by Theorem 2.1, we just need to solve the symmetric eigenvalue problem

$$L_{\mathrm{n}}^{\mathrm{v}} y = \lambda y. \quad (3.5)$$

The eigenvectors corresponding to the k smallest eigenvalues provide the solution Y of (3.4), by which we recover the solution $H = D^{-\frac{1}{2}}VY$ of the fair SC minimization problem (2.16).

Although this variant of FairSC avoids computing matrix square root of a dense matrix, the other drawbacks of FairSC remain, namely explicit computation of the nullspace of C^T and eigenvalue computation of the dense matrix $L_{\mathrm{n}}^{\mathrm{v}}$.

3.3 Scalable FairSC algorithm

We now show how to reformulate the eigenvalue problem (3.5) to address the remaining pitfalls of FairSC. We begin with the eigenvalue problem of $L_{\mathrm{n}}^{\mathrm{v}}$ in (3.5):

$$\left(V^T L_{\mathrm{n}} V\right) y = \lambda y.$$

A left multiplication of V leads to

$$(VV^T L_{\mathrm{n}} VV^T)Vy = \lambda \cdot Vy, \quad (3.6)$$

where on the left side $VV^T \cdot Vy \equiv Vy$ due to the fact $V^TV = I$. Denote by $P = VV^T$ a projection matrix onto the range space of V (i.e., nullspace of C^T). Then (3.6) leads to the following *projected eigenvalue problem*

$$L_{\mathrm{n}}^{\mathrm{p}} x = \lambda x, \quad (3.7)$$

where $x = Vy$ and $L_{\mathrm{n}}^{\mathrm{p}} = PL_{\mathrm{n}}P \in \mathbb{R}^{n\times n}$. Consequently, an eigenvalue λ of $L_{\mathrm{n}}^{\mathrm{v}}$ in (3.5) is also an eigenvalue of $L_{\mathrm{n}}^{\mathrm{p}}$ in (3.7). A major advantage of the projected eigenvalue problem (3.7) is that it may avoid the computation of the nullspace of C^T by exploiting the fact that the projection matrix

$$P = I - U_2 U_2^T, \quad (3.8)$$

where $U_2 \in \mathbb{R}^{n\times(h-1)}$ is an orthonormal basis of the range C. ($[V, U_2] \in \mathbb{R}^{n\times n}$ is orthogonal) This is especially beneficial since C is a tall and skinny matrix, where $U_2 \in \mathbb{R}^{n\times(h-1)}$ is much smaller than $V \in \mathbb{R}^{n\times(n-h+1)}$. In addition, to compute the eigenvalues of $L_{\mathrm{n}}^{\mathrm{p}}$ by an iterative eigensolver, we only need the matrix-vector product $L_n^{\mathrm{p}} w$ for a given vector w and the matrix $L_{\mathrm{n}}^{\mathrm{p}}$ is never formed explicitly. The product Pw can be applied without formulating U_2; see implementation detail in Section 3.4.

For FairSC, we need the k smallest eigenvalues of the matrix $L_{\mathrm{n}}^{\mathrm{v}}$ in (3.5). The following proposition connects the eigenstructures of the matrices $L_{\mathrm{n}}^{\mathrm{v}}$ and $L_{\mathrm{n}}^{\mathrm{p}}$.

Proposition 3.1. *Suppose $L_{\mathrm{n}}^{\mathrm{v}}$ in* (3.5) *has the eigendecomposition*

$$L_{\mathrm{n}}^{\mathrm{v}} = Y\Lambda_{\mathrm{v}}Y^T, \quad (3.9)$$

where $\Lambda_{\mathrm{v}} = diag(\lambda_1, \lambda_2, \ldots, \lambda_{n-h+1})$ contains the eigenvalues, and Y is an orthogonal matrix of order $n-h+1$ containing eigenvectors. Then the matrix $L_{\mathrm{n}}^{\mathrm{p}}$ from (3.7) *has the eigendecomposition*

$$L_{\mathrm{n}}^{\mathrm{p}} = \begin{bmatrix} U_1 & U_2 \end{bmatrix} \begin{bmatrix} \Lambda_{\mathrm{v}} & \\ & \mathbf{0}_{h-1,h-1} \end{bmatrix} \begin{bmatrix} U_1^T \\ U_2^T \end{bmatrix}, \quad (3.10)$$

where $U = [U_1, U_2] \in \mathbb{R}^{n\times n}$ is orthogonal with $U_1 = VY \in \mathbb{R}^{n\times(n-h+1)}$ and $U_2 \in \mathbb{R}^{n\times(h-1)}$ being an arbitrary orthonormal basis of the range of C.

Proof. See Appendix A.2. □

The following is a direct consequence of Proposition 3.1.

Corollary 3.1. *Let $L_{\mathrm{n}}^{\mathrm{v}}$ and $L_{\mathrm{n}}^{\mathrm{p}}$ be defined as in* (3.4) *and* (3.7)*. Then*

(i) If (λ, y) is an eigenpair of $L_{\mathrm{n}}^{\mathrm{v}}$, then (λ, x) with $x = Vy$ is an eigenpair of $L_{\mathrm{n}}^{\mathrm{p}}$.

(ii) If (λ, x) is an eigenpair of $L_{\mathrm{n}}^{\mathrm{p}}$ and $C^Tx = 0$, then (λ, y) with $y = V^Tx$ is an eigenpair of $L_{\mathrm{n}}^{\mathrm{v}}$.

Let us return to the eigenvalue problem (3.5). Since the matrix $L_{\mathrm{n}}^{\mathrm{v}}$ is positive semi-definite, it has $n-h+1$ ordered eigenvalues $0 \le \lambda_1 \le \cdots \le \lambda_{n-h+1}$. By (3.10), the projected matrix $L_{\mathrm{n}}^{\mathrm{p}}$ has n ordered eigenvalues: $\underbrace{0 = \cdots = 0}_{h-1} \le \lambda_1 \le \cdots \le \lambda_{n-h+1}$, where the first $h-1$ zero eigenvalues (counting multiplicity) have eigenvectors in the range of C.

In the simple case of $\lambda_1 > 0$, the k smallest eigenvalues $\lambda_1, \ldots, \lambda_k$ of L_{n}^{v} corresponds to the k smallest *positive eigenvalues* of $L_{\mathrm{n}}^{\mathrm{v}}$. To find those eigenvalues, we can first compute $K = k + h - 1$ smallest eigenvalues of $L_{\mathrm{n}}^{\mathrm{p}}$ by an eigensolver, and then select the desired k eigenpairs corresponding to non-zero eigenvalues (alternatively, select those eigenvalues with eigenvectors orthogonal to C). However, if $\lambda_1 = 0$ (or $\lambda_1 \approx 0$), then this simple selection scheme does not work, as the eigenvector corresponding to $\lambda_1 = 0$ is mixed (or numerically mixed) with the eigenspace of the $h-1$ zero eigenvalues. This eigenspace mixing issue happens, in particular, if the solution is computed by an iterative method with low accuracy.

To address the eigenspace mixing issue, we turn to the second major contribution of this work, namely a novel use of Hotelling's deflation. In the following, we first discuss Hotelling's deflation (Hotelling, 1943), which is also known as explicit external deflation and is suitable for high-performance computing; see, e.g., Parlett (1998); Yamazaki et al. (2019). The main idea of Hotelling's deflation is summarized in the following proposition.

Proposition 3.2. *Let the eigenvalue decomposition of a symmetric matrix $A \in \mathbb{R}^{n\times n}$ be given by*

$$A = Q\Lambda Q^T = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 & \\ & \Lambda_2 \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}, \quad (3.11)$$

where $\Lambda_1 \in \mathbb{R}^{k\times k}$ and $\Lambda_2 \in \mathbb{R}^{(n-k)\times(n-k)}$ contain eigenvalues, and $Q_1 \in \mathbb{R}^{n\times k}$ and $Q_2 \in \mathbb{R}^{n\times(n-k)}$ are orthonormal eigenvectors. For a given shift $\sigma \in \mathbb{R}$, define the shifted matrix

$$A_\sigma = A + \sigma Q_1 Q_1^T.$$

Then the eigenvalue decomposition of A_σ has the following form

$$A_\sigma = \begin{bmatrix} Q_1 & Q_2 \end{bmatrix} \begin{bmatrix} \Lambda_1 + \sigma I & \\ & \Lambda_2 \end{bmatrix} \begin{bmatrix} Q_1^T \\ Q_2^T \end{bmatrix}. \quad (3.12)$$

Proof. See Appendix A.3. □

Suppose we are interested in the eigenvalues Λ_2 and the corresponding eigenvectors Q_2. By Proposition 3.2, if we choose the shift σ sufficiently large, eigenvalues in Λ_2 will always correspond to the $n-k$ smallest eigenvalues of A_σ, since the unwanted eigenvalues Λ_1 are shifted away to $\Lambda_1 + \sigma I$. The corresponding eigenvectors remain unchanged. This is exactly what we need to untangle the unwanted $h-1$ zero eigenvalues of $L_{\mathrm{n}}^{\mathrm{p}}$ in (3.10) from the rest of the eigenvalues of λ_i.

Recall the eigenvalue decomposition of $L_{\mathrm{n}}^{\mathrm{p}}$ in (3.10). To shift away the unwanted $h-1$ zero eigenvalues, we can apply Hotelling's deflation with a shift σ to obtain

$$L_{\mathrm{n}}^{\sigma} := L_{\mathrm{n}}^{\mathrm{p}} + \sigma U_2 U_2^T, \quad (3.13)$$

where recall that U_2 is an orthonormal basis for the range of C. If the shift σ is chosen such that $\sigma > \lambda_k$, where λ_k is the k-th smallest eigenvalue in Λ_{v}, then our desired k smallest eigenvalues in Λ_{v} are corresponding to the k smallest eigenvalues of L_{n}^{σ}. Consequently, the eigenspace mixing issue is solved. On the other hand, by (3.8), the shifted matrix in (3.13) can be expressed as follows

$$L_{\mathrm{n}}^{\sigma} = PL_{\mathrm{n}}P + \sigma(I-P) = P(L_{\mathrm{n}} - \sigma I)P + \sigma I. \quad (3.14)$$

Then the matrix-vector multiplication with L_{n}^{σ} only requires operations with P and L_{n}. This is extremely beneficial for large-scale fair SC models.

3.4 Algorithm and implementation

As we described in the previous section, Hotelling's deflation with a proper choice of σ resolves the eigenspace mixing issue for the projected eigenvalue problem (3.7). To summarize, the solution of the constrained trace minimization (2.16) can now be characterized by the following proposition.

Proposition 3.3. *Let $L_{\mathrm{n}}^{\sigma} \in \mathbb{R}^{n\times n}$ be defined as in* (3.14) *and assume σ is sufficiently large such that $\sigma > \lambda_k(L_{\mathrm{n}}^{\mathrm{v}})$, where $\lambda_k(L_{\mathrm{n}}^{\mathrm{v}})$ is the k-th smallest eigenvalue of $L_{\mathrm{n}}^{\mathrm{v}}$ in* (3.5)*. Then H is a solution to the trace minimization* (2.16) *if and only if $H = D^{-\frac{1}{2}}X$, where $X = [x_1, x_2, \ldots, x_k] \in \mathbb{R}^{n\times k}$ contains the k eigenvectors corresponding to the k smallest eigenvalues of L_{n}^{σ}.*

The final algorithm based on projected eigenproblem (3.7) and Hotelling's deflation is presented in Algorithm 3, called scalable FairSC, s-FairSC in short.

Algorithm 3 Scalable FairSC (s-FairSC)

Input: weighted adjacency matrix $W \in \mathbb{R}^{n\times n}$; degree matrix $D \in \mathbb{R}^{n\times n}$; group-membership matrix $F \in \mathbb{R}^{n\times(h-1)}$; shift $\sigma \in \mathbb{R}$; $k \in \mathbb{N}$

Output: a clustering of indices $1:n$ into k clusters

1: compute the Laplacian matrix $L = D - W$;
2: set $L_{\mathrm{n}} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$, and $C = D^{-\frac{1}{2}}F$;
3: compute the k smallest eigenvalues of L_{n}^{σ} in (3.14) and the corresponding eigenvectors as columns of $X \in \mathbb{R}^{n\times k}$;
4: apply k-means clustering to the rows of $H = D^{-\frac{1}{2}}X$.

Implementation issues. A few implementation issues regarding s-FairSC (Algorithm 3) are in order. (i) For computing eigenvalue of L_{n}^{σ}, we can use an iterative eigensolver, such as `eigs` in MATLAB, which is based on ARPACK (Lehoucq et al., 1998), an implicitly restarted Arnoldi method. The eigensolver only needs to access L_{n}^{σ} through the matrix-vector multiplication $L_{\mathrm{n}}^{\sigma}w$. By (3.14),

$$L_{\mathrm{n}}^{\sigma}w = P(L_{\mathrm{n}}(Pw)) - \sigma Pw + \sigma w.$$

(ii) The projection P in (3.8) can be written as $P = I - C(C^TC)^{-1}C^T$, see Golub et al. (2000). Consequently,

$$Pw = (I - C(C^TC)^{-1}C^T)w = w - Cz, \quad (3.15)$$

where z is the solution to the least-squares problem

$$\min_z \|Cz - w\|_2.$$

For small to moderate size problems, direct LS solver can be applied to computing z. For large scale problems, iterative methods such as LSQR (Paige and Saunders, 1982) can be applied; this is in line with the inner-outer iteration methods for eigenvalue computation, see e.g., Golub et al. (2000). (iii) For an appropriate choice of shift σ, one can use an estimation for the largest eigenvalue of L_{n}. Such a shift also guarantees the numerical stability of Hotelling's deflation (Lin et al., 2021).

Time Complexity. The complexity of s-FairSC is dominated by computing k eigenpairs of the matrix L_{n}^{σ}. To

use a modern Krylov subspace eigensolver, say the function `eigs` in MATLAB, the two leading costs are (1) the matrix-vector product $L_{\mathrm{n}}^{\sigma}w$, and (2) the orthonormalization of basis vectors of Krylov subspace eigensolver. For (1), the complexity is $O(nm + nh^2)$ and for (2), it is $O(nk^2)$, where $n = |V|$, $m = |W|$ of graph $\mathcal{G}(V, W)$, h is the number of groups and assume that the product Pw for the projection matrix P is computed by a direct least squares solver since h is typically small. Therefore, the complexity of s-FairSC is $O(n(m + h^2 + k^2))$, where the constant of $\mathcal{O}(\cdot)$ depends on the number of restarts of subspace iterations, usually about 10 to 20. Using the same analysis, the complexity of SC (without fairness constraints) is $O(n(m + k^2))$. Since h is typically small, say $h = 10$, it explains observations that s-FairSC is as fast as SC; see numerical results in Section 4.2.

3.5 Related work

The idea of transforming the optimization problem (2.16) to an equivalent eigenvalue problem is very natural. For the case of $k = 1$, the projected eigenvalue problem (3.7) was considered in Golub (1973) and (Golub and Van Loan, 1996, p. 621).

The projected eigenvalue problem (3.7) is a form of so-called *constrained eigenvalue problems*, which is more generally formulated as $Ax = \lambda Mx$ subject to $C^T x = 0$, where A and M are symmetric and M is positive definite. The constrained eigenvalue problems are found in many applications. There are a number of approaches available; see Arbenz and Drmac (2002) for an algorithm for positive semidefinite A with a known nullspace; Baker and Lehoucq (2009) for a preconditioning technique; Golub et al. (2000) for a Lanczos process with inner-outer iterations to handle large matrices; and Porcelli et al. (2015) for a solution procedure within the structural finite-element code NOSA-ITACA. For constrained eigenvalue problems, the matrix C is typically corresponding to the nullspace of A, and the constraint $C^T x = 0$ is to avoid computing "null eigenvectors". Since the entire nullspace of A is avoided, there is no eigenvector selection issue as in our problem (3.7). Simoncini (2003) proposed a reformulation of the constrained eigenproblem based on null eigenvalue shifting. Her approach essentially includes Hotelling's deflation as a special case, but with a goal to shift away the entire nullspace of A. By discussion in Section 3, we show that Hotelling's deflation is also capable of splitting the unwanted null vectors from those desired ones.

4 EXPERIMENTS

In this section, we present experimental results on the proposed s-FairSC (Algorithm 3). Similar to SC and FairSC, s-FairSC is implemented in MATLAB®.[2] The results are obtained from a MacBook Pro with an 8-core i9 processor @2.3 GHz, 16 GB memory, and 16 MB L3 cache.

4.1 Datasets

Modified Stochastic block model (m-SBM). The stochastic block model (SBM) (Holland et al., 1983) is a random graph model with planted blocks (ground-truth clustering). It is widely used to generate synthetic networks for clustering and community detection (Rohe et al., 2011; Balakrishnan et al., 2011; Lei and Rinaldo, 2015; Sarkar and Bickel, 2015). To take group fairness into account, we use a modified SBM (m-SBM) proposed by Kleindessner et al. (2019) to generate the test graph $\mathcal{G}(V, W)$. In m-SBM, n vertices are assigned to k prescribed (ground-truth) clusters $V = C_1 \cup \cdots \cup C_k$, and between any pair of vertices, an edge is placed with a probability that depends only on the clusters of the two vertices (see Appendix B.1 for details). Let $V = \widehat{C}_1 \cup \cdots \cup \widehat{C}_k$ be a computed clustering. The discrepancy between the computed and ground-truth clustering is measured by the *error rate* of clustering (proportion of misclustered vertices):

$$\mathrm{Err}(\widehat{H} - H) := \frac{1}{n} \min_{J \in \Pi_k} \|\widehat{H}J - H\|_F^2, \qquad (4.1)$$

where H and $\widehat{H}$ are the ground-truth and computed cluster indicator matrices, respectively, and Π_k is the set of all possible $k \times k$ permutation matrices.

FacebookNet. FacebookNet[3] is a dataset that collects Facebook friendship relations between students in a high school in France in 2013. This social network dataset was studied for information propagation and opinion formation Mastrandrea et al. (2015) and for clustering (Crawford and Milenković, 2018; Kleindessner et al., 2019; Chodrow et al., 2021). In graph $\mathcal{G}(V, W)$, V is the set of students ($n = |V| = 155$), and an edge represents a friendship between two students. Students are divided by gender into two groups $V = V_1 \cup V_2$, with $|V_1| = 70$ of girls and $|V_2| = 85$ of boys.

LastFMNet. LastFMNet[4] (Rozemberczki and Sarkar, 2020) is a real-world dataset that contains mutual follower relations among users of Last.fm, a recommender-system-based online radio and music community in Asia. LastFMNet was collected from public API in 2020 and used to

[2] SC and FairSC code: `https://github.com/matthklein/fair_spectral_clustering`. s-FairSC code: `https://github.com/jiiwang/scalable_fair_spectral_clustering`

[3] `http://www.sociopatterns.org/datasets/high-school-contact-and-friendship-networks/`

[4] `http://snap.stanford.edu/data/feather-lastfm-social.html`

study the distribution of vertex features on graphs. In graph $\mathcal{G}(V, W)$, V is the set of users with $n = |V| = 5576$, and an edge represents a mutual follower friendship between two users. LastFMNet also records nationalities of the users $V = V_1 \cup \cdots \cup V_6$ with $|V_1| = 1073$, $|V_2| = 505$, $|V_3| = 645$, $|V_4| = 1266$, $|V_5| = 558$ and $|V_6| = 1529$. $\mathcal{G}(V, W)$ has 19587 edges, and the density is 0.00013.

Random Laplacian. To create a random Laplacian of graph $\mathcal{G}(V, W)$, we first generate a random symmetric weight matrix $W \in \mathbb{R}^{n\times n}$ with prescribed sparsity s and $n = |V|$, and then set the degree matrix $D = \mathrm{diag}(W\mathbf{1}_n)$ and the Laplacian $L = D - W$. The matrix $F \in \mathbb{R}^{n\times(h-1)}$ in the constraints of the fair SC model (2.16) is also constructed as a random matrix. Here, F is not for the group-membership information but only acts as a placeholder. We will use this dataset to show the scalability of algorithms.

4.2 Experimental results

Experiment 1. This experiment is conducted on the m-SBM to compare the error rate (4.1) and running time of SC, FairSC, and s-FairSC. Figure 1 depicts the computation results. SC and s-FairSC are tested for model sizes from $n = 1000$ to 10000. FairSC stops at $n = 4000$ due to its high computational cost, echoing results reported in Kleindessner et al. (2019).

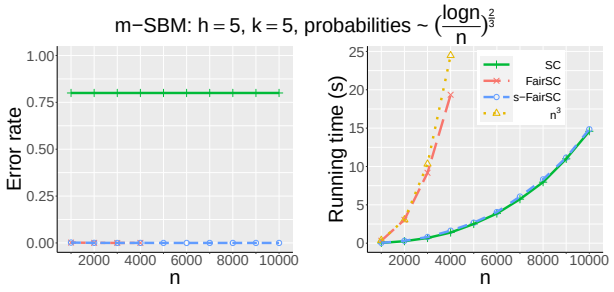

Figure 1: Error rate and running time (in seconds) of SC, FairSC and s-FairSC of an m-SBM with $h = 5$, $k = 5$, and edge connectivity probabilities proportional to $(\frac{\log n}{n})^{\frac{2}{3}}$.

From Figure 1, we observe that both FairSC and s-FairSC successfully retrieve the fair ground-truth clustering, but SC fails. We can see that s-FairSC is as good as FairSC in terms of error rate for the computed clustering. But its running time is only a fraction of that of FairSC; e.g., for $n = 4000$, s-FairSC is 12× faster than FairSC. We also observe that s-FairSC is as scalable as SC, but the latter does not account for the fairness constraints.

Experiment 2. We use the FacebookNet dataset to quantify the group fairness in the computed clustering. Table 1 records the quantities from Definition 2.1.

	SC	FairSC	s-FairSC
$\frac{\lvert V_1\rvert}{\lvert V\rvert}$	0.4516		
$\frac{\lvert V_1\cap\widehat{C}_1\rvert}{\lvert\widehat{C}_1\rvert}$	0.6528	0.3537	0.3537
$\frac{\lvert V_1\cap\widehat{C}_2\rvert}{\lvert\widehat{C}_2\rvert}$	0.2771	0.5616	0.5616
$\frac{\lvert V_2\rvert}{\lvert V\rvert}$	0.5484		
$\frac{\lvert V_2\cap\widehat{C}_1\rvert}{\lvert\widehat{C}_1\rvert}$	0.3472	0.6463	0.6463
$\frac{\lvert V_2\cap\widehat{C}_2\rvert}{\lvert\widehat{C}_2\rvert}$	0.7229	0.4384	0.4384

Table 1: Fractions of group membership within each cluster for the recovered clustering $V = \widehat{C}_1 \cup \widehat{C}_2$.

The average balance introduced in Chierichetti et al. (2017) has been used to measure fairness in clustering. Given a clustering $V = C_1 \cup \cdots \cup C_k$ and group partition $V = V_1 \cup \cdots \cup V_h$, the balance of cluster C_ℓ for $\ell = 1, 2, \cdots, k$ is defined as

$$\mathrm{balance}(C_\ell) := \min_{s\neq s'\in\{1,\cdots,h\}} \frac{|V_s \cap C_\ell|}{|V_{s'} \cap C_\ell|} \in [0, 1]. \quad (4.2)$$

The average balance is then given by

$$\text{Average\_Balance} := \frac{1}{k}\sum_{l=1}^{k} \mathrm{balance}(C_\ell). \quad (4.3)$$

A higher balance implies a fairer clustering; see Appendix B.2 for an explanation.

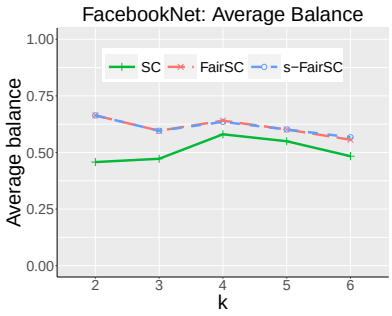

Figure 2: Average_Balance of SC, FairSC, and s-FairSC on FacebookNet as a function of the number k of clusters.

By Table 1 and Figure 2, we observe that since problem (2.13) is relaxed to problem (2.14), the equality in (2.5) does not hold. Nevertheless, both FairSC and s-FairSC have improved fairness compared to SC. In addition, FairSC and s-FairSC produce almost identical results.

Experiment 3. In this experiment, we use LastFMNet to compare the running time of SC, FairSC, and s-FairSC. We also measure average balance (4.3) to evaluate the fairness of clustering by the algorithms. The running time as a function of the number k of clusters is illustrated in Figure 3.

We observe that when $k \geq 5$, s-FairSC is $7\times$ faster than FairSC, and it is as fast as SC. Figure 4 shows the values of Average_Balance as a function of the number k of clusters. Both FairSC and s-FairSC have higher values of Average_Balance than SC, indicating they have improved fairness compared to SC.

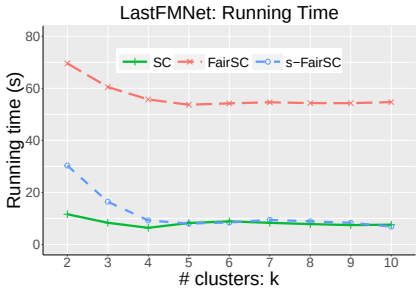

Figure 3: Running time (in seconds) of SC, FairSC, and s-FairSC on LastFMNet as a function of the number k of clusters.

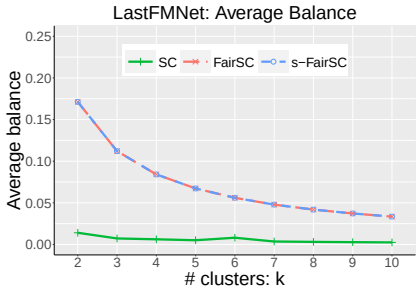

Figure 4: Average_Balance of SC, FairSC, and s-FairSC on LastFMNet as a function of the number k of clusters.

Experiment 4. In this experiment, we use random Laplacian to demonstrate that s-FairSC has a similar scalability as SC. Figure 5 reports the running time of SC (Algorithm 1) for solving the SC model (2.14) and the s-FairSC (Algorithm 3) for solving the fair SC model (2.16). Here, the model sizes range from 5000 to 10000 with the number of groups $h = 5$ and different numbers of clusters $k = 5, 8, 10$. We observe that s-FairSC is only slightly more expensive than SC and is as scalable as SC.

5 CONCLUDING REMARKS

FairSC (Algorithm 2) is able to recover fairer clustering, but sacrifices the performance and scalability. In this paper, we presented a scalable FairSC (s-FairSC, Algorithm 2) by incorporating nullspace projection and Hotelling's deflation. All computational kernels of s-FairSC only involve the sparse matrix-vector products, so the algorithm can fully exploit the sparsity of the fair SC model (2.16)

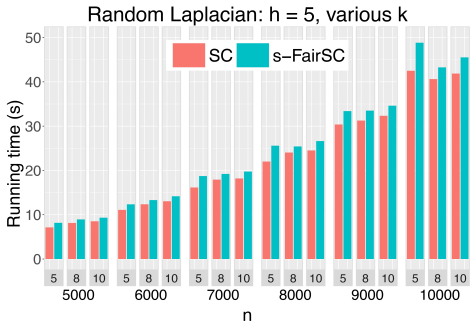

Figure 5: Running time (in seconds) of SC and s-FairSC on random Laplacian with $h = 5$ and $k \in \{5, 8, 10\}$.

and is scalable in the sense that it only has a marginal increase in computational costs compared to SC without fairness constraints.

We note that the non-overlap of groups leads to the full rank of the group indicator matrix H, which simplifies the rest of presentation substantially. An interesting extension to the current work is to incorporate group overlapping. The overlap of groups may lead to the rank deficiency of F. In this case, a simple approach is to perform rank-revealing factorization of F first, and the rest of the discussion will hold. However, it is a subject of further study on how to avoid the rank-revealing factorization, and maintain the sparsity of F in computation. Another intriguing problem is the development of scalable algorithms to solve the group fairness condition (2.5) in a less stringent manner. For instance, the following notion of group fairness clustering with lower and upper bounds is introduced in Bera et al. (2019):

$$\beta_s \leq \frac{|V_s \cap C_\ell|}{|C_\ell|} \leq \alpha_s \quad \text{for } s = 1, 2, \cdots, h, \tag{5.1}$$

where β_s and α_s are lower and upper bounds for group V_s, respectively, and $0 < \beta_s \leq \alpha_s < 1$. A further topic is to extend the s-FairSC to individual fairness, where any two individuals who are similar with respect to a specific sensitive attribute should be treated similarly (Dwork et al., 2012; Zemel et al., 2013). An SC model with individual fairness constraints is devised in Gupta and Dukkipati (2022). However, the existing algorithm (Gupta and Dukkipati, 2022) is not scalable due to the high costs of its computational kernels.

Acknowledgements

Wang and Bai was supported in part by NSF grant 1913364. Lu was supported by NSF grant 2110731. Davidson was supported in part by NSF grant 1910306, and a gift from Google. We would like to thank anonymous reviewers for their constructive comments that have significantly improved the presentation.

References

A. Agarwal, M. Dudík, and Z. S. Wu. Fair regression: Quantitative definitions and reduction-based algorithms. In *International Conference on Machine Learning*, pages 120–129. PMLR, 2019.

S. Aghaei, M. J. Azizi, and P. Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 1418–1426, 2019.

A. Amini, A. P. Soleimany, W. Schwarting, S. N. Bhatia, and D. Rus. Uncovering and mitigating algorithmic bias through learned latent structure. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 289–295, 2019.

P. Arbenz and Z. Drmac. On positive semidefinite matrices with known null space. *SIAM Journal on Matrix Analysis and Applications*, 24(1):132–149, 2002. doi: 10.1137/S0895479800381331. URL https://doi.org/10.1137/S0895479800381331.

F. Bach and M. Jordan. Learning spectral clustering. *Advances in neural information processing systems*, 16, 2003.

F. R. Bach and M. I. Jordan. Learning spectral clustering, with application to speech separation. *The Journal of Machine Learning Research*, 7:1963–2001, 2006.

Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst (editors). *Templates for the solution of Algebraic Eigenvalue Problems: A Practical Guide*. SIAM, Philadelphia, 2000.

C. G. Baker and R. B. Lehoucq. Preconditioning constrained eigenvalue problems. *Linear algebra and its applications*, 431(3-4):396–408, 2009.

S. Balakrishnan, M. Xu, A. Krishnamurthy, and A. Singh. Noise thresholds for spectral clustering. *Advances in Neural Information Processing Systems*, 24, 2011.

S. Bera, D. Chakrabarty, N. Flores, and M. Negahbani. Fair algorithms for clustering. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/fc192b0c0d270dbf41870a63a8c76c2f-Paper.pdf.

F. Chierichetti, R. Kumar, S. Lattanzi, and S. Vassilvitskii. Fair clustering through fairlets. *Advances in Neural Information Processing Systems*, 30, 2017.

P. S. Chodrow, N. Veldt, and A. R. Benson. Generative hypergraph clustering: From blockmodels to modularity. *Science Advances*, 7(28):eabh1303, 2021.

A. Chouldechova and A. Roth. The frontiers of fairness in machine learning. *arXiv preprint arXiv:1810.08810*, 2018.

J. Crawford and T. Milenković. Cluenet: Clustering a temporal network based on topological similarity rather than denseness. *PloS one*, 13(5):e0195993, 2018.

I. Davidson and S. S. Ravi. Making existing clusterings fairer: Algorithms, complexity results and insights. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3733–3740, 2020.

E. Deadman, N. J. Higham, and R. Ralha. Blocked schur algorithms for computing the matrix square root. In *International Workshop on Applied Parallel Computing*, pages 171–182. Springer, 2012.

C. Dwork, M. Hardt, T. Pitassi, O. Reingold, and R. Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226, 2012.

K. Fan. On a theorem of weyl concerning eigenvalues of linear transformations i. *Proceedings of the National Academy of Sciences of the United States of America*, 35(11):652, 1949.

M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian. Certifying and removing disparate impact. In *proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 259–268, 2015.

A. W. Flores, K. Bechtel, and C. T. Lowenkamp. False positives, false negatives, and false analyses: A rejoinder to machine bias: There's software used across the country to predict future criminals. and it's biased against blacks. *Fed. Probation*, 80:38, 2016.

G. H. Golub. Some modified matrix eigenvalue problems. *SIAM Review*, 15(2):318–334, 1973.

G. H. Golub and C. F. Van Loan. *Matrix computations*. Johns Hopkins University Press, 1996.

G. H. Golub, Z. Zhang, and H. Zha. Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner–outer iterations. *Linear Algebra And Its Applications*, 309(1-3):289–306, 2000.

S. Gupta and A. Dukkipati. Consistency of constrained spectral clustering under graph induced fair planted partitions. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=FHgpw2Cn__.

M. Hardt, E. Price, and N. Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

N. J. Higham and A. H. Al-Mohy. Computing matrix functions. *Acta Numerica*, 19:159–208, 2010.

P. W. Holland, K. B. Laskey, and S. Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983. ISSN 0378-8733. doi: https://doi.org/10.1016/0378-8733(83)90021-7. URL https://www.sciencedirect.com/science/article/pii/0378873383900217.

R. A. Horn and C. R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

H. Hotelling. Some new methods in matrix calculation. *The Annals of Mathematical Statistics*, 14(1):1–34, 1943.

M. Kleindessner, S. Samadi, P. Awasthi, and J. Morgenstern. Guarantees for spectral clustering with fairness constraints. In *International Conference on Machine Learning*, pages 3458–3467. PMLR, 2019.

K. Lang. Fixing two weaknesses of the spectral method. *Advances in Neural Information Processing Systems*, 18, 2005.

R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998.

J. Lei and A. Rinaldo. Consistency of spectral clustering in stochastic block models. *The Annals of Statistics*, 43(1):215–237, 2015.

C.-P. Lin, D. Lu, and Z. Bai. Backward stability of explicit external deflation for the symmetric eigenvalue problem. *arXiv preprint arXiv:2105.01298*, 2021.

R. Mastrandrea, J. Fournet, and A. Barrat. Contact patterns in a high school: A comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PLOS ONE*, 10:1–26, 09 2015. doi: 10.1371/journal.pone.0136497. URL https://doi.org/10.1371/journal.pone.0136497.

N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.

A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001.

C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations and sparse least squares. *ACM Transactions on Mathematical Software (TOMS)*, 8(1):43–71, 1982.

B. N. Parlett. *The symmetric eigenvalue problem*. SIAM, 1998.

F. Pethig and J. Kroenung. Biased humans,(un) biased algorithms? *Journal of Business Ethics*, pages 1–16, 2022.

M. Porcelli, V. Binante, M. Girardi, C. Padovani, and G. Pasquinelli. A solution procedure for constrained eigenvalue problems and its application within the structural finite-element code NOSA-ITACA. *Calcolo*, 52(2):167–186, 2015.

K. Rohe, S. Chatterjee, and B. Yu. Spectral clustering and the high-dimensional stochastic blockmodel. *The Annals of Statistics*, 39(4):1878–1915, 2011.

B. Rozemberczki and R. Sarkar. Characteristic Functions on Graphs: Birds of a Feather, from Statistical Descriptors to Parametric Models. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, page 1325–1334. ACM, 2020.

S. Samadi, U. Tantipongpipat, J. H. Morgenstern, M. Singh, and S. Vempala. The price of fair pca: One extra dimension. *Advances in neural information processing systems*, 31, 2018.

P. Sarkar and P. J. Bickel. Role of normalization in spectral clustering for stochastic blockmodels. *The Annals of Statistics*, 43(3):962–990, 2015.

J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

V. Simoncini. Algebraic formulations for the solution of the nullspace-free eigenvalue problem using the inexact shift-and-invert lanczos method. *Numerical linear algebra with applications*, 10(4):357–375, 2003.

F. Tung, A. Wong, and D. A. Clausi. Enabling scalable spectral clustering for image segmentation. *Pattern Recognition*, 43(12):4069–4076, 2010.

D. Wagner and F. Wagner. Between min cut and graph bisection. In *International Symposium on Mathematical Foundations of Computer Science*, pages 744–750. Springer, 1993.

I. Yamazaki, Z. Bai, D. Lu, and J. Dongarra. Matrix powers kernels for thick-restart lanczos with explicit external deflation. In *2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 472–481. IEEE, 2019.

R. Zemel, Y. Wu, K. Swersky, T. Pitassi, and C. Dwork. Learning fair representations. In *International conference on machine learning*, pages 325–333. PMLR, 2013.

H. Zhang, S. Basu, and I. Davidson. A framework for deep constrained clustering-algorithms and advances. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 57–72. Springer, 2019.

A MISSING PROOFS

In this section, we provide proofs that are missing in the main manuscript.

A.1 Proof of Lemma 2.1

For (i): recall that each row of G contains exactly one nonzero entry, and it equals 1. Hence,

$$G\mathbf{1}_h = \mathbf{1}_n \quad \text{and} \quad \mathbf{1}_n^T G \mathbf{1}_h = n. \tag{A.1}$$

Since G has orthogonal columns, the first equation above implies

$$\mathbf{1}_h = G^\dagger \mathbf{1}_n, \tag{A.2}$$

where $G^\dagger := (G^T G)^{-1} G^T$ denotes the pseudo inverse of G. For $\mathrm{rank}(F_0) = h - 1$, it is sufficient to show that the nullspace of F_0 is of dimension one. Let $x \in \mathbb{R}^p$ be a null vector of F_0, i.e., $F_0 x = 0$. By the definition of F_0, we have

$$Gx = \alpha \cdot \mathbf{1}_n \quad \text{with} \quad \alpha := (\mathbf{1}_n^T G x)/n.$$

A multiplication of $G^\dagger$ to the equation, together with (A.2), leads to $x = \alpha \mathbf{1}_h$. On the other hand, $\mathbf{1}_h$ is a null vector of F_0:

$$F_0 \cdot \mathbf{1}_h = G\mathbf{1}_h - \mathbf{1}_n(\mathbf{1}_n^T G \mathbf{1}_h)/n = 0,$$

where we used (A.1). Consequently, the nullspace of F_0 is spanned by the vector $\mathbf{1}_h$. By the rank-nullity theorem in linear algebra, $\mathrm{rank}(F_0) = h - 1$. It follows from $F_0 \cdot \mathbf{1}_h = 0$ that the last column of F_0 is a linear combination of the first $h - 1$ columns. Consequently, we have $\mathrm{rank}(F_0) = \mathrm{rank}(F)$.

For (ii): it follows from (i) that F_0 and F have the same range space. Hence, $F_0^T y = 0$ if and only if $F^T y = 0$. Then (ii) follows from (2.8).

A.2 Proof of Proposition 3.1

We just need to verify that (3.10) is an eigenvalue decomposition of the matrix $L_{\mathrm{n}}^{\mathrm{p}}$. First, since V is a basis of the nullspace of C^T and U_2 is a basis of the range of C, we have

$$V^T U_2 = 0 \quad \text{and} \quad U_1^T U_2 = Y^T (V^T U_2) = 0.$$

A quick verification shows $U = [U_1, U_2]$ satisfy $U^T U = I_n$. Therefore U is orthogonal.

On the other hand, it follows from $L_{\mathrm{n}}^{\mathrm{v}} = V^T L_{\mathrm{n}} V$ and $L_{\mathrm{n}}^{\mathrm{p}} = P L_{\mathrm{n}} P$ that

$$L_{\mathrm{n}}^{\mathrm{p}} = V L_{\mathrm{n}}^{\mathrm{v}} V^T.$$

Hence,

$$L_{\mathrm{n}}^{\mathrm{p}} U_1 = (V L_{\mathrm{n}}^{\mathrm{v}} V^T)(VY) = V(L_{\mathrm{n}}^{\mathrm{v}} Y) = V(Y \Lambda_{\mathrm{v}}) = U_1 \Lambda_{\mathrm{v}},$$

where we used (3.9) in the third equality. On the other hand, $V^T U_2 = 0$ implies

$$L_{\mathrm{n}}^{\mathrm{p}} U_2 = V L_{\mathrm{n}}^{\mathrm{v}} V^T U_2 = 0.$$

Consequently, $L_{\mathrm{n}}^{\mathrm{p}}[U_1, U_2] = [U_1, U_2] \cdot \mathrm{blockdiag}(\Lambda_{\mathrm{v}}, \mathbf{0}_{h-1,h-1})$, i.e., the eigenvalue decomposition (3.10) of $L_{\mathrm{n}}^{\mathrm{p}}$.

A.3 Proof of Proposition 3.2

It follows from (3.11) that $AQ_1 = Q_1 \Lambda_1$ and $AQ_2 = Q_2 \Lambda_2$. Consequently, $A_\sigma Q_1 = AQ_1 + \sigma Q_1 = Q_1(\Lambda_1 + \sigma I)$ and $A_\sigma Q_2 = AQ_2 = Q_2 \Lambda_2$.

B ADDITIONAL EXPERIMENTS AND DISCUSSIONS

B.1 m-SBM Dataset

In this section, we first describe the standard stochastic block model (SBM). Then, we discuss a modification to accommodate group fairness.

B.1.1 SBM

In an SBM with n vertices and k blocks, each vertex is assigned to one block (*a cluster*) to prescribe a clustering, and edges are placed between vertex pairs with probabilities dependent only on the block membership of the vertices.

Following (Lei and Rinaldo, 2015), to generate a random graph $\mathcal{G}(V, W)$ with a ground-truth clustering $V = C_1 \cup \cdots \cup C_k$ by an SBM, we need a pair of parameters (u, P). The vector $u = [u_1, u_2, \ldots, u_k] \in \mathbb{N}^k$ stores the sizes of each block, i.e., each element u_i denotes the number of vertices in C_i, so that $\sum_{i=1}^{k} u_i = n$, where $n = |V|$. Once u is assigned, we can represent the ground-truth clustering by an indicator matrix $H \in \{0, 1\}^{n \times k}$ as defined in (2.2). P is a symmetric probability matrix $P = (p_{ij}) \in \mathbb{R}^{n \times n}$ defining the edge connectivity with

$$p_{ij} = \begin{cases} a, & \text{if } v_i \text{ and } v_j \text{ are in the same cluster,} \\ b, & \text{if } v_i \text{ and } v_j \text{ are in different clusters.} \end{cases} \tag{B.1}$$

We require $a > b$ so that two vertices within a same cluster have a higher chance to be joined by an edge than between clusters.

Next, let α and β be the weights for within-cluster and between-cluster edges, respectively. Then the weighted adjacency matrix $W = (w_{ij}) \in \{0, \alpha, \beta\}^{n \times n}$ of graph $\mathcal{G}$ is generated by

$$w_{ij} = \begin{cases} \text{Bernoulli}(p_{ij}), & \text{if } i \neq j, \\ 0, & \text{if } i = j, \end{cases} \tag{B.2}$$

where Bernoulli(p_{ij}) is a random variable satisfying the Bernoulli distribution with probability p_{ij} such that

$$\begin{cases} P_r(w_{ij} = \alpha) = p_{ij} = 1 - P_r(w_{ij} = 0), & \text{if } v_i \text{ and } v_j \text{ are in the same cluster,} \\ P_r(w_{ij} = \beta) = p_{ij} = 1 - P_r(w_{ij} = 0), & \text{if } v_i \text{ and } v_j \text{ are in different clusters.} \end{cases} \tag{B.3}$$

The SBM graph is then given by $\mathcal{G}(V, W)$.

Example B.1. Let $k = 3, n = 6$, we set vector $u = [u_1, u_2, u_3] = [2, 2, 2]$, parameters $\alpha = 3, \beta = 1$ for the edge weight, and parameters $a = 0.6, b = 0.2$ for the probability matrix P. First, we define a ground-truth clustering from u using the cluster indicator matrix H as follows

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T.$$

Based on H, we then use (B.1), (B.2) and (B.3) to generate the adjacency matrix W with given α, β, and a, b. Figure 6 depicts the SBM $\mathcal{G}(V, W)$ and the underlying ground-truth clustering $V = C_1 \cup C_2 \cup C_3$. □

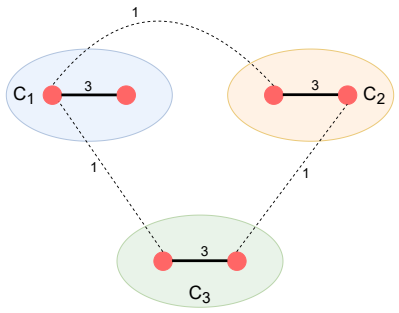

Figure 6: The SBM $\mathcal{G}(V, W)$ and the ground-truth clustering $V = C_1 \cup C_2 \cup C_3$ (Example B.1).

B.1.2 m-SBM

To take group fairness into account, we utilize a modified SBM (m-SBM) proposed in Kleindessner et al. (2019). In m-SBM, n vertices are partitioned into h disjoint groups such that $V = V_1 \cup \cdots \cup V_h$, and are assigned to k clusters for a prescribed fair ground-truth clustering $V = C_1 \cup \cdots \cup C_k$. An edge between a pair of vertices is placed with probabilities dependent only on whether the terminal vertices belong to the same cluster or group.

Specifically, let us first define $u = [u_1, u_2, \ldots, u_k]$ as the block vector consisting of k blocks. Each block $u_i \in \mathbb{N}^h$ contains h elements, and each element $u_i^{(j)}$ of u_i equals the number of vertices in $V_j \cap C_i$, i.e.,

$$u_i^{(j)} = |V_j \cap C_i|.$$

Consequently, we have

$$\sum_{i=1}^{k} u_i^{(j)} = |V_j|, \quad \sum_{j=1}^{h} u_i^{(j)} = |C_i|, \quad \text{and} \quad \sum_{i=1}^{k}\sum_{j=1}^{h} u_i^{(j)} = |V|.$$

To satisfy the fairness condition (2.5), for $j = 1, 2, \cdots, h$, the elements of the vector u should be chosen such that

$$\frac{u_i^{(j)}}{\sum_{j=1}^{h} u_i^{(j)}} = \frac{\sum_{i=1}^{k} u_i^{(j)}}{|V|} \quad \text{for any } i = 1, 2, \cdots, k. \tag{B.4}$$

Once u is set, we have

- Group-membership vectors $g^{(s)}$, for $s = 1, 2, \cdots, h$, as defined in (2.4), and the corresponding group-membership matrix F as defined in Lemma 2.1;
- Clustering indicator matrix $H \in \{0, 1\}^{n\times k}$ as defined in (2.2), containing the fair ground-truth clustering.

The probability matrix $P = (p_{ij}) \in \mathbb{R}^{n\times n}$ for edge connectivity is defined as follows:

$$p_{ij} = \begin{cases} a, & \text{if } v_i, v_j \text{ are in the same cluster and group,} \\ b, & \text{if } v_i, v_j \text{ are in different clusters but the same group,} \\ c, & \text{if } v_i, v_j \text{ are in the same cluster but different groups,} \\ d, & \text{if } v_i, v_j \text{ are in different clusters and groups,} \end{cases} \tag{B.5}$$

where $a > b > c > d$. Let α be the weight of within-cluster edges and β be the weight of between-cluster edges and $\alpha > \beta$, then the adjacency matrix of the m-SBM $\mathcal{G}(V, W)$ is given by $W = (w_{ij}) \in \{0, \alpha, \beta\}^{n\times n}$ as follows:

$$w_{ij} = \begin{cases} \text{Bernoulli}(p_{ij}), & \text{if } i \neq j, \\ 0, & \text{if } i = j, \end{cases} \tag{B.6}$$

where Bernoulli(p_{ij}) is a random variable satisfying the Bernoulli distribution with probability p_{ij} such that

$$\begin{cases} P_r(w_{ij} = \alpha) = p_{ij} = 1 - P_r(w_{ij} = 0), & \text{if } v_i \text{ and } v_j \text{ are in the same cluster,} \\ P_r(w_{ij} = \beta) = p_{ij} = 1 - P_r(w_{ij} = 0), & \text{if } v_i \text{ and } v_j \text{ are in different clusters.} \end{cases} \tag{B.7}$$

Example B.2. Let $k = 3$, $h = 2$ and $n = 10$, we set the block vector $u = [u_1, u_2, u_3] = [(2, 2), (2, 2), (1, 1)]$, parameters $\alpha = 3$ and $\beta = 1$ for the edge weight, and parameters $a = 0.6, b = 0.4, c = 0.2, d = 0.1$ for the probability matrix P. By the vector u, we have the following group membership vectors $g^{(s)}$ and the indicator matrix H of the fair ground-truth clustering

$$g^{(1)} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T,$$
$$g^{(2)} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}^T,$$
$$H = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T.$$

By (B.5), (B.6) and (B.7), with given weights α, β, probabilities a, b, c, d, group information $g^{(1)}, g^{(2)}$, and the cluster information in H, we can generate the adjacency matrix W. The m-SBM $\mathcal{G}(V, W)$ is shown in Figure 7. □

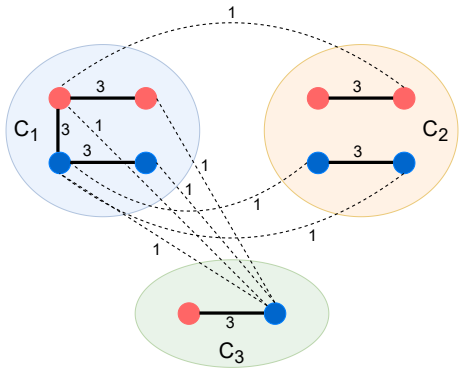

Figure 7: An m-SBM $\mathcal{G}(V, W)$ and fair ground-truth clustering $V = C_1 \cup C_2 \cup C_3$ with respect to the group partition $V = V_1 \cup V_2$, where V_1 is the set of red vertices and V_2 is the set of blue vertices (Example B.2).

B.2 Fairness Measured by Balance

For the metric of balance discussed in Section 4.2, we claimed "a higher balance implies a fairer clustering". Here we give a brief justification. First, let us recall the definitions of balance and average balance.

Definition B.1. Given a clustering $V = C_1 \cup \cdots \cup C_k$ and a non-overlapping group partition $V = V_1 \cup \cdots \cup V_h$, the *balance* of cluster C_ℓ for $\ell = 1, 2, \cdots, k$ is defined as

$$\text{Balance}(C_\ell) := \min_{s \neq s' \in \{1,\cdots,h\}} \frac{|V_s \cap C_\ell|}{|V_{s'} \cap C_\ell|}. \tag{4.2}$$

The *average balance* is defined as

$$\text{Average\_Balance} = \frac{1}{k} \sum_{\ell=1}^{k} \text{Balance}(C_\ell). \tag{4.3}$$

It follows from Kleindessner et al. (2019) that for any clustering, we have

$$\min_{\ell \in \{1,\cdots,k\}} \text{Balance}(C_\ell) \leq \min_{s \neq s' \in \{1,\cdots,h\}} \frac{|V_s|}{|V_{s'}|}. \tag{B.8}$$

Assume a clustering reaches the upper bound (B.8), i.e.,

$$\frac{|V_s \cap C_\ell|}{|V_{s'} \cap C_\ell|} = \frac{|V_s|}{|V_{s'}|}, \tag{B.9}$$

for $\ell = 1, \ldots, k$ and $s, s' = 1, \ldots, h$ and $s \neq s'$. Then we will have the group fairness as defined in (2.5), i.e.,

$$\frac{|V_s \cap C_\ell|}{|C_\ell|} = \frac{|V_s|}{|V|}, \tag{B.10}$$

for $\ell = 1, \ldots, k$ and $s = 1, \ldots, h$. To justify the equation above, we derive from (B.9) that

$$|V_s \cap C_\ell| \cdot |V_{s'}| = |V_s| \cdot |V_{s'} \cap C_\ell|. \tag{B.11}$$

Therefore,

$$\sum_{s \neq s' \in \{1,\ldots,h\}} |V_s \cap C_\ell| \cdot |V_{s'}| = \sum_{s \neq s' \in \{1,\ldots,h\}} |V_s| \cdot |V_{s'} \cap C_\ell|.$$

Consequently,

$$(|C_\ell| - |V_{s'} \cap C_\ell|)|V_{s'}| = (|V| - |V_{s'}|)|V_{s'} \cap C_\ell|,$$

or equivalently

$$\frac{|V_{s'} \cap C_\ell|}{|C_\ell|} = \frac{|V_{s'}|}{|V|}.$$

Multiplication of the above equation to (B.9) leads directly to the group fairness (2.5):

$$\frac{|V_s \cap C_\ell|}{|C_\ell|} = \frac{|V_s|}{|V|}.$$

Combining (4.2), (B.8), and (B.9), we can conclude that a higher value of Average_Balance indicates a fairer clustering.

B.3 Hyperparameter Tuning

s-FairSC (Algorithm 3) takes a few parameters as input, namely, parameter k (the number of clusters), parameter h (the number of groups), and shift parameter σ for Hotelling's deflation. In this section, we illustrate the effect of these parameters on the performance of our algorithm.

Parameter k. In Experiment 4 on Random Laplacian dataset, we report the performance with respect to the parameter k; see Figure 5.

Parameter h. We provide additional experiments on the performance with respect to the parameter h. Recall that the number of groups h is associated with the number of linear constraints in the fair SC model (2.16). We perform experiments with fixed probabilities a, b, c, d and number of clusters k, but different group numbers h. Figure 8 depicts the error rates and running time of SC, FairSC, and s-FairSC. SC and s-FairSC are tested for model sizes from $n = 1000$ to 10000. However, FairSC stops at $n = 4000$ due to its high computational cost, echoing results reported in Kleindessner et al. (2019).

From Figure 1, we observe that SC has high error rates and fails to recover the fair ground-truth clustering, while both FairSC and s-FairSC are able to retrieve the fair ground-truth clustering and s-FairSC is as accurate as FairSC. However, the running time of s-FairSC is only a fraction of that of FairSC. For instance, when $n = 4000$, s-FairSC is about $10\times$ to $12\times$ faster than FairSC. We also observe that s-FairSC is as scalable as SC without fairness constraints. Noticeably, we observe that when $h = 10$, s-FairSC is even faster than SC. For example, when $n = 10000$, s-FairSC takes only 62% time of SC. We anticipate that it is due to the factors such as the sparsity distribution of the random graph $\mathcal{G}(V, W)$, and the eigenvalue distribution of L_{n}^{σ} in (3.14). This issue is subject to further investigation.

Parameter σ. In our implementation,we use $\|L_{\mathrm{n}}\|_1$ as shift σ. Theoretical justification for the choice of the shift can be found in Lin et al. (2021). Given the equivalence of matrix norms, there is no significant difference with respect to the choice of the matrix norm.

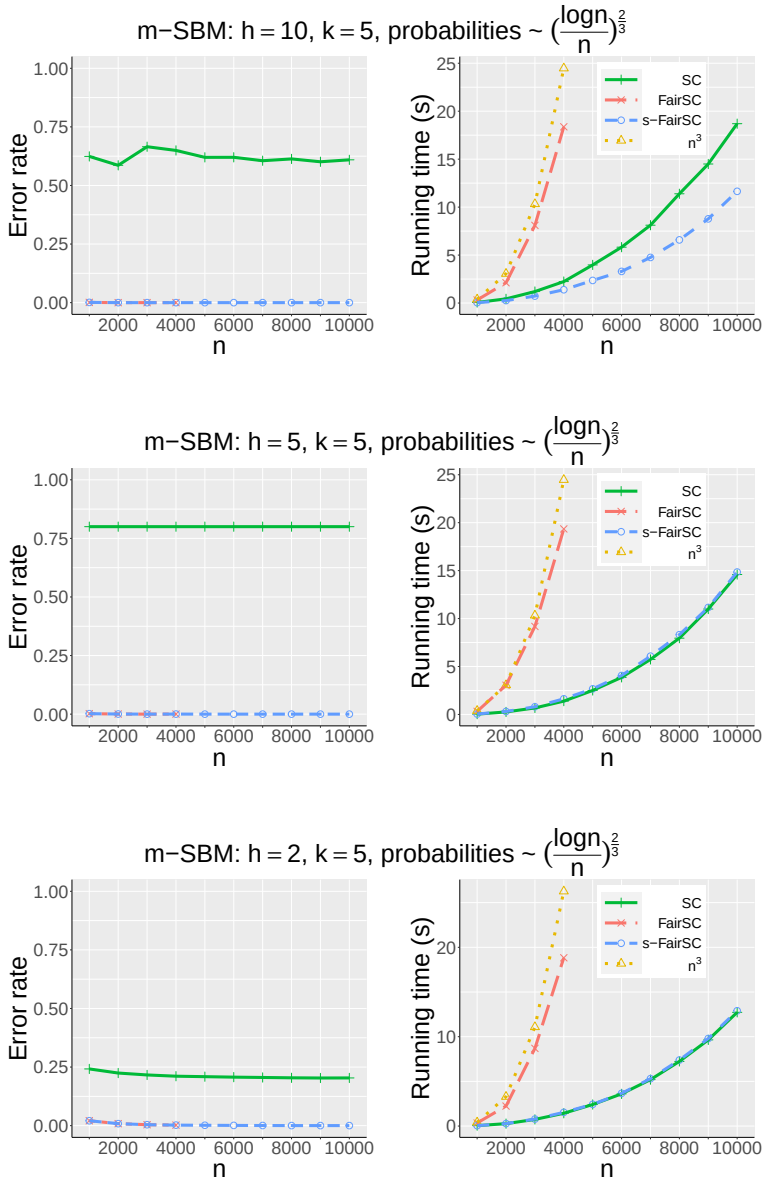

Figure 8: Error rates and running time (in seconds) of SC, FairSC, and s-FairSC on m-SBM with $h \in \{2, 5, 10\}$, edge connectivity probabilities proportional to $(\frac{\log n}{n})^{\frac{2}{3}}$, specifically $a = 10 \times (\frac{\log n}{n})^{2/3}$, $b = 7 \times (\frac{\log n}{n})^{2/3}$, $c = 4 \times (\frac{\log n}{n})^{2/3}$, $d = (\frac{\log n}{n})^{2/3}$.