Spectral and Semidefinite Relaxations of the CLUHSIC Algorithm

Wen-Yun Yang*†‡ James T. Kwok† Bao-Liang Lu‡§

Abstract

CLUHSIC is a recent clustering framework that unifies the geometric, spectral and statistical views of clustering. In this paper, we show that the recently proposed discriminative view of clustering, which includes the DIFFRAC and DisKmeans algorithms, can also be unified under the CLUHSIC framework. Moreover, CLUHSIC involves integer programming and one has to resort to heuristics such as iterative local optimization. In this paper, we propose two relaxations that are much more disciplined. The first one uses spectral techniques while the second one is based on semidefinite programming (SDP). Experimental results on a number of structured clustering tasks show that the proposed method significantly outperforms existing optimization methods for CLUHSIC. Moreover, it can also be used in semi-supervised classification. Experiments on real-world protein subcellular localization data sets clearly demonstrate the ability of CLUHSIC in incorporating structural and evolutionary information.

1 Introduction

Clustering has been an invaluable data analysis tool and is now widely used in diverse domains ranging from engineering, medical science, earth science, social science to economics. Over the decades, a battery of clustering algorithms have been developed. In general, these can be considered as representing three different views of clustering [24]: (1) geometric view, which includes methods like k-means; (2) spectral view, which includes methods such as normalized cut and various spectral clustering algorithms; (3) statistical view, which includes methods such as information bottleneck.

Recently, Song *et al.* [24] proposed the CLUHSIC ("Clustering using HSIC") framework that unifies these different views of clustering. It is based on statistical dependence using the Hilbert Schmidt Independence Criterion (HSIC) [9]. Besides, correlations and structural relationships among clusters can be utilized as side information for clustering. This will be useful when, for example, the user specifies that the clusters should form a hierarchy (e.g., document clusters in a document hierarchy), chain (e.g., image clusters in a video along the time axis) or even graphs. Very recently, Blaschko and Gretton [2] further extended CLUHSIC so that learning of an output taxonomy can be simultaneously performed with data clustering.

Despite its generality and success, there are two issues associated with CLUHSIC. First, while CLUHSIC unifies many existing views of clustering, a novel discriminative view of clustering has been recently proposed. Examples include the DIFFRAC [1], which uses a discriminative cost function as the objective, and DisKmeans [27], which performs subspace selection and discriminant analysis simultaneously. It is unclear whether this view can also be unified under CLUHSIC.

The second issue is that CLUHSIC involves optimizing an integer program which is NP-hard. To obtain an approximate solution, Song *et al.* [24] relied on a heuristic that iteratively updates the partitioning based on greedy local optimization. Later, two approaches that are more disciplined, namely, spectral relaxation and nonnegative matrix factorization (NMF), are also studied in [23]. However, the spectral relaxation in [23] is only applicable to the special case where the output has no structure. As for NMF, it needs to modify the kernel matrix $\mathbf{K}$ so that all its entries are non-negative. However, this makes $\mathbf{K}$ non-centered and CLUHSIC is no longer maximizing the HSIC as is supposed. Moreover, empirically, its performance with NMF is often poor [23]. Recently, Khot and Noar [12] proposed a constant factor polynomial time approximation algorithm for solving this integer program. However, this is only designed for the unnormalized version of CLUHSIC (whose performance is often inferior than the normalized version) and no experimental results are provided.

In this paper, we address the first issue by showing that the discriminative view of clustering can indeed be included under CLUHSIC (Section 3). As for the second issue, we propose two relaxations. The first one is spectral relaxation (Section 4) which, unlike the one studied in [23], is applicable to the general case where the output has non-trivial structure. However, it turns out that

*University of California, Los Angeles, California, USA

†Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong

‡Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China

§MOE-Microsoft Key Laboratory for Intelligent Computing and Intelligent Systems, Shanghai Jiao Tong University, Shanghai, China

this relaxation yields the same solution as the spectral relaxation of k-means and so is not useful in structured clustering. The second relaxation is based on semidefinite programming (SDP) (Section 5). Experimental results show that it significantly outperforms the other optimization heuristics for CLUHSIC. Moreover, CLUHSIC can also be extended for semi-supervised learning and experiments on protein subcellular localization also show similarly encouraging results (Section 6).

Notation The transpose of vector/matrix is denoted by the superscript ^{T}, and the identity matrix by $\mathbf{I}$. The vectors of all ones and zeros are denoted by $\mathbf{1}$ and $\mathbf{0}$, respectively. Moreover, $\mathrm{tr}(\mathbf{A})$ denotes the trace of matrix $\mathbf{A} = [A_{ij}]$, $\mathrm{rank}(\mathbf{A})$ is the rank of $\mathbf{A}$, $\|\mathbf{A}\|_F = \sqrt{\mathrm{tr}(\mathbf{A}^T\mathbf{A})}$ is its Frobenius norm, $\mathrm{vec}(\cdot)$ is the vectorizing operator that stacks the columns of a matrix in a column vector, $\mathbf{A} \geq \mathbf{0}$ denotes that every element of $\mathbf{A}$ is non-negative, and $\mathbf{A} \succeq 0$ denotes that $\mathbf{A}$ is symmetric, positive semidefinite (psd).

2 Clustering using HSIC (CLUHSIC)

CLUHSIC [24] is a clustering algorithm based on maximizing the (possibly highly nonlinear) dependence between input data and output cluster labels. Given a sample $\mathcal{S} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)\}$, the linear dependence between input $\mathbf{x}_i$'s and output $\mathbf{y}_i$'s can be easily estimated by simple statistics such as linear correlation. However, nonlinear dependencies are more difficult to measure. A recently proposed dependence (or, more precisely, independence) measure is the Hilbert Schmidt Independence Criterion (HSIC) [9]. Specifically, let ϕ and λ be the feature maps on the input and output, respectively. Denote the corresponding reproducing kernel Hilbert space (RKHS) by $\mathcal{F}$ and $\mathcal{G}$, and the corresponding kernels by $k(\cdot,\cdot)$ and $l(\cdot,\cdot)$. HSIC is defined as the square of the Hilbert-Schmidt norm $\|\cdot\|_{HS}$ of the cross-covariance operator $\mathbf{C}_{xy}$ from $\mathcal{F}$ to $\mathcal{G}$ [8]:

$$\mathbf{C}_{xy} = \mathbf{E}_{\mathbf{xy}}\left[(\phi(\mathbf{x}) - \mathbf{E}[\phi(\mathbf{x})]) \otimes (\lambda(\mathbf{y}) - \mathbf{E}[\lambda(\mathbf{y})])\right].$$

Here, $\otimes$ is the tensor product and $\mathbf{E}[\cdot]$ is the expectation operator. It can be shown that

$$\begin{aligned}\mathrm{HSIC}(\mathcal{F},\mathcal{G}) &= \|\mathbf{C}_{xy}\|^2_{HS}\\ &= \mathbf{E}_{\mathbf{xx'yy'}}[k(\mathbf{x},\mathbf{x}')l(\mathbf{y},\mathbf{y}')] + \mathbf{E}_{\mathbf{xx'}}[k(\mathbf{x},\mathbf{x}')]\mathbf{E}_{\mathbf{yy'}}[l(\mathbf{y},\mathbf{y}')]\\ &\quad -2\mathbf{E}_{\mathbf{xy}}[\mathbf{E}_{\mathbf{x'}}k(\mathbf{x},\mathbf{x}')][\mathbf{E}_{\mathbf{y'}}l(\mathbf{y},\mathbf{y}')]].\end{aligned}$$

Given the sample $\mathcal{S}$, an empirical estimate of HSIC is

$$(m-1)^{-2}\mathrm{tr}(\mathbf{HKHL}), \tag{2.1}$$

where $\mathbf{K} = [k(\mathbf{x}_i,\mathbf{x}_j)], \mathbf{L} = [l(\mathbf{y}_i,\mathbf{y}_j)]$ are kernel matrices defined on $\{\mathbf{x}_1,\ldots,\mathbf{x}_m\}$ and $\{\mathbf{y}_1,\ldots,\mathbf{y}_m\}$, respectively, and $\mathbf{H} = \mathbf{I} - \frac{1}{m}\mathbf{1}\mathbf{1}'$ is the so-called centering matrix. In the sequel, we will always assume that $\mathbf{K}$ is centered and so (2.1) can be simply written as

$$(m-1)^{-2}\mathrm{tr}(\mathbf{KL}). \tag{2.2}$$

Recent studies show that HSIC has several advantages over other independence measures [9, 23]. First, its empirical estimate in (2.2) is easy to compute. Moreover, it has good uniform convergence guarantees and very little bias even in high dimensions.

To use HSIC in clustering, one first defines a kernel matrix $\mathbf{A} \in \mathbb{R}^{c\times c}$ on a set of c clusters. This is used to model the prior structural relationships and correlation among clusters. In general, kernel entries for clusters that are structurally close to each other will be assigned high values. For example, for the chain structure in Figure 1(a), one can use the output kernel matrix $\begin{bmatrix} 2 & 1 & 0 & 0\\ 1 & 2 & 1 & 0\\ 0 & 1 & 2 & 1\\ 0 & 0 & 1 & 2\end{bmatrix}$. Here, the leftmost cluster in Figure 1(a) corresponds to the first row/column of the kernel matrix, the second-left cluster corresponds to the second row/column, and so on. For more complicated structures such as rings (Figure 1(b)) and trees (Figure 1(c)), the corresponding kernel matrices can be defined as $\begin{bmatrix} 2 & 1 & 0 & 1\\ 1 & 2 & 1 & 0\\ 0 & 1 & 2 & 1\\ 1 & 0 & 1 & 2\end{bmatrix}$ and $\begin{bmatrix} 2 & 1 & 0 & 0\\ 1 & 2 & 0 & 0\\ 0 & 0 & 2 & 1\\ 0 & 0 & 1 & 2\end{bmatrix}$, respectively.

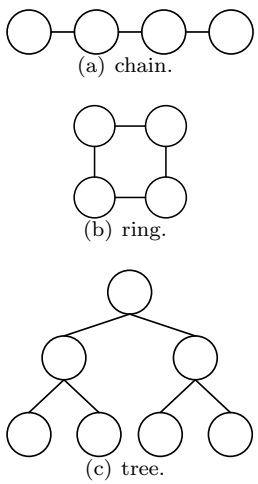


Figure 1: Some common structures among clusters.

Let $\mathbf{\Pi} \in \mathbb{R}^{m\times c}$ be a binary partition matrix such that the ith row specifies the assignment of the ith pattern to one of the c clusters. CLUHSIC aims at finding the cluster assignment $\mathbf{\Pi}$ that maximizes the dependence (as measured by HSIC) between the input kernel matrix $\mathbf{K}$ and the output kernel matrix $\mathbf{L} =$

$\mathbf{\Pi A \Pi}^T$ defined on the labels. Using (2.2), this leads to the following optimization problem:

$$\begin{aligned} (2.3) \qquad \max_{\mathbf{\Pi}} \quad & \operatorname{tr}(\mathbf{K \Pi A \Pi}^T) \\ \text{s.t.} \quad & \mathbf{\Pi 1} = \mathbf{1},\ \Pi_{ij} \in \{0,1\}. \end{aligned}$$

In clustering, it is often necessary to normalize each cluster's contribution to the objective by its size [13]. In CLUHSIC, $\mathbf{\Pi}$ can be normalized such that each column (cluster) has the same norm. The optimization problem for the normalized version of (2.3) is then:

$$\begin{aligned} (2.4) \qquad \max_{\mathbf{P}} \quad & \operatorname{tr}(\mathbf{KPAP}^T) \\ \text{s.t.} \quad & \mathbf{P} = \mathbf{\Pi D},\ \mathbf{\Pi 1} = \mathbf{1},\ \Pi_{ij} \in \{0,1\}, \end{aligned}$$

where $\mathbf{D} = (\mathbf{\Pi}^T\mathbf{\Pi})^{-1/2}$ is a diagonal matrix. Empirically, this normalized version often performs better, and so will be of our primary interest here.

3 CLUHSIC and Discriminative Clustering

The formulation in (2.3) and (2.4) is very flexible. In particular, by using different $\mathbf{A}$'s in the output label space, many traditional clustering algorithms such as the k-means, weighted k-means, kernel k-means, hierarchical clustering, and spectral clustering can all be subsumed under this CLUHSIC framework [24]. In this section, we will show that the recently proposed discriminative clustering approach, including the DIFFRAC [1] and DisKmeans [27] algorithms, can also be considered as special cases of CLUHSIC.

3.1 DIFFRAC [1] Suppose that we are given a set of d-dimensional unlabeled data $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]^T \in \mathbb{R}^{m \times d}$ coming from c underlying classes. With the use of a linear model $f(\mathbf{x}) = \mathbf{W}^T\mathbf{x}$ (where $\mathbf{W} \in \mathbb{R}^{d \times c}$), DIFFRAC simultaneously solves for the partition matrix $\mathbf{\Pi}$ and the weight matrix $\mathbf{W}$ by optimizing the following linear discriminative cost function:

$$(3.5) \qquad \min_{\mathbf{\Pi}, \mathbf{W}} \ \frac{1}{m}\|\mathbf{\Pi} - \mathbf{XW}\|_F^2 + \kappa\|\mathbf{W}\|_F^2,$$

where κ is a tradeoff parameter. Note that if $\mathbf{\Pi}$ were known, (3.5) reduces to the standard regularized linear regression problem. Hence, the idea of discriminative clustering is to adopt a cost function originally used for classification as a clustering criterion.

3.1.1 DIFFRAC is a Special Case of CLUHSIC It is easy to show that (3.5) can be reduced to

$$(3.6) \qquad \max_{\mathbf{\Pi}} \ \operatorname{tr}((\tilde{\mathbf{K}} - \mathbf{I})\mathbf{\Pi\Pi}^T),$$

where $\tilde{\mathbf{K}} = \mathbf{X}\left(\mathbf{X}^T\mathbf{X} + m\kappa\mathbf{I}\right)^{-1}\mathbf{X}^T$. The derivation is similar to that in [1] and so is not shown here because of the lack of space. The objective function can be further rewritten as

$$\begin{aligned} \operatorname{tr}((\tilde{\mathbf{K}} - \mathbf{I})\mathbf{\Pi\Pi}^T) &= \operatorname{tr}(\tilde{\mathbf{K}}\mathbf{\Pi\Pi}^T) - \operatorname{tr}(\mathbf{\Pi\Pi}^T) \\ &= \operatorname{tr}(\tilde{\mathbf{K}}\mathbf{\Pi\Pi}^T) - m. \end{aligned}$$

On dropping the constant m, this is thus the same as CLUHSIC with the use of a transformed (input) kernel matrix $\tilde{\mathbf{K}}$ and an output kernel matrix $\mathbf{A} = \mathbf{I}$.

3.1.2 Adding Structure Information to DIFFRAC Since DIFFRAC implicitly uses $\mathbf{A} = \mathbf{I}$, it cannot utilize structure information among the cluster labels. Here, we show that DIFFRAC can also be extended for the incorporation of structure information.

To achieve this, we replace the Frobenius norm in (3.5) with the weighted Frobenius norm $\|\cdot\|_{M,N}$. In general, for a matrix $\mathbf{Z} \in \mathbb{R}^{m \times n}$, one can define the matrix norm

$$\|\mathbf{Z}\|_{M,N} = \|\mathbf{M}^{\frac{1}{2}}\mathbf{Z}\mathbf{N}^{-\frac{1}{2}}\|_F,$$

where $\mathbf{M} \in \mathbb{R}^{m \times m}, \mathbf{N} \in \mathbb{R}^{n \times n}$ are positive definite matrices [29]. In our context, in order to encode the correlations between outputs as defined via $\mathbf{A}$, we choose $\mathbf{M} = \mathbf{I}$ and $\mathbf{N} = \mathbf{A}^{-1}$. Then $\|\mathbf{Z}\|^2_{\mathbf{I},\mathbf{A}^{-1}} = \operatorname{tr}(\mathbf{ZAZ}^T)$ captures the correlation of the various components of $\mathbf{Z}$. On substituting this weighted Frobenius norm in (3.5), and following a similar derivation as in [1], it can be shown that we will obtain an optimization problem which is analogous to (3.6)

$$\begin{aligned} (3.7) \quad & \max_{\mathbf{\Pi}} \ \operatorname{tr}\left((\tilde{\mathbf{K}} - \mathbf{I})\mathbf{\Pi A \Pi}^T\right) \\ & = \max_{\mathbf{\Pi}} \ \left\{\operatorname{tr}(\tilde{\mathbf{K}}\mathbf{\Pi A \Pi}^T) - \operatorname{tr}(\mathbf{\Pi A \Pi}^T)\right\}. \end{aligned}$$

Typically, the self-similarities of the clusters are the same, so $A_{ii} = d$ for some $d > 0$. Then, $\operatorname{tr}(\mathbf{\Pi A \Pi}^T) = md$, and (3.7) again reduces to

$$\max_{\mathbf{\Pi}} \ \operatorname{tr}(\tilde{\mathbf{K}}\mathbf{\Pi A \Pi}^T),$$

which is of the form of CLUHSIC.

3.2 DisKmeans [27] DisKmeans achieves discriminative clustering by performing subspace selection and linear discriminant analysis (LDA) simultaneously. Let the normalized partition matrix be $\mathbf{P}$. The between-cluster scatter of the data in the original input space is $\mathbf{S}_b = \mathbf{X}^T\mathbf{PP}^T\mathbf{X}$ while its within-cluster variance $\mathbf{S}_w = \mathbf{X}^T\mathbf{X} - \mathbf{X}^T\mathbf{PP}^T\mathbf{X}$. Let the projection matrix be $\mathbf{R} \in \mathbb{R}^{d \times b}$, where $b < d$ is the dimensionality of the projected space. DisKmeans then finds $\mathbf{P}$ and $\mathbf{R}$ such that the between-cluster scatter of the projected data $\mathbf{R}^T\mathbf{S}_b\mathbf{R}$ is maximized while its within-cluster variance $\mathbf{R}^T\mathbf{S}_w\mathbf{R}$ (or the total scatter $\mathbf{R}^T(\mathbf{S}_w + \mathbf{S}_b)\mathbf{R}$) is minimized as in standard LDA.

3.2.1 DisKmeans is a Special Case of CLUHSIC Since DisKmeans is equivalent to kernel k-means with a particular kernel [27] and kernel k-means is a special case of CLUHSIC [24], it is thus immediate that CLUHSIC also includes DisKmeans.

3.2.2 Adding Structure Information to DisKmeans As in Section 3.1.2, we can also extend DisKmeans by incorporating the correlation between clusters defined in $\mathbf{A}$. We replace the between-cluster scatter $\mathbf{S}_b$ by its correlated variant

$$\tilde{\mathbf{S}}_b = \mathbf{X}^T\mathbf{P}\mathbf{A}\mathbf{P}^T\mathbf{X}, \tag{3.8}$$

and similarly the within-cluster scatter $\mathbf{S}_w$ by

$$\tilde{\mathbf{S}}_w = \mathbf{X}^T\mathbf{X} - \mathbf{X}^T\mathbf{P}\mathbf{A}\mathbf{P}^T\mathbf{X}. \tag{3.9}$$

DisKmeans then maximizes the between-cluster scatter and minimizes the regularized total scatter of the projected data, i.e.,

$$\max_{\mathbf{P},\mathbf{R}} \operatorname{tr}\left((\mathbf{R}^T(\tilde{\mathbf{S}}_w + \tilde{\mathbf{S}}_b + \lambda\mathbf{I})\mathbf{R})^{-1}\mathbf{R}^T\tilde{\mathbf{S}}_b\mathbf{R}\right), \tag{3.10}$$

where λ is a regularization parameter. Using the representer theorem, the optimal $\mathbf{R}$ can be expressed as

$$\mathbf{R} = \mathbf{X}^T\mathbf{B} \tag{3.11}$$

for some $\mathbf{B} \in \mathbb{R}^{m\times b}$. Substituting (3.8), (3.9) and (3.11) into (3.10), we obtain the optimization problem

$$\max_{\mathbf{P},\mathbf{B}} \operatorname{tr}[(\mathbf{B}^T(\mathbf{K}\mathbf{K} + \lambda\mathbf{K})\mathbf{B})^{-1}\mathbf{B}^T\mathbf{K}\mathbf{P}\mathbf{A}\mathbf{P}\mathbf{K}\mathbf{B}], \tag{3.12}$$

where $\mathbf{K} = \mathbf{X}^T\mathbf{X}$. It can be shown that we can factor out the matrix $\mathbf{B}$ from (3.12) and obtain the following proposition.

PROPOSITION 1. *The optimal* $\mathbf{P}$ *in (3.12) can be obtained from the following trace maximization problem*

$$\mathbf{P}^* = \max_{\mathbf{P}} tr\left[\left(\mathbf{I} - (\mathbf{I} + \frac{1}{\lambda}\mathbf{K})^{-1}\right)\mathbf{P}\mathbf{A}\mathbf{P}^T\right].$$

Proof. The proof follows a similar derivation as in [27] and is not shown here because of the lack of space.

Thus, this extension can also be regarded as a variant of CLUHSIC with the transformed input kernel matrix $\mathbf{I} - (\mathbf{I} + \frac{1}{\lambda}\mathbf{K})^{-1}$ and output kernel matrix $\mathbf{A}$.

4 Spectral Relaxation

Recall that CLUHSIC leads to the optimization problem (2.4) which is a difficult integer program. In order to improve its computational efficiency, Song *et al.* [24] relied on a heuristic that iteratively updates the partitioning based on greedy local optimization. Obviously, a more disciplined optimization approach is highly desirable. As discussed in Section 1, a preliminary study on the spectral relaxation of CLUHSIC has been investigated in [23]. However, it only considers the special case where $\mathbf{A} = \mathbf{I}$, which unfortunately only corresponds to the spectral relaxation of standard k-means with no output structure [28].

4.1 The General Case $\mathbf{A} \neq \mathbf{I}$ In this section, we consider the general case where $\mathbf{A} \neq \mathbf{I}$. By ignoring the special structure of $\mathbf{P}$ in (2.4) and let it be an arbitrary orthonormal matrix, we obtain the following relaxed trace maximization problem:

$$\begin{aligned} \max_{\mathbf{P}} \quad & \operatorname{tr}(\mathbf{K}\mathbf{P}\mathbf{A}\mathbf{P}^T) \\ \text{s.t.} \quad & \mathbf{P}^T\mathbf{P} = \mathbf{I}. \end{aligned} \tag{4.13}$$

Our main tool is a technical result from [21].

THEOREM 1. *(Theorem 3.1 of [21]) Let the eigenvalues of* $\mathbf{A}$ *(resp.* $\mathbf{B}$*) be* $\lambda_1 \geq \cdots \geq \lambda_m \geq 0$ *(resp.* $\mu_1 \geq \cdots \geq \mu_c \geq 0$*). Then,* $\max_{\mathbf{X}^T\mathbf{X}=\mathbf{I}} tr(\mathbf{A}\mathbf{X}\mathbf{B}\mathbf{X}^T) = \sum_{i=1}^{m} \lambda_i\mu_i$.

The optimal $\mathbf{P}$ in (4.13) can then be obtained by the following proposition.

PROPOSITION 2. *Let the eigenvalues of* $\mathbf{K}$ *(resp.* $\mathbf{A}$*) be* $\lambda_1 \geq \cdots \geq \lambda_m \geq 0$ *(resp.* $\mu_1 \geq \cdots \geq \mu_c \geq 0$*) and the matrix containing the eigenvectors be* $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_m]$ *(resp.* $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_c]$*). Then,*

$$\max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} tr(\mathbf{K}\mathbf{P}\mathbf{A}\mathbf{P}^T) = \sum_{i=1}^{c} \lambda_i\mu_i,$$

and the optimal is

$$\mathbf{P}^* = [\mathbf{u}_1, \cdots, \mathbf{u}_c]\mathbf{V}^T. \tag{4.14}$$

Proof. Let $\hat{\mathbf{P}} \in \mathbb{R}^{m\times m}$ be an orthonormal matrix. It can be partitioned as $\hat{\mathbf{P}} = [\mathbf{P}, \mathbf{P}_\perp]$ where $\mathbf{P}_\perp$ is the orthogonal complement of $\mathbf{P}$. Let $\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \in \mathbb{R}^{m\times m}$. It is easy to verify that $\mathbf{K}\hat{\mathbf{P}}\hat{\mathbf{A}}\hat{\mathbf{P}}^T = \mathbf{K}\mathbf{P}\mathbf{A}\mathbf{P}^T$, and hence

$$\max_{\hat{\mathbf{P}}^T\hat{\mathbf{P}}=\mathbf{I}} \operatorname{tr}(\mathbf{K}\hat{\mathbf{P}}\hat{\mathbf{A}}\hat{\mathbf{P}}^T) = \max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \operatorname{tr}(\mathbf{K}\mathbf{P}\mathbf{A}\mathbf{P}^T).$$

Moreover, let $\mathbf{\Lambda} = \operatorname{diag}([\mu_1, \ldots, \mu_c])$, then $\hat{\mathbf{A}}$ can be eigen-decomposed as

$$\begin{aligned} \hat{\mathbf{A}} &= \hat{\mathbf{V}}\hat{\mathbf{\Lambda}}\hat{\mathbf{V}}^T \\ &= \begin{bmatrix} \mathbf{V} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{\Lambda} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{V}^T & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \end{aligned}$$

Using Theorem 1 above, we have

$$\max_{\hat{\mathbf{P}}^T\hat{\mathbf{P}}=\mathbf{I}} \mathrm{tr}(\mathbf{K}\hat{\mathbf{P}}\hat{\mathbf{A}}\hat{\mathbf{P}}^T) = \sum_{i=1}^{c} \lambda_i\mu_i$$

when $\hat{\mathbf{P}} = \mathbf{U}\hat{\mathbf{V}}^T$, which corresponds to the optimal $\mathbf{P}$ of $\mathbf{U}\begin{bmatrix}\mathbf{V}^T\\ \mathbf{0}\end{bmatrix} = [\mathbf{u}_1,\ldots,\mathbf{u}_c]\mathbf{V}^T$. □

Recall that the spectral relaxation of k-means [28] leads to a similar trace maximization problem

$$\max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \mathrm{tr}(\mathbf{K}\mathbf{P}\mathbf{P}^T). \tag{4.15}$$

Its optimal $\mathbf{P}$ is given by $[\mathbf{u}_1,\cdots,\mathbf{u}_c]\mathbf{S}$, where the $\mathbf{u}_i$'s are as defined in Proposition 2 and $\mathbf{S}$ is an arbitrary orthogonal matrix. Now, since the matrix $\mathbf{V}$ in (4.14) is orthonormal (and thus orthogonal), the optimal $\mathbf{P}$ solution of (4.13), i.e., $\mathbf{P}^*$ in (4.14), is thus also optimal for (4.15). In other words, the spectral relaxations of CLUHSIC and k-means yield the same cluster assignment. Unfortunately, this also implies that spectral relaxation cannot gain additional information from $\mathbf{A}$.

4.2 Concave Convex Procedure Recently, in the context of kernelized sorting, Quadrianto *et al.* [19] proposed the use of the concave convex procedure to optimize over the set of permutation matrices. While the $\mathbf{P}$ here is not a permutation matrix, we find that a similar approach can also be used for CLUHSIC. Specifically, let the estimate of $\mathbf{P}$ at the tth iteration be $\mathbf{P}_t$. We replace the objective $\mathrm{tr}(\mathbf{K}\mathbf{P}\mathbf{A}\mathbf{P}^T)$ in (2.4) by its first-order approximation at $\mathbf{P}_t$

$$\begin{aligned}&\mathrm{tr}(\mathbf{K}\mathbf{P}_t\mathbf{A}\mathbf{P}_t^T) + 2\mathrm{tr}[(\mathbf{P}-\mathbf{P}_t)^T\mathbf{K}\mathbf{P}_t\mathbf{A}]\\ =\;& 2\mathrm{tr}(\mathbf{P}^T\mathbf{B}_t) + \text{constant},\end{aligned} \tag{4.16}$$

where $\mathbf{B}_t = \mathbf{K}\mathbf{P}_t\mathbf{A}$. This linearized objective is then maximized w.r.t. $\mathbf{P}$ and the process re-iterated until convergence. Interestingly, we will show in the following that this procedure leads to the same spectral relaxation in Section 4.1.

First, maximizing the linearized objective in (4.16) is obviously the same as

$$\max_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \mathrm{tr}(\mathbf{P}^T\mathbf{B}_t). \tag{4.17}$$

Since

$$\begin{aligned}\|\mathbf{P}-\mathbf{B}_t\|_F^2 &= \mathrm{tr}((\mathbf{P}-\mathbf{B}_t)^T(\mathbf{P}-\mathbf{B}_t))\\ &= \mathrm{tr}(\mathbf{P}^T\mathbf{P} - \mathbf{P}^T\mathbf{B}_t - \mathbf{B}_t^T\mathbf{P} + \mathbf{B}_t^T\mathbf{B}_t)\\ &= -2\mathrm{tr}(\mathbf{P}^T\mathbf{B}_t) + \text{constant},\end{aligned}$$

(4.17) is also equivalent to: $\min_{\mathbf{P}^T\mathbf{P}=\mathbf{I}} \|\mathbf{P}-\mathbf{B}_t\|_F$, which is a matrix nearness problem[1] [10]. From matrix theory, its optimal $\mathbf{P}^*$ can be obtained from the polar decomposition[2] of $\mathbf{B}_t$ [10]. Besides, when the above iteration converges (i.e., $\mathbf{P}_t = \mathbf{P}_{t+1}$), $\mathbf{P}^*$ satisfies (4.17), and so

$$\max_{\mathbf{P}^{*T}\mathbf{P}^*=\mathbf{I}} \mathrm{tr}(\mathbf{P}^{*T}\mathbf{B}) = \max_{\mathbf{P}^{*T}\mathbf{P}^*=\mathbf{I}} \mathrm{tr}(\mathbf{K}\mathbf{P}^*\mathbf{A}\mathbf{P}^{*T}),$$

which is the same as the spectral relaxation in (4.13). This equivalence is also confirmed in the experiments.

5 Semidefinite Relaxation

In this section, we propose another disciplined optimization approach for solving the difficult integer program in CLUHSIC. This relaxation results in a semidefinite program (SDP), which optimizes a linear objective over the cone of positive semidefinite matrices. SDP relaxations have been highly successful in machine learning and data mining. They can construct tight convex approximations in many hard combinatorial optimization problems [3].

As will be shown in Section 5.1, the resultant optimization problem is a standard SDP and can be solved in polynomial time by off-the-shelf interior point solvers [3]. However, these general-purpose solvers can be slow. In Section 5.2, we show that the matrix variable $\mathbf{Q}$ in the SDP should ideally have rank one, and so it is natural to consider a low-rank SDP solver instead. SDP-LR [4] eliminates the psd constraint by replacing $\mathbf{Q}$ with a factorization of the form $\mathbf{Y}\mathbf{Y}^T$, where $\mathbf{Y}$ is a low-rank matrix, and then applies a fast first-order nonlinear optimizer. Empirically, this approach directly exploits the low-rank structure and significantly faster than existing interior-point solvers. It has also been successfully used in many machine learning and data mining problems such as embedding, clustering and semi-supervised learning [5, 14, 22]. Finally, the recovery of the cluster assignment from the obtained matrix variable will be discussed in Section 5.3.

5.1 Optimization Problem Recall that DIFFRAC can be considered as a special case of CLUHSIC (Section 3.1). A SDP relaxation has already been proposed for DIFFRAC, which is based on relaxing $\mathbf{\Pi}\mathbf{\Pi}^T$ in (3.6) as a matrix variable [1]. However, with the incorpora-

[1] Here, the problem consist of finding the orthonormal matrix $\mathbf{P}$ that is nearest to $\mathbf{B}_t$, where distance is measured by the Frobenius norm.

[2] For $\mathbf{A} \in \mathbb{R}^{m\times n}$, $m \geq n$, there exists a matrix $\mathbf{U} \in \mathbb{R}^{m\times n}$ with orthonormal columns and a unique positive semi-definite matrix $\mathbf{H} \in \mathbb{R}^{n\times n}$ such that $\mathbf{A} = \mathbf{U}\mathbf{H}$. This is called the polar decomposition of $\mathbf{A}$ [10].

tion of structure information in CLUHSIC, $\mathbf{\Pi\Pi}^T$ becomes $\mathbf{\Pi A \Pi}^T$ and one cannot simply replace $\mathbf{\Pi A \Pi}^T$ by a matrix variable. The relaxation of CLUHSIC's optimization problem into an SDP is thus more involved.

First, we will need the following property of the Kronecker product [16]:

LEMMA 1. *For* $\mathbf{A} \in R^{m\times n}, \mathbf{B} \in \mathbb{R}^{n\times p}$ *and* $\mathbf{C} \in \mathbb{R}^{p\times q}$,

$$(\mathbf{C}' \otimes \mathbf{A})vec(\mathbf{B}) = vec(\mathbf{ABC}).$$

We can then rewrite the objective in (2.4) as

$$\begin{aligned} \mathrm{tr}(\mathbf{KPAP}^T) &= \langle \mathrm{vec}(\mathbf{P}), \mathrm{vec}(\mathbf{KPA})\rangle \\ &= \mathrm{vec}(\mathbf{P})^T(\mathbf{A}\otimes\mathbf{K})\mathrm{vec}(\mathbf{P}) \\ &= \mathrm{tr}((\mathbf{A}\otimes\mathbf{K})\mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T) \\ &= \mathrm{tr}((\mathbf{A}\otimes\mathbf{K})\mathbf{Q}), \end{aligned}$$

where $\mathbf{Q} = \mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T$. Thus, the CLUHSIC problem in (2.4) can be equivalently formulated in terms of $\mathbf{Q}$, as

(5.18)
$$\max_{\mathbf{Q}} \quad \mathrm{tr}((\mathbf{A}\otimes\mathbf{K})\mathbf{Q})$$
$$\text{s.t.} \quad \mathbf{Q} \in \mathcal{Q} = \left\{ \mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T \;\middle|\; \begin{array}{l} \mathbf{P} = \mathbf{\Pi D}, \\ \mathbf{\Pi 1} = \mathbf{1}, \\ \Pi_{ij} \in \{0,1\} \end{array} \right\}.$$

Note that this combinatorial optimization is still NP-hard [7]. However, the objective is now linear in the matrix variable $\mathbf{Q}$. In the following, we will show how an efficient convex relaxation can be obtained.

First, we introduce some notations. Let $\mathbf{Q}_{[ij]}$ be the (ij)th block of the matrix $\mathbf{Q} = \mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T$, i.e.,

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{[11]} & \cdots & \mathbf{Q}_{[1c]} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{[c1]} & \cdots & \mathbf{Q}_{[cc]} \end{bmatrix} = \begin{bmatrix} \mathbf{p}_1\mathbf{p}_1^T & \cdots & \mathbf{p}_1\mathbf{p}_c^T \\ \vdots & \ddots & \vdots \\ \mathbf{p}_c\mathbf{p}_1^T & \cdots & \mathbf{p}_c\mathbf{p}_c^T \end{bmatrix},$$

where $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_c]$. Define the two linear operators $\mathrm{bdiag} : \mathbb{R}^{mc\times mc} \to \mathbb{R}^{m\times m}$ and $\mathrm{odiag} : \mathbb{R}^{mc\times mc} \to \mathbb{R}^{c\times c}$ such that

$$\mathrm{bdiag}(\mathbf{Q}) = \sum_{i=1}^{c} \mathbf{Q}_{[ii]}, \tag{5.19}$$

$$(\mathrm{odiag}(\mathbf{Q}))_{ij} = \mathrm{tr}(\mathbf{Q}_{[ij]}), \quad i,j = 1,\ldots,c. \tag{5.20}$$

It is easy to show that for $\mathbf{Q} = \mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T$,

$$\mathrm{bdiag}(\mathbf{Q}) = \mathbf{PP}^T, \tag{5.21}$$

$$\mathrm{odiag}(\mathbf{Q}) = \mathbf{P}^T\mathbf{P}. \tag{5.22}$$

To obtain an efficient convex relaxation, we replace the non-convex constraint $\mathbf{Q} = \mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T$ by the positive semidefinite constraint $\mathbf{Q} \succeq 0$. By using the following properties of $\mathbf{P}$, we can also replace the other constraints in $\mathcal{Q}$ by convex constraints on $\mathbf{Q}$.

1. Columns of $\mathbf{P}$ are orthonormal. Using (5.21), this is equivalent to the constraint $\mathrm{odiag}(\mathbf{Q}) = \mathbf{I}$.
2. Each column of $\mathbf{PP}^T$ has a constant sum of one. Using (5.22), this is equivalent to the constraint $\mathrm{bdiag}(\mathbf{Q})\mathbf{1} = \mathbf{1}$.
3. $\Pi_{ij} \in \{0,1\}$ is relaxed to the constraint $\mathbf{Q} \geq \mathbf{0}$.

Combining these three constraints with the positive semidefinite constraint, we arrive at the following SDP relaxation of (5.18):

$$\max_{\mathbf{Q}} \quad \mathrm{tr}((\mathbf{A}\otimes\mathbf{K})\mathbf{Q}) \tag{5.23}$$
$$\text{s.t.} \quad \mathbf{Q} \in \mathcal{C} = \left\{ \begin{array}{l} \mathbf{Q} \succeq 0, \mathbf{Q} \geq \mathbf{0}, \\ \mathrm{odiag}(\mathbf{Q}) = \mathbf{I}, \\ \mathrm{bdiag}(\mathbf{Q})\mathbf{1} = \mathbf{1} \end{array} \right\}.$$

This is a standard SDP and can be solved in polynomial time by off-the-shelf solvers.

5.2 Low-Rank SDP Recall that a crucial step in the above relaxation is to replace the constraint $\mathbf{Q} = \mathrm{vec}(\mathbf{P})\mathrm{vec}(\mathbf{P})^T$ by $\mathbf{Q} \succeq 0$. Interestingly, the following proposition shows that one can recover the original CLUHSIC formulation by adding back the rank-one constraint on $\mathbf{Q}$.

For any $\mathbf{Q} \in \mathcal{C} \cap \mathcal{R}$, where $\mathcal{C}$ is defined in (5.23) and $\mathcal{R} = \{\mathbf{Q} : \mathrm{rank}(\mathbf{Q}) = 1\}$, we can first decompose it into $\mathbf{Q} = \mathrm{vec}(\widetilde{\mathbf{P}})\mathrm{vec}(\widetilde{\mathbf{P}})^T$, then we will prove that this $\widetilde{\mathbf{P}}$ has the following properties, which eventually guarantee that $\mathbf{Q} \in \mathcal{Q}$.

LEMMA 2. $\widetilde{\mathbf{P}} \geq 0$ *or* $\widetilde{\mathbf{P}} \leq 0$, *and there is only one nonzero element in each row of* $\widetilde{\mathbf{P}}$.

Proof. The constraint $\mathbf{Q} = \mathrm{vec}(\widetilde{\mathbf{P}})\mathrm{vec}(\widetilde{\mathbf{P}})^T \geq \mathbf{0}$ in $\mathcal{C}$ implies that the product between every two elements of $\widetilde{\mathbf{P}}$ must be non-negative. This implies either $\widetilde{\mathbf{P}} \geq \mathbf{0}$ or $\widetilde{\mathbf{P}} \leq \mathbf{0}$. Moreover, the constraint $\mathrm{odiag}(\mathbf{Q}) = \mathbf{I}$ in $\mathcal{C}$ is equivalent to $\widetilde{\mathbf{P}}^T\widetilde{\mathbf{P}} = \mathbf{I}$. This column orthogonality of $\widetilde{\mathbf{P}}$ then ensures that each row of $\mathbf{P}$ has only one nonzero element. □

Let the ith column of $\widetilde{\mathbf{P}}$ be $\widetilde{\mathbf{p}}_i$, then $\widetilde{\mathbf{P}} = [\widetilde{\mathbf{p}}_1, \widetilde{\mathbf{p}}_2, \ldots, \widetilde{\mathbf{p}}_c]$. Denote the jth element of $\widetilde{\mathbf{p}}_i$ by $\widetilde{p}_{ij}$.

LEMMA 3. *Given the* u*th column vector* $\widetilde{\mathbf{p}}_u$, *for any* $\widetilde{p}_{u\alpha} \neq 0$ *and* $\widetilde{p}_{u\beta} \neq 0$ *where* $1 \leq \alpha, \beta \leq m$, $(\alpha \neq \beta)$, *then* $\widetilde{p}_{u\alpha} = \widetilde{p}_{u\beta}$. *In other words, the nonzero elements in each column of* $\widetilde{\mathbf{P}}$ *are equal to each other.*

Proof. The constraint $\mathrm{bdiag}(\mathbf{Q})\mathbf{1} = \sum_{i=1}^{c} \mathbf{Q}_{[ii]} = \mathbf{1}$, is equivalent to

$$\sum_{i=1}^{c}\sum_{j=1}^{m} \widetilde{p}_{ir}\widetilde{p}_{ij} = 1, \text{ for } 1 \leq r \leq m. \tag{5.24}$$

From Lemma 2, there is only one nonzero element in each row. Without loss of generality, assume that the αth and βth rows both have nonzero elements in the uth column. Then, from (5.24), we could get the following two equations,

$$\sum_{i=1}^{c}\sum_{j=1}^{m}\widetilde{p}_{i\alpha}\widetilde{p}_{ij} = \widetilde{p}_{u\alpha}\sum_{j=1}^{m}\widetilde{p}_{uj} = 1,$$
$$\sum_{i=1}^{c}\sum_{j=1}^{m}\widetilde{p}_{i\beta}\widetilde{p}_{ij} = \widetilde{p}_{u\beta}\sum_{j=1}^{m}\widetilde{p}_{uj} = 1,$$

as $\widetilde{p}_{i\alpha} = 0$ and $\widetilde{p}_{i\beta} = 0$ for $i \neq u$. Therefore we can obtain that $\widetilde{p}_{u\alpha}$ and $\widetilde{p}_{u\beta}$ have the equal value of $1/\sum_{j=1}^{m}\widetilde{p}_{uj}$. □

PROPOSITION 3. *Problem (2.4) is equivalent to*

$$\begin{aligned} &\max_{\mathbf{Q}} && tr((\mathbf{A}\otimes\mathbf{K})\mathbf{Q}) \\ &s.t. && \mathbf{Q}\in\mathcal{C}\cap\mathcal{R}, \end{aligned} \tag{5.25}$$

where $\mathcal{C}$ is defined in (5.23) and $\mathcal{R} = \{\mathbf{Q} : rank(\mathbf{Q}) = 1\}$.

Proof. Note that problem (2.4) is equivalent to problem (5.18), thus we turn to prove the equivalence between problems (5.18) and (5.25).

As the objective functions in (5.25) and (5.18) are the same, we only need to show the equivalence of feasible regions, i.e., $\mathcal{Q} = \mathcal{C}\cap\mathcal{R}$. Now, it is obvious that $\mathcal{Q}\subset\mathcal{C}\cap\mathcal{R}$. So, we will only show that $\mathcal{Q}\supset\mathcal{C}\cap\mathcal{R}$.

For any $\mathbf{Q}\in\mathcal{C}\cap\mathcal{R}$, we decompose it as $\mathbf{Q} = \mathrm{vec}(\widetilde{\mathbf{P}})\mathrm{vec}(\widetilde{\mathbf{P}})^T$ since it is a rank-one matrix. From Lemmas 2 and 3, we know that this $\widetilde{\mathbf{P}}$ satisfies three properties: (1) $\widetilde{\mathbf{P}}\geq\mathbf{0}$ or $\widetilde{\mathbf{P}}\leq\mathbf{0}$; (2) There is only one nonzero element in each row; (3) The nonzero elements in each column are equal to each other. Obviously, any $\widetilde{\mathbf{P}}$ with these three properties also satisfies the definition of $\mathbf{P}$ in (2.4), which in turn leads to a matrix $\mathbf{Q} = \mathrm{vec}(\widetilde{\mathbf{P}})\mathrm{vec}(\widetilde{\mathbf{P}})^T\in\mathcal{Q}$. □

Because of this low-rank nature of the optimal $\mathbf{Q}$ solution, we now consider the use of a low-rank SDP solver [4], which is empirically much faster than existing interior-point solvers.

5.2.1 Low-Rank Representation On replacing $\mathbf{Q}$ by a low-rank representation $\mathbf{Y}\mathbf{Y}^T$, where $\mathbf{Y}\in\mathbb{R}^{mc\times r}$ and r is the rank, (5.23) becomes

$$\begin{aligned} &\max_{\mathbf{Y}} && \mathrm{tr}(\mathbf{Y}^T(\mathbf{A}\otimes\mathbf{K})\mathbf{Y}) \\ &\text{s.t.} && \mathrm{odiag}(\mathbf{Y}\mathbf{Y}^T) = \mathbf{I},\ \mathrm{bdiag}(\mathbf{Y}\mathbf{Y}^T)\mathbf{1} = \mathbf{1}, \\ & && \mathbf{Y}\geq\mathbf{0}. \end{aligned} \tag{5.26}$$

Note that this formulation is indeed tighter than (5.23) since $\mathbf{Y}\geq\mathbf{0}$ is a stronger constraint than $\mathbf{Y}\mathbf{Y}^T\geq\mathbf{0}$. Moreover, the following lemmas show that the two constraints in (5.26) can both be rewritten as a set of constraints of the form $\mathrm{tr}(\mathbf{Y}^T\mathbf{A}_i\mathbf{Y}) = b_i$.

LEMMA 4. *The constraint $odiag(\mathbf{Y}\mathbf{Y}^T) = \mathbf{I}$ is equivalent to the set of constraints*

$$tr(\mathbf{Y}^T\mathbf{A}^o_{i,j}\mathbf{Y}) = \delta[i=j],\ 1\leq i,j\leq c, \tag{5.27}$$

where $\mathbf{A}^o_{i,j}\in\mathbb{R}^{mc\times mc}$ and $\delta[i=j] = \begin{cases} 1 & i=j \\ 0 & otherwise \end{cases}$. Each $\mathbf{A}^o_{i,j}$ is partitioned into c^2 $m\times m$ submatrices in the same manner as (5.19). Its (ij)th block is $\mathbf{I}$ and all the other blocks are $\mathbf{0}$, i.e.,

$$\mathbf{A}^o_{i,j} = \begin{bmatrix} & \vdots & \\ \cdots & \mathbf{A}^o_{[ij]} = \mathbf{I} & \cdots \\ & \vdots & \end{bmatrix}. \tag{5.28}$$

Proof. The original constraint can be rewritten as c^2 constraints[3], each of which corresponds to one matrix element $\mathrm{odiag}(\mathbf{Y}\mathbf{Y}^T)_{ij}$. It is then easy to check that $\mathrm{odiag}(\mathbf{Y}\mathbf{Y}^T)_{ij} = \mathrm{tr}(\mathbf{Y}^T\mathbf{A}^o_{i,j}\mathbf{Y})$ when $\mathbf{A}^o_{i,j}$ is defined as in (5.28). □

LEMMA 5. *The constraint $bdiag(\mathbf{Y}\mathbf{Y}^T)\mathbf{1} = \mathbf{1}$ is equivalent to the set of constraints*

$$tr(\mathbf{Y}^T\mathbf{A}^l_i\mathbf{Y}) = 1,\ 1\leq i\leq m, \tag{5.29}$$

where each $\mathbf{A}^l_i\in\mathbb{R}^{mc\times mc}$ is block-diagonal, and every $m\times m$ block on its diagonal has the i-th column equals $\mathbf{1}$ and all the other columns are $\mathbf{0}$.

Proof. As in Lemma 4, we rewrite the original constraint as a set of constraints, each on one of its m elements. It is then easy to check that $(\mathrm{bdiag}(\mathbf{Y}\mathbf{Y}^T)\mathbf{1})_i = \mathrm{tr}(\mathbf{Y}^T\mathbf{A}^l_i\mathbf{Y})$ when $\mathbf{A}^l_i$ is as defined in the lemma. □

According to Proposition 3, one should set r, the rank of $\mathbf{Y}$, to one. However, as the optimization problem is difficult, empirical results show that the use of $r = 1$ can easily get stuck in poor local minima. Hence, we further relax the problem by enlarging the feasible region with the use of a larger r. We bias the optimizer towards a rank-one solution by favoring $\mathbf{Y}\mathbf{Y}^T$

[3]Indeed, as $\mathrm{odiag}(\mathbf{Y}\mathbf{Y}^T)$ is symmetric, only $c(c+1)/2$ constraints are needed. However, for simplicity of exposition, we write it as c^2 constraints here.

that has a large projection $\mathrm{tr}(\mathbf{1}\mathbf{1}^T\mathbf{Y}\mathbf{Y}^T)$ onto this space. The resultant optimization problem thus becomes

$$(5.30)\quad \begin{aligned} \max_{\mathbf{Y}} \quad & \mathrm{tr}(\mathbf{Y}^T(\mathbf{A}\otimes\mathbf{K})\mathbf{Y}) + \gamma \mathrm{tr}(\mathbf{Y}^T\mathbf{1}\mathbf{1}^T\mathbf{Y}) \\ \text{s.t.} \quad & \mathrm{tr}(\mathbf{Y}^T\mathbf{A}_i\mathbf{Y}) = b_i \ \text{ in (5.27) and (5.29)}, \\ & \mathbf{Y} \geq \mathbf{0}, \end{aligned}$$

and $\gamma > 0$ balances the contributions from the two terms. Preliminary experiments indicate that the performance is not sensitive to γ and so it is always fixed at 0.1. Moreover, as will be seen in Section 6, r only needs to be a small number for good clustering performance. This also agrees with the fact that ideally r should be 1. Hence, in the experiments, r is always fixed to 10 for all the data sets.

5.2.2 Computational Cost The main computational cost is on computing the Lagrangian and its gradient, in which their most expensive step is on the computing of $\mathrm{tr}(\mathbf{Y}^T(\mathbf{A}\otimes\mathbf{K})\mathbf{Y})$ and $\mathrm{tr}(\mathbf{Y}^T\mathbf{1}\mathbf{1}^T\mathbf{Y})$. This can be done in $O(rzc^2)$ time, where z is the number of nonzero entries in matrix $\mathbf{K}$.

5.3 Recovering the Cluster Assignment After obtaining $\mathbf{Y}$ from (5.30), the last step is to recover the binary partition matrix $\mathbf{\Pi}$. Note that unlike traditional clustering algorithms, CLUHSIC has to ensure that the objects are in the correct clusters and also that the extracted clusters preserve the structure in $\mathbf{A}$. As an illustrating example, suppose that in Figure 2(a), one has to cluster the six images into three clusters along a chain. We assume that each cluster should contain 2 images. For the four possible $\mathbf{\Pi}$ solutions below:

$$\text{(a)} \begin{bmatrix} 1&0&0\\1&0&0\\0&1&0\\0&1&0\\0&0&1\\0&0&1 \end{bmatrix}; \quad \text{(b)} \begin{bmatrix} 0&0&1\\0&0&1\\0&1&0\\0&1&0\\1&0&0\\1&0&0 \end{bmatrix};$$

$$\text{(c)} \begin{bmatrix} 1&0&0\\1&0&0\\0&1&0\\0&0&1\\0&0&1\\0&0&1 \end{bmatrix}; \quad \text{(d)} \begin{bmatrix} 1&0&0\\1&0&0\\0&0&1\\0&0&1\\0&1&0\\0&1&0 \end{bmatrix}.$$

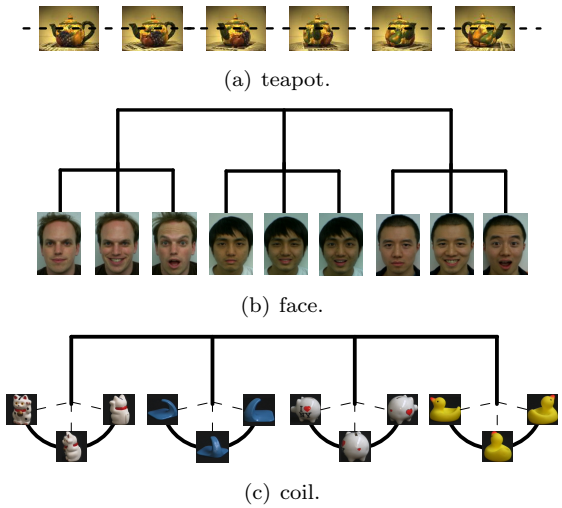
Solutions (a) and (b) are both correct (as both successfully capture the adjacent relationships between clusters), even though they are in different orders. Solution (c) is incorrect because the third and fourth elements are not in the same cluster. Solution (d) is also incorrect because the clusters are in the wrong order, which violates the structural relationships of the clusters in a chain. On the other hand, traditional clustering methods disregard the structural relationships and will consider (a), (b), (d) as correct.

To recover $\mathbf{\Pi}$, we first find the rank-one matrix that is nearest to $\mathbf{Y}\mathbf{Y}^T$ in terms of Euclidean distance. This is simply $\lambda\boldsymbol{\pi}\boldsymbol{\pi}'$, where λ is the largest eigenvalue of $\mathbf{Y}\mathbf{Y}^T$ and $\boldsymbol{\pi}$ is the corresponding eigenvector. We then resize $\sqrt{\lambda}\boldsymbol{\pi} \in \mathbb{R}^{mc}$ to a matrix $\mathbf{\Pi} \in \mathbb{R}^{m\times c}$. This is transformed to a non-negative orthonormal matrix by alternately projecting $\mathbf{\Pi}$ onto

1. the Stiefel manifold: Let the polar decomposition of $\mathbf{\Pi}$ be $\mathbf{U}\mathbf{H}$, where $\mathbf{U} \in \mathbb{R}^{m\times c}$ has orthonormal columns and $\mathbf{H} \in \mathbb{R}^{c\times c}$ is a Hermitian positive semi-definite matrix. Then the nearest matrix with orthonormal columns is $\mathbf{U}$ [10]; and

2. the set of nonnegative matrices, by simply setting the negative matrix elements to zeros.

This is repeated until convergence. Alternating projection in Stiefel manifold has been highly successful in practice (e.g., [25, 15]. In the experiments, we observe that it typically converges in fewer than 10 iterations. Finally, each pattern (which corresponds to a row in $\mathbf{\Pi}$) is then assigned to the cluster (a column in $\mathbf{\Pi}$) which has the largest value on that row.



(a) teapot.

(b) face.

(c) coil.

Figure 2: Data sets used in the clustering experiment.

6 Experiments

In Section 6.1, we first report experiments on clustering. Similar to DIFFRAC, CLUHSIC can also be used for semi-supervised learning by enforcing the similarity/dissimilarity information as constraints on the partition matrix. Hence, in Section 6.2, we will apply CLUHSIC on the task of semi-supervised protein subcellular localization.

6.1 Clustering

6.1.1 Data Sets The first data set is the teapot data[4] used in [24]. We consider a chain structure by using 100 images with rotation angles from 1° to 180°. They are grouped into 5 well-separated clusters, each having 20 images (Figure 2(a)). The Gaussian kernel is used on the input, and the kernel matrix

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix}$$

as suggested for chains in [24], is used on the output.

The second data set is the facial expression data also used in [24]. It consists of 185 images of three types of facial expressions from three subjects (Figure 2(b)). The Gaussian kernel is used on the input, and the kernel matrix

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}$$

as suggested for hierarchies in [24], is used on the output.

The third data set is the COIL-100 data[5], which consists of images of objects taken at different poses. Here, we use 4 objects (number 14, 36, 48, 74). For each object, we select 24 images from 1° to 180°, which are then grouped in 3 well-separated clusters. The Gaussian kernel is used on the input. As for the output structure, its upper level is tree-structured with 4 nodes (one for each object), while its lower level is chain-structured (Figure 2(c)). Hence, we use the kernel matrix

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}.$$

6.1.2 Setup We compare the following optimization approaches of CLUHSIC: (1) local greedy search (greedy) [24]; (2) NMF [23]; and (3) the proposed low-rank SDP relaxation (SDPLR), with $r = 10$. Spectral relaxation will not be compared as it yields the same solution as the spectral relaxation of k-means. We also compare with other well-known clustering methods, including (1) k-means (KM); (2) kernel k-means (KKM); (3) normalized cut[6] (NCut); and (4) DIFFRAC [7].

The following measures are used in performance evaluation: (1) accuracy (acc); (2) normalized mutual information (NMI), whose value is always between zero and one; (3) chain loss (for chain structures), which is the difference in relative positions from the ground truth; and (4) tree loss (for hierarchical structures), which is the height of the first common ancestor of the true and predicted cluster labels in the hierarchy. These loss functions have been commonly used in structured prediction [26]. For both the accuracy and NMI, a higher value indicates better clustering quality; whereas for the chain/tree loss, the lower the better. Moreover, to reduce statistical variability, all the results reported here are based on averages over 50 repetitions.

Recall that, unlike CLUHSIC, the clustering algorithms KM, KKM, NCut and DIFFRAC are "structure-less" in that they do not indicate how the obtained clusters are ordered in the structure (Section 5.3). For these algorithms, we follow the common practice of permuting the obtained clusters to best match the ground truth [28]. Hence, these "structure-less" algorithms can have an unfair advantage over CLUHSIC by trying a lot more permutations than those allowed in the CLUHSIC solution. Thus, the results reported for these "structure-less" methods should be for reference only. For the face data set, this unfair advantage can be reduced by implementing a variant of the "structure-less" algorithm that observes the hierarchical structure: We first use the structure-less algorithm to cluster the data set into 3 clusters. Then, we further divide each cluster into 3 sub-clusters.

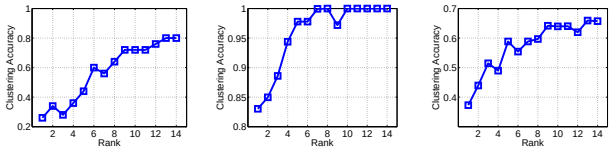
6.1.3 Results Clustering results are shown in Table 1. As can be seen, SDPLR significantly outperforms the other optimization methods for CLUHSIC. The improvements are statistically significant at a 0.01 level of significance. It even beats the "structure-less" methods on face, even though these methods have an unfair advantage over CLUHSIC. Moreover, note that some methods yield good NMI but poor accuracy and loss value. This is because NMI only measures the mutual information between the set of obtained clusters and the set of true clusters, but does not consider the structural relationships among clusters. So, even when the clusters are wrongly placed in the structure, the NMI value can still be high (e.g., solution (d) in Section 5.3 is one such example).

We also study the use of different ranks ($r = \text{rank}(\mathbf{Y})$) on the accuracy of the SDP relaxation. As can be seen from Figure 3, although ideally $\mathbf{Y}$ should be rank one (Proposition 3), using $r = 1$ will easily get stuck in a poor local minimum. The clustering accuracy gradually improves with a larger r, which then flattens off at around $r = 10$. On the other hand, as the number of optimization variables increases with r, the CPU time also increases. Empirically, as shown in Figure 4, the time increases linearly with r.

[4] http://www.cs.usyd.edu.au/∼lesong/software/cluhsic.zip
[5] http://www1.cs.columbia.edu/CAVE/databases/
[6] http://www.cis.upenn.edu/∼jshi/software/
[7] http://www.di.ens.fr/∼fbach/diffrac/

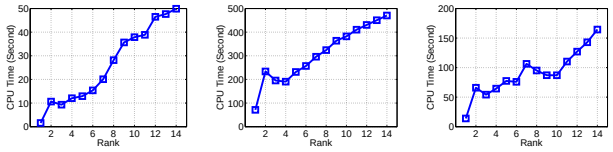
Table 1: Clustering performance on the data sets. Note that KM, KKM, NCut and DIFFRAC cannot preserve the output structure so their results are for reference only. For details, please see the text.

	teapot			face			coil		
method	acc (%)	NMI (%)	loss	acc (%)	NMI (%)	loss	acc (%)	NMI (%)	loss
KM	86.78	93.56	1.27	57.94	81.05	0.95	67.29	87.02	1.05
KKM	89.24	93.62	1.26	72.67	88.10	0.94	74.35	87.35	1.15
NCut	97.30	97.60	1.00	85.11	91.79	0.80	84.88	94.00	1.03
DIFFRAC	100.00	100.00	0.00	95.68	95.18	0.43	91.67	94.17	1.00
(CLUHSIC)									
greedy	32.46	97.85	1.23	65.26	92.85	0.56	38.85	91.16	0.80
NMF	39.54	96.56	1.17	80.89	96.52	0.34	43.44	91.88	0.67
SDPLR	**72.00**	**100.00**	**0.48**	**100.00**	**100.00**	**0.00**	**56.79**	**94.73**	**0.26**



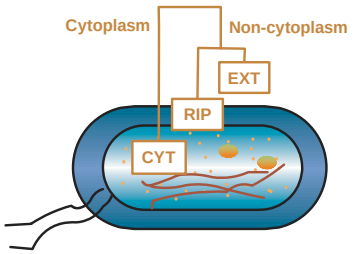
(a) teapot. (b) face. (c) coil.

Figure 3: Effect of the rank r on accuracy.

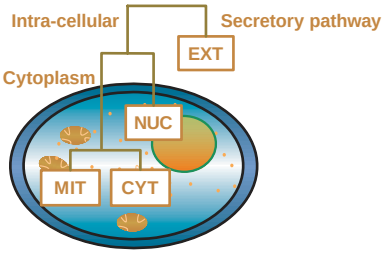


(a) teapot. (b) face. (c) coil.

Figure 4: Effect of the rank r on time.



(a) Prokaryotic cell.



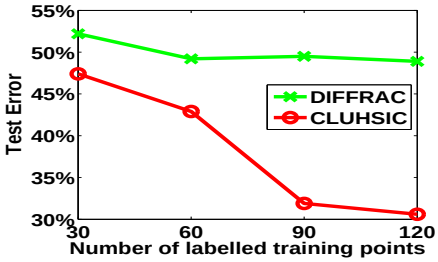
(b) Eukaryotic cell.

Figure 5: Biological sorting structures. Here, EXT stands for extra-cellular; NUC for nucleus; CYT for cytosol; MIT for mitochondria; and RIP for periplasm.

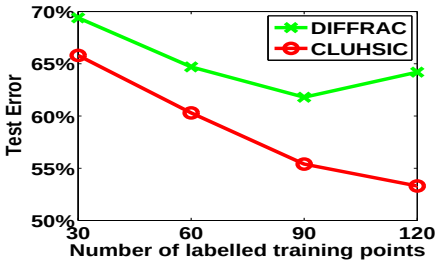
6.2 Semi-Supervised Protein Subcellular Localization A major challenge in modern biology is to advance the knowledge of the expression, regulation, and function of the entire set of proteins encoded by an organism, i.e., its proteome. This information will be invaluable for understanding how complex biological processes occur at a molecular level, how they differ in various cell types, and how they are altered in disease states. Protein subcellular localization is one of the key steps to determining its biological function, as proteins must be localized in the same subcellular compartment in order to cooperate towards a common function.

In the past, researchers only used the amino acid composition to predict subcellular localization, without any explicit experimental knowledge of the protein under investigation [6]. Furthermore, these methods assume that all the subcellular locations are equivalent. However, in reality, some locations are more similar to each other than others. For example, endoplasmic reticulum is closer to extra-cellular than to nuclear due to proximity in the space of the biological sorting machinery.

Recently, Nair and Rost [17] showed that structural and evolutionary information can significantly improve prediction accuracy. Here, we demonstrate that CLUHSIC can also benefit from the use of such structural information. Figure 5 shows the biological sorting structures of the prokaryotic and eukaryotic cells, respectively [17]. The hierarchical architecture has been designed to mimic the biological protein sorting mechanism as closely as possible. The branches of the tree represent intermediate stages in the sorting machinery while the nodes represent the decision points in the sorting machinery.

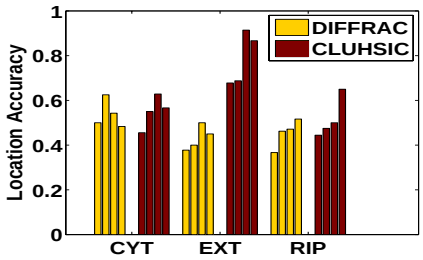


(a) prokaryotic.

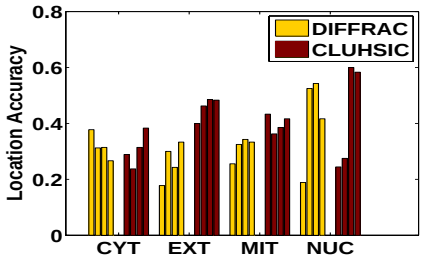


(b) eukaryotic.

Figure 6: Results on semi-supervised protein subcellular localization.



(a) prokaryotic.



(b) eukaryotic.

Figure 7: Location accuracies (recall) at various subcellular locations.

6.2.1 Setup We use a popular data set from [20], and randomly select 100 protein sequences from each subcellular location. The amino acid composition, which contains the percentage of each type of amino acid in the protein, is used to construct a 20-dimensional feature for each protein sequence [11, 18]. We follow a transductive learning setting, and the total number of labeled samples is varied from 30 to 120. Results are averaged over five repetitions.

6.2.2 Results As can be seen from Figure 6, CLUHSIC yields significant improvement over the "structureless" DIFFRAC in terms of the testing error. Figure 7 compares the location accuracies (recall) at various subcellular locations. Within each group, the 4 bars are arranged in increasing number of labeled samples. Note that the most significant improvement is on the EXT (extra-cellular) location, which is regarded as structurally very distinct from the others, as shown in Figure 5. This confirms our proposal that strong structural information can benefit the learning task.

7 Conclusion

In this paper, we first unify the recent discriminative view of clustering into the CLUHSIC family. We then study the integer program in the CLUHSIC algorithm and propose two relaxations that are more disciplined than the greedy local optimization heuristic used in [24]. The first one is spectral relaxation which leads to the same solution as the spectral relaxations for k-means. The second one is SDP relaxation, which can naturally be solved by an efficient low-rank SDP solver. Experimental results show that the SDP relaxation obtains significantly improved clustering performance than the existing optimization methods for CLUHSIC. Moreover, it can also be used in semi-supervised learning. Experiments on protein subcellular localization also confirm that incorporating structural and evolutionary information can significantly benefit the learning task.

Since CLUHSIC requires solving a difficult integer program, there is still much room for improvement. The local greedy optimization heuristic used in [24] can only be used on small data sets. The proposed approach shows encouraging results on mid-sized data sets. However, the computational cost will still be high on large-scale data. This will be further investigated in the future.

Acknowledgments

This research was partially supported by the Research Grants Council of the Hong Kong Special Administrative Region under grant 614508, the Natural Science Foundation of China under grants 60773090 and 90820018, and the National Basic Research Program of China under grant 2009CB320901.

References

[1] F. Bach and Z. Harchaoui. DIFFRAC: A discriminative and flexible framework for clustering. In *Advances in Neural Information Processing Systems 20*, pages 49–56. MIT Press, 2008.

[2] M. Blaschko and A. Gretton. Learning taxonomies by dependence maximization. In *Advances in Neural Information Processing Systems 21*, pages 153–160. MIT Press, 2009.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[4] S. Burer and R. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[5] T. De Bie and N. Cristianini. Fast SDP relaxations of graph cut clustering, transduction, and other combinatorial problems. *Journal of Machine Learning Research*, 7:1409–1436, 2006.

[6] P. Donnes and A. Hoglund. Predicting protein subcellular localization: Past, present, and future. *Genomics Proteomics Bioinformatics*, 2:209–215, 2004.

[7] A. M. Frieze and M. Jerrum. Improved approximation algorithms for max k-cut and max bisection. In *Integer Programming and Combinatorial Optimization*, volume 920, pages 1–13, 1995.

[8] K. Fukumizu, F. Bach, and M. Jordan. Dimensionality reduction for supervised learning with reproducing kernel Hilbert spaces. *Journal of Machine Learning Research*, 5:73–99, 2004.

[9] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf. Measuring statistical dependence with Hilbert-Schmidt norms. In *Proceedings of the Sixteenth International Conference on Algorithmic Learning Theory*, pages 63–77, Singapore, October 2005.

[10] N. Higham. Matrix nearest problems and applications. In M. Gover and S. Barnett, editors, *Applications of Matrix Theory*, pages 1–27. Oxford University Press, 1989.

[11] S. Hua and Z. Sun. Support vector machine approach for protein subcellular localization prediction. *Bioinformatics*, 17:721–728, 2001.

[12] S. Khot and A. Naor. Approximate kernel clustering. In *Proceedings of the Forty-Ninth Symposium on Foundations of Computer Science*, pages 561–570, Philadelphia, PA, USA, October 2008.

[13] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: A kernel approach. *Machine Learning*, 74(1):1–22, 2009.

[14] B. Kulis, A. Surendran, and J. Platt. Fast low-rank semidefinite programming for embedding and clustering. In *Proceedings of the Eleventh International Workshop on Artificial Intelligence and Statistics*, pages 235–242, San Juan, Puerto Rico, March 2007.

[15] A. Lewis and J. Malick. Alternating projections on manifolds. *Mathematics Of Operations Research*, 33(1):216–234, 2008.

[16] H. Lütkepohl. *Handbook of Matrices*. John Wiley and Sons, 1996.

[17] R. Nair and B. Rost. Mimicking cellular sorting improves prediction of subcellular localization. *Journal of Molecular Biology*, 348:85–100, 2005.

[18] K. J. Park and M. Kanehisa. Prediction of protein subcellular locations by support vector machines using compositions of amino acids and amino acid pairs. *Bioinformatics*, 19:1656–1663, 2003.

[19] N. Quadrianto, L. Song, and A. Smola. Kernelized sorting. In *Advances in Neural Information Processing Systems 21*, pages 1289–1296. MIT Press, 2009.

[20] A. Reinhardt and T. Hubbard. Using neural networks for prediction of subcellular location of proteins. *Nucleic Acids Research*, 26:2230–2236, 1998.

[21] F. Rendl and H. Wolkowicz. Applications of parametric programming and eigenvalue maximization to the quadratic assignment problem. *Mathematical Programming*, 53:63–78, 1992.

[22] B. Shaw and T. Jebara. Structure preserving embedding. In *Proceedings of the Twenty-Sixth International Conference on Machine Learning*, Montreal, Quebec, Canada, June 2009.

[23] L. Song. *Learning via Hilbert Space Embedding of Distributions*. PhD thesis, University of Sydney, 2008. http://www.cs.cmu.edu/∼lesong/papers/lesong_thesis.pdf.

[24] L. Song, A. Smola, A. Gretton, and K. Borgwardt. A dependence maximization view of clustering. In *Proceedings of the Twenty-Fourth International Conference on Machine Learning*, pages 815–822, Corvallis, Oregon, USA, June 2007.

[25] J. Tropp, I. Dhillon, R. Heath, and T. Strohmer. Designing structured tight frames via an alternating projection method. *IEEE Transactions on Information Theory*, 51(1):188–209, Jan. 2005.

[26] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 6:1453–1484, 2005.

[27] J. Ye, Z. Zhao, and M. Wu. Discriminative k-means for clustering. In *Advances in Neural Information Processing Systems 20*, pages 1649–1656. MIT Press, 2008.

[28] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Spectral relaxation for k-means clustering. In *Advances in Neural Information Processing Systems 14*, pages 1057–1064. MIT Press, 2002.

[29] Z. Zhour and A. Kilicman. The representation and approximation for the weighted Minkowski inverse in Minkowski space. *Mathematical and Computer Modelling*, 47:363–371, 2008.