**Title:**

# A new theoretical framework for K-means-type clustering

**Authors:**

Jiming Peng and Yu Xia

**AdvOl-Report No. 2004/6**

July 2004, Hamilton, Ontario, Canada

# A new theoretical framework for K-means-type clustering

Jiming Peng [*] Yu Xia

July 9, 2004

**Abstract**

One of the fundamental clustering problems is to assign $n$ points into $k$ clusters based on the minimal sum-of-squares(MSSC), which is known to be NP-hard. In this paper, by using matrix arguments, we first model MSSC as a so-called 0-1 semidefinite programming (SDP). The classical K-means algorithm can be interpreted as a special heuristics for the underlying 0-1 SDP. Moreover, the 0-1 SDP model can be further approximated by the relaxed and polynomially solvable linear and semidefinite programming. This opens new avenues for solving MSSC. The 0-1 SDP model can be applied not only to MSSC, but also to other scenarios of clustering as well. In particular, we show that the recently proposed normalized k-cut and spectral clustering can also be embedded into the 0-1 SDP model in various kernel spaces.

## 1 Introduction

Clustering is one of major issues in data mining and machine learning with many applications arising from different disciplines including text retrieval, pattern recognition and web mining[12, 15]. Roughly speaking, clustering involves partition a given data set into subsets based on the closeness or similarity among the data. Typically, the similarities among entities in a data set are measured by a specific proximity function, which can be make precise in many ways. This results in many clustering problems and algorithms as well.

Most clustering algorithms belong to two classes: hierarchical clustering and partitioning. The hierarchical approach produces a nested series of partitions consisting of clusters either disjoint or included one into the other. Those

[*]Advanced optimization Lab, Department of Computing and Software McMaster University, Hamilton, Ontario L8S 4K1, Canada. Email:pengj@mcmaster.ca, xiay@optlab..mcmaster.ca. The research of the first author was partially supported by the grant # RPG 249635-02 of the National Sciences and Engineering Research Council of Canada (NSERC) and a PREA award. This research was also Supported by the MITACS project "New Interior Point Methods and Software for Convex Conic-Linear Optimization and Their Application to Solve VLSI Circuit Layout Problems".

clustering algorithms are either agglomerative or divisive. An agglomerative clustering algorithm starts with every singleton entity as a cluster, and then proceeds by successively merging clusters until a stopping criterion is reached. A divisive approach starts with an initial cluster with all the entities in it, and then performs splitting until a stopping criterion is reached. In hierarchical clustering, an objective function is used locally as the merging or splitting criterion. In general, hierarchical algorithms can not provide optimal partitions for their criterion. In contrast, partitional methods assume given the number of clusters to be found and then look for the optimal partition based on the object function. Partitional methods produce only one partition. Most partitional methods can be further classified as deterministic or stochastic, depending on whether the traditional optimization technique or a random search of the state space is used in the process. There are several different ways to separate various clustering algorithms, for a comprehensive introduction to the topic, we refer to the book [12, 15], and for more recent results, see survey papers [4] and [13].

Among various criterion in clustering, the minimum sum of squared Euclidean distance from each entity to its assigned cluster center is the most intuitive and broadly used. Both hierarchical and partitional procedures for MSSC have been investigated. For example, Ward's [27] agglomerative approach for MSSC has a complexity of $O(n^2 \log n)$ where $n$ is the number of entities. The divisive hierarchical approach is more difficult. In [9], the authors provided an algorithm running in $O(n^{d+1} \log n)$ time, where $d$ is the dimension of the space to which the entities belong.

However, in many applications, assuming a hierarchical structure in partitioning based on MSSC is unpractical. In such a circumstance, the partitional approach directly minimizing the sum of squares distance is more applaudable. The traditional way to deal with this problem is to use some heuristics such as the well-known K-means [18]. To describe the algorithm, let us go into a bit more details.

Given a set $S$ of $n$ points in a $d$-dimensional Euclidean space, denoted by

$$S = \{\mathbf{s}_i = (s_{i1}, \cdots, s_{id})^T \in \mathbf{R}^d \quad i = 1, \cdots, n\}$$

the task of a partitional MSSC is to find an assignment of the $n$ points into $k$ disjoint clusters $\mathcal{S} = (S_1, \cdots, S_k)$ centered at cluster centers $\mathbf{c}_j$ $(j = 1, \cdots, k)$ based on the total sum-of-squared Euclidean distances from each point $\mathbf{s}_i$ to its assigned cluster centroid $\mathbf{c}_i$, i.e.,

$$f(S, \mathcal{S}) = \sum_{j=1}^{k} \sum_{i=1}^{|S_j|} \left\| \mathbf{s}_i^{(j)} - \mathbf{c}_j \right\|^2,$$

where $|S_j|$ is the number of points in $S_j$, and $\mathbf{s}_i^{(j)}$ is the $i^{th}$ point in $S_j$. Note that if the cluster centers are known, then the function $f(S, \mathcal{S})$ achieves its minimum when each point is assigned to its closest cluster center. Therefore, MSSC can be described by the following bilevel programming problem (see for instance

[2, 19]).

$$\min_{\mathbf{c}_1,\cdots,\mathbf{c}_k} \sum_{i=1}^{n} \min\{\|\mathbf{s}_i - \mathbf{c}_1\|^2, \cdots, \|\mathbf{s}_i - \mathbf{c}_k\|^2\}. \tag{1}$$

Geometrically speaking, assigning each point to the nearest center fits into a framework called *Voronoi Program*, and the resulting partition is named *Voronoi Partition*. On the other hand, if the points in cluster $S_j$ are fixed, then the function

$$f(S_j, \mathcal{S}_j) = \sum_{i=1}^{|S_j|} \left\| \mathbf{s}_i^{(j)} - \mathbf{c}_j \right\|^2$$

is minimal when

$$\mathbf{c}_j = \frac{1}{|S_j|} \sum_{i=1}^{|S_j|} \mathbf{s}_i^{(j)}.$$

The classical K-means algorithm [18], based on the above two observations, is described as follows:

**K-means clustering algorithm**

**(1)** Choose $k$ cluster centers randomly generated in a domain containing all the points,
**(2)** Assign each point to the closest cluster center,
**(3)** Recompute the cluster centers using the current cluster memberships,
**(4)** If a convergence criterion is met, stop; Otherwise go to step 2.

Another way to model MSSC is based on the assignment. Let $X = [x_{ij}] \in \Re^{n\times k}$ be the assignment matrix defined by

$$x_{ij} = \begin{cases} 1 & \text{If } \mathbf{s}_i \text{ is assigned to } S_j; \\ 0 & \text{Otherwise.} \end{cases}$$

As a consequence, the cluster center of the cluster $S_j$, as the mean of all the points in the cluster, is defined by

$$\mathbf{c}_j = \frac{\sum_{l=1}^{n} x_{lj}\mathbf{s}_l}{\sum_{l=1}^{n} x_{lj}}.$$

Using this fact, we can represent (1) as

$$\min_{x_{ij}} \sum_{j=1}^{k}\sum_{i=1}^{n} x_{ij} \left\| \mathbf{s}_i - \frac{\sum_{l=1}^{n} x_{lj}\mathbf{s}_l}{\sum_{l=1}^{n} x_{lj}} \right\|^2 \tag{2}$$

$$S.T. \sum_{j=1}^{k} x_{ij} = 1 \ (i = 1, \cdots, n) \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} \geq 1 \ (j = 1, \cdots, k) \tag{4}$$

$$x_{ij} \in \{0, 1\} \ (i = 1, \cdots, n;\ j = 1, \cdots, k) \tag{5}$$

The constraint (3) ensures that each point $\mathbf{s}_i$ is assigned to one and only one cluster, and (4) ensures that there are exactly $k$ clusters. This is a mixed integer programming with nonlinear objective [8], which is NP-hard. The difficulty of the problem consists of two parts. First, the constraints are discrete. Secondly the objective is nonlinear and nonconvex. Both the difficulties in the objective as well as in the constraints make MSSC extremely hard to solve.

Many different approaches have been proposed for attacking (2) both in the communities of machine learning and optimization [1, 8, 3]. Most methods for (2) are heuristics that can locate only a good local solution, not the exact global solution for (2). Only a few works are dedicated to the exact algorithm for (2) as listed in the references of [3].

Approximation methods provide a useful approach for (2). There are several different ways to approximate (2). For example, by solving the so-called K-medians problem we can obtain a 2-approximately optimal solution for (2) in $O(n^{d+1})$ time [10]. In [22], Mutousek proposed a geometric approximation method that can find an $(1+\epsilon)$ approximately optimal solution for (2) in $O(n \log^k n)$ time, where the constant hidden in the big-O notation depends polynomially on $\epsilon$. Another efficient way of approximation is to attack the original problem (typically NP-hard) by solving a relaxed polynomially solvable problem. This has been well studied in the field of optimization, in particular, in the areas of combinatorial optimization and semidefinite programming [5]. We noted that recently, Xing and Jordan [29] considered the SDP relaxation for the so-called normalized k-cut spectral clustering.

In the present paper, we focus on developing approximation methods for (2) based on linear and semidefinite programming (LP/SDP) relaxation. A crucial step in relaxing (2) is to rewrite the objective in (2) as a simple convex function of matrix argument that can be tackled easily, while the constraint set still enjoy certain geometric properties. This was possibly first suggested in [6] where the authors owed the idea to an anonymous referee. However, the authors of [6] did not explore the idea in depth to design any usable algorithm. A similar effort was made in [30] where the authors rewrote the objective in (2) as a convex quadratic function in which the argument is a $n \times k$ orthonormal matrix.

Our model follows the same stream as in [6, 30]. However, different from the approach [30] where the authors used only a quadratic objective and simple spectral relaxation, we elaborate more on how to characterize (2) exactly by means of matrix arguments. In particular, we show that MSSC can be modelled as the so-called 0-1 semidefinite programming (SDP), which can be further relaxed to polynomially solvable linear programming (LP) and SDP. Several different relaxation forms are discussed. We also show that variants of K-means can be viewed as heuristics for the underlying 0-1 SDP.

Our model provides novel avenues not only for solving MSSC, but also for solving clustering problems based on some other criterions. For example, the clustering based on normalized cuts can also be embedded into our model. Moreover, our investigation reveals some interesting links between the well-known K-means and some recently proposed algorithms like spectral clustering.

The paper is organized as follows. In Section 2, we show that MSSC can be modelled as 0-1 SDP, which allows convex relaxation such as SDP and LP. In Section 3, we discuss algorithms and challenges for solving our 0-1 SDP model. Section 4 devotes to the discussion on the links between our model and some other recent models for clustering. Finally we close the paper by few concluding remarks.

## 2 Equivalence of MSSC to 0-1 SDP

In this section, we establish the equivalence between MSSC and 0-1 SDP. We start with a brief introduction to SDP and 0-1 SDP.

In general, SDP refers to the problem of minimizing (or maximizing) a linear function over the intersection of a polyhedron and the cone of symmetric and positive semidefinite matrices. The canonical SDP takes the following form

$$\textbf{(SDP)} \begin{cases} \min & \mathrm{Tr}\,(\mathrm{WZ}) \\ S.T. & \mathrm{Tr}\,(\mathrm{B_iZ}) = \mathrm{b_i} \quad \mathrm{for i} = 1, \cdots, \mathrm{m} \\ & Z \succeq 0 \end{cases}$$

Here $\mathrm{Tr}\,(.)$ denotes the trace of the matrix, and $Z \succeq 0$ means that $Z$ is positive semidefinite. If we replace the constraint $Z \succeq 0$ by the requirement that $Z^2 = Z$, then we end up with the following problem

$$\textbf{(0-1 SDP)} \begin{cases} \min & \mathrm{Tr}\,(\mathrm{WZ}) \\ S.T. & \mathrm{Tr}\,(\mathrm{B_iZ}) = \mathrm{b_i} \quad \mathrm{for i} = 1, \cdots, \mathrm{m} \\ & Z^2 = Z, Z = Z^T \end{cases}$$

We call it 0-1 SDP owing to the similarity of the constraint $Z^2 = Z$ to the classical 0-1 requirement in integer programming.

We next show that MSSC can be modelled as 0-1 SDP. By rearranging the items in the objective of (2), we have

$$(6) \qquad f(S, \mathcal{S}) = \sum_{i=1}^{n} \|\mathbf{s}_i\|^2 \left( \sum_{j=1}^{k} x_{ij} \right) - \sum_{j=1}^{k} \frac{\left\| \sum_{i=1}^{n} x_{ij} \mathbf{s}_i \right\|^2}{\sum_{i=1}^{n} x_{ij}}$$
$$= \mathrm{Tr}\left(\mathrm{W_S^T W_S}\right) - \sum_{\mathrm{j=1}}^{\mathrm{k}} \frac{\left\| \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{x_{ij}} \mathbf{s}_{\mathrm{i}} \right\|^2}{\sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{x_{ij}}},$$

where $W_S \in \Re^{n \times d}$ denotes the matrix whose $i$th row is the vector $\mathbf{s}_i$. Since $X$ is an assignment matrix, we have

$$X^T X = \mathrm{diag}\,(\sum_{i=1}^{n} x_{i1}^2, \cdots, \sum_{i=1}^{n} x_{ik}^2) = \mathrm{diag}\,(\sum_{i=1}^{n} x_{i1}, \cdots, \sum_{i=1}^{n} x_{ik}).$$

Let

$$Z := [z_{ij}] = X(X^T X)^{-1} X^T,$$

we can write (6) as $\mathrm{Tr}\left(\mathrm{W_S W_S^T (I - Z)}\right) = \mathrm{Tr}\left(\mathrm{W_S^T W_S}\right) - \mathrm{Tr}\left(\mathrm{W_S^T W_S Z}\right)$. Obviously $Z$ is a projection matrix satisfying $Z^2 = Z$ with nonnegative elements. For any integer $m$, let $e_m$ be the all one vector in $\Re^m$. We can write the constraint (3) as

$$Xe^k = e^n.$$

It follows immediately

$$Ze^n = ZXe^k = Xe^k = e^n.$$

Moreover, the trace of $Z$ should equal to $k$, the number of clusters, i.e.,

$$\mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}.$$

Therefore, we have the following 0-1 SDP model for MSSC

$$\begin{aligned} \min \quad & \mathrm{Tr}\left(\mathrm{W_S W_S^T (I - Z)}\right) \\ & Ze = e, \mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}, \\ & Z \geq 0, Z = Z^T, Z^2 = Z. \end{aligned} \tag{7}$$

We first give a technical result about positive semidefinite matrix that will be used in our later analysis.

**Lemma 2.1** *For any symmetric positive semidefinite matrix $Z \in \Re^{n \times n}$, there exists an index $i_0 \in \{1, \cdots, n\}$ such that*

$$Z_{i_0 i_0} = \max_{i,j} Z_{ij}.$$

**Proof:** For any positive semidefinite matrix $Z$, it is easy to see that

$$Z_{ii} \geq 0, \quad i = 1, \cdots, n.$$

Suppose the statement of the lemma does not hold, i.e., there exists $i_0 \neq j_0$ such that

$$Z_{i_0 j_0} = \max_{i,j} Z_{ij} > 0.$$

Then the submatrix

$$\begin{pmatrix} Z_{i_0 i_0} & Z_{i_0 j_o} \\ Z_{j_0 i_0} & Z_{j_0 j_0} \end{pmatrix}$$

is not positive semidefinite. This contradicts to the assumptuion in the lemma.

Now we are ready to establish the equivalence between the models (7) and (2).

**Theorem 2.2** *Solving the 0-1 SDP problem (7) is equivalent to finding a global solution of the integer programming problem (2).*

**Proof:** From the construction of the 0-1 SDP model (7), we know that one can easily construct a feasible solution for (7) from a feasible solution of (2). Therefore, it remains to show that from a global solution of (7), we can obtain a feasible solution of (2).

Suppose that $Z$ is a global minimum of (7). Obviously $Z$ is positive semidefinite. From Lemma 2.1 we conclude that there exists an index $i_1$ such that

$$Z_{i_1 i_1} = \max\{Z_{ij} : 1 \le i, j \le n\} > 0.$$

Let us define the index set

$$\mathcal{I}_1 = \{j : Z_{i_1 j} > 0\}.$$

Since $Z^2 = Z$, we have

$$\sum_{j \in \mathcal{I}_1} (Z_{i_1 j})^2 = Z_{i_1 i_1},$$

which implies

$$\sum_{j \in \mathcal{I}_1} \frac{Z_{i_1 j}}{Z_{i_1 i_1}} Z_{i_1 j} = 1.$$

From the choice of $i_1$ and the constraint

$$\sum_{j=1}^{n} Z_{i_1 j} = \sum_{j \in \mathcal{I}_1} Z_{i_1 j} = 1,$$

we can conclude that

$$Z_{i_1 j} = Z_{i_1 i_1}, \quad \forall j \in \mathcal{I}_1.$$

This further implies that the submatrix $Z_{\mathcal{I}_1 \mathcal{I}_1}$ is a matrix whose elements are all equivalent, and we can decompose the matrix $Z$ into a bock matrix with the following structure

$$Z = \begin{pmatrix} Z_{\mathcal{J}_1 \mathcal{J}_1} & 0 \\ 0 & Z_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1} \end{pmatrix}, \tag{8}$$

where $\bar{\mathcal{I}}_1 = \{i : i \notin \mathcal{I}_1\}$. Since $\sum_{i \in \mathcal{I}_1} Z_{ii} = 1$ and $(Z_{\mathcal{I}_1 \mathcal{I}_1})^2 = Z_{\mathcal{I}_1 \mathcal{I}_1}$, we can consider the reduced 0-1 SDP as follows

$$\begin{aligned} \min \;\; & \mathrm{Tr}\left(\left(\mathrm{W_S W_S^T}\right)_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1} (\mathrm{I} - \mathrm{Z})_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1}\right) \\ & Z_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1} e = e, \mathrm{Tr}\left(\mathrm{Z}_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1}\right) = \mathrm{k} - 1, \\ & Z_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1} \ge 0, Z^2_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1} = Z_{\bar{\mathcal{I}}_1 \bar{\mathcal{I}}_1}. \end{aligned} \tag{9}$$

Repeating the above process, we can show that if $Z$ is a global minimum of the 0-1 SDP, then it can be decomposed into a diagonal block matrix as

$$Z = \mathrm{diag}\,(Z_{\mathcal{I}_1\mathcal{I}_1}, \cdots, Z_{\mathcal{I}_k\mathcal{I}_k}),$$

where each block matrix $Z_{\mathcal{I}_l\mathcal{I}_l}$ is a nonnegative projection matrix whose elements are equal, and the sum of each column or each row equals to 1.

Now let us define the assignment matrix $X \in \Re^{n\times k}$

$$X_{ij} = \begin{cases} 1 & \text{if } i \in \mathcal{I}_j \\ 0 & \text{otherwise} \end{cases}$$

One can easily verify that $Z = X(X^TX)^{-1}X^T$. Our above discussion illustrates that from a feasible solution of (7), we can obtain an assignment matrix that satisfies the condition in (2). This finishes the proof of the theorem.

By comparing (7) with (2), we find that the objective in (7) is linear, while the constraint in (7) is still nonlinear, even more complex than the 0-1 constraint in (2). The most difficult part in the constraint of (7) is the requirement that $Z^2 = Z$. Several different ways for solving (7) will be discussed in the next section.

## 3 Algorithms for solving 0-1 SDP

In this section, we focus on various algorithms for solving the 0-1 SDP model (7). From a viewpoint of the algorithm design, we can separate these algorithms into two groups. The first group consists of the so-called feasible iterative algorithms, while the second group contains approximation algorithms (might be infeasible at some stage in the process) based on relaxation. It is worthwhile pointing out that our discussion will focus on the design of the algorithm as well as the links among various techniques, not on the implementation details of the algorithm and numerical testing.

### 3.1 Feasible Iterative Algorithms

We first discuss the so-called feasible iterative algorithms in which all the iterates are feasible regarding the constraints in (7), while the objective is reduced step by step until some termination criterion is reached. A general procedure for feasible iterative algorithms can be described as follows:

**Feasible Iterative Algorithm**

**Step 1:** Choose a starting matrix $Z^0$ satisfying all the constraints in (7),

**Step 2:** Use a heuristics to update the matrix $Z^k$ such that the value of the objective function in (7) is decreased,

**Step 3:** Check the termination criterion. If the criterion is reached, then stop; Otherwise go to Step 2.

We point out that the classical K-means algorithm described in the introduction can be interpreted as a special feasible iterative scheme for attacking (7). To see this, let us recall our discussion on the equivalence between MSSC and (7), one can verify that, at each iterate, all the constraints in (7) are satisfied by the matrix transformed from the K-means algorithm. It is also easy to see that, many variants of the K-means algorithm such as the variants proposed in [11, 14], can also be interpreted as specific iterative schemes for (7).

## 3.2 Approximation Algorithms Based on LP/SDP Relaxations

In the section we discuss the algorithms in the second group that are based on LP/SDP relaxation. We starts with a general procedure for those algorithm.

**Approximation Algorithm Based on Relaxation**

**Step 1:** Choose a relaxation model for (7),
**Step 2:** Solve the relaxed problem for an approximate solution,
**Step 3:** Use a rounding procedure to extract a feasible solution to (7) from the approximate solution.

The relaxation step has an important role in the whole algorithm. For example, if the approximation solution obtained from Step 2 is feasible for (7), then it is exactly an optimal solution of (7). On the other hand, when the approximation solution is not feasible regarding (7), we have to use a rounding procedure to extract a feasible solution. In what follows we discuss how to design a rounding procedure.

First, we note that when $Z^*$ is a solution of (7), it can be shown that the matrix $Z^*W_S$ contains $k$ different rows, and each of these $k$ different rows represents one center in the final clusters. A good approximate solution, although it might not be feasible for (7), should give us some indications on how to locate a feasible solution. Motivated by the above-mentioned observation, we can cast the rows of the matrix $ZW_S$ as a candidate set for the potential approximate centers in the final clustering. This leads to the following rounding procedure.

**A Rounding Procedure**

**Step 0:** Input: an approximate solution $Z$ and the matrix $W_S$,
**Step 1:** Select $k$ rows from the rows of the matrix $ZW_S$ [1] as the initial centers,
**Step 2:** Apply the classical K-means to the original MSSC using the selected initial centers.

We mention that in [29], Xing and Jordan proposed a rounding procedure based on the singular value decomposition $Z = U^TU$ of $Z$. In their approach,

[1] For example, we can select $k$ rows from $ZW_S$ based on the frequency of the row in the matrix, or arbitrarily select $k$ centers.

Xing and Jordan first cast the rows of $U^T$ as points in the space, and then they employed the classical K-means to cluster those points.

Another way for extracting a feasible solution is to utilize branch and cut. In order to use branch and cut, we recall the fact that any feasible solution $Z$ of (7) satisfies the following condition

$$Z_{ij}(Z_{ij} - Z_{ii}) = 0, \quad i, j = 1, \cdots, n.$$

If an approximate solution meets the above requirement, then it is a feasible solution and thus an optimal solution to (7). Otherwise, suppose that there exist indices $i, j$ such that

$$Z_{ij}(Z_{ii} - Z_{ij}) \neq 0,$$

then we can add cut $Z_{ii} = Z_{ij}$ or $Z_{ij} = 0$ to get two subproblems. By combining such a branch-cut procedure with our linear relaxation model, we can find the exact solution to (7) in finite time, as the number of different branches is at most $2^{n^2}$.

To summarize, as shown in our above discussion, finding a good approximation (or a nice relaxation) is essential for the success of approximation algorithms. This will be the main focus in the following subsections.

### 3.2.1 Relaxations based on SDP

In this subsection, we describe few SDP-based relaxations for (7). First we recall that in (7), the argument $Z$ is stipulated to be a projection matrix, i.e., $Z^2 = Z$, which implies that the matrix $Z$ is a positive semidefinite matrix whose eigenvalues are either 0 or 1. A straightforward relaxation to (7) is replacing the requirement $Z^2 = Z$ by the relaxed condition

$$I \succeq Z \succeq 0.$$

Note that in (7), we further stipulate that all the entries of $Z$ are nonnegative, and the sum of each row(or each column) of $Z$ equals to 1. This means the eigenvalues of $Z$ is always less than 1. In this circumstance, the constraint $Z \preceq I$ becomes superfluous and can be waived. Therefore, we obtain the following SDP relaxation for MSSC

$$(10) \qquad \begin{aligned} \min \ & \mathrm{Tr}\left(\mathrm{W_S W_S^T (I - Z)}\right) \\ & Ze = e, \mathrm{Tr}\left(\mathrm{Z}\right) = \mathrm{k}, \\ & Z \geq 0, Z \succeq 0. \end{aligned}$$

The above problem is feasible and bounded below. We can apply many existing optimization solvers such as interior-point methods to solve (10). It is known that an approximate solution to (10) can be found in polynomial time.

We noted that in [29], the model (10) with a slightly different linear constraint [2] was used as a relaxation to the so-called normalized k-cut clustering.

[2]In [29], the constraint $Ze = e$ in (7) is replaced by $Zd = d$ where $d$ is a positive scaling vector associated with the affinity matrix, and the constraint $Z \preceq I$ can not be waived.

As we shall show in section 4, the model (10) can always provides better approximation to (7) than the spectral clustering. This was also observed and pointed out by Xing and Jordan [29].

However, we would like to point out here that although there exist theoretically polynomial algorithm for solving (10), most of the present optimization solvers are unable to handle the problem in large size efficiently.

Another interesting relaxation to (7) is to further relax (10) by dropping some constraints. For example, if we remove the nonnegative requirement on the elements of $Z$, then we obtain the following simple SDP problem

$$\begin{aligned} \min \quad & \mathrm{Tr}\left(\mathrm{W_S W_S^T (I - Z)}\right) \\ & Ze = e, \mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}, \\ & I \succeq Z \succeq 0. \end{aligned} \tag{11}$$

The above problem can be equivalently stated as

$$\begin{aligned} \max \quad & \mathrm{Tr}\left(\mathrm{W_S W_S^T Z}\right) \\ & Ze = e, \mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}, \\ & I \succeq Z \succeq 0. \end{aligned} \tag{12}$$

In the sequel we discuss how to solve (12). Note that if $Z$ is a feasible solution for (12), then we have

$$\frac{1}{\sqrt{n}} Ze = \frac{1}{\sqrt{n}} e,$$

which implies $\frac{1}{\sqrt{n}}e$ is an eigenvector of $Z$ with eigenvalue 1. Therefore, we can write any feasible solution of (12) $Z$ as

$$Z = Q\Gamma Q^T + \frac{1}{n} ee^T,$$

where $Q \in \Re^{n\times(n-1)}$ is a matrix satisfying the condition:

**C.1** The matrix $[Q : \frac{1}{\sqrt{n}}e]$ is orthogonal,

and $\Gamma = \mathrm{diag}\,(\gamma_1, \cdots, \gamma_{n-1})$ is a nonnegative diagonal matrix. It follows

$$k - 1 = \mathrm{Tr}\,(\mathrm{Z}) - 1 = \mathrm{Tr}\left(\mathrm{Q\Gamma Q^T}\right) = \mathrm{Tr}\left(\mathrm{Q^T Q \Gamma}\right) = \mathrm{Tr}\,(\Gamma)\,.$$

Therefore, we can reduce (12) to

$$\begin{aligned} \max \quad & \mathrm{Tr}\left(\mathrm{W_S W_S^T Q\Gamma Q^T}\right) = \mathrm{Tr}\left(\mathrm{Q^T W_S W_S^T Q \Gamma}\right) \\ & \mathrm{Tr}\,(\Gamma) = \mathrm{k} - 1, \\ & I_{n-1} \succeq \Gamma \succeq 0. \end{aligned} \tag{13}$$

Let $\lambda_1(Q^T W_S W_S^T Q), \cdots, \lambda_{n-1}(Q^T W_S W_S^T Q)$ be the eigenvalues of the matrix $Q^T W_S W_S^T Q$ listed in the order of decreasing values. The optimal solution of (13) can be achieved if and only if

$$\mathrm{Tr}\left(\mathrm{Q^T W_S W_S^T Q \Gamma}\right) = \sum_{\mathrm{i=1}}^{\mathrm{k-1}} \lambda_{\mathrm{i}}(\mathrm{Q^T W_S W_S^T Q}).$$

Note that for any matrix $Q$ satisfying Condition C.1, the summation of the first $k-1$ largest eigenvalues of the matrix $Q^T W_S W_S^T Q$ are independent of the choice of $Q$. This gives us an easy way to solve (13) and correspondingly (12). The algorithmic scheme for solving (12) can be described as follows:

**Algorithm:**

**Step 1:** Choose a matrix $Q$ satisfying C.1,

**Step 2:** Use singular value decomposition method to compute the first $k-1$ largest eigenvalues of the matrix $Q^T W_S W_S^T Q$ and their corresponding eigenvectors $v_1, \cdots, v_{k-1}$,

**Step 3:** Set

$$Z = \frac{1}{n} e^T e + \sum_{i=1}^{k-1} (Qv_i)^T Qv_i.$$

It should be mentioned that if $k = 2$, then Step 2 in the above algorithm uses the eigenvector corresponding to the largest eigenvalue of $Q^T W_S W_S^T Q$. This eigenvector has an important role in Shi and Malik' work [25] (See also [28]) for image segmentation where the clustering problem with $k = 2$ was discussed.

### 3.2.2 LP Relaxation

In this subsection, we propose an LP relaxation for (7). First we observe that if $\mathbf{s}_i$ and $\mathbf{s}_j$, $\mathbf{s}_j$ nd $\mathbf{s}_k$ belong to the same clusters, then $\mathbf{s}_i$ and $\mathbf{s}_k$ belong to the same cluster. In such a circumstance, from the definition of the matrix $Z$ we can conclude that

$$Z_{ij} = Z_{jk} = Z_{ik} = Z_{ii} = Z_{jj} = Z_{kk}.$$

Such a relationship can be partially characterized by the following inequality

$$Z_{ij} + Z_{ik} \leq Z_{ii} + Z_{jk}.$$

Correspondingly, we can define a metric polyhedron MET[3] by

$$\mathrm{MET} = \{Z = [z_{ij}] : z_{ij} \leq z_{ii}, \quad z_{ij} + z_{ik} \leq z_{ii} + z_{jk}\}.$$

Therefore, we have the following new model

$$(14) \qquad \begin{aligned} \min \ & \mathrm{Tr}\left(\mathrm{W_S W_S^T (I - Z)}\right) \\ & Ze = e, \mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}, \\ & Z \geq 0, \\ & Z \in \mathrm{MET}. \end{aligned}$$

[3]A similar polyhedron MET had been used by Karisch and Rendl, Leisser and Rendl in their works [16, 17] on graph partitioning. We changed slightly the definition of MET in [17] to adapt to our problem.

If the optimal solution of (14) is not a feasible solution of (7), then we need to refer to the rounding procedure that we described earlier to extract a feasible solution for (7).

Solving (14) directly for large-size data set is clearly unpractical due to the huge amount ($O(n^3)$) of constraints. In what follows we report some preliminary numerical results for small-size data set. Our implementation is done on an IBM RS-6000 workstation and the package CPLEX 7.1 with AMPL interface is used to solve the LP model (14).

The first data set we use to test our algorithm is the Soybean data (small) from the UCI Machine Learning Repository [4], see also [21]. This data set has 47 instances and each instance has 35 normalized attributes. It is known this data set has 4 clusters. As shown by the following table, for $k$ from 2 to 4, we found the exact clusters by solving (14).

**The Soybean data**

| k | Objective | CPU time(s) |
|---|---|---|
| 2 | 404.4593 | 4.26 |
| 3 | 215.2593 | 1.51 |
| 4 | 205.9637 | 1.68 |

The second test set is the Ruspini data set from [24]. This data set, consisting of 75 points in $\Re^2$ with four groups, is popular for illustrating clustering techniques [15]. The numerical result is listed as follows:

**The Ruspini's data**

| k | Objective | CPU time(s) |
|---|---|---|
| 2 | 893380 | 27.81 |
| 3 | 510630 | 66.58 |
| 4 | 12881 | 7.22 |
| 5 | 10127 | 9.47 |

We observed that in our experiments, for all cases $k = 2, \cdots, 5$, the solution of (14) is not feasible for (7). However, the resulting matrix is quite close to a feasible solution of (7). Therefore, we use the classical K-means to get the final clusters. After a few iterations, the algorithm terminated and reported the numerical results that match the best known results in the literature for the same problem.

The third test set is the Späth's postal zones data [26]. This data set contains 89 entities and each entity has 3 features. Correspondingly, we transform all the entities into points in $\Re^3$. It is known that the data set has 7 groups. In all cases that $k$ runs from 2 to 9, we were able to find the exact solution of (7) via solving (14).

[4] *http://www.ics.uci.edu/ mlearn/MLRepository.html*

**The Spath's Postal Zone data**

| k | Objective | CPU time(s) |
|---|---|---|
| 2 | $6.0255 * 10^{11}$ | 283.26 |
| 3 | $2.9451 * 10^{11}$ | 418.07 |
| 4 | $1.0447 * 10^{11}$ | 99.54 |
| 5 | $5.9761 * 10^{10}$ | 60.67 |
| 6 | $3.5908 * 10^{10}$ | 52.55 |
| 7 | $2.1983 * 10^{10}$ | 61.78 |
| 8 | $1.3385 * 10^{10}$ | 26.91 |
| 9 | $7.8044 * 10^{9}$ | 18.04 |

It is worthwhile mentioning that, as shown in the tables, the running time of the algorithm does not increase as the cluster number $k$ increases. Actually, from a theoretical viewpoint, the complexity of the algorithm for solving (14) is independent of $k$. This indicates our algorithm is scalable to large data set, while how to solve (14) efficiently still remains a challenge. In contrast, the complexity of the approximation algorithms in [22] increases with respect to $k$.

## 4 Relations to Other Clustering Methods

In the previous sections, we proposed and analyzed the 0-1 SDP model for MSSC. In this section, we consider the more general 0-1 SDP model for clustering

$$(15) \qquad \begin{aligned} \max \ & \mathrm{Tr}\,(\mathrm{WZ}) \\ & Ze = e, \mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}, \\ & Z \geq 0, Z^2 = Z, Z = Z^T, \end{aligned}$$

where $W$ is the so-called affinity matrix whose entries represent the similarities or closeness among the entities in the data set. In the MSSC model, we use the geometric distance between two points to characterize the similarity between them. In this case, we have $W_{ij} = \mathbf{s}_i^T \mathbf{s}_j$. However, we can also use a general function $\phi(\mathbf{s}_i, \mathbf{s}_j)$ to describe the similarity relationship between $\mathbf{s}_i$ and $\mathbf{s}_j$. For example, let us choose

$$(16) \qquad W_{ij} = \phi(\mathbf{s}_i, \mathbf{s}_j) = \exp^{-\frac{\|\mathbf{s}_i - \mathbf{s}_j\|^2}{\sigma}}, \quad \sigma > 0.$$

In order to apply the classical K-means algorithm to (15), we first use the singular eigenvalue decomposition method to decompose the matrix $W$ into the product of two matrices, i.e., $W = U^T U$. In this case, each column of $U$ can be cast as a point in a suitable space. Then, we can apply the classical K-means method for MSSC model to solving problem (15). This is exactly the procedure what the recently proposed spectral clustering follows. However, now we can interpret the spectral clustering as a variant of MSSC in a different kernel space. It is worthwhile mentioning that certain variants of K-means can be adapted to solve (15) directly without using the SVD decomposition of the affinity matrix.

We note that recently, the k-ways normalized cut and spectral clustering received much attention in the machine learning community, and many interesting results about these two approaches have been reported [7, 20, 23, 25, 28, 29, 30]. In particular, Zha et'al [30] discussed the links between spectral relaxation and K-means. Similar ideas was also used in [23]. An SDP relaxation for normalized k-cut was discussed [29]. The relaxed SDP in [29] takes a form quite close to (10). As we pointed out in Section 3, the main difference between the relaxed model in [29] and (10) lies in the constraint.

In fact, with a closer look at the model for normalized k-cut in [29], one can find that it is a slight variant of the model (15). To see this, let us recall the exact model for normalized k-cut [29]. Let $W$ be the affinity matrix defined by (16) and $X$ be the assignment matrix in the set $\mathcal{F}_k$ defined by

$$\mathcal{F}_k = \{X : Xe^k = e^n, x_{ij} \in \{0,1\}\}.$$

Let $d = We^n$ and $D = \operatorname{diag}(d)$. The exact model for normalized k-cut in [29] can be rewritten as

$$\max_{X\in\mathcal{F}_k} \ \mathrm{Tr}\left((\mathrm{X^T D X})^{-1}\mathrm{X^T W X}\right) \tag{17}$$

If we define

$$Z = D^{\frac{1}{2}} X (X^T D X)^{-1} X^T D^{\frac{1}{2}},$$

then we have

$$Z^2 = Z, Z^T = Z, Z \geq 0, Zd^{\frac{1}{2}} = d^{\frac{1}{2}}.$$

Following a similar process as in the proof of Theorem 2.2, we can show that the model (17) equals to the following 0-1 SDP:

$$\begin{aligned} \max \ & \mathrm{Tr}\left(\mathrm{D}^{-\frac{1}{2}}\mathrm{W}\mathrm{D}^{-\frac{1}{2}}\mathrm{Z}\right) \\ & Zd^{\frac{1}{2}} = d^{\frac{1}{2}}, \mathrm{Tr}\,(\mathrm{Z}) = \mathrm{k}, \\ & Z \geq 0, Z^2 = Z, Z = Z^T. \end{aligned} \tag{18}$$

The only difference between (15) and (18) is the introduction of the scaling matrix $D$. However, our new unified model (15) provides more insight for clustering problem and opens new avenues for designing new efficient clustering methods. It is also interesting to note that when we use SDP relaxation to solve (15), the constraint $Z \preceq I$ can be waived without any influence on the solution, while such a constraint should be kept in the SDP relaxation for (18). This will definitely impact the numerical efficiency of the approach. It will be helpful to compare these two models in real application to see what is the role of the scaling matrix $D$.

## 5 Conclusions

In this paper, we reformulated the classical MSSC as a 0-1 SDP. Our new model not only provides a unified framework for several existing clustering approaches, but also opens new avenues for clustering. Several LP/SDP relaxations are suggested to attack the underlying 0-1 SDP. Preliminary numerical tests indicate that these approaches are feasible, and have a lot of potential for further improvement.

Several important issues regarding the new framework remain open. The first is how to estimate the approximate rate of the approximation solution obtained from the relaxed LP/SDP problems. Secondly, the issue of how to design a rounding procedure without using the classical K-means heuristics to extract a feasible solution deserves further study. Thirdly, for specific clustering problem, how to choose a suitable affinity matrix, or in other words, how to find a suitable kernel space needs to be investigated. The last, but also the most important issue, is to develop efficient optimization algorithms for solving the relaxed problems so that these techniques can be applied to large size data set. We hope future study can help us to address these questions.

**Acknowledgement** The authors thank the two anonymous referees for their useful comments.

## References

[1] Agarwal, P.K. and Procopiuc. (2002). Exact and approximation algorithms for clustering. *Algorithmica*, 33, 201-226.

[2] Bradley,P.S., Fayyad, U.M., and Mangasarian, O.L.(1999). Mathematical Programming for data mining: formulations and challenges. *Informs J. Comput.,* 11, 217-238.

[3] Du Merle, O., Hansen, P., Jaumard, B. and Mladenović, N. (2000). An interior-point algorithm for minimum sum of squares clustering. *SIAM J. Sci. Comput.,* 21, 1485-1505.

[4] Ghosh J.(2003). Scalable Clustering. In N. Ye, Editor, The Handbook of Data Mining, Lawrence Erlbaum Associate, Inc, pp. 247-277.

[5] Goemans, M.X. (1997). Semidefinite programming in combinatorial optimization. *Mathematical Programming.* 79, 143–161.

[6] Gordon, A.D. and Henderson, J.T. (1977). Al algorithm for Euclidean sum of squares classification. *Biometrics.* 33, 355-362.

[7] Gu, M., Zha, H., Ding, C., He, X. and Simon, H. (2001). Spectral relaxation models and structure analysis for k-way graph Clustering and bi-clustering. Penn State Univ Tech Report.

[8] Hansen, P. & Jaumard B. (1997). Cluster analysis and mathematical programming. *Math. Programming*, 79(B), 191-215.

[9] Hansen, P., Jaumard, B. and Mladenović, N. (1998). Minumum sum of squares clustering in a low dimensional space. *J. Classification*, 15, 37-55.

[10] Hasegawa, S., Imai, H., Inaba, M., Katoh, N. and Nakano, J. (1993). Efficient algorithms for variance-based k-clustering. In *Proc. First Pacific Conf. Comput. Graphics Appl., Seoul, Korea.* 1, 75-89. World Scientific. Singapore.

[11] Howard, H.(1966). Classifying a population into homogeneous groups. In Lawrence, J.R. Eds., *Opertational Research in Social Science*, Tavistock Publ., London.

[12] Jain, A.K., & Dubes, R.C. (1988). *Algorithms for clustering data.* Englewood Cliffs, NJ: Prentice Hall.

[13] Jain, A.K., Murty, M.N. and Flynn, P.J. (1999). Data clustering: A review. *ACM Computing Surveys*, 31, 264-323.

[14] Jancey, R.C.(1966). Multidimensional group analysis. *Australian J. Botany*, 14, 127-130.

[15] Kaufman, L. and Peter Rousseeuw, P. (1990). Finding Groups in Data, an Introduction to Cluster Analysis, John Wiley.

[16] Karisch, S.E. and Rendl, F. (1998). Semidefinite programming and graph equipartition. *Fields Institute Communications.* 18, 77-95.

[17] Leisser, A. and Rendl, F. (2003). Graph partitioning using linear and semidefinite programming. *Mathematical Programming (B)*, 95,91-101.

[18] McQueen, J.(1967). Some methods for classification and analysis of multivariate observations. *Computer and Chemistry*, 4, 257-272.

[19] Mangasarian, O.L. (1997). Mathematical programming in data mining. *Data Min. Knowl. Discov.,* 1, 183-201.

[20] Meila, M. and Shi, J. (2001). A random walks view of spectral segmentation. Int'l Workshop on AI & Stat.

[21] Michalski, R.S. and Chilausky, R.L. (1980a). Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. *International Journal of Policy Analysis and Information Systems*, 4(2), 125-161.

[22] Matousek, J. (2000). On approximate geometric k-clustering. *Discrete Comput. Geom.*, 24, 61-84.

[23] Ng, A.Y., Jordan, M.I. and Weiss, Y. (2001). On spectral clustering: Analysis and an algorithm. *Proc. Neural Info. Processing Systems, NIPS*, 14.

[24] Ruspini, E.H. (1970). Numerical methods for fuzzy clustering. *Inform. Sci.*, 2, 319-350.

[25] Shi,J. and Malik, J. (2000). Normalized cuts and image segmentation. *IEEE. Trans. on Pattern Analysis and Machine Intelligence*, 22, 888-905.

[26] Späth, H. (1980). *Algorithms for Data Reduction and Classification of Objects*, John Wiley & Sons, Ellis Horwood Ltd.

[27] Ward, J.H. (1963). Hierarchical grouping to optimize an objective function. *J. Amer. Statist. Assoc.,* 58, 236-244.

[28] Weiss, Y. (1999). Segmentation using eigenvectors: a unifying view. *Proceedings IEEE International Conference on Computer Vision*, 975-982.

[29] Xing, E.P. and Jordan, M.I. (2003). On semidefinite relaxation for normalized k-cut and connections to spectral clustering. Tech Report CSD-03-1265, UC Berkeley.

[30] Zha, H., Ding, C., Gu, M., He, X. and Simon, H. (2002). Spectral Relaxation for K-means Clustering. In Dietterich, T., Becker, S. and Ghahramani, Z. Eds., *Advances in Neural Information Processing Systems 14*, pp. 1057-1064. MIT Press.