Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2012, Article ID 764760, 12 pages
doi:10.1155/2012/764760

Research Article

An Efficient Approach to Solve the Large-Scale Semidefinite Programming Problems

Yongbin Zheng,[1] Yuzhuang Yan,[1] Sheng Liu,[2] Xinsheng Huang,[1] and Wanying Xu[1]

[1] *College of Mechatronics and Automation, National University of Defense Technology, Changsha 410073, China*
[2] *College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China*

Correspondence should be addressed to Yongbin Zheng, zybnudt@nudt.edu.cn

Received 21 March 2011; Accepted 17 April 2011

Academic Editor: Shengyong Chen

Solving the large-scale problems with semidefinite programming (SDP) constraints is of great importance in modeling and model reduction of complex system, dynamical system, optimal control, computer vision, and machine learning. However, existing SDP solvers are of large complexities and thus unavailable to deal with large-scale problems. In this paper, we solve SDP using matrix generation, which is an extension of the classical column generation. The exponentiated gradient algorithm is also used to solve the special structure subproblem of matrix generation. The numerical experiments show that our approach is efficient and scales very well with the problem dimension. Furthermore, the proposed algorithm is applied for a clustering problem. The experimental results on real datasets imply that the proposed approach outperforms the traditional interior-point SDP solvers in terms of efficiency and scalability.

1. Introduction

Semidefinite programming (SDP) is a technique widely used in modeling of complex systems and some important issues in computer vision and machine learning. Examples of application include model reduction [1], modeling of nonlinear systems [2, 3], optimal control [4], clustering [5–7], robust Euclidean embedding [8], kernel matrix learning [9], and metric learning [10]. It can be seen in [5, 7] that SDP relaxation can produce more accurate estimates than spectral methods. However, the SDP optimization suffers extremely high time complexity. Current SDP solvers utilizing interior-point methods scale as $O(n^{4.5})$ or worse

[1, 6, 10], where n is the number of rows (or columns) of the semidefinite matrix. As a result, SDP can only run on small datasets. It is meaningful to propose a scalable SDP solver.

Inspired by column generation [11], which is a classical technology in optimization literature for solving large scale problem, Shen et al. proposed matrix generation in [10] for solving a semidefinite metric learning problem. By using matrix generation, the particular semidefinite program in metric learning was converted into a sequence of linear programs, and it is possible to use the well-developed linear programming technology.

In this paper, we propose the matrix generation-based iteration approach to solve general SDP optimization problems. At each iteration, the exponentiated gradient (EG) algorithm [12] is applied to solve the special structure subproblem. Experiments are also carried out on some real datasets to evaluate the proposed method's efficacy and efficiency.

The method proposed here can be seen as an extension of column generation to solve SDP problems. The proposed matrix generation method also has the drawback of column generation. It converges slowly of one wants to achieve high accuracy. This is the so-called "tailing effect". In practice, for many applications, we do not need a very accurate solution. Typically, a moderately accurate solution suffices. On the other hand, the proposed method can also be considered as a generalization of EG to the matrix case. EG is used to solve problems with a vector optimization variable, and the vector must be on a simplex. Here, we generalize EG in the sense that the proposed method solves optimization with a semidefinite matrix whose eigenvalues are on a simplex.

We present our main results in the next section.

2. The Algorithm

2.1. Notation

The following notations will be used throughout the paper:

(i) a bold lower-case letter ($\mathbf{x}$): a column vector,

(ii) an upper-case letter (X): a matrix,

(iii) $A \succcurlyeq 0$: a positive semidefinite (p.s.d.) matrix,

(iv) $\mathbf{a} \succcurlyeq \mathbf{b}$: the component-wise inequality between two vectors,

(v) $\mathbb{R}^{m\times n}$: the vector space of real matrices of size $m \times n$,

(vi) $\mathbb{S}$: the space of real matrices,

(vii) $\mathbb{S}^n$: the space of symmetric matrices of size $n \times n$,

(viii) $\mathbb{S}^n_+$: the space of symmetric positive semidefinite matrices of size $n \times n$,

(ix) $\langle A, B\rangle = \mathbf{Tr}(A^\top B)$: the inner product defined on the above spaces,

(x) $\mathbf{Tr}(\cdot)$: the trace of a matrix,

(xi) $\Delta_n := \mathbf{x} \in \mathbb{R}^n \mid \mathbf{x} \succcurlyeq 0,\ \mathbf{1}^\top\mathbf{x} = 1$: the n-dimensional simplex set for vectors,

(xii) $\mathcal{D}_n := \{X \in \mathbb{S}^n_+ \mid X \succcurlyeq 0, \mathbf{Tr}(X) = 1\}$: the density matrix set,

(xiii) $\mathcal{Q}_n := \{X \in \mathbb{S}^n_+ \mid\ X \succcurlyeq 0, \mathbf{Tr}(X) = 1,\ \mathbf{rank}(X) = 1\}$: the dyad set of matrices.

2.2. Solving Optimization Problem with SDP Constraints Using Matrix Generation

We are interested in the following general convex problem:

$$\min_X f(X), \quad \text{s.t. } X \in \mathcal{P}_n. \tag{2.1}$$

Here, $f(\cdot)$ is a convex function defined in $\mathbb{S}_+$.

We need to extend the Fenchel conjugate [13] to matrices. For a function $f(\cdot) : \mathbb{R}^n \longrightarrow (-\infty, \infty]$, the Fenchel conjugate $f^*(\cdot):\mathbb{R}^n \longrightarrow (-\infty, \infty]$ is the lower semicontinuous and convex function

$$f^*(\mathbf{u}) = \sup_{\mathbf{x}\in\mathbb{R}^n} \left\{ \mathbf{u}^\top \mathbf{x} - f(\mathbf{x}) \right\}. \tag{2.2}$$

Now, we impose the following two conditions on $f(\cdot)$:

(i) $f(\cdot)$ is differentiable everywhere,

(ii) the gradient $f'(\mathbf{x})$ is continuous and monotonically increasing at each direction.

In other words, $f(\cdot)$ is differentiable and strictly convex. In this case, the Fenchel conjugate is also called Legendre transform and has the following important result:

$$\mathbf{u} = f'(\mathbf{x}). \tag{2.3}$$

By analogy, we can define a Fenchel conjugate for a matrix

$$f^*(U) = \sup_{X\in\mathbb{S}} \{ \langle U, X \rangle - f(X) \}, \tag{2.4}$$

as in the vector space $\mathbb{R}^n$. $f^*(\cdot)$ defined in (2.4) must be lower semicontinuous and convex, because $f^*(\cdot)$ is a supremum of linear continuous functions of U.

We are ready to reformulate our problem (2.1). It is proved in [10] that a p.s.d. matrix can always be decomposed as a linear convex combination of a set of rank-one matrices, we decompose $X \in \mathbb{S}_+$ into a convex combination of a set of dyads

$$X = \sum_{j=1}^{J} \theta_j Z_j, \tag{2.5}$$

with $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_J]^\top \in \Delta_J$, $Z_j \in Q_n$, for all j. So, we can write (2.1) into

$$\begin{aligned} \min_{\boldsymbol{\theta}} \ & f(X), \quad \text{s.t. } X = \sum_{j=1}^{J} \theta_j Z_j, \\ & \boldsymbol{\theta} \succcurlyeq 0, \qquad 1^\top \boldsymbol{\theta} = 1, \\ & Z_j \in Q_n, \quad \forall j. \end{aligned} \tag{2.6}$$

Note that here, $\boldsymbol{\theta}$ and X are the optimization variables. Although X is redundant in (2.6), we need to keep it to arrive at the important Lagrange dual problem later.

This above problems is still very hard to solve, since it has nonconvex rank constraints, and the variable J is indefinite, because there are an indefinite number of rank-one matrices. However, if we somehow know matrices Z_j $(j = 1, \ldots)$ a priori, then all the constraints imposed on Z_j $(j = 1, \ldots)$ can be dropped, and the problem becomes a linear program.

We apply matrix generation to solve the problem. Matrix generation is an extension of column generation to nonpolyhedral semidefinite constraints for solving difficult large-scale optimization problems [10]. It is a method to avoid considering all variables of a problem explicitly, and only a small subset of the entire variable set is considered. Once the problem with the small subset variables is solved, the question of if there are any other variables that can be included to improve the solution should be answered. For convex programs, the primal variables correspond to the dual constraints. The dual can be used to solve the above subproblem.

The Lagrangian is

$$L(\theta, X, U, t, \mathbf{p}) = f(X) - \left\langle U, X - \sum_{j=1}^{J} \theta_j Z_j \right\rangle - \mathbf{p}^\top \boldsymbol{\theta} - t\left(\mathbf{1}^\top \boldsymbol{\theta} - 1\right), \tag{2.7}$$

with $\mathbf{p} \succcurlyeq 0$, and the dual is

$$\inf_{\boldsymbol{\theta}, X} L = \inf_{X}\left(f(X) - \langle U, X\rangle\right) + \underbrace{(\boldsymbol{\varphi} - \mathbf{p} - t\mathbf{1})^\top \boldsymbol{\theta}}_{\text{must be } \mathbf{0}} + t = t - f^*(U), \tag{2.8}$$

where we have defined $\boldsymbol{\varphi}$ with $\varphi_j = \langle U, Z_j\rangle$, $j = 1 \cdots J$. The dual problem of (2.6) is

$$\max_{t, U} t - f^*(U), \quad \text{s.t. } \boldsymbol{\varphi} \succcurlyeq t\mathbf{1}. \tag{2.9}$$

Essentially the dual is

$$\max_{t, U} t - f^*(U), \quad \text{s.t. } \langle U, Z_j\rangle \geq t, \ \forall j = 1 \cdots J. \tag{2.10}$$

We now only consider a small subset of the variables in the primal; that is, only a subset of Z (denoted by $\overline{Z}$) is used. The LP solved using $\overline{Z}$ is usually termed restricted master problem (RMP). Because the primal variables correspond to the dual constraints, solving RMP is equivalent to solve a relaxed version of the dual problem [10]. If all the constraints that we have not added to the dual problem are kept, solving the restricted problem is equivalent to solve the original problem. Otherwise, if there exists at least one constraint that is violated, the violated constraints correspond to variables in primal that are not in RMP. Adding these variables to RMP leads to a new RMP problem that needs to be reoptimized. Here, we have a iterative algorithm that either finds a new Z' such that

$$\langle U, Z'\rangle < t, \tag{2.11}$$

where t is the solution of the current restricted problem, or it guarantees that such a Z' does not exist. To make convergence fast, we find the one by solving the following optimization problem:

$$Z' = \arg\min_{Z}\left\{\langle U, Z'\rangle\right\}, \quad \text{s.t. } Z \in Q_n. \tag{2.12}$$

The optimal value Z' is exactly $vv^\top$, where v is the eigenvector of U that corresponds to the smallest eigenvalue. For the time being, Z' is added to the dyads, and the only variable to estimate is $\boldsymbol{\theta}$. The primal can be writhen as

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}), \quad \text{s.t. } \boldsymbol{\theta} \in \Delta_J. \tag{2.13}$$

We generalize the EG algorithm to the matrix case and use it to solve the above problem.

At optimality, because of (2.3), we have the connection between the primal and dual variables

$$U^\star = f'(X^\star). \tag{2.14}$$

We also know that at least one of the dual constraints takes the equality at optimality, which enable us to obtain $t^\star$

$$t^\star = \min_{j=1\cdots J} \{\langle U^\star, Z_j \rangle\}, \tag{2.15}$$

$t^\star$ can be used as the iteration stopping criteria.

2.3. The Algorithm Implementation

2.3.1. The Matrix Generation

We propose the implementation of the algorithm for the general convex problem (2.1). To be more general, we extend the SDP constraint $X \in \mathcal{P}_n = \{X \in \mathbb{S}_+^n \mid X \succcurlyeq 0, \mathbf{Tr}(X) = 1\}$ to $X \in \{X \in \mathbb{S}_+^n \mid X \succcurlyeq 0, \mathbf{Tr}(X) = k\}$, as shown in(2.16)

$$\min_{X} f(X), \quad \text{s.t. } X \succcurlyeq 0, \quad \mathbf{Tr}(X) = k, \tag{2.16}$$

where k is a positive integer.

The detailed implementation of the algorithm is proposed in Algorithm 1. We can observe, at each iteration, that a new matrix is generated, hence the algorithm is named matrix generation. Besides, at each iteration, only the most significant eigenvalue and its corresponding eigenvector are needed. This makes the algorithm efficient.

2.3.2. The EG Algorithm

In [12], it is shown that EG style updates can converge very quickly compared to other methods. It has been successfully applied to structured prediction problems such as structured support vector machines and conditional random field. We generalize the EG algorithm to matrix to solve the special structure convex optimization problem in MG algorithm. We first give a brief introduction to the EG algorithm [12, 14, 15]. The EG algorithm efficiently solves the convex optimization problem [16]

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}), \quad \text{s.t. } \boldsymbol{\theta} = [\theta_1, \ldots, \theta_J]^\top \in \Delta_J = \left\{\boldsymbol{\theta} \in R^J : \mathbf{1}^\top \boldsymbol{\theta} = k, \boldsymbol{\theta} \succeq 0\right\}, \tag{2.17}$$

Initialize:
(i) The maximum number of integrations $J_{\max}$.
(ii) The pre-set tolerance value ε (e.g., 10^{-5}).
(iii) $t_0^* = 0$.
Iteration 1:
(1) randomly select $Z_1 \in Q_n$; $\boldsymbol{\theta} = k$.
(2) then compute: $X = \boldsymbol{\theta} Z_1$, $U^* = f'(X)$, $t_1^* = \langle U, Z_1 \rangle$.
While *iteration* $J = 2, 3, \ldots, J_{\max}$ **do**
(1) If abs$(t_J^* - t_{J-1}^*) < \varepsilon$ then break (problem solved);
(2) Find a new Z_J by finding the eigenvector v corresponding to the smallest eigenvalue of U^* (v is normalized by $v \longleftarrow (v/\|v\|_2)$ and $Z_J = v * v^\top$).
(3) Find a new $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_J]^\top \in \Delta_J = \{\boldsymbol{\theta} \in R^J : \mathbf{1}^\top \boldsymbol{\theta} = k, \boldsymbol{\theta} \geq 0\}$, by solving the following spectral structure problem by EG algorithm as described in Section 2.3.3:

$$\min_{\boldsymbol{\theta}} f(\boldsymbol{\theta}), \quad \text{s.t. } \boldsymbol{\theta} \in R^J, \ \mathbf{1}^\top \boldsymbol{\theta} = k, \ \boldsymbol{\theta} \geq 0$$

where $f(\boldsymbol{\theta})$ is obtained from $f(X)$ by substituting X with $\sum_{j=1}^{J} \theta_j Z_j$
(4) Compute $X = \sum_{j=1}^{J} \theta_j Z_j$, $U^* = f'(X)$, $t_J^* = \min\{\langle U^*, Z_J \rangle\}$
Output: The p.s.d. matrix $X = \sum_{j=1}^{J} \theta_j Z_j$ and minimal value $f(X)$.

Algorithm 1: The matrix generation.

under the assumption that the object function $f(\cdot)$ is a convex Lipschitz continuous function with Lipschitz constant L_f w.r.t. a fixed given norm $\|\cdot\|$. The mathematical definition of L_f is that $|f(\mathbf{x}) - f(\mathbf{z})| \leq L_f \|\mathbf{x} - \mathbf{z}\|$ holds for any $\mathbf{x}, \mathbf{z}$ in the domain of $f(\cdot)$. The detail of the EG algorithm is shown in Algorithm 2.

2.3.3. Evaluation of the Algorithm

In this section, we evaluate the convergence, running speed, and memory consumption of the algorithm by solving

$$\min_X \|K - X\|_F^2, \quad \text{s.t. } X \succcurlyeq 0, \ X\mathbf{1}^\top = \mathbf{1}, \ X = X^\top, \ \mathbf{Tr}(X) = k, \tag{2.18}$$

where $\|X\|_F = \sum_{ij} X_{ij}^2$ is the Frobenius norm, K is a $n \times n$ symmetric p.s.d. matrix, and k is a positive integer. This problem is important in the issue of the affinity matrix normalization of spectral clustering [17].

The above problem is approximate to

$$\min_X \left(\|K - X\|_F^2 + \delta \left\| X\mathbf{1}^\top - \mathbf{1} \right\|_2^2 \right), \quad \text{s.t. } X \succcurlyeq 0, \ \mathbf{Tr}(X) = k, \tag{2.19}$$

where δ is a scalar, for example, $\delta = 10^3$.

Before reporting the results, we compare the proposed algorithm with the convex optimization solver, namely, SDPT3 [18] which is used as internal solvers in the disciplined convex programming software CVX [19, 20]. The CVX toolbox can solve standard problems

Initialize:
(i) $\boldsymbol{\theta}^0 \in$ the interior of $\Delta_J = \{\boldsymbol{\theta} \in R^J : \mathbf{1}^\top \boldsymbol{\theta} = k, \boldsymbol{\theta} \succeq 0\}$.
(ii) The maximum number of integrations $i_{\max}$
While *iteration* $i = 1, 2, \ldots, i_{\max}$ **do**
(1) Generate the sequence $\{\boldsymbol{\theta}^i\}$ with:

$$\boldsymbol{\theta}_j^i = \frac{\boldsymbol{\theta}_j^{i-1} \exp[-\tau_i f_j'(\boldsymbol{\theta}^{i-1})]}{\sum_{j=1}^J \boldsymbol{\theta}_j^{i-1} \exp[-\tau_i f_j'(\boldsymbol{\theta}^{i-1})]}$$

where τ_i is step-size, which can be determined by $\tau_i = (\sqrt{2 \log J}/L_f)(1/\sqrt{i})$ following [15], $f'(\boldsymbol{\theta}) = [f_1'(\boldsymbol{\theta}), \ldots, f_J'(\boldsymbol{\theta})]^T$ is the gradient of $f(\cdot)$.
(2) Stop if some stopping criteria are met.
Output: $\boldsymbol{\theta}^i$ and $f(\boldsymbol{\theta})$.

Algorithm 2: The EG algorithm: solving the special structure problem shown in (2.17).

such as linear programs (LPs), quadratic programs (QPs), second-order cone programs (SOCPs), and semidefinite programs (SDPs).

(1) *Convergence*

We randomly generate a 50×50 p.s.d. matrix K in (2.18); let k be 5. In Figure 1, we plot the optimal value obtained by the proposed algorithm at each iteration (the blue curve). The red dash line shows the ground truth obtained by directly solving the original problem in (2.16) using the CVX toolbox. We can see that the algorithm converges to the near-optimal solutions quickly.

(2) *Running Time*

We solve the problem in (2.16) using the proposed algorithm and the CVX toolbox, respectively, and the running time is shown in Figure 2. The blue curve is the running time of the proposed algorithm versus the matrix dimension, and the red curve is the running time of CVX versus the matrix dimension. We can see that the running time of CVX increases quickly with the matrix dimension and the running time of the proposed algorithm is a fraction of the CVX and scales very well with the matrix dimension. The proposed algorithm has a clear advantage on computational efficiency. One reason is that at each iteration, only the most significant eigenvalue and its corresponding eigenvector are needed in the proposed algorithm.

(3) *Memory Consumption*

Figure 3 shows the memory consumption of the proposed algorithm and the CXV toolbox. The blue curve is the approximate memory consumption of the proposed algorithm versus the matrix dimension, and the red curve is the approximate memory consumption of CVX versus the matrix dimension. It can be seen that the memory consumption of CVX increases quickly with the matrix dimension n and the memory consumption of the proposed algorithm is a fraction of the CVX and scales very well with the dimension.

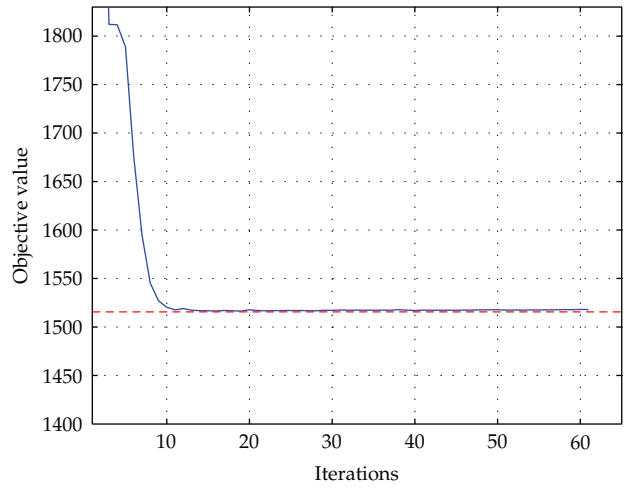

Figure 1: The convergence of the proposed algorithm. The blue curve shows the optimal value obtained by the proposed algorithm at each iteration, and the dash line shows the ground truth obtained by directly solving the original problem in (2.16) using the CVX toolbox.

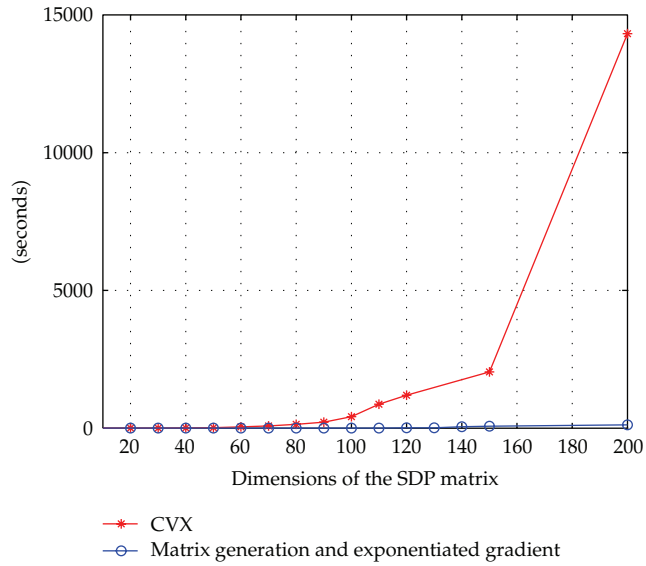

Figure 2: Running time of the proposed algorithm. The blue curve shows the running time of the proposed algorithm versus the matrix dimension, and the red curve shows the running time of CVX versus the matrix dimension.

In summary, the proposed algorithm can converge to the near-optimal solutions quickly and scale very well with the dimension, which are very important for solving large-scale SDP problems. As a result, the proposed method is potential to be applied to computer vision to deal with large matrix of 3D visual data [21] and clustering in data analysis. In next section, we will show how to apply the proposed method to clustering problems.

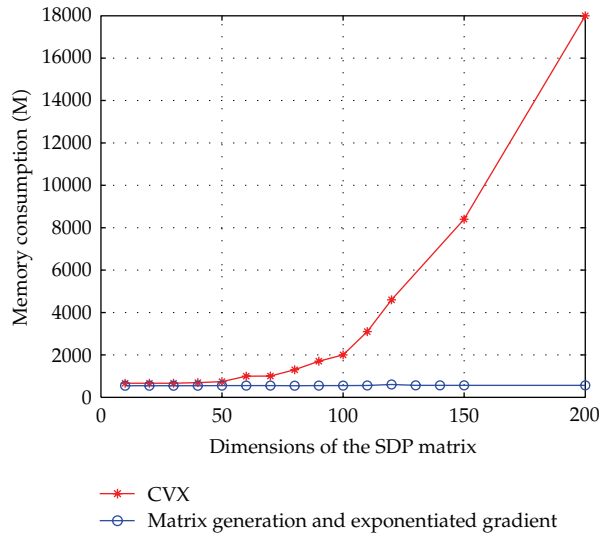

Figure 3: Memory consumption of the proposed algorithm. The blue curve shows the approximate memory consumption of the proposed algorithm versus the matrix dimension, and the red curve shows the approximate memory consumption of CVX versus the matrix dimension.

Table 1: UCI datasets used together with some characteristics and the clustering results achieved using the different methods.

Dataset	k	Size	Dims.	Accuracy			
				K-means	Spectral clustering	SDP relaxation	
						Ours	CVX
Soybean	4	47	36	0.723	0.745	0.766	0.766
Iris	3	150	5	0.893	0.933	0.947	0.953
Wine	3	178	13	0.702	0.725	0.730	0.730
SPECTF heart	2	267	44	0.625	0.802	0.816	0.816

3. The Application

The problem of partitioning data points into a number of distinct sets, known as the clustering problem, is one of the most fundamental and significant topics in data analysis and machine learning. From an optimization viewpoint, clustering aims at the minimal sum-of-squares (MSSC) based on some definition of similarly or distance. But the variable is a 0-1 discrete indicator matrix $Y \in \mathbb{R}^{m\times n}$ (n is the number of samples, and m is the number of clusters), thus the optimization becomes a mixed integer programming with nonlinear objective, which is NP-hard. The computation time of NP-hard problems is too heavy to use in practical engineering applications and should be accelerated by some approximation methods. K-means and spectral clustering are the two most well-known algorithms to approximately solve the clustering problem. However, the former is merely able to achieve local optimum through an indirect way, and the later, which can obtain a global optimum through eigenvalue decomposition, is unfortunately a weak relaxation of the N-cut problem [7]. Recent research of semidefinite programming (SDP) relaxation on clustering proposed that the minimal sum-of-squares (MSSC) [22] is possible to be relaxed as SDP problems, which is more direct than K-means and tighter than spectral clustering [7].

Table 2: Time consumptions (seconds) of the proposed algorithm and CVX on clustering the selected UCI datasets.

	Soybean	Iris	Wine	SPECTF heart
Ours	1.34	2.879	4.22	21.29
CVX	1.21	4.629	6.95	28.00

In this section, we apply the proposed algorithm to solve the SDP relaxed clustering problem (0-1 SDP for clustering) [22]

$$\min_X \mathbf{Tr}\ (W(I-X)), \quad \text{s.t. } X\mathbf{1}=\mathbf{1},\ \mathbf{Tr}(X)=m,\ X\geq 0,\ X\succeq 0, \tag{3.1}$$

where W is the so-called affinity matrix whose entries represent the similarities or closeness among the entities in the dataset, and m is the class number of the dataset. According to [22], it is reasonable to relax the nonnegative constraint in (3.1)

$$\min_X \mathbf{Tr}\ (W(I-X)), \quad \text{s.t. } X\mathbf{1}=\mathbf{1},\ \mathbf{Tr}(X)=m,\ X\succeq 0. \tag{3.2}$$

We solve the problem in (3.2) using the algorithm described in Algorithm 1. To test the algorithm, we carry out some experiments on four real datesets collected from UCI machine learning repository (http://www.ics.uci.edu/~mlearn/MLRepository.html.) The datasets are listed in Table 1 together with some their characteristics. k, Size, and Dim are the class number, the number of instances, and the number of attributes of each dataset, respectively. All the experiments are done by using Matlab on a personal computer with a 2.93 GHz processor and a 2 G memory. The accuracy of clustering each dataset of our algorithm is compared with K-means, spectral clustering, and CVX (here CVX means the SDP problem in (3.2) is solved using the CVX toolbox). The results are shown in Table 1. We can see that

(i) the proposed algorithm performs better than K-means,

(ii) the proposed algorithm achieves higher accuracy than the spectral clustering that is because the proposed algorithm is the SDP relaxation of the clustring problem and is tighter than the spectral relaxation,

(iii) the proposed algorithm is as good as CVX. This means it can solve the SDP problem as accurately as the CVX toolbox.

We also compare the speed of the proposed algorithm with CVX, and the results are shown in Table 2. It shows that the proposed algorithm is much faster than CVX.

As observed from the experiments, it can be found that the proposed algorithm performs very well on these datasets and can efficiently solve the 0-1 SDP problem for clustering. Both speed and accuracy are improved as compared with traditional interior-point SDP solvers. We plan to further extend the approach to more general problems and look for other applications such as modeling [23, 24] and stabilizing some kinds of dynamical systems [25] model reduction.

4. Conclusion

This paper proposed an approach to solve the optimization problem with SDP constraints using matrix generation and exponentiated gradient. The process is faster and more scalable than the traditional SDP approach. The results of numerical experiments show that the method scales very well with the problem dimensions. We also applied the approach to solve the SDP relaxation clustering problem. Experimental results on real datasets show that the algorithm outperforms much in both speed and accuracy as compared with interior-point SDP solvers.

References

[1] A. Sootla, *Model reduction using semidefinite programming*, Licentiate Thesis, Department of Automatic Control, Lund University, Lund, Sweden, 2009.

[2] Z. Mahmood, B. Bond, T. Moselhy, A. Megretski, and L. Daniel, "Passive reduced order modeling of multiport interconnects via semidefinite programming," in *Proceedings of the International Conference on Design, Automation and Test in Europe (DATE '10)*, pp. 622–625, 2010.

[3] S. Y. Chen and Q. Guan, "Parametric shape representation by a deformable NURBS model for cardiac functional measurements," *IEEE Transactions on Biomedical Engineering*, vol. 58, no. 3, part 1, pp. 480–487, 2011.

[4] P. S. Ansell, K. D. Glazebrook, I. Mitrani, and J. Niño-Mora, "Semidefinite programming approach to the optimal control of a single server queueing system with imposed second moment constraints," *Journal of the Operational Research Society*, vol. 50, no. 7, pp. 765–773, 1999.

[5] E. P. Xing and M. I. Jordan, "On semidefinite relaxations for normalized k-cut and connections to spectral clustering," Tech. Rep., University of California at Berkeley, Berkeley, Calif, USA, 2003.

[6] B. Kulis, A. C. Surendran, and J. C. Platt, "Fast low-rank semidefinite programming for embedding and clustering," in *Proceedings of the International Workshop on Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 2007.

[7] T. Zhou and D. Tao, "Fast gradient clustering," in *Proceedings of the Advances in Neural Information Processing Systems, Workshop on Discrete Optimization in Machine Learning*, pp. 1–6, 2009.

[8] L. Cayton and S. Dasgupta, "Robust euclidean embedding," in *Proceedings of the International Conference on Machine Learning*, 2006.

[9] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. El Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semidefinite programming," *Journal of Machine Learning Research (JMLR)*, vol. 5, pp. 27–72, 2003/04.

[10] C. Shen, A. Welsh, and L. Wang, "PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning," in *Proceedings of the Advances in Neural Information Processing Systems*, pp. 1473–1480, Vancouver, Canada, 2008.

[11] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[12] A. Globerson, T. Y. Koo, X. Carreras, and M. Collins, "Exponentiated gradient algorithms for log-linear structured prediction," in *Proceedings of the 24th International Conference on Machine Learning (ICML '07)*, vol. 227, pp. 305–312, 2007.

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

[14] J. Kivinen and M. K. Warmuth, "Additive versus exponentiated gradient updates for linear prediction," *Information and Computation*, vol. 132, no. 1, pp. 1–64, 1997.

[15] A. Beck and M. Teboulle, "Mirror descent and nonlinear projected subgradient methods for convex optimization," *Operations Research Letters*, vol. 31, no. 3, pp. 167–175, 2003.

[16] C. Shen, P. Wang, and H. Li, "LACBoost and FisherBoost: optimally building cascade classifiers," in *Proceedings of the 11th European Conference on Computer Vision (ECCV '10)*, vol. 2 of *Lecture Notes in Computer Science (LNCS) 6312*, pp. 608–621, Springer, Crete Island, Greece, September 2010.

[17] R. Zass and A. Shashua, "Doubly stochastic normalization for spectral clustering," in *Proceedings of theAdvances in Neural Information Processing Systems*, 2006.

[18] R. H. Tütüncü, K. C. Toh, and M. J. Todd, "Solving semidefinite-quadratic-linear programs using SDPT3," *Mathematical Programming*, vol. 95, no. 2, pp. 189–217, 2003.

[19] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 1.21," 2010, http://cvxr.com/cvx.

[20] M. C. Grant and S. P. Boyd, "Graph implementations for nonsmooth convex programs," in *Recent Advances in Learning and Control*, V. Blondel, S. Boyd, and H. Kimura, Eds., vol. 371 of *Lecture Notes in Control and Informmation Sciences*, pp. 95–110, Springer, London, UK, 2008.

[21] S. Y. Chen, Y. F. Li, and J. Zhang, "Vision processing for realtime 3-D data acquisition based on coded structured light," *IEEE Transactions on Image Processing*, vol. 17, no. 2, pp. 167–176, 2008.

[22] J. Peng and Y. Wei, "Approximating *k*-means-type clustering via semidefinite programming," *SIAM Journal on Optimization*, vol. 18, no. 1, pp. 186–205, 2007.

[23] S. Y. Chen and Y. F. Li, "Vision sensor planning for 3-D model acquisition," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 35, no. 5, pp. 894–904, 2005.

[24] B. N. Bond, Z. Mahmood, Y. Li et al., "Compact modeling of nonlinear analog circuits using system identification via semidefinite programming and incremental stability certification," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 29, no. 8, pp. 1149–1162, 2010.

[25] M. Li, S. C. Lim, and S. Chen, "Exact solution of impulse response to a class of fractional oscillators and its stability," *Mathematical Problems in Engineering*, vol. 2011, Article ID 657839, 9 pages, 2011.