Local and Global Optimization Techniques in Graph-based Clustering

Daiki Ikami Toshihiko Yamasaki Kiyoharu Aizawa
The University of Tokyo, Japan
{ikami, yamasaki, aizawa}@hal.t.u-tokyo.ac.jp

Abstract

The goal of graph-based clustering is to divide a dataset into disjoint subsets with members similar to each other from an affinity (similarity) matrix between data. The most popular method of solving graph-based clustering is spectral clustering. However, spectral clustering has drawbacks. Spectral clustering can only be applied to macro-average-based cost functions, which tend to generate undesirable small clusters. This study first introduces a novel cost function based on micro-average. We propose a local optimization method, which is widely applicable to graph-based clustering cost functions. We also propose an initial-guess-free algorithm to avoid its initialization dependency. Moreover, we present two global optimization techniques. The experimental results exhibit significant clustering performances from our proposed methods, including 100% clustering accuracy in the COIL-20 dataset.

1. Introduction

Clustering is an important technique with many applications in computer vision and machine learning, such as image segmentation [19] and motion segmentation [6]. This study focuses on graph-based clustering, which is the optimization problem maximizing the following objective function:

$$E(\mathbf{Z}) = \sum_{j=1}^{k} \frac{\mathbf{z}_j^\top \mathbf{A} \mathbf{z}_j}{f_j(\mathbf{Z})}, \tag{1}$$

where $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the affinity matrix of n data points, and $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_k] \in \mathcal{Z}$,$\mathcal{Z} = \{\mathbf{Z} \in \{0,1\}^{n \times k} | \mathbf{Z}\mathbf{1} = \mathbf{1}\}$ is an assignment matrix, in which k is the number of clusters and $\mathbf{1}$ is the all-one vector. $f(\mathbf{Z})$ denotes the functions defined by each cost function described in Section 1.1. In contrast to representative-based clustering (e.g., k-means clustering and Gaussian mixture model clustering), graph-based clustering enables clustering on associations between data, such as by the k-nearest neighbor (k-NN) graph for $\mathbf{A}$, and shows a good clustering performance.

Clustering performances are mainly determined by three primary factors in graph-based clustering: construction of the affinity matrix $\mathbf{A}$, the objective function, and optimization method. We briefly review the related works regarding these three subjects.

Notation: In this study, bold uppercase letters (*e.g.* $\mathbf{X}$), bold lowercase letters (*e.g.* $\mathbf{x}$), and plain lowercase letters (*e.g.* x) denote matrices, column vectors, and scalars, respectively. $\mathbf{0}$, $\mathbf{1}$, and $\mathbf{I}_n$ are the all-zero vector, all-one vector, and $n \times n$ identity matrix, respectively. $\mathbf{X}[S, T]$ is a submatrix in $\mathbf{X}$ with row subset S and column subset T.

1.1. Related Work

Affinity matrix: The choice of an affinity matrix is important in graph-based clustering. The most simple approaches are distance-based, such as k-NN graph or the Gaussian kernel of Euclidean distances [24, 26]. Other methods are learning an affinity matrix from the data, such as based on sparse [6, 25] or low-rank [13, 23] self-representation. In these methods, an affinity matrix is represented by a coefficient matrix of linear combinations of data. Some recent studies integrated the two problems of learning affinity and clustering into one optimization problem [17, 12].

Objective function: If $f_j(\mathbf{Z})$ is a constant and $a_{ij} \geq 0$ for all i, j, we can easily find that the global optimal solution is obtained by assigning all data points to a single cluster. $f_j(\mathbf{Z})$ is designed to avoid this trivial solution and encourage the cluster size not to be too large. For example, $f_j(\mathbf{Z}) = \mathbf{1}^\top \mathbf{z}_j$ is used in a macro-average association, while $f_j(\mathbf{Z}) = \mathbf{1}^\top \mathbf{A} \mathbf{z}_j$ is used in a normalized cut [19]. Please refer to [20, 5] for more details. These objective functions are based on macro average (i.e., lth cluster's energy $\mathbf{z}_j^\top \mathbf{A} \mathbf{z}_j$ is divided by the lth cluster size). Optimizing such macro-average-based cost functions tends to yield undesirable small clusters.

Accordingly, Chen *et al*. proposed a objective function to balance the cluster sizes [3], which in this study we call balanced association (BA), to overcome undesirable small clusters problem. The objective function of BA is

$$\sum_{j=1}^{k} \mathbf{z}_j^\top \mathbf{A} \mathbf{z}_j + \lambda \|\mathbf{Z}^\top \mathbf{Z}\|_F^2 = \sum_{j=1}^{k} \mathbf{z}_j^\top \left(\mathbf{A} - \lambda \mathbf{1}\mathbf{1}^\top\right) \mathbf{z}_j. \tag{2}$$

Table 1: Cost functions and their optimization methods. DLO and GIA are our proposed methods.

Name	$E(\mathbf{Z})$	Opt. method
macro-AA	$\sum_{j=1}^{k} \frac{\mathbf{z}_j^\top \mathbf{A}\mathbf{z}_j}{\mathbf{1}^\top \mathbf{z}_j}$	BO, SC, DLO
NC	$\sum_{j=1}^{k} \frac{\mathbf{z}_j^\top \mathbf{A}\mathbf{z}_j}{\mathbf{1}^\top \mathbf{A}\mathbf{z}_j}$	BO, SC, DLO
BA	$\sum_{j=1}^{k} \mathbf{z}_j^\top \left(\mathbf{A} - \lambda \mathbf{1}\mathbf{1}^\top\right) \mathbf{z}_j$	ADMM, DLO
micro-AA (proposed)	$\frac{1}{\sum_{j=1}^{k} (\mathbf{z}_j^\top \mathbf{1})^p} \sum_{j=1}^{k} \mathbf{z}_j^\top \mathbf{A}\mathbf{z}_j$	DLO, GIA

Parameter λ controls the cluster sizes. The cluster sizes are more uniform with the larger λ. Note that we can transform Eq. (2) into Eq. (1) by $\mathbf{A}' = \mathbf{A} - \lambda \mathbf{1}\mathbf{1}^\top$ and $f_j(\mathbf{Z}) = 1$ for all l. Chen *et al*. also proposed a self-tuning method of λ. Table 1 summarizes the objective functions.

Optimization methods: Eq. (1) is an NP-hard combinational optimization problem; thus, global optimization is difficult. For macro-AA and NC, spectral clustering (SC) [19, 24] that uses spectral relaxation to solve Eq. (1) is the most widely used method to solve Eq. (1). For example, we can rewrite the problem of the macro-AA cost as follows:

$$E(\mathbf{Z}) = \mathrm{tr}(\mathbf{Y}^\top \mathbf{A}\mathbf{Y}), \text{ where } \mathbf{Y} = \left[\frac{\mathbf{z}_1}{\|\mathbf{z}_1\|_2}, \cdots, \frac{\mathbf{z}_k}{\|\mathbf{z}_k\|_2}\right]. \quad (3)$$

We consider a relaxed constraint $\mathbf{Z} \in \mathcal{Z}', \mathcal{Z}' = \{\mathbf{Z} \in \mathbb{R}^{n \times k} | \mathbf{Z}^\top \mathbf{Z} = \mathbf{I}_k\}$ instead of the constraint $\mathbf{Z} \in \mathcal{Z}$. We can then solve Eq. (3) on $\mathbf{Z}$ by k large eigenvectors of $\mathbf{A}$. k-means clustering is performed on row-vectors of relaxed $\mathbf{Z}$ to satisfy $\mathbf{Z} \in \mathcal{Z}$. SC is widely used because of its easiness and effectiveness. However, SC has some drawbacks. First, SC does an approximation, which causes performance degradation. Second, SC can only be applied to macro-average-based cost functions. Third, SC usually has a computational complexity of $O(n^3)$ because of eigen decomposition.

The other method is the bound optimization (BO) method [4, 5, 20], which is also known as the kernel k-means algorithm. BO repeats the construction of an upper bound function and minimization of the upper bound function:

$$\mathbf{Z}^{(t+1)} = \arg\max_{\mathbf{Z}} E_t(\mathbf{Z})$$

$$E_t(\mathbf{Z}) = \sum_{j=1}^{k} \mathbf{z}_j^\top \left(\frac{\left(\mathbf{z}_j^{(t)}\right)^\top \mathbf{K}\mathbf{z}_j^{(t)}}{\left(\mathbf{w}^\top \mathbf{z}_j^{(t)}\right)^2} \mathbf{w} - \frac{2\mathbf{K}\mathbf{z}_j^{(t)}}{\mathbf{w}^\top \mathbf{z}_j^{(t)}} \right), \quad (4)$$

where $\mathbf{K} = \mathbf{A} + \delta \mathbf{I}_n, \mathbf{w} = \mathbf{1}$ in macro-AA and $\mathbf{K} = \mathbf{A} + \delta\, \mathrm{diag}(\mathbf{S}\mathbf{1}), \mathbf{w} = \mathbf{A}\mathbf{1}$ in NC. To satisfy that Eq. (4) is strictly a bound function, sufficiently large δ is required such that $\mathbf{K}$ is a positive semi-definite matrix. However, a solution often does not move in a strict bound function, as described in Section 5. Dhillon *et al*. proposed a two-step process for clustering: first, perform SC for an initial solution, then apply BO [4].

For BA, Chen *et al*. proposed the alternating direction method of multipliers (ADMM) to solve Eq. (2) [3]. Eq. (2) can be transformed into the following equivalent problem:

$$\min_{\mathbf{Z} \in \mathcal{Z}, \mathbf{Y} = \mathbf{Z}} \mathrm{tr}(\mathbf{Z}^\top \hat{\mathbf{A}}\mathbf{Y}) \quad (5)$$

where $\hat{\mathbf{A}} = \lambda \mathbf{1}\mathbf{1}^\top - \mathbf{A}$. The augmented Lagrangian function of Eq. (2) is presented as follows:

$$L(\mathbf{Z}, \mathbf{Y}, \mathbf{L}) = \mathrm{tr}(\mathbf{Z}^\top \hat{\mathbf{A}}\mathbf{Y}) + \langle \mathbf{L}, \mathbf{Z} - \mathbf{Y} \rangle + \frac{\mu}{2} \|\mathbf{Z} - \mathbf{Y}\|_F^2 \quad (6)$$

where $\mathbf{L}$ is the Lagrange multipliers, and $\langle \cdot, \cdot \rangle$ is the sum of the element product. The optimization is performed by an alternating update of $\mathbf{Z}$, $\mathbf{Y}$, and $\mathbf{L}$ with increasing μ.

1.2. Contributions

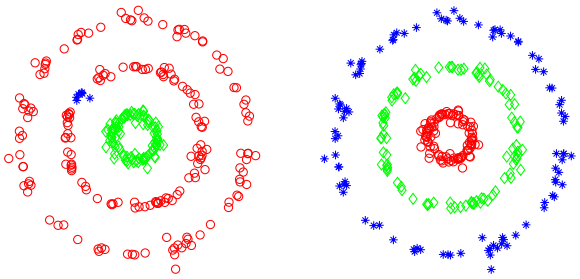
The existing optimization methods are only applicable to limited cost functions and do not achieve global optimization. We propose herein a local optimization method, called the direct local optimization (DLO), for a general graph-clustering cost function Eq. (1) without approximation. We also propose techniques to achieve global optimization and summarize our contributions as follows:

1. We introduce a novel cost function, called the micro average association (micro-AA). In contrast to macro-average-based costs, micro-AA can prevent generating undesirable small clusters.
2. We propose DLO, which is a local optimization method for Eq. (1) without approximation. DLO is guaranteed to increase the objective value, and can be applied to wide graph-based clustering cost functions as in Table 1. DLO requires a good initial solution to achieve good clustering. We also propose an initial-guess-free algorithm to solve this problem.
3. Our proposed methods may be trapped into a bad local optimum. To overcome this problem, We introduce two global optimization techniques for graph clustering.
4. We evaluate the performances with different objective functions, initializations, and local optimization methods with various datasets. Our experiments demonstrate that the proposed optimization method achieves a significant improvement in most datasets (i.e., 100% clustering accuracy in the COIL-20 dataset).

2. Micro-average association

As described in Section 1.1, macro-average-based cost functions, such as NC, tend to overestimate small clusters as in Fig. 1(a). In this section, we propose an objective function based on micro average association:

$$E(\mathbf{Z}) = \frac{1}{\sum_{j=1}^{k} \left(\mathbf{z}_j^\top \mathbf{1}\right)^p} \sum_{j=1}^{k} \mathbf{z}_j^\top \mathbf{A}\mathbf{z}_j, \quad (7)$$



(a) Optimal clustering based on NC (b) Optimal clustering based on micro-AA

Fig 1: Different clustering results optimized by NC and micro-AA. We construct the affinity matrix by k-NN graph with $k = 5$. The objective values of Fig. 1(a) and Fig. 1(b) are 3.000 and 2.996 in NC and 0.1386 and 0.1664 in micro-AA, respectively.

This is equivalent to Eq. (1) by $f_k(\mathbf{Z}) = \sum_{j=1}^{K} \left(\mathbf{z}_j^\top \mathbf{1}\right)^p$ for all $k = 1, \cdots, K$. Parameter p controls the cluster sizes. $\sum_{j=1}^{k} \left(\mathbf{z}_j^\top \mathbf{1}\right)^p$ is equal to constant value n if $p = 1$, and we obtain the clustering result that all data belong to a single cluster. A large p encourages the cluster sizes to be uniform. Especially all clusters have the same size at $p \to \infty$. We call this cost function Eq. (7) the micro average association (micro-AA) cost.

The main advantage of micro-AA is the avoidance of undesirable small clustering results (Fig. 1(b)) as with BA. Compared with BA, micro-AA experimentally achieves better clustering performances, as described in Section 5. However, micro-AA is an NP hard combinational optimization as with other graph-based clustering cost functions. Moreover, the existing techniques to solve graph-based clustering, such as spectral relaxation, cannot be applied to micro-AA. In this paper we propose an optimization method that can solve general graph-based clustering problems, including micro-AA.

3. Optimization methods for graph clustering

We first propose the direct local optimization (DLO) method, which is an iterative optimization method guaranteed to increase the objective value Eq. (1). DLO requires a good initial solution to achieve good clustering. We also propose the greedy incremental algorithm (GIA), which does not require an initial solution. GIA starts from all empty clusters and repeats assigning an element to a cluster.

Without loss of generality, we assume the symmetric affinity matrix $\mathbf{A} = \mathbf{A}^\top$ because Eq. (1) with a non-symmetric matrix $\mathbf{A}$ is equivalent to that with $(\mathbf{A}+\mathbf{A}^\top)/2$. We denote a column vector and a row vector of $\mathbf{Z}$ by $\mathbf{z}_{\bullet,i}$ and $\mathbf{z}_{i,\bullet}$, respectively. That is to say, $\mathbf{Z} = [\mathbf{z}_{\bullet,1}, \cdots, \mathbf{z}_{\bullet,k}] = [\mathbf{z}_{1,\bullet}, \cdots, \mathbf{z}_{n,\bullet}]^\top$.

Algorithm 1 Direct Local optimization

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, initial solution $\mathbf{Z}^{(1)}$
Initialize: $t = 1$
1: Compute $F^{(1)}(l, c)$ for all l and c
2: **while** not converged **do**
3: $l^*, c^* = \arg\max_{l,c} F^{(t)}(l, c)$
4: Update $\mathbf{Z}^{(t+1)}$ by Eq. (9)
5: Update $F^{(t+1)}$ by Eq. (12)
6: $t \leftarrow t + 1$
7: **end while**
Output: $\mathbf{Z}$

3.1. Direct local optimization (DLO)

We propose the iterative local optimization method for Eq. (1), which is guaranteed to increase the objective value at each iteration. Our algorithm is a kind of coordinate descent method: given a solution $\mathbf{Z}^{(t)}$ at iteration t, we optimize a single row vector $\mathbf{z}_{l,\bullet}$ on the other fixed $\mathbf{z}_{i,\bullet}$ for $\{i | 1 \le i \le n, i \ne l\}$:

$$\mathbf{z}^* = \arg\max_{\mathbf{z}} E(\mathbf{z}_{1,\bullet}^{(t)}, \cdots, \mathbf{z}_{l-1,\bullet}^{(t)}, \mathbf{z}, \mathbf{z}_{l+1,\bullet}^{(t)}, \cdots, \mathbf{z}_{n,\bullet}^{(t)}) \quad (8)$$

We then update the solution by

$$\mathbf{z}_{i,\bullet}^{(t+1)} = \begin{cases} \mathbf{z}_{i,\bullet}^{(t)} & (i \ne l) \\ \mathbf{z}^* & (i = l) \end{cases} \quad (9)$$

Eq. (8) can be solved by computing all possible k objective values. Given the current solution $\mathbf{Z}^{(t)}$, we define $F^{(t)}(l, c)$ as follows:

$$F^{(t)}(l, c) = E(\mathbf{Z}'), \text{ where } z'_{ij} = \begin{cases} z_{ij}^{(t)} & \text{if } i \ne l \\ 1 & \text{if } i = l \text{ and } j = c \\ 0 & \text{otherwise} \end{cases}. \quad (10)$$

$F^{(t)}(l, c)$ is the energy of changing lth element to belong to cluster c. Given l, we can compute $F^{(t)}(l, c)$ for all c. Therefore, we can solve Eq. (8) as follows:

$$c^* = \arg\max_{c} F^{(t)}(l, c), \quad \mathbf{z}^* = \text{onehot}(c^*), \quad (11)$$

where $\text{onehot}(c)$ is a one-hot vector, whose cth element is 1, and the others are 0. The update rule of Eq. (9) and Eq. (11) always increases the objective value of Eq. (1) at each iteration.

We summarize the algorithm in Algorithm 1 and describe how to choose coordinate l and compute $F^{(t)}(l, c)$ in the followings.

Coordinate selection: Coordinate selection methods are categorized into three types in coordinate descent methods [16, 18]. The randomized coordinate descent randomly chooses l from the set of $\{1, \cdots, n\}$. The cyclic coordinate descent chooses l in a cyclic order. Meanwhile, the greedy coordinate descent selects the best coordinate to improve

the objective function value.

The greedy coordinate descent is generally time consuming because of the overhead computation of selecting the best coordinate, but it achieves better performances than the random coordinate descent in problems with a small overhead computation for choosing the best coordinate. We adopted the greedy coordinate selection because we can efficiently choose the best coordinate as described below:

Efficient computation for $F^{(t)}(l,c)$: Although a naive computation for $F^{(t)}(l,c)$ is time consuming, we can efficiently compute $F^{(t)}(l,c)$ by computation results at iteration $t-1$. Let $\hat{c}$ be the cluster to which the lth element belongs at iteration t. We can rewrite $F^{(t)}(l,c)$ as follows:

$$F^{(t)}(l,c) = \sum_{j=1}^{k}\left(\frac{\left(\mathbf{z}_{\bullet,l}^{(t)}\right)^{\top}\mathbf{A}\mathbf{z}_{\bullet,l}^{(t)}}{f_j(\mathbf{Z}')}\right) - \frac{2\mathbf{a}_l^{\top}\mathbf{z}_{\bullet,\hat{c}}}{f_{\hat{c}}(\mathbf{Z}')} + \frac{2\mathbf{a}_l^{\top}\mathbf{z}_{\bullet,c}}{f_c(\mathbf{Z}')}$$
$$= \sum_{j=1}^{k}\left(\frac{G_l^{(t)}}{f_j(\mathbf{Z}')}\right) - \frac{2H^{(t)}(l,\hat{c})}{f_{\hat{c}}(\mathbf{Z}')} + \frac{2H^{(t)}(l,c)}{f_c(\mathbf{Z}')} \tag{12}$$

where

$$G_j^{(t)} = \left(\mathbf{z}_{\bullet,j}^{(t)}\right)^{\top}\mathbf{A}\mathbf{z}_{\bullet,j}^{(t)}, \quad H^{(t)}(l,j) = \mathbf{a}_l^{\top}\mathbf{z}_{\bullet,j}^{(t)} \tag{13}$$

The naive computational complexities of $G_k^{(t)}$ and $H_{l,c}^{(t)}$ are $O(n^2)$ and $O(n)$, respectively. However, we can efficiently compute $G_k^{(t)}$ and $H_{l,c}^{(t)}$ by

$$G_j^{(t+1)} = \begin{cases} G_j^{(t)} & (j\neq\hat{c} \text{ and } j\neq c) \\ G_j^{(t)} - 2H_{l,\hat{c}}^{(t)} & (j=\hat{c}) \\ G_j^{(t)} + 2H_{l,c}^{(t)} & (j=c) \end{cases}, \tag{14}$$

$$H^{(t+1)}(l,j) = \begin{cases} H^{(t)}(l,j) & (j\neq\hat{c} \text{ and } j\neq c) \\ H^{(t)}(l,j) - a_{l,l*} & (j=\hat{c}) \\ H^{(t)}(l,j) + a_{l,l*} & (j=c) \end{cases}. \tag{15}$$

By these update rules, both computational complexities of $G_k^{(t)}$ and $H_{l,c}^{(t)}$ are $O(1)$. We can also compute $f_k(\mathbf{Z}')$ in $O(1)$ for all the objective functions in Table 1. Therefore, the computational cost of $F^{(t)}(l,c)$ for all l,c is $O(kn^2)$ at the first iteration and $O(kn)$ at the other iterations.

Relation to existing algorithms: Our algorithm is similar to Hartigan's algorithm [10, 21] in k-means clustering. Especially, the objective function is equivalent to that of k-means clustering if we use the negative squared Euclidean distance for $\mathbf{A}$ and $f_k(\mathbf{Z}) = \mathbf{1}^{\top}\mathbf{z}_k$ in Eq. (1), and in this case DLO with the random coordinate selection is completely equivalent to Hartigan's algorithm.

The other local optimization method used in graph clustering is BO. BO has the parameter, which controls the strictness of the upper bound function, as described in Section 1.1. Although BO requires a proper determination of this parameter to achieve good optimization performances, our algorithm is parameter-free and guaranteed to increase the objective value. Moreover, our algorithm can be applied for the objective functions in Table 1, while BO is only applicable to macro-AA and NC in Table 1.

Algorithm 2 Greedy incremental algorithm (GIA).

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$
Initialize: $\mathbf{Z}^{(1)} = \mathbf{0}$, $T = \{1,\cdots,n\}$, $F^{(1)}(l,c) = s_{l,l}\ \forall c$
1: **for** $t = 1$ to n **do**
2: $l^*, c^* = \arg\max_{l\in T, c} F^{(t)}(l,c)$
3: Update $\mathbf{Z}^{(t+1)}$
4: Update $F^{(t+1)}(l,c)$
5: $T \leftarrow T \setminus l^*$
6: DLO for $S[\bar{T},\bar{T}]$
7: **end for**
Output: $\mathbf{Z}$

3.2. Greedy incremental algorithm (GIA)

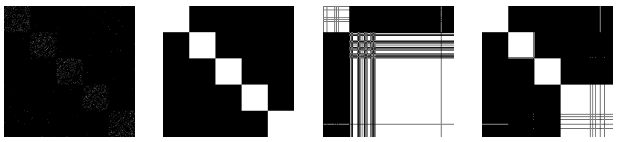
DLO is a local optimization method; thus, a good initial solution is required for global optimization. SC is a method of obtaining an initial solution. However, SC does not work in some cases described in Section 5. In this section, we propose an initial solution-free algorithm, namely the greedy incremental algorithm (GIA).

All clusters are initially set to be empty. At each iteration, a single element is assigned to a cluster by a greedy strategy. We first initialize $\mathbf{Z}^{(0)} = \mathbf{0}$ and $F_0(l,c) = s_{l,l}$ for all l and c, respectively. We also initialize unassigned elements set as $T = \{1,\cdots,n\}$. As in the same manner in Section 3.1, we can compute $F^{(t)}(l,c)$, which is the cost to assign the lth element into the cth cluster. The assignment that maximizes the objective value is then obtained by $l^*, c^* = \arg\max_{l\in T, c} F^{(t)}(l,c)$[1]. $\mathbf{Z}^{(t)}$ and T are then updated.

We summarize the GIA algorithm in Algorithm 2. We perform DLO for already assigned elements for each iteration. This DLO process improves the optimization performance. The total computational complexity of GIA is $O(n^2k + lnk)$, where l is the total number of iterations of DLO. Note that we do not need to compute $F^{(t)}(l,c)$ from scratch in the DLO process because it is computed in GIA. Moreover, l is generally smaller than n, at least in our experiments.

GIA works well in solving micro-AA. However, unfortunately, we found that GIA often failed into a poor local optimum in solving macro-AA, NC, and BA. In solving macro-AA and NC, assigning an element to the largest cluster is best because small clusters are overestimated from the terms $1/\mathbf{1}^{\top}\mathbf{z}_k$ and $1/\mathbf{1}^{\top}\mathbf{A}\mathbf{z}_k$, respectively. Therefore, GIA tends to generate small clusters with a single large cluster. We show the results in Fig. 2. In solving macro-AA by GIA,

[1]One of the solutions is randomly chosen if multiple solutions exist. For example of $t = 1$, $F^{(1)}(l,c) = 0$ for all l,c. Therefore, l,c are uniformly selected from $\{1,n\}$ and $\{1,\cdots,k\}$, respectively.



(a) Input $\mathbf{A}$. (b) Ground truth. (c) Solved on NC. (d) Solved on BA.

Fig 2: Clustering results solved by GIA. The global optimum clustering on both the NC and BA costs is shown in (b). GIA fails to obtain the best clustering, as in (c) and (d).

Algorithm 3 Graduated optimization.

Input: Affinity matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$

1: Obtain $\mathbf{Z}^{(g)}$ by solve Eq. (1) on $\mathbf{A}^g$
2: **for** $t = 1$ to $g - 1$ **do**
3: Obtain $\mathbf{Z}^{(g-t)}$ by DLO on $\mathbf{A}^{g-t}$ from $\mathbf{Z}^{(g)}$
4: **end for**

Output: $\mathbf{Z}^{(1)}$

assigning an element to the largest cluster is the best assignment because of the term $\mathbf{z}_k^\top \mathbf{A}\mathbf{z}_k$. Therefore, GIA tends to generate larger clusters than the optimum clustering.

4. Global optimization techniques for graph clustering

Most graph-clustering optimization methods, including our proposals, are often trapped into a poor local minimum because of the nonconvexity of the problem Eq. (1). Therefore, global optimization techniques are required. Running the algorithm multiple times and choosing the best results is a popular method. In this study, we introduce two global optimization techniques for graph clustering.

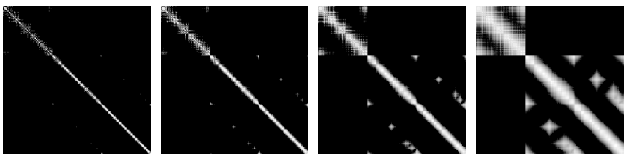
4.1. Graduated optimization (GO)

We find that GIA tends to be trapped into a bad local minimum if $\mathbf{A}$ is a high-sparse matrix. We overcome this problem by applying graduated optimization (GO) [1] to solve graph clustering. In solving a nonconvex function $f(x)$, GO first solves a smoothed function $f_s(x)$, which (is expected to) has less minima, then gradually transforms $f_s(x)$ into $f(x)$ with updating a solution. We construct a smoothed function in graph clustering by a smoothed affinity matrix of $\mathbf{A}^g$. For example of $g = 2$, the (i, j) element of $\mathbf{A}^2$ is presented as follows:

$$\left(\mathbf{A}^2\right)_{ij} = \sum_l a_{il} a_{lj}. \tag{16}$$

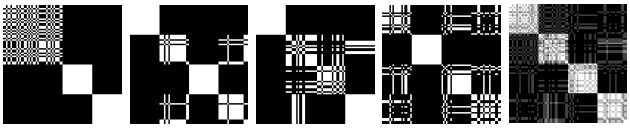
Eq. (16) means that $(\mathbf{A}^2)_{ij} > 0$ if there exists k such that $a_{ik}a_{kj} > 0$ although $a_{ij} = 0$. We show the examples in Fig. 3.

According to the manner of GO, we first use a large g to smooth the objective function to distinct poor local optima, then gradually decrease g to 1 to solve the original objective function. We summarize the algorithm in Algorithm 3.



(a) $\mathbf{A}$ (original) (b) $\mathbf{A}^2$ (c) $\mathbf{A}^4$ (d) $\mathbf{A}^8$

Fig 3: Examples of the matrices $\mathbf{A}^g$ used in GO.



(a) $\mathbf{\Theta}_1$ (b) $\mathbf{\Theta}_2$ (c) $\mathbf{\Theta}_3$ (d) $\mathbf{\Theta}_4$ (e) $\mathbf{\Theta}_{en}$

Fig 4: Clustering ensemble. (a)–(d) are local optimal solutions. (e) is co-association matrix.

4.2. Clustering ensemble (CE)

Although the solution obtained by GIA is often a poor local minimum, it tends to be partially well clustered as in Fig. 4. In this case, the clustering ensemble [22] is a good approach of achieving a high clustering performance.

We compute the co-association matrix [7] $\Theta_{en} = 1/n \sum_{i=1}^{n} \mathbf{Z}_i^\top \mathbf{Z}_i$ from n solutions $\mathbf{Z}_1, \cdots, \mathbf{Z}_n$. $\mathbf{\Theta}_{en}$ has continuous values. Hence, we need to transform $\mathbf{\Theta}_{en}$ to $\mathbf{\Theta} = \mathbf{Z}^\top \mathbf{Z}$, where $\mathbf{Z} \in \mathcal{Z}$, to obtain clustering. We can achieve this transformation by solving Eq. (1) on $\mathbf{A} = \mathbf{\Theta}_{en}$. Our motivation is to perform global optimization on graph-based clustering. Therefore, we finally perform DLO on original $\mathbf{A}$ from the solution obtained on $\mathbf{\Theta}_{en}$.

5. Experiments

We evaluated the performance of our proposed method. All the experiments were run on an Intel Core i7-6500U CPU (2.50 GHz) with 8 GB RAM and implemented in MATLAB.

5.1. Experimental detalis

We used three objective functions, namely NC, BA, and micro-AA, and three initialization methods, namely random initialization, SC, and GIA. Spectral relaxation can only be applied to NC; hence, SC was applied to NC for all the three objective functions. GIA was applied to the micro-AA cost for all the three objective functions because GIA works only on micro-AA, as described in Section 3.2.

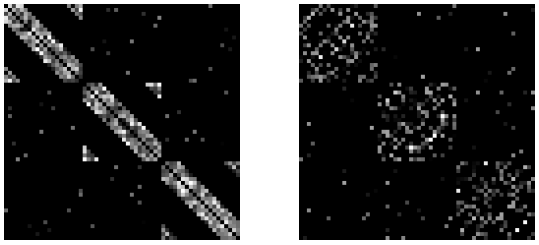
We used a fixed value or a linearly increasing value for parameter δ in BO. For example, $\delta = [0.0, 0.2]$ means that we used $\delta = 0.0$ at the first iteration and $\delta = 0.2$ at the last iteration. We used $\lambda = \lambda' \sum_{i,j} A_{ij}/n^2$ in BA. For the parameters μ in ADMM, we used the update rule of $\mu = \rho\mu$ at each iteration with $\rho = 1.02$ for all experiments. All

Table 2: Optimization performances for the NC cost on synthetic data. We report normalized objective values, such that the maximum value becomes one. The best and second-best values are emphasized in bold and italic, respectively.

Input	Random initialization			SC initialization			GIA initialization		
	DLO	BO		DLO	BO		DLO	BO	
		$\delta = [0.0, 0.2]$	$\delta = 0$		$\delta = [0.0, 0.2]$	$\delta = 0$		$\delta = [0.0, 0.2]$	$\delta = 0$
$\mathbf{S}_{diag}$	0.921	0.943	0.917	0.967	0.973	0.966	**1.000**	**1.000**	*0.999*
$\mathbf{S}_{rand}$	0.899	0.880	0.472	*0.999*	**1.000**	0.990	0.990	0.998	0.990

Table 3: Optimization performances for the BA cost on synthetic data. We report normalized objective values, such that the maximum value becomes one. The best and second-best values are emphasized in bold and italic, respectively.

Input	Random initialization				SC initialization				GIA initialization			
	DLO	ADMM			DLO	ADMM			DLO	ADMM		
		$\mu = 0.2$	$\mu = 0.6$	$\mu = 1.0$		$\mu = 0.2$	$\mu = 0.6$	$\mu = 1$		$\mu = 0.2$	$\mu = 0.6$	$\mu = 1$
$\mathbf{S}_{diag}$	0.886	0.938	0.959	0.908	0.927	0.933	0.958	0.928	**1.000**	*0.998*	**1.000**	**1.000**
$\mathbf{S}_{rand}$	0.864	0.936	0.901	0.74748	**1.000**	0.933	*0.995*	0.982	0.832	0.903	0.869	0.721



(a) $\mathbf{A}_{diag}$ $(q = 8)$ (b) $\mathbf{A}_{rand}$ $(p = 0.2)$

Fig 5: Examples of $\mathbf{A}_{diag}$ and $\mathbf{A}_{rand}$.

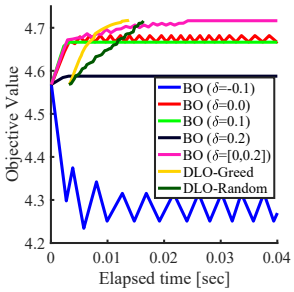


Fig 6: Objective values of NC by each method.

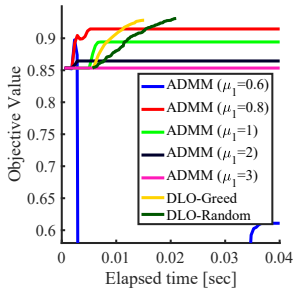


Fig 7: Objective values of BA by each method.

methods were run from 20 initial solutions for each matrix. The results with the best objective value are reported.

5.2. Experiments on synthetic data

We generated a synthetic affinity matrix $\mathbf{A}$ as follows:

$$\mathbf{A}' = \text{blockdiag}(\mathbf{B}_1, \cdots, \mathbf{B}_k) + \mathbf{N} \in \mathbb{R}^{mk \times mk}, \quad (17)$$

where $\mathbf{N}$ is a random sparse matrix with density 0.01. We constructed two types of $\mathbf{B}_i \in \mathbb{R}^{m \times m}$: q-th diagonal matrix ($\mathbf{A}_{diag}$) and random sparse matrix with density pm^2 ($\mathbf{A}_{rand}$). All non-zero elements of matrices $\mathbf{N}$ and $\mathbf{B}_i$ for all $i = 1, \cdots, k$ were uniformly sampled from $[0, 1]$. We finally computed $\mathbf{A} = (\mathbf{A}' + (\mathbf{A}')^\top)/2$ and set the diagonal element of $\mathbf{A}$ as 0. Fig. 5 shows the examples of $\mathbf{A}_{diag}$ and $\mathbf{A}_{rand}$ with $m = 20$, $k = 3$.

Comparing GIA with SC and random initialization: Table 2 and Table 3 present the objective values of NC and BA by each optimization method, respectively. The random initialization is worst in most cases. GIA outperformed SC in $\mathbf{A}_{diag}$. Conversely, SC initialization worked well in $\mathbf{A}_{rand}$. We can conclude that which initialization was better between SC and GIA depended on the affinity matrix characteristics.

Comparing DLO with BO and ADMM: Table 2 shows that DLO achieved comparable performances with BO. Note that DLO is a parameter-free algorithm, while BO has a parameter that significantly affects the optimization performances. Fig. 6 illustrates the objective values versus the elapsed time in one case of our experiments. As shown in Fig. 6, a large δ (e.g., $\delta = 0.1$ and $\delta = 0.2$) monotonically increased the objective values. However, an overly large δ (e.g., $\delta = 2$) achieved a poor optimization performance because a large δ enforces to not move the solution. In contrast, $\delta = 0$ did not converge because Eq. (4) with $\delta = 0$ was not a strict bound function. However, the optimization performance was better than a large delta (e.g., $\delta = 0.2$ and $\delta = 0.1$). The best method was to increase δ at each iteration, as in $\delta = [0.0, 0.2]$.

Table 3 shows that ADMM with proper μ achieved a better optimization performances than DLO, except for the case of the SC initialization for $\mathbf{S}_{rand}$. However, ADMM was not guaranteed to increase the objective value at each iteration. An overly small μ_1 breaks a good initial solution (Fig. 7).

Computational cost: We evaluated the computational costs of GIA with SC and ADMM. We generated different sizes of $\mathbf{A}_{diag}$ as $m = \{20, 50, 100, 200, 500\}$ with $k = 10$. Fig. 8 shows the computational costs of the three methods. GIA achieved a comparable computation cost with ADMM and outperformed SC. These results are in accordance with the fact that GIA has computational complexities of $O(n^2)$,

Table 4: Clustering results on MNIST. We used micro-AA and optimized by GIA and DLO from the SC initialization. We show the objective values and clustering accuracies (in parentheses).

Affinity	GIA				DLO from SC			
		with CE	with GO	GO + CE		with CE	with GO	GO + CE
SSC	45.56 (77.0)	47.21 (91.3)	**47.44 (93.7)**	47.41 (93.1)	47.08 (90.2)	47.12 (90.6)	*47.42 (93.2)*	47.38 (92.4)
k-NN	35.77 (90.4)	35.89 (93.4)	**35.91 (93.91)**	*35.90 (93.6)*	35.48 (86.7)	35.38 (85.9)	35.73 (89.7)	35.38 (84.6)

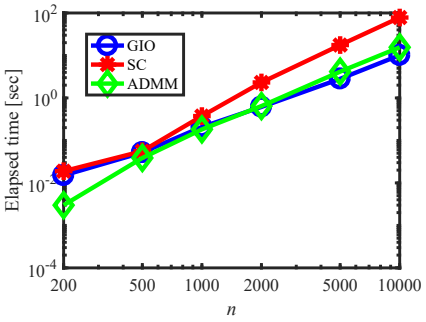


Fig 8: Computational costs of the three methods on different affinity matrix sizes.

while SC has a complexity of $O(n^3)$.

5.3. Experiments on the benchmark datasets

We used two types of affinity matrices in this section. First is the k-nearest neighbor (k-NN) graph with self-tuned scale [26]:

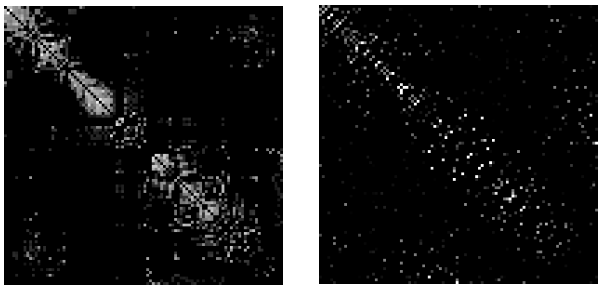
$$a_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{\sigma_i \sigma_j}\right) & (j \in k\text{-NN}(i)) \\ 0 & (\text{otherwise}) \end{cases} \quad (18)$$

where σ_i is the distance between point $\mathbf{x}_i$ and its kth nearest neighbor. The second type is the sparse self-representation (SSR) used in sparse subspace clustering [25, 6]. We used You *et al.*'s method [25] for MNIST and Elhamifar *et al.*'s method [6] for Extended YaleB and Hopkins 155.

MNIST: The MNIST dataset contains images of handwritten digits from 0 to 9. We followed the pre-process of [25]. We extracted 3,472-dimensional feature vectors by scattering convolution network [2], then reduced the dimension to 500 by PCA. We used $k = 10$ for the k-NN graph and the same parameters in [25] for SSR.

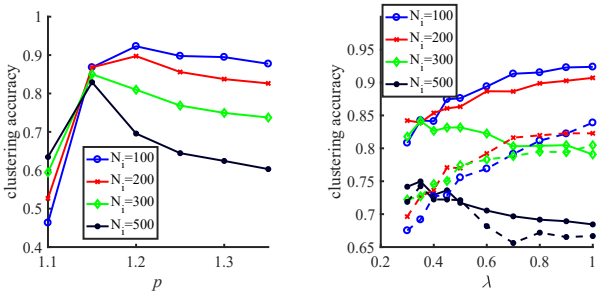
Comparing GIA with SC: We randomly selected totally 1,000 images, 100 images for each 10 digits. We run 10 times on different sets and report the average scores. We compared the performances of GIA and DLO from the SC initialization with and without graduated optimization (GO) and clustering ensemble (CE).

Table 4 shows the results. Without GO and CE, we found that SC performed well on the SSR matrix, whereas GIA performed well on the k-NN graph. Fig. 9 shows the part of matrix $\mathbf{A}$ in a single digit. The k-NN graph matrix seemed to have diagonal characteristics, whereas the SSR matrix is similar to the random matrix. These results indicate that



(a) $\mathbf{A}$ by k-NN. (b) $\mathbf{A}$ by SSR.

Fig 9: Affinity matrices by k-NN and SSR. We sorted the rows and columns to maximize the sum of the k-diagonal elements.



(a) Clustering by micro-AA with different p. (b) Clustering by BA with different λ'.

Fig 10: Clustering results on the imbalanced MNIST dataset. The solid and broken lines represent the clustering accuracies by DLO from GIA ($p = 1.2$) and ADMM from the random initialization, respectively.

GIA works in diagonal matrix and SC works in random matrix as with the synthetic data experiments.

We found that all methods can be improved by GO. CE also improved the performance in GIA. However, SC with CE was not effective. SC was assumed to obtain the narrow local minima because of the spectral relaxation with k-means clustering.

Imbalanced MNIST: We showed the performances of micro-AA and BA on imbalanced data. We randomly selected $N_i \in 150, 200, 300, 500$ images for one digit and 100 images for the other digits. Therefore, a total of 1050, 1100, 1200, and 1400 images were selected.

Fig. 10(a) and 10(b) show the clustering accuracy solved on micro-AA cost with different p and BA cost with different λ, respectively. Micro-AA with a small p achieved

Table 5: Results on the benchmark datasets. We show the clustering accuracies. Except for spectral clustering (Spect. Clust) and ADMM, we used DLO from initialization methods of GIA or SC.

dataset	micro-AA ($p = 1.2$)			NC			BA ($\lambda' = 0.8$)				Spect.
	SC	GIA	CE (GIA)	SC	GIA	CE (GIA)	ADMM	SC	GIA	CE (GIA)	Clust.
COIL-20	72.78	*95.56*	**100.0**	70.83	86.67	**100.0**	62.71	71.32	*95.56*	**100.0**	56.87
COIL-100	63.24	80.82	**89.85**	60.42	80.74	*83.50*	50.33	61.51	80.75	80.63	61.49
MNIST-05	67.78	77.65	*83.52*	67.75	65.46	82.03	65.24	67.81	77.71	**83.69**	66.87
Umist	68.17	76.87	**80.00**	73.39	70.78	76.17	64.17	68.70	65.74	*79.30*	73.04
USPS	77.34	80.95	*94.53*	83.23	80.92	82.17	51.74	77.34	80.96	**96.57**	68.35
Extended Yale-B	**96.25**	95.31	**96.25**	*96.09*	95.31	**96.25**	62.34	**96.25**	95.31	**96.25**	*96.09*
Hopkins 155	90.10	91.51	88.57	*94.22*	93.20	91.15	85.76	86.84	87.29	87.34	**94.24**

Table 6: Descriptions of the datasets and used affinity matrix. We show the minimum/maximum number of samples per class.

Dataset	Classes	Samples per class	**A**
COIL-20	20	72	k-NN
COIL-100	100	72	k-NN
MNIST-05	10	315-393	k-NN
Umist	20	19-48	k-NN
USPS	10	708-1553	k-NN
Extended Yale-B	10	72	SSR
Hopkins 155	3	33-443[2]	SSR

good clustering accuracies in imbalanced datasets, such as $p = 1.15$ in $N_i = 500$. Micro-AA with $p = 1.1$ tended to generate a large cluster because of the small power of balancing cluster sizes, as in $N_i = 100$. For BA, DLO from GIA ($p = 1.2$) achieved better performances than ADMM. BA with a small λ solved by DLO and GIA also achieved a comparable performance with micro-AA in the imbalanced datasets.

Experiments on the other benchmark datasets: We evaluated the clustering methods on seven datasets: COIL-20 [15], COIL-100 [14] (20 and 100 classes of different objects), MNIST-05 [3] (handwritten digits sampled from MNIST), UMIST [9] (face images of 20 individuals), USPS [11] (handwritten digits), Extended Yale-B [8] (face images of 38 individuals), and Hopkins 155[4] (two or three motion segmentations). For the Extended Yale-B dataset, we followed the experimental setting of [6] and reported the averaging scores from three trials. For the Hopkins 155 dataset, we used 35 out of 155 videos, which contained three motion segments and reported the average of all 35 results. For the affinity matrices of the k-NN graphs, we used $k = 4$ for COIL-20, COIL-100, and Umist, $k = 10$ for MNIST-05 and USPS, and the same parameters in [6] for Extended Yale-B and Hopkins 155. Table 6 summarizes the descriptions of these seven datasets.

[2]We show the number with the maximum ratio in one motion data.
[3]http://www.escience.cn/people/fpnie/index.html
[4]http://www.vision.jhu.edu/data/hopkins155/

Table 5 shows the clustering accuracies. We used the random initialization for ADMM. In the five datasets using the k-NN affinity matrices, GIA outperformed other optimization methods in most cases of all the three cost functions. CE improves the clustering accuracies in most cases, especially 100% clustering accuracy in the COIL-20 dataset. For the cost functions, micro-AA and BA outperformed NC, except for Hopkins 155. Hopkins 155 is a highly imbalanced dataset, which has the maximum ratio of cluster sizes over 13 (Table 6). Therefore, micro-AA and BA did not perform well because these cost functions overly balanced the cluster sizes.

6. Conclusion

In this study, we presented novel objective function and optimization techniques for graph-based clustering. First, we introduced micro-AA, which was based on micro-average, to avoid undesirable small clusters. For optimization, we proposed DLO, which is guaranteed to monotonically increase the objective value for general objective functions used in graph-based clustering. We also proposed GIA, which was an initialization-free optimization method with a computational complexity of $O(n^2)$ and global optimization techniques (i.e., GO and CE). Our experiments showed that the proposed techniques significantly improved the clustering performances and provided guidelines on when each method works well.

Although we investigated the relation between the optimization methods and the affinity matrix structures, we did not strongly focus on how to construct an affinity matrix. As a future work, we will try to apply our optimization techniques for the joint formulation of affinity matrix construction and graph-based clustering.

Acknowledgements. This research is partially supported by CREST (JPMJCR1686).

References

[1] A. Blake and A. Zisserman. *Visual reconstruction*. 1987. 5

[2] J. Bruna and S. Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013. 7

[3] X. Chen, J. Zhexue Haung, F. Nie, R. Chen, and Q. Wu. A self-balanced min-cut algorithm for image clustering. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 1, 2

[4] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 551–556. ACM, 2004. 2

[5] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE transactions on pattern analysis and machine intelligence*, 29(11), 2007. 1, 2

[6] E. Elhamifar and R. Vidal. Sparse subspace clustering: Algorithm, theory, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 35(11):2765–2781, 2013. 1, 7, 8

[7] A. L. Fred and A. K. Jain. Combining multiple clusterings using evidence accumulation. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):835–850, 2005. 5

[8] A. Georghiades, P. Belhumeur, and D. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE Trans. Pattern Anal. Mach. Intelligence*, 23(6):643–660, 2001. 8

[9] D. GRAHAM and N. ALLINSON. Characterising virtual eigensignatures for general purpose face recognition. *NATO ASI series. Series F: computer and system sciences*, pages 446–456, 1998. 8

[10] J. A. Hartigan. Clustering algorithms (probability & mathematical statistics), 1975. 4

[11] J. J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 1994. 8

[12] C.-G. Li, C. You, and R. Vidal. Structured sparse subspace clustering: A joint affinity learning and subspace clustering framework. *IEEE Transactions on Image Processing*, 26(6):2988–3001, 2017. 1

[13] G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):171–184, 2013. 1

[14] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil 100). *Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*, 1996. 8

[15] S. A. Nene, S. K. Nayar, and H. Murase. Columbia object image library (coil-20). 1996. 8

[16] Y. Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012. 3

[17] F. Nie, X. Wang, M. I. Jordan, and H. Huang. The constrained laplacian rank algorithm for graph-based clustering. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1969–1976. AAAI Press, 2016. 1

[18] J. Nutini, M. Schmidt, I. Laradji, M. Friedlander, and H. Koepke. Coordinate descent converges faster with the gauss-southwell rule than random selection. In *International Conference on Machine Learning*, pages 1632–1641, 2015. 3

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000. 1, 2

[20] M. Tang, D. Marin, I. B. Ayed, and Y. Boykov. Kernel Cuts: MRF meets kernel & spectral clustering. *arXiv preprint arXiv:1506.07439*, 2015. 1, 2

[21] M. Telgarsky and A. Vattani. Hartigan's method: k-means clustering without voronoi. In *International Conference on Artificial Intelligence and Statistics*, pages 820–827, 2010. 4

[22] S. Vega-Pons and J. Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(03):337–372, 2011. 5

[23] R. Vidal and P. Favaro. Low rank subspace clustering (lrsc). *Pattern Recognition Letters*, 43:47–61, 2014. 1

[24] U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007. 1, 2

[25] C. You, D. Robinson, and R. Vidal. Scalable sparse subspace clustering by orthogonal matching pursuit. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3918–3927, 2016. 1, 7

[26] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in neural information processing systems*, pages 1601–1608, 2005. 1, 7