Research Reports on Mathematical and Computing Sciences
Series B : Operations Research

Department of Mathematical and Computing Sciences
Tokyo Institute of Technology
2-12-1 Oh-Okayama, Meguro-ku, Tokyo 152, Japan

Exploiting Sparsity in Primal-Dual Interior-Point Methods for Semidefinite Programming.

Katsuki Fujisawa†, Masakazu Kojima⋆, Kazuhide Nakata‡

January 1997, B-324

Abstract. The Helmberg-Rendl-Vanderbei-Wolkowicz/Kojima-Shindoh-Hara/Monteiro and the Nesterov-Todd search directions have been used in many primal-dual interior-point methods for semidefinite programs. This paper proposes an efficient method for computing the two directions when a semidefinite program to be solved is large scale and sparse.

Key words Interior-Point Methods, Semidefinite Programming, Sparsity

† fujisawa@is.titech.ac.jp
⋆ kojima@is.titech.ac.jp
‡ nakata@is.titech.ac.jp

1 Introduction.

This paper deals with two types of search directions which have been used in many primal-dual interior-point methods [1, 2, 6, 7, 8, 9, 11, 12, 13, 14, 16, 17, 19, etc.] for semidefinite programming. The one is the HRVW/KSH/M direction which was independently proposed by two groups of researchers, Helmberg-Rendl-Vanderbei-Wolkowicz [6] and Kojima-Shindoh-Hara [8], and later rediscovered by Monteiro [12] in a new formulation. The other is the NT direction by Nesterov and Todd [13, 14]. The AHO direction given by Alizadeh, Haeberly and Overton [1] is another important direction in primal-dual interior-point methods for semidefinite programming. Global and local convergence of primal-dual interior-point methods using the HRVW/KSH/M, the NT and the AHO directions have been studied extensively in many papers [1, 2, 6, 7, 8, 9, 11, 12, 13, 14, 16, 19, etc.], and some computational results on the three directions have been reported in the papers [1, 2, 6, 17, etc.]. Some computer programs [3, 5, 18] of primal-dual interior-point methods using the three directions are available through the Internet. The purpose of this paper is to present an efficient method for computing the HRVW/KSH/M and the NT directions when a semidefinite program to be solved is large scale and sparse.

Let $R^{n\times n}$ denote the set of all $n \times n$ real matrices. We regard $R^{n\times n}$ the n^2-dimensional Euclidean space. Let $\mathcal{S}^n$ denote the set of all $n \times n$ symmetric real matrices; $\mathcal{S}^n$ forms a $n(n+1)/2$-dimensional linear subspace of $R^{n\times n}$. For each pair of $\boldsymbol{X}$ and $\boldsymbol{Z}$ in $R^{n\times n}$, $\boldsymbol{X} \bullet \boldsymbol{Z}$ stands for the inner product of $\boldsymbol{X}$ and $\boldsymbol{Z}$, *i.e.*, Tr $\boldsymbol{X}^T\boldsymbol{Z}$, the trace of $\boldsymbol{X}^T\boldsymbol{Z}$. We write $\boldsymbol{X} \succ \boldsymbol{O}$ if $\boldsymbol{X} \in \mathcal{S}^n$ is positive definite, *i.e.*, $\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{u} > 0$ for every nonzero $\boldsymbol{u} \in R^n$, and $\boldsymbol{X} \succeq \boldsymbol{O}$ if $\boldsymbol{X} \in \mathcal{S}^n$ is positive semidefinite, *i.e.*, $\boldsymbol{u}^T\boldsymbol{X}\boldsymbol{u} \geq 0$ for every $\boldsymbol{u} \in R^n$. Here $\boldsymbol{O}$ denotes the $n \times n$ zero matrix.

Let $\boldsymbol{C} \in \mathcal{S}^n$, $\boldsymbol{A}_i \in \mathcal{S}^n$ $(1 \leq i \leq m)$ and $a_i \in R$ $(1 \leq i \leq m)$. Consider the semidefinite program and its dual:

$$\left.\begin{array}{lll} \mathcal{P}: & \text{minimize} & \boldsymbol{C} \bullet \boldsymbol{X} \\ & \text{subject to} & \boldsymbol{A}_i \bullet \boldsymbol{X} = a_i \ (1 \leq i \leq m),\ \boldsymbol{X} \succeq \boldsymbol{O}. \\ \mathcal{D}: & \text{maximize} & \sum_{i=1}^{m} a_i y_i \\ & \text{subject to} & \sum_{i=1}^{m} \boldsymbol{A}_i y_i + \boldsymbol{Z} = \boldsymbol{C},\ \boldsymbol{Z} \succeq \boldsymbol{O}. \end{array}\right\} \tag{1}$$

Throughout the paper we assume that the set of $n\times n$ symmetric matrices $\boldsymbol{A}_i$ $(1 \leq i \leq m)$ is linearly independent. This implies that $m \leq n(n+1)/2$. We say that $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ is a feasible solution (an interior-feasible solution, or an optimal solution, respectively.) of the SDP (1) if $\boldsymbol{X}$ is a feasible solution (an interior-feasible solution, i.e., a feasible solution satisfying $\boldsymbol{X} \succ \boldsymbol{O}$, or a minimum solution, respectively) of $\mathcal{P}$ and $(\boldsymbol{y}, \boldsymbol{Z})$ is a feasible solution (an interior-feasible solution, i.e., a feasible solution satisfying $\boldsymbol{Z} \succ \boldsymbol{O}$, or a maximum solution, respectively) of $\mathcal{D}$.

Generic Primal-Dual Interior-Point Method :

Step 0: Set up a stopping criteria, and choose an initial point $(\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$ such that $\boldsymbol{X}^0 \succ \boldsymbol{O}$ and $\boldsymbol{Z}^0 \succ \boldsymbol{O}$. Let $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z}) = (\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$.

Step 1: If the current iterate $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ satisfies the stopping criteria, stop the iteration.

Step 2: Choose a search direction $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$.

Step 3: Choose a primal step length α_p and a dual step length α_d such that

$$\boldsymbol{X} + \alpha_p d\boldsymbol{X} \succ \boldsymbol{O} \text{ and } \boldsymbol{Z} + \alpha_d d\boldsymbol{Z} \succ \boldsymbol{O}.$$

Let

$$\boldsymbol{X} = \boldsymbol{X} + \alpha_p d\boldsymbol{X} \text{ and } (\boldsymbol{y}, \boldsymbol{Z}) = (\boldsymbol{y}, \boldsymbol{Z}) + \alpha_d (d\boldsymbol{y}, d\boldsymbol{Z}).$$

Step 4: Go to Step 1.

Most of the primal-dual interior-point methods developed so far may be obtained if we specify

- a stopping criteria at Step 0,
- an initial point $(\boldsymbol{X}^0, \boldsymbol{y}^0, \boldsymbol{Z}^0)$ at Step 0,
- a search direction $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ at Step 2,
- a primal step length α_p and a dual step length α_d at Step 3

in the generic primal-dual interior-point method described above. In Section 2, we present the HRVW/KSH/M and the NT directions which we employ at Step 3. In Section 3, we propose an efficient method for computing the two directions when some or all of the data matrices $\boldsymbol{A}_i$ $(1 \leq i \leq m)$ are sparse. Section 4 is devoted to computational results on the proposed method.

2 Search Directions.

The HRVW/KSH/M direction is described as a solution $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ of the three types of equations

$$\boldsymbol{A}_i \bullet d\boldsymbol{X} = p_i \ (1 \leq i \leq m), \ d\boldsymbol{X} \in \mathcal{S}^n, \tag{2}$$

$$\sum_{i=1}^{m} \boldsymbol{A}_i dy_i + d\boldsymbol{Z} = \boldsymbol{D}, \ d\boldsymbol{Z} \in \mathcal{S}^n, \tag{3}$$

$$\widehat{d\boldsymbol{X}}\,\boldsymbol{Z} + \boldsymbol{X}\,d\boldsymbol{Z} = \boldsymbol{K}', \ \widehat{d\boldsymbol{X}} \in R^{n\times n}, \ d\boldsymbol{X} = (\widehat{d\boldsymbol{X}} + \widehat{d\boldsymbol{X}}^T)/2. \tag{4}$$

Here we regard $\widehat{d\boldsymbol{X}} \in R^{n\times n}$ as an auxiliary variable matrix, and $p_i \in R$, $\boldsymbol{D} \in \mathcal{S}^n$ and $\boldsymbol{K}' \in R^{n\times n}$ denote a scalar constant, a $n \times n$ constant symmetric matrix, and a $n \times n$ constant matrix, respectively, which are determined by the current point $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ and some other factors. They differ from one method to another. For example, we take

$$p_i = 0 \ (1 \leq i \leq m), \ \boldsymbol{D} = \boldsymbol{O} \text{ and } \boldsymbol{K}' = \beta \frac{\boldsymbol{X} \bullet \boldsymbol{Z}}{n} \boldsymbol{I} - \boldsymbol{X}\boldsymbol{Z} \text{ for some } \beta \in (0, 1)$$

in the feasible-path-following method using the HRVW/KSH/M direction. But their actual values are not relevant in the succeeding discussions. See the papers [1, 2, 6, 7, 8, 9, 11, 12, 16, 17, 19, etc.] for various primal-dual interior-point methods using the HRVW/KSH/M direction. Under the linear independence assumption on the set $\{\boldsymbol{A}_i : 1 \leq i \leq m\}$ of constraint matrices, it is known [8] that for any $\boldsymbol{X} \succ \boldsymbol{O}$, $\boldsymbol{Z} \succ \boldsymbol{O}$, $p_i \in R$ $(1 \leq i \leq m)$, $\boldsymbol{D} \in \mathcal{S}^n$, and $\boldsymbol{K}' \in R^{n\times n}$, the system of equations (2), (3) and (4) has a unique solution $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$.

The NT direction shares the equations (2) and (3) with the HRVW/KSH/M direction, but we replace (4) by

$$d\boldsymbol{X}\boldsymbol{W}^{-1} + \boldsymbol{W}d\boldsymbol{Z} = \boldsymbol{K}", \tag{5}$$

where $\boldsymbol{W} = \boldsymbol{X}^{1/2}(\boldsymbol{X}^{1/2}\boldsymbol{Z}\boldsymbol{X}^{1/2})^{-1/2}\boldsymbol{X}^{1/2} \in \mathcal{S}^n$, and $\boldsymbol{K}" \in R^{n\times n}$ are determined by the current point $(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{Z})$ and some other factors. The NT direction is given as a solution $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ of the system equations (2), (3) and (5). The same comments on the values of $p_i \in R$, $\boldsymbol{D} \in \mathcal{S}^n$ and $\boldsymbol{K}" \in R^{n\times n}$ and on the existence and uniqueness of the direction apply to the NT direction as in the case of the HRVW/KSH/M direction. See the paper [17] for more details.

3 Computing Directions.

Todd-Toh-Tütüncü [17] presented some numerically stable methods for computing the AHO, the NT and the HRVW/KSH/M directions, and reported numerical results showing that the numerical stability depends on how we compute the directions. Alizadeh, Haeberly and Overton [2] also proposed some numerically stable methods for the three directions. We will investigate below how to compute the HRVW/KSH/M and the NT directions. However, the main issue here is not the numerical stability but the computational efficiency when some or all of the data matrices $\boldsymbol{A}_i$ $(1 \le i \le m)$ of the SDP are sparse.

In the HRVW/KSH/M direction case, we reduce the system of equations (2), (3) and (4) to

$$\left.\begin{array}{l} \boldsymbol{B}'d\boldsymbol{y} = \boldsymbol{b}', \\ d\boldsymbol{Z} = \boldsymbol{D} - \sum_{j=1}^{m} \boldsymbol{A}_j dy_j, \\ \widehat{d\boldsymbol{X}} = (\boldsymbol{K}' - \boldsymbol{X}d\boldsymbol{Z})\,\boldsymbol{Z}^{-1},\ d\boldsymbol{X} = (\widehat{d\boldsymbol{X}} + \widehat{d\boldsymbol{X}}^T)/2, \end{array}\right\} \tag{6}$$

where

$$B'_{ij} = \boldsymbol{X}\boldsymbol{A}_i\boldsymbol{Z}^{-1} \bullet \boldsymbol{A}_j \ (1 \le i \le m,\ 1 \le j \le m), \tag{8}'$$

$$b'_i = (\boldsymbol{X}\boldsymbol{D} - \boldsymbol{K}')\boldsymbol{Z}^{-1} \bullet \boldsymbol{A}_i + p_i \ (1 \le i \le m).$$

The $n\times n$ matrix $\boldsymbol{X}$, the $n\times n$ matrix $\boldsymbol{Z}^{-1}$ and the $m\times m$ matrix $\boldsymbol{B}'$ are symmetric, and dense in general even when all $\boldsymbol{A}_i$ $(1 \le i \le m)$ are sparse. Hence solving the system of equations (6) in $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ by using a direct method such as the Cholesky factorization requires $O(m^3) + O(n^3)$ arithmetic operations. On the other hand, if we use the above formulae for the coefficient matrix $\boldsymbol{B}'$ and the right hand side vector $\boldsymbol{b}'$ in a straightforward way, the computation of $\boldsymbol{B}'$ requires $O(mn^3 + m^2n^2)$ arithmetic operations and the computation of $\boldsymbol{b}'$ $O(n^3 + mn^2)$ arithmetic operations, respectively. Therefore computing the coefficient matrix $\boldsymbol{B}'$ is more crucial than computing $\boldsymbol{b}'$ and solving $\boldsymbol{B}'d\boldsymbol{y} = \boldsymbol{b}'$ in the entire computation of the HRVW/KSH/M direction. We will investigate how efficiently we compute the matrix $\boldsymbol{B}'$ when some or all of the matrices $\boldsymbol{A}_i$ $(1 \le i \le m)$ are sparse.

We can reduce the system of equations (2), (3) and (5) describing the NT direction to a similar system of equations as (6):

$$\left.\begin{array}{l} \boldsymbol{B}"d\boldsymbol{y} = \boldsymbol{b}", \\ d\boldsymbol{Z} = \boldsymbol{D} - \sum_{j=1}^{m} \boldsymbol{A}_j dy_j, \\ d\boldsymbol{X} = (\boldsymbol{K}" - \boldsymbol{W}d\boldsymbol{Z})\,\boldsymbol{W}, \end{array}\right\} \tag{7}$$

where

$$B"_{ij} = \boldsymbol{W}\boldsymbol{A}_i\boldsymbol{W} \bullet \boldsymbol{A}_j \ (1 \le i \le m,\ 1 \le j \le m), \tag{8}"$$

$$b"_i = (\boldsymbol{W}\boldsymbol{D} - \boldsymbol{K}")\boldsymbol{W} \bullet \boldsymbol{A}_i + p_i \ (1 \le i \le m).$$

Similar comments apply to the computation of the $m \times m$ symmetric matrix $\boldsymbol{B}"$ in (8)", the computation of the right hand side vector $\boldsymbol{b}"$, and solving the system (7) of equations in the NT direction $(d\boldsymbol{X}, d\boldsymbol{y}, d\boldsymbol{Z})$ as in the case of the HRVW/KSH/M direction above.

The remainder of this section is devoted to an efficient computation of the $m \times m$ symmetric matrices $\boldsymbol{B}'$ and $\boldsymbol{B}"$. To deal with both directions simultaneously, we consider

$$B_{ij} = \boldsymbol{T}\boldsymbol{A}_i\boldsymbol{U} \bullet \boldsymbol{A}_j \tag{8}$$

$(1 \le i \le m,\ 1 \le j \le m)$, where $\boldsymbol{T} \in \mathcal{S}^n$ and $\boldsymbol{U} \in \mathcal{S}^n$, If we take $\boldsymbol{T} = \boldsymbol{X}$ and $\boldsymbol{U} = \boldsymbol{Z}^{-1}$, then $\boldsymbol{B} = \boldsymbol{B}'$. If $\boldsymbol{T} = \boldsymbol{U} = \boldsymbol{W}$ then $\boldsymbol{B} = \boldsymbol{B}"$. Note that $\boldsymbol{B}$ is symmetric; hence we only need to compute the upper triangular part of $\boldsymbol{B}$, i.e., B_{ij} $(1 \le i \le j \le m)$.

There are some factors which take part in the CPU time for computing the matrix $\boldsymbol{B}$ such as multiplications, additions and access time to elements of the sparse data matrices $\boldsymbol{A}_i$ $(1 \le i \le m)$. The number of additions is a similar order as the number of multiplications (see Remark (B) of Section 5 for more details). But access time to elements of the sparse data matrix $\boldsymbol{A}_i$ is much affected by the data structure for $\boldsymbol{A}_i$. First we focus our attention to the number of multiplications, and later we incorporate an overhead of access time to elements of the sparse data matrices $\boldsymbol{A}_i$ $(1 \le i \le m)$ into our consideration.

Let f_i denote the number of nonzero elements in $\boldsymbol{A}_i$. Let $\boldsymbol{\Sigma}$ denote the set of permutations of the indices $1, 2, \ldots, m$ (*i.e.*, one-to-one mappings from $\{1, 2, \ldots, m\}$ onto itself). Each $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$ determines an order in computation of the elements B_{ij} $(1 \le i \le m,\ 1 \le j \le m)$ such that

$$\left.\begin{array}{llll} \overrightarrow{B_{\sigma(1)\sigma(1)}}, & B_{\sigma(1)\sigma(2)}, & \ldots & B_{\sigma(1)\sigma(m)}, \\ & \overrightarrow{B_{\sigma(2)\sigma(2)}}, & \ldots & B_{\sigma(2)\sigma(m)}, \\ & & \overrightarrow{\cdots} & \cdots \\ & & & B_{\sigma(m)\sigma(m)}. \end{array}\right\} \tag{9}$$

Recall that $\boldsymbol{B}$ is symmetric; hence $B_{\sigma(j)\sigma(i)} = B_{\sigma(i)\sigma(j)}$ $(1 \le i < j \le m)$.

Let $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$ and $i \in \{1, 2, \ldots, m\}$ be fixed. We introduce three kinds of formulae for computing $B_{\sigma(i)\sigma(j)}$ $(i \le j \le m)$ below.

$\mathcal{F}$-1: Compute $\boldsymbol{F}_i = \boldsymbol{A}_{\sigma(i)}\boldsymbol{U}$, which requires $nf_{\sigma(i)}$ multiplications, and $\boldsymbol{G}_i = \boldsymbol{T}\boldsymbol{F}_i$, which requires n^3 multiplications. For each $j = i, i+1, \ldots, m$, compute $B_{\sigma(i)\sigma(j)} = \boldsymbol{G}_i \bullet \boldsymbol{A}_{\sigma(j)}$, which requires $f_{\sigma(j)}$ multiplications. Then the total number of multiplications to compute all $B_{\sigma(i)\sigma(j)}$ $(i \le j \le m)$ turns out to be

$$nf_{\sigma(i)} + n^3 + \sum_{i \le j \le m} f_{\sigma(j)}. \tag{10}$$

$\mathcal{F}$-2: Compute $\boldsymbol{F}_i = \boldsymbol{A}_{\sigma(i)}\boldsymbol{U}$, which requires $nf_{\sigma(i)}$ multiplications. For each $j = i, i+1, \ldots, m$, compute

$$B_{\sigma(i)\sigma(j)} = \sum_{\alpha=1}^{n}\sum_{\beta=1}^{n}[\boldsymbol{A}_{\sigma(j)}]_{\alpha\beta}\left(\sum_{\gamma=1}^{n} T_{\alpha\gamma}[\boldsymbol{F}_i]_{\gamma\beta}\right),$$

which requires $(n+1)f_{\sigma(j)}$ multiplications. Then the total number of multiplications to compute all $B_{\sigma(i)\sigma(j)}$ $(i \le j \le m)$ is given by

$$nf_{\sigma(i)} + (n+1)\sum_{i\le j\le m} f_{\sigma(j)}. \tag{11}$$

$\mathcal{F}$-3: For each $j = i, i+1, \ldots, m$, compute

$$B_{\sigma(i)\sigma(j)} = \sum_{\gamma=1}^{n}\sum_{\epsilon=1}^{n}\left(\sum_{\alpha=1}^{n}\sum_{\beta=1}^{n}[\boldsymbol{A}_{\sigma(i)}]_{\alpha\beta}T_{\alpha\gamma}U_{\beta\epsilon}\right)[\boldsymbol{A}_{\sigma(j)}]_{\gamma\epsilon},$$

which requires $(2f_{\sigma(i)}+1)f_{\sigma(j)}$ multiplications. Hence the total number of multiplications to compute all $B_{\sigma(i)\sigma(j)}$ $(i \le j \le m)$ is given by

$$(2f_{\sigma(i)}+1)\sum_{i\le j\le m} f_{\sigma(j)} \tag{12}$$

We now introduce an overhead of access time to elements of the sparse data matrices $\boldsymbol{A}_i$ $(1 \le i \le m)$. We assume that

(i) all $\boldsymbol{A}_i$ $(1 \le i \le m)$ are stored in a common sparse data matrix structure,

(ii) the matrices $\boldsymbol{T}$ and $\boldsymbol{U}$ are stored in the standard matrix array since they are generally dense,

(iii) an access to each element of a matrix stored in the sparse data matrix structure is slower than an access to each element of a matrix stored in the standard matrix array.

Based on these assumption, we modify (10), (11) and (12) to define "the weighted number $d_{ki}(\boldsymbol{\sigma})$ of multiplications" to compute all $B_{\sigma(i)\sigma(j)}$ $(i \le j \le m)$ by formula $\mathcal{F}$-k $(k = 1, 2, 3)$:

$$d_{1i}(\boldsymbol{\sigma}) = \kappa n f_{\sigma(i)} + n^3 + \kappa\sum_{i\le j\le m} f_{\sigma(j)}, \tag{10'}$$

$$d_{2i}(\boldsymbol{\sigma}) = \kappa n f_{\sigma(i)} + \kappa(n+1)\sum_{i\le j\le m} f_{\sigma(j)}, \tag{11'}$$

$$d_{3i}(\boldsymbol{\sigma}) = \kappa(2\kappa f_{\sigma(i)}+1)\sum_{i\le j\le m} f_{\sigma(j)}. \tag{12'}$$

Here $\kappa \ge 1$ is a constant. We may regard κ as an overhead caused by access to an element of sparse data matrices $\boldsymbol{A}_i$ $(1 \le i \le m)$. If $\kappa = 1$ then (10)', (11)' and (12)' coincide with their original definitions (10), (11) and (12). We employ the weighted number of multiplications for measuring the efficiency of the formulae $\mathcal{F}$-1, $\mathcal{F}$-2 and $\mathcal{F}$-3. We will take $\kappa = 1.5$ in Section 4 where we report numerical results on formulae $\mathcal{F}$-1, $\mathcal{F}$-2, $\mathcal{F}$-3 and a combination of them for computing the matrix $\boldsymbol{B}$.

Let $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$. Define

$$d_{*i}(\boldsymbol{\sigma}) = \min\{d_{1i}(\boldsymbol{\sigma}),\ d_{2i}(\boldsymbol{\sigma}),\ d_{3i}(\boldsymbol{\sigma})\}\ (1 \le i \le m), \tag{13}$$

$$d_{*}(\boldsymbol{\sigma}) = \sum_{1\le i\le m} d_{*i}(\boldsymbol{\sigma}). \tag{14}$$

Then, for each $i = 1, 2, \ldots, m$), $d_{*i}(\boldsymbol{\sigma})$ denotes the minimum weighted number of multiplications to compute $B_{\sigma(i)\sigma(j)}$ $(i \leq j \leq m)$ by using any of formulae $\mathcal{F}$-1, $\mathcal{F}$-2 and $\mathcal{F}$-3, and their sum $d_*(\boldsymbol{\sigma})$ over all $i = 1, 2, \ldots, m$ denotes the minimum weighted number of multiplications that are required to compute all the elements of the matrix $\boldsymbol{B}$ in the order (9). $d_*(\boldsymbol{\sigma})$ depends on $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$, *i.e.*, how we permute the indices $1, 2, \ldots, m$ before applying formulae $\mathcal{F}$-1, $\mathcal{F}$-2 and $\mathcal{F}$-3 to computation of the elements of the matrix $\boldsymbol{B}$. We want to choose a permutation that minimizes $d_*(\boldsymbol{\sigma})$ over all $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$. The theorem below states that the minimum of $d_*(\boldsymbol{\sigma})$ over all $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$ is attained if we sort the indices according to $f_1, f_2, \ldots, f_m$ in a descending order.

Theorem 3.1. *(i) $\boldsymbol{\sigma}^*$ is a minimizer of $d_*(\boldsymbol{\sigma})$ over all $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$ if and only if it satisfies*

$$f_{\sigma^*(1)} \geq f_{\sigma^*(2)} \geq \ \cdots \ \geq f_{\sigma^*(m)}. \tag{15}$$

(ii) Suppose that $\boldsymbol{\sigma}^* \in \boldsymbol{\Sigma}$ satisfies (15). Then there exist $q_1 \in \{0, 1, 2, \ldots, m\}$ and $q_2 \in \{q_1, q_1 + 1, \ldots, m\}$ such that

$$\left.\begin{array}{llll} d_{1i}(\boldsymbol{\sigma}^*) \leq d_{2i}(\boldsymbol{\sigma}^*), & d_{1i}(\boldsymbol{\sigma}^*) \leq d_{3i}(\boldsymbol{\sigma}^*) & \text{if } 0 < i \leq q_1, \\ d_{2i}(\boldsymbol{\sigma}^*) < d_{1i}(\boldsymbol{\sigma}^*), & d_{2i}(\boldsymbol{\sigma}^*) \leq d_{3i}(\boldsymbol{\sigma}^*) & \text{if } q_1 < i \leq q_2, \\ d_{3i}(\boldsymbol{\sigma}^*) < d_{1i}(\boldsymbol{\sigma}^*), & d_{3i}(\boldsymbol{\sigma}^*) < d_{2i}(\boldsymbol{\sigma}^*) & \text{if } q_2 < i \leq m. \end{array}\right\} \tag{16}$$

Proof: See Appendix. ∎

Based on the theorem above, we now propose:

Combined Formula $\mathcal{F}$-*(κ):

Step A: Count the number f_i of nonzero elements in $\boldsymbol{A}_i$ $(1 \leq i \leq m)$.

Step B: Sort the set of indices $1, 2, \ldots, m$ according to $f_1, f_2, \ldots, f_m$ in a descending order as given in (15), where $\boldsymbol{\sigma}^*$ is a permutation of the index set $\{1, 2, \ldots, m\}$. For each $i = 1, 2, \ldots, m$, compute $d_{1i}(\boldsymbol{\sigma}^*)$ by (10)', $d_{2i}(\boldsymbol{\sigma}^*)$ by (11)' and $d_{3i}(\boldsymbol{\sigma}^*)$ by (12)'. Find $q_1 \in \{0, 1, 2, \ldots, m\}$ and $q_2 \in \{q_1, q_1 + 1, \ldots, m\}$ satisfying (16).

Step C: For every $i \in \{1, 2, \ldots, m\}$,

- use formula $\mathcal{F}$-1 if $0 < i \leq q_1$, or
- use formula $\mathcal{F}$-2 if $q_1 < i \leq q_2$, or
- use formula $\mathcal{F}$-3 otherwise (i.e., if $q_2 < i \leq m$)

to compute $B_{\sigma^*(i)\sigma^*(j)}$ $(i \leq j \leq m)$.

We can easily incorporate these three steps into a generic primal-dual interior-point method using the HRVW/KSH/M direction or the NT direction. We place Step A where we read data matrices $\boldsymbol{A}_i$ $(1 \leq i \leq m)$, and Step B right after Step A. Hence we execute both steps once in the method. On the other hand, we place Step C where we compute a direction. Hence we need to execute Step C repeatedly at each iteration of the primal-dual interior-point method.

4 Numerical Results.

In this section, we present some numerical results on formulae $\mathcal{F}$-1, $\mathcal{F}$-2, $\mathcal{F}$-3 and $\mathcal{F}$-$*(\kappa)$ for four kinds of SDPs. One is a randomly generated SDP, and the others are SDP relaxations of the quadratic assignment problem, the graph partitioning problem and the maximum clique problem. We applied a modified version of the SDPA [5] using the HRVW/KSH/M search direction to all SDPs. The numerical results were computed on DEC Alpha (CPU 21164-300MHz with 256MB memory). In general, the overhead parameter κ caused by access to an element of sparse data matrices $\boldsymbol{A}_i$ $(1 \leq i \leq m)$ depends on a data structure for $\boldsymbol{A}_i$ $(1 \leq i \leq m)$. We employed a sparse data structure illustrated in Figure 1. We made preliminary numerical experiments to estimate $\kappa = 1.5$.

Figure 1: Data Structure for Sparse Data Matrices.

$$A_i = \begin{pmatrix} 11 & 0 & 13 & 0 & 0 \\ 0 & 0 & 0 & 0 & 25 \\ 13 & 0 & 33 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 25 & 0 & 0 & 0 \end{pmatrix}$$

$$\Downarrow$$

row	$\rightarrow$	1	1	2	3
column	$\rightarrow$	1	3	5	3
value	$\rightarrow$	11	13	25	33

4.1 Randomly Generated SDPs.

Let p be a positive integer. Define a function $f(\cdot; m, n, p) : \{1, 2, \ldots, m\} \rightarrow \{1, 2, \ldots, n^2\}$ by

$$f(i; m, n, p) = \left[\frac{(n^2 - 1)(m - i)^p}{(m - 1)^p} + 1 \right],$$

where $[a]$ denotes the largest integer not less than a. To construct an SDP, we randomly generated a $n \times n$ data matrix $\boldsymbol{A}_i \in \mathcal{S}^n$ which expectedly had $f(i; m, n, p)$ nonzero elements $(1 \leq i \leq m)$ (If $\boldsymbol{A}_i = \boldsymbol{O}$ then we modified it so that it had at least one nonzero element). The parameter p controls how fast the expected number of nonzero elements in the $n \times n$ symmetric matrix $\boldsymbol{A}_i$ decreases as i increases from 1 to m. For any choice of p, $\boldsymbol{A}_1$ has n^2 nonzero elements and $\boldsymbol{A}_m$ at least one nonzero element. We took $p = \log_2 n$, so that $\boldsymbol{A}_{[m/2]}$ expectedly had about n nonzeros. Table 1 shows numerical results. The columns $\mathcal{F}$-1, $\mathcal{F}$-2, $\mathcal{F}$-3 and $\mathcal{F}$-$*(1.5)$ denote the average CPU time in second to compute the $m \times m$ matrix $\boldsymbol{B}$ per iteration when we used the formulae $\mathcal{F}$-1, $\mathcal{F}$-2, $\mathcal{F}$-3 and $\mathcal{F}$-$*(1.5)$ (the combined formula $\mathcal{F}$-$*(\kappa)$ with $\kappa = 1.5$), respectively. The last column denotes "the average CPU time of one iteration in second - the average CPU time to compute $\boldsymbol{B}$ in second" when we used formula $\mathcal{F}$-$*(1.5)$. One iteration consists of computation of the matrix $\boldsymbol{B}$, a search direction, a step length, a new iterate, etc. Each average CPU time in Table 1 was taken over the first 10 iterations. Recall also that q_1 and q_2 denote the positive numbers determined by (16), i.e., in the combined formula $\mathcal{F}$-$*(1.5)$, to compute $B_{\sigma^*(i)\sigma^*(j)}$ $(i \leq j \leq m)$, we used $\mathcal{F}$-1 if

Table 1: Randomly Generated SDPs.

p	n	m	$\mathcal{F}$-1	$\mathcal{F}$-2	$\mathcal{F}$-3	$\mathcal{F}$-$*$(1.5)	q_1	q_2	one iteration $-\mathcal{F}$-$*$(1.5)
5	32	32	0.07	0.08	1.65	0.04	11	20	0.08
5	32	64	0.14	0.30	5.87	0.09	26	40	0.10
5	32	128	0.30	1.02	23.19	0.22	62	80	0.14
6	64	64	1.10	1.79	71.56	0.59	22	39	0.71
6	64	128	2.24	7.02	279.45	1.33	52	76	0.77
6	64	256	4.75	23.81	-	3.14	118	152	1.09
7	128	128	19.89	-	-	11.98	49	79	6.43
7	128	256	42.82	-	-	24.07	99	147	7.06

$0 < i \leq q_1$, $\mathcal{F}$-2 if $q_1 < i \leq q_2$ and $\mathcal{F}$-3 otherwise (i.e., if $q_2 < i \leq m$). We see from Table 1 that

- formula $\mathcal{F}$-$*$(1.5) works efficiently, and
- as m increases, the CPU time to compute $\boldsymbol{B}$ becomes more crucial in the CPU time of one iteration.

Figures 2 and 3 compare the weighted number $d_{*i}(\boldsymbol{\sigma}^*) = \min\{d_{1i}(\boldsymbol{\sigma}^*),\ d_{2i}(\boldsymbol{\sigma}^*),\ d_{3i}(\boldsymbol{\sigma}^*)\}$ of multiplications with the average CPU time to compute $B_{\sigma^*(i)\sigma^*(j)}$ $(i \leq j \leq m)$ per iteration for the case of $p = 7$, $n = 128$ and $m = 256$. We observe that our theoretical efficiency measure $d_{ki}(\boldsymbol{\sigma}^*)$ for formula $\mathcal{F}$-k works quite nicely $(k = 1, 2, 3)$.

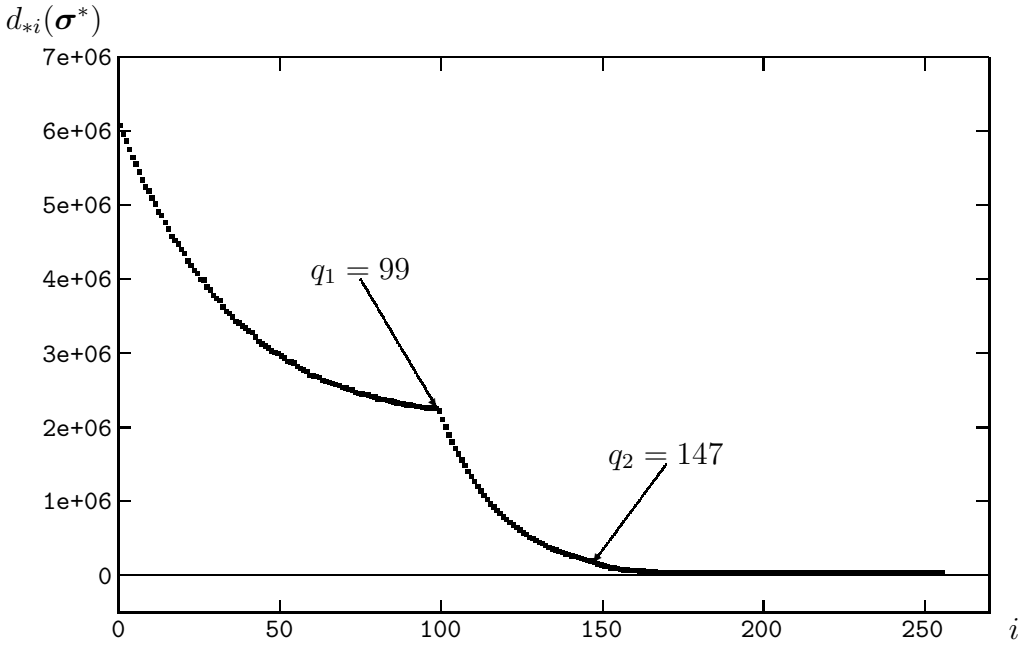

Figure 2: Weighted number $d_{*i}(\sigma^*)$ of multiplications for the case: $p = 7$, $n = 128$ and $m = 256$

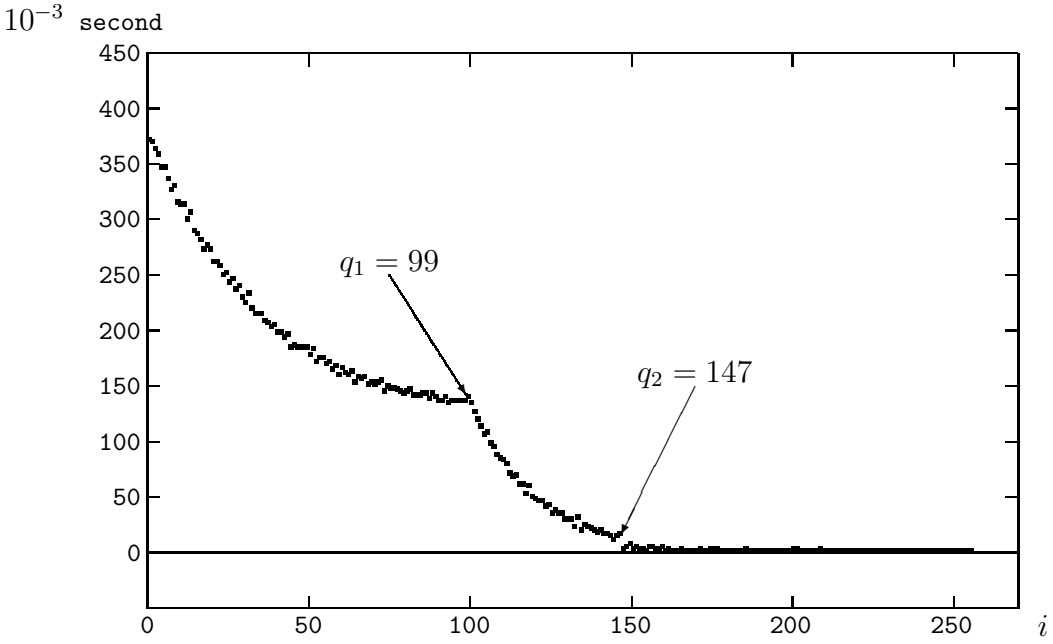

Figure 3: Average CPU time to compute $B_{\sigma^*(i)\sigma^*(j)}$ $(i \leq j \leq m)$ for the case: $p = 7$, $n = 128$ and $m = 256$

Table 2: Quadratic Assignment Problems.

s	n	m	$\mathcal{F}$-1	$\mathcal{F}$-2	$\mathcal{F}$-3	$\mathcal{F}$-$*$(1.5)	$q_1 = q_2$	one iteration $-\mathcal{F}$-$*$(1.5)
5	26	136	0.14	0.12	2.15	0.04	10	0.10
6	37	229	0.67	0.48	11.73	0.13	12	0.34
7	50	358	3.33	1.91	56.52	0.42	14	1.32
8	65	529	8.29	4.65	-	0.90	16	3.16
9	82	748	22.89	11.79	-	1.99	18	11.08
10	101	1021	61.28	29.47	-	4.50	20	36.75

4.2 Quadratic Assignment Problems.

As a SDP relaxation of quadratic assignment problems (see, for example, [15, 20]), we generated SDPs of the form (1) with $n = s^2+1$, $m = 2s+s^3+1$, $2s$ $\boldsymbol{A}_i$'s having s^4 nonzero elements, s^2 $\boldsymbol{A}_i$'s having 3 nonzero elements, $(s^3 - s^2)$ $\boldsymbol{A}_i$'s having 2 nonzero elements and one $\boldsymbol{A}_i$ having 1 nonzero element. Here s denotes the size of a quadratic assignment problem. In this case, we have that

$$
\begin{aligned}
&f_{\sigma^*(i)} = s^4 \ (1 \le i \le 2s),\\
&f_{\sigma^*(i)} \in \{1,2,3\} \ (2s+1 \le i \le 2s+s^3+1),\\
&d_{1i}(\boldsymbol{\sigma}^*) < d_{2i}(\boldsymbol{\sigma}^*) \ \text{ and } d_{1i}(\boldsymbol{\sigma}^*) < d_{3i}(\boldsymbol{\sigma}^*) \ (1 \le i \le 2s),\\
&d_{3i}(\boldsymbol{\sigma}^*) < d_{1i}(\boldsymbol{\sigma}^*) \ \text{ and } d_{3i}(\boldsymbol{\sigma}^*) < d_{2i}(\boldsymbol{\sigma}^*) \ (2s+1 \le i \le 2s+s^3+1).
\end{aligned}
$$

Therefore $q_1 = q_2 = 2s$. Table 2 shows numerical results. The combined formula $\mathcal{F}$-$*$(1.5) works very efficiently.

4.3 Graph Partitioning Problems

As SDP relaxation of graph partitioning problems, we generated SDPs of the form (1) with $n = s$, $m = s+1$, one $\boldsymbol{A}_i$ having s^2 nonzero elements and s $\boldsymbol{A}_i$'s having 1 nonzero element. Here s denotes the number of nodes. In this case, we have that

$$
\begin{aligned}
&f_{\sigma^*(1)} = s^2,\\
&f_{\sigma^*(i)} = 1 \ (2 \le i \le s+1),\\
&d_{11}(\boldsymbol{\sigma}^*) < d_{21}(\boldsymbol{\sigma}^*) \ \text{ and } d_{11}(\boldsymbol{\sigma}^*) < d_{31}(\boldsymbol{\sigma}^*),\\
&d_{3i}(\boldsymbol{\sigma}^*) < d_{1i}(\boldsymbol{\sigma}^*) \ \text{ and } d_{3i}(\boldsymbol{\sigma}^*) < d_{2i}(\boldsymbol{\sigma}^*) \ (2 \le i \le s+1).
\end{aligned}
$$

Therefore $q_1 = 1$ and $q_2 = 1$. Table 3 shows numerical result, where t denotes the number of edges. The combined formula $\mathcal{F}$-$*$(1.5) works efficiently.

4.4 Maximum Clique Problems.

As SDP relaxation of maximum clique problems, we generated SDPs of the form (1) with $n = s$, $m = u+1$, one $\boldsymbol{A}_i$ having s nonzero elements and u $\boldsymbol{A}_i$'s having 2 nonzero elements.

Table 3: Graph Partitioning Problems.

s	t	n	m	$\mathcal{F}$-1	$\mathcal{F}$-2	$\mathcal{F}$-3	$\mathcal{F}$-$*$(1.5)	one iteration $-\mathcal{F}$-$*$(1.5)
124	1271	124	125	21.15	0.56	22.13	0.36	6.36
250	2421	250	251	360.09	5.07	356.96	2.99	59.50
500	5120	500	501	7247.16	52.04	6341.55	29.29	607.23

Table 4: Maximum Clique Problems.

s	u	n	m	$\mathcal{F}$-1	$\mathcal{F}$-2	$\mathcal{F}$-3	$\mathcal{F}$-$*$(1.5)	one iteration $-\mathcal{F}$-$*$(1.5)
150	561	150	562	123.15	5.35	0.36	0.36	13.20
150	1101	150	1102	238.90	19.70	1.46	1.46	44.75
200	621	200	622	336.58	9.30	0.51	0.48	28.82
200	992	200	993	540.16	28.73	1.27	1.32	50.28
250	651	250	652	980.63	19.01	0.59	0.61	63.43
250	972	250	973	1396.54	40.86	1.54	1.33	80.04
300	943	300	944	2472.22	43.01	1.40	1.29	120.21

Here s denotes the number of nodes and u denotes the number of pairs of nodes having no edge between them (or "$n(n-1)/2$ − the number of edges"). In this case, we have that

$$
\begin{aligned}
&f_{\sigma^*(1)} = s,\\
&f_{\sigma^*(i)} = 2 \ (2 \le i \le u+1),\\
&d_{21}(\boldsymbol{\sigma}^*) < d_{11}(\boldsymbol{\sigma}^*) \ \text{ and } d_{21}(\boldsymbol{\sigma}^*) < d_{31}(\boldsymbol{\sigma}^*),\\
&d_{3i}(\boldsymbol{\sigma}^*) < d_{1i}(\boldsymbol{\sigma}^*) \ \text{ and } d_{3i}(\boldsymbol{\sigma}^*) < d_{2i}(\boldsymbol{\sigma}^*) \ (2 \le i \le u+1).
\end{aligned}
$$

Therefore $q_1 = 0$ and $q_2 = 1$. Table 4 shows numerical result. Again the combined formula $\mathcal{F}$-$*$(1.5) works efficiently.

5 Concluding Remarks.

(A) We have assumed the matrix $\boldsymbol{F}_i = \boldsymbol{A}_{\sigma(i)}\boldsymbol{U}$ to be stored as a dense matrix when we count the number n^3 of multiplications to compute $\boldsymbol{G}_i = \boldsymbol{T}\boldsymbol{F}_i$ in formula $\mathcal{F}$-1, and when we count the number $(n+1)f_{\sigma(j)}$ of multiplications to compute

$$
B_{\sigma(i)\sigma(j)} = \sum_{\alpha=1}^{n}\sum_{\beta=1}^{n}[\boldsymbol{A}_{\sigma(j)}]_{\alpha\beta}\left(\sum_{\gamma=1}^{n} T_{\alpha\gamma}[\boldsymbol{F}i]_{\gamma\beta}\right).
$$

It should be noted that a γth row of the matrix $\boldsymbol{F}_i = \boldsymbol{A}_{\sigma(i)}\boldsymbol{U}$ becomes the zero vector whenever the γth row of $\boldsymbol{A}_{\sigma(i)}$ is the zero vector $(1 \le \gamma \le n)$. Let $g_{\sigma(i)}$ denote the number of nonzero rows of the symmetric data matrix $\boldsymbol{A}_{\sigma(i)}$. Then $\sqrt{f_{\sigma(i)}} \le g_{\sigma(i)} \le \min\{n,\ f_{\sigma(i)}\}$

since $\boldsymbol{A}_{\sigma(i)}$ is symmetric. If we incorporate the sparsity of the matrix $\boldsymbol{F}_i = \boldsymbol{A}_{\sigma(i)}\boldsymbol{U}$ into our consideration, we replace (10) by

$$nf_{\sigma(i)} + g_{\sigma(i)}n^2 + \sum_{i \le j \le m} f_{\sigma(j)},$$

and (11) by

$$nf_{\sigma(i)} + (g_{\sigma(i)} + 1) \sum_{i \le j \le m} f_{\sigma(j)},$$

respectively. Such a modification is effective when $g_{\sigma(i)} < n$. But we also mention that if $g_{\sigma(i)}$ is sufficiently small (or more precisely, if $g_{\sigma(i)} = O(1)$), formula $\mathcal{F}$-3 looks more efficient than formulae $\mathcal{F}$-1 and $\mathcal{F}$-2 even with such a modification; hence formulae $\mathcal{F}$-1 and $\mathcal{F}$-2 are not likely to be used.

(B) We can evaluate the numbers of additions in formulae $\mathcal{F}$-1, $\mathcal{F}$-2 and $\mathcal{F}$-3 to compute $B_{\sigma(i)\sigma(j)}$ $(i \le j \le m)$. The numbers of additions in formulae $\mathcal{F}$-1 and $\mathcal{F}$-2 coincide with the numbers of multiplications in formulae $\mathcal{F}$-1 and $\mathcal{F}$-2, respectively. See (10) and (11). But the number of additions in formulae $\mathcal{F}$-3 is smaller than the number of multiplication given in (12);

$$\begin{aligned} \text{“the number of multiplication”} &= (2f_{\sigma(i)} + 1) \sum_{i \le j \le m} f_{\sigma(j)} \\ &> (f_{\sigma(i)} + 1) \sum_{i \le j \le m} f_{\sigma(j)} = \text{“the number of additions”}. \end{aligned}$$

We learned through our numerical experiments that formula $\mathcal{F}$-3 works more efficiently than what we expect from our theoretical analysis in Section 4. In fact, we observe a little jump of the curve at $q_2 = 147$ in Figure 3, and we also observe in Table 4 that formula $\mathcal{F}$-3 works as efficiently as formula $\mathcal{F}$-$*$(1.5). These facts may be partially explained by the number of additions which we have not taken account of.

(C) We can combine our method for computing the coefficient matrix $\boldsymbol{B} = \boldsymbol{B}'$ of the system of equations in (6) (or the coefficient matrix $\boldsymbol{B} = \boldsymbol{B}''$ of the system of equations in (7)) in Section 3 with indirect or iteration methods (such as the conjugate direction method) for solving the system of equations $\boldsymbol{B}d\boldsymbol{y} = \boldsymbol{b}$. In those methods we repeatedly compute $\boldsymbol{u} = \boldsymbol{B}\boldsymbol{v}$ for different $\boldsymbol{v}$'s in R^m. It should be noted that each element u_i is of the form $u_i = \sum_{1 \le j \le m} B_{ij}v_j$ $(1 \le i \le m)$. Therefore we can compute $\boldsymbol{u} = \boldsymbol{B}\boldsymbol{v}$ without storing the entire matrix $\boldsymbol{B}$. This advantage of iteration methods is crucial when the number m of the equality constraints of the SDP (1) is too large to store the entire matrix $\boldsymbol{B}$ in the computer memory. In fact, the conjugate gradient method was applied to large scale SDPs arising from relaxations of the quadratic assignment problem ([10, 20, etc.]).

(D) Suppose that the data matrices $\boldsymbol{A}_i$ $(0 \le i \le m)$ of the SDP (1) share a common block diagonal matrix structure as follows:

$$\boldsymbol{A}_i = \begin{pmatrix} \boldsymbol{A}_i^0 & \boldsymbol{O} & \cdots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{A}_i^1 & \cdots & \boldsymbol{O} \\ \cdot & \cdot & \cdots & \cdot \\ \boldsymbol{O} & \boldsymbol{O} & \cdots & \boldsymbol{A}_i^\ell \end{pmatrix}. \tag{17}$$

Here $\boldsymbol{A}_i^k$ denotes a $n_k \times n_k$ symmetric matrix ($0 \leq k \leq \ell$). In this case we can represent the coefficient matrix $\boldsymbol{B} = \boldsymbol{B}'$ of the system of equations in (6) (or the coefficient matrix $\boldsymbol{B} = \boldsymbol{B}''$ of the system of equations in (7)) as

$$\boldsymbol{B} \;=\; \sum_{0 \leq k \leq \ell} \boldsymbol{B}^k \;\text{ and }\; B_{ij}^k = \boldsymbol{T}^k \boldsymbol{A}_i^k \boldsymbol{U}^k \bullet \boldsymbol{A}_j^k \;(0 \leq k \leq \ell).$$

Therefore we can apply the method given in Section 3 to computation of each $\boldsymbol{B}^k$ separately ($0 \leq k \leq \ell$). Such a block diagonal matrix structure often appears in semidefinite programs arising from the system and control theory. See [4]. Also, when we are given inequality constraints

$$\boldsymbol{A}_i^0 \bullet \boldsymbol{X}^0 \leq b_i \;(1 \leq i \leq m) \;\text{ and }\; \boldsymbol{X}^0 \succeq \boldsymbol{O},$$

we can convert them to equality constraints

$$\boldsymbol{A}_i \bullet \boldsymbol{X} = b_i \;(1 \leq i \leq m) \;\text{ and }\; \boldsymbol{X} \succeq \boldsymbol{O}$$

by defining the matrices $\boldsymbol{A}_i$ ($1 \leq i \leq m$) by (17) with

$$\ell = m,\; n_k = 1 \;(1 \leq k \leq m) \;\text{ and }\; \boldsymbol{A}_i^k = \begin{cases} 1 & \text{if } k = i, \\ 0 & \text{otherwise}, \end{cases}$$

where $\boldsymbol{X}$ is a variable symmetric matrix having the same block diagonal structure as $\boldsymbol{A}_i$;

$$\boldsymbol{X} = \begin{pmatrix} \boldsymbol{X}^0 & \boldsymbol{O} & \ldots & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{X}^1 & \ldots & \boldsymbol{O} \\ . & . & \ldots & . \\ \boldsymbol{O} & \boldsymbol{O} & \ldots & \boldsymbol{X}^\ell \end{pmatrix}.$$

References

[1] F. Alizadeh, J.-P.A. Haeberly and M.L. Overton, "Primal-dual interior-point methods for semidefinite programming," Working Paper, 1994.

[2] F. Alizadeh, J.-P.A. Haeberly and M.L. Overton, "Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results," Report 721, Computer Science Dept. New York University, New York, May 1996.

[3] N. Brixius, F.A. Potra and R. Sheng, "Solving semidefinite programming in mathematica," Reports on Computational Mathematics, No 97/1996, Dept. of Mathematics, University of Iowa, October 1996. Available at http://www.cs.uiowa.edu/ brixius/sdp.html.

[4] S. Boyd, L.E. Ghaoui, E. Feron and V. Balakrishnan, *Linear Matrix Inequalities in System and Control Theory,* (SIAM, Philadelphia, 1994).

[5] K. Fujisawa, M. Kojima and K. Nakata, "SDPA (Semidefinite Programming Algorithm) – User's Manual –," Technical Report B-308, Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan, December 1995, Revised August 1996. Available via anonymous ftp at ftp.is.titech.ac.jp in pub/OpRes/software/SDPA.

[6] C. Helmberg, F. Rendl, R.J. Vanderbei and H. Wolkowicz, "An interior-point method for semidefinite programming," *SIAM Journal on Optimization* **6** (1996) 342–361.

[7] M. Kojima, M. Shida and S. Shindoh, "Local Convergence of Predictor-Corrector Infeasible-Interior-Point Algorithms for SDPs and SDLCPs" Research Report #306, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152, Japan, December 1995, Revised October 1996.

[8] M. Kojima, S. Shindoh and S. Hara, "Interior-point methods for the monotone semidefinite linear complementarity problems," to appear in *SIAM Journal on Optimization.*

[9] C.-J. Lin and R. Saigal, "A predictor-corrector method for semi-definite linear programming," Working paper, Department of Industrial and Operations Engineering, The University of Michigan, Ann Arbor, Michigan 48109-2117, October 1995.

[10] C.-J. Lin and R. Saigal, "Predictor corrector methods for semidefinite programming," Informs Atlanta Fall Meeting, November 3-6, 1996.

[11] Z.-Q. Luo, J.F. Sturm and S. Zhang, "Superlinear convergence of a symmetric primal-dual path following algorithm for semidefinite programming," to appear in *SIAM Journal on Optimization.*

[12] R.D.C. Monteiro, "Primal-Dual Path Following Algorithms for Semidefinite Programming," to appear in *SIAM Journal on Optimization.*

[13] Yu.E. Nesterov and M.J. Todd, "Self-Scaled Cones and Interior-Point Methods in Nonlinear Programming," Working Paper, CORE, Catholic University of Louvain, Louvain-la-Neuve, Belgium, April 1994, to appear in *Mathematics of Operations Research.*

[14] Yu.E. Nesterov and M.J. Todd, "Primal-dual interior-point methods for self-scaled cones," Technical Report 1125, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853-3801, USA, 1995.

[15] S.Poljak, F.Rendl and H.Wolkowicz, "A recipe for semidefinite relaxation for (0,1) quadratic programming," *Journal of Global Optimization* **7** (1995) 51–73.

[16] F.A. Potra and R. Sheng, "Superlinear convergence of infeasible-interior-point algorithms for semidefinite programming," Department of Mathematics, University of Iowa, Iowa City, IA 52242, April 1996.

[17] M.J. Todd, K.C. Toh and R.H. Tütüncü, "On the Nesterov-Todd direction in semidefinite programming," Technical Report, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, New York 14853-3801, USA, March 1996, Revised May 1996.

[18] K.C. Toh, M.J. Todd and R.H. Tütüncü, "SDPT3 – a MATLAB software package for semidefinite programming," Dept. of Mathematics, National University of Singapore, 10 Kent Ridge Crescent, Singapore, December 1996. Available at http://www.math.nus.sg/ mattohkc/index.html.

[19] Y. Zhang, "On Extending Primal-Dual Interior-Point Algorithms from Linear Programming to Semidefinite Programming," to appear in *SIAM Journal on Optimization.*

[20] Q. Zhao, S.E. Karisch, F. Rendl and H. Wolkowicz, "Semidefinite programming relaxations for the quadratic assignment problem," CORR Report 95-27, Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Ontario, Canada, September 1996.

Appendix: Proof of Theorem 3.1.

We first prove two lemmas.

Lemma A.1. *Let $\boldsymbol{\sigma}$ be a permutation of $\{1,2,\ldots,m\}$. Suppose that $f_{\sigma(\ell)} < f_{\sigma(\ell+1)}$ for some $\ell \in \{1,2,\ldots,m-1\}$. Define the permutation $\boldsymbol{\tau}$ by*

$$\tau(i) = \begin{cases} \sigma(i) & \text{if } 1 \le i < \ell \ \text{ or } \ell+1 < i \le m, \\ \sigma(\ell+1) & \text{if } i = \ell, \\ \sigma(\ell) & \text{if } i = \ell+1. \end{cases}$$

Then $d_*(\boldsymbol{\sigma}) > d_*(\boldsymbol{\tau})$.

Proof: For simplicity of notation, we assume that $\sigma(i) = i$ $(i = 1,2,\ldots,m)$. We see from the assumption that

$$\begin{aligned} &\tau(i) = \begin{cases} i & \text{if } 1 \le i < \ell \ \text{ or } \ell+1 < i \le m, \\ \ell+1 & \text{if } i = \ell, \\ \ell & \text{if } i = \ell+1, \end{cases} \\ &f_\ell < f_{\ell+1}, \\ &d_{*i}(\boldsymbol{\sigma}) = d_{*i}(\boldsymbol{\tau}) \ \text{ if } 1 \le i < \ell \ \text{ or } \ell+1 < i \le m. \end{aligned} \tag{18}$$

Hence it suffices to show that the inequality

$$d_{*\ell}(\boldsymbol{\sigma}) + d_{*(\ell+1)}(\boldsymbol{\sigma}) > d_{*\ell}(\boldsymbol{\tau}) + d_{*(\ell+1)}(\boldsymbol{\tau}) \tag{19}$$

holds under the assumption (18). Furthermore we know by definition that

$$\begin{aligned} d_{q\ell}(\boldsymbol{\tau}) &\ge d_{*\ell}(\boldsymbol{\tau}) \ \text{ for any } q \in \{1,2,3\}, \\ d_{r(\ell+1)}(\boldsymbol{\tau}) &\ge d_{*(\ell+1)}(\boldsymbol{\tau}) \text{ for any } r \in \{1,2,3\}. \end{aligned}$$

Therefore the inequality (19) follows if

$$d_{*\ell}(\boldsymbol{\sigma}) + d_{*(\ell+1)}(\boldsymbol{\sigma}) \ge d_{q\ell}(\boldsymbol{\tau}) + d_{r(\ell+1)}(\boldsymbol{\tau}) \tag{20}$$

holds for some $q,\ r \in \{1,2,3\}$. To derive (20) for some $q,\ r \in \{1,2,3\}$, we take q and r as follows:

(a) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{1\ell}(\boldsymbol{\sigma})$ and $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{1(\ell+1)}(\boldsymbol{\sigma})$ then let $q = 1$ and $r = 1$.

(b) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{1\ell}(\boldsymbol{\sigma})$ and $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{2(\ell+1)}(\boldsymbol{\sigma})$ then let $q = 1$ and $r = 2$.

(c-1) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{1\ell}(\boldsymbol{\sigma})$, $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma})$ and $2\kappa \sum_{\ell \le j \le m} f_j \ge n$ then let $q = 1$ and $r = 3$.

(c-2) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{1\ell}(\boldsymbol{\sigma})$, $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma})$ and $2\kappa \sum_{\ell \le j \le m} f_j < n$ then let $q = 3$ and $r = 3$.

(d) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{2\ell}(\boldsymbol{\sigma})$ and $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{1(\ell+1)}(\boldsymbol{\sigma})$ then let $q = 1$ and $r = 2$.

(e) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{2\ell}(\boldsymbol{\sigma})$ and $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{2(\ell+1)}(\boldsymbol{\sigma})$ then let $q = 2$ and $r = 2$.

(f-1) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{2\ell}(\boldsymbol{\sigma})$, $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma})$ and $2\kappa \sum_{\ell \le j \le m} f_j \ge n$ then let $q = 2$ and $r = 3$.

(f-2) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{2\ell}(\boldsymbol{\sigma})$, $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma})$ and $2\kappa \sum_{\ell \le j \le m} f_j < n$ then let $q = 3$ and $r = 3$.

(g) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{3\ell}(\boldsymbol{\sigma})$ and $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{1(\ell+1)}(\boldsymbol{\sigma})$ then let $q = 1$ and $r = 3$.

(h-1) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{3\ell}(\boldsymbol{\sigma})$, $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{2(\ell+1)}(\boldsymbol{\sigma})$ and $2\kappa f_{\ell+1} \ge n$ then let $q = 2$ and $r = 3$.

(h-2) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{3\ell}(\boldsymbol{\sigma})$, $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{2(\ell+1)}(\boldsymbol{\sigma})$ and $2\kappa f_{\ell+1} < n$ then let $q = 3$ and $r = 3$.

(i) If $d_{*\ell}(\boldsymbol{\sigma}) = d_{3\ell}(\boldsymbol{\sigma})$ and $d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma})$ then let $q = 3$ and $r = 3$.

In each of the cases, it is easy to derive the inequality (20). Here we only deal with cases (c-1) and (c-2). Suppose that

$$d_{*\ell}(\boldsymbol{\sigma}) = d_{1\ell}(\boldsymbol{\sigma}),\ d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma}) \ \text{ and } 2\kappa \sum_{\ell \le j \le m} f_j \ge n.$$

Then

$$\begin{aligned} &\Big(d_{1\ell}(\boldsymbol{\sigma}) + d_{3(\ell+1)}(\boldsymbol{\sigma})\Big) - \Big(d_{1\ell}(\boldsymbol{\tau}) + d_{3(\ell+1)}(\boldsymbol{\tau})\Big) \\ &= \ \kappa(f_{\ell+1} - f_\ell)\left(2\kappa \sum_{\ell \le j \le m} f_j - n\right) + \kappa(f_{\ell+1} - f_\ell) \ > \ 0. \end{aligned}$$

Thus we have shown the inequality (20) in case (c-1). Now assume that

$$d_{*\ell}(\boldsymbol{\sigma}) = d_{1\ell}(\boldsymbol{\sigma}),\ d_{*(\ell+1)}(\boldsymbol{\sigma}) = d_{3(\ell+1)}(\boldsymbol{\sigma}) \ \text{ and } 2\kappa \sum_{\ell \le j \le m} f_j < n.$$

Then

$$\begin{aligned} &\Big(d_{1\ell}(\boldsymbol{\sigma}) + d_{3(\ell+1)}(\boldsymbol{\sigma})\Big) - \Big(d_{3\ell}(\boldsymbol{\tau}) + d_{3(\ell+1)}(\boldsymbol{\tau})\Big) \\ &= \ \kappa f_\ell \left(n - 2\kappa \sum_{\ell \le j \le m} f_j\right) + n^3 + \kappa(f_{\ell+1} - f_\ell) \ > \ 0. \end{aligned}$$

Thus we have shown the inequality (20) in case (c-2). We can derive the other cases similarly. The details are omitted. ∎

Lemma A.2. *Let $\boldsymbol{\sigma}$ be a permutation of $\{1, 2, \ldots, m\}$ and $i \in \{1, 2, \ldots, m-1\}$. Suppose that $f_{\sigma(i)} \ge f_{\sigma(i+1)}$ for some $i \in \{1, 2, \ldots, m-1\}$.*

(a) If $d_{2i}(\boldsymbol{\sigma}) \le d_{1i}(\boldsymbol{\sigma})$ then $d_{2(i+1)}(\boldsymbol{\sigma}) < d_{1(i+1)}(\boldsymbol{\sigma})$.

(b) If $d_{3i}(\boldsymbol{\sigma}) \le d_{1i}(\boldsymbol{\sigma})$ then $d_{3(i+1)}(\boldsymbol{\sigma}) < d_{1(i+1)}(\boldsymbol{\sigma})$.

(c) If $d_{3i}(\boldsymbol{\sigma}) \le d_{2i}(\boldsymbol{\sigma})$ then $d_{3(i+1)}(\boldsymbol{\sigma}) < d_{2(i+1)}(\boldsymbol{\sigma})$.

Proof: For simplicity of notation, we assume that $\sigma(j) = j$ $(j = 1, 2, \ldots, m)$, and use the notation d_{1j}, d_{2j} and d_{3j} for $d_{1j}(\boldsymbol{\sigma})$, $d_{2j}(\boldsymbol{\sigma})$ and $d_{3j}(\boldsymbol{\sigma})$, respectively.

(a) By (10)' and (11)' we see that

$$\left(d_{1(i+1)} - d_{2(i+1)}\right) - (d_{1i} - d_{2i}) \;=\; n\kappa f_i > 0.$$

Hence $(d_{1i} - d_{2i}) \ge 0$ implies $\left(d_{1(i+1)} - d_{2(i+1)}\right) > 0$.

(b) By (10)' and (12)' we see that

$$\left(d_{1(i+1)} - d_{3(i+1)})\right) \;=\; n^3 + \kappa f_{i+1}\left(n - 2\kappa \sum_{i+1\le j\le m} f_j\right).$$

Hence, if $\left(n - 2\kappa \sum_{i+1\le j\le m} f_j\right) \ge 0$ then $\left(d_{1(i+1)} - d_{3(i+1)}\right) > 0$. Now assume that $\left(n - 2\kappa \sum_{i+1\le i\le m} f_j\right) < 0$. Then

$$\begin{aligned}&\left(d_{1(i+1)} - d_{3(i+1)}\right) - (d_{1i} - d_{3i})\\ &\quad= \; 2(\kappa f_i)^2 + \kappa(f_i - f_{i+1})\left(2\kappa \sum_{i+1\le j\le m} f_j - n\right) \;>\; 0.\end{aligned}$$

Hence $(d_{1i} - d_{3i}) \ge 0$ implies $\left(d_{1(i+1)} - d_{3(i+1)}\right) > 0$.

(c) By (11)' and (12)', we see that if $n \ge 2\kappa f_{i+1}$ or $n \ge 2\kappa \sum_{i+1\le j\le m} f_j$ then

$$\begin{aligned}\left(d_{2(i+1)} - d_{3(i+1)}\right) \;&=\; n\kappa f_{i+1} + (n - 2\kappa f_{i+1})\,\kappa \sum_{i+1\le j\le m} f_j\\ &=\; \left(n - 2\kappa \sum_{i+1\le j\le m} f_j\right)\kappa f_{i+1} + n\kappa \sum_{i+1\le j\le m} f_j \;>\; 0.\end{aligned}$$

Now assume that $n < 2\kappa f_{i+1}$ and $n < 2\kappa \sum_{i+1\le j\le m} f_j$. Then

$$\begin{aligned}&\left(d_{2(i+1)} - d_{3(i+1)}\right) - (d_{2i} - d_{3i})\\ &\quad=\; \kappa(f_i - f_{i+1})\left(2\kappa \sum_{i+1\le j\le m} f_j - n\right) + \kappa f_i\,(2\kappa f_i - n)\\ &\quad\ge\; f_i\,(2f_{i+1} - n) \;>\; 0.\end{aligned}$$

Hence $(d_{2i} - d_{3i}) \ge 0$ implies $\left(d_{2(i+1)} - d_{3(i+1)}\right) > 0$. ∎

Now we are ready to prove Theorem 3.1. Let $\boldsymbol{\sigma}^*$ be a minimizer of $d_*(\boldsymbol{\sigma})$ over $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$. Assume on the contrary that $\boldsymbol{\sigma}^*$ does not satisfy the relation (15). Then there is an index $\ell \in \{1, 2, \ldots, m-1\}$ such that $f_{\sigma^*(\ell)} < f_{\sigma^*(\ell+1)}$. Hence, applying Lemma A.1, we can find a $\boldsymbol{\tau} \in \boldsymbol{\Sigma}$ such that $d_*(\boldsymbol{\tau}) < d_*(\boldsymbol{\sigma}^*)$. This is a contradiction. Thus we have shown that $\boldsymbol{\sigma}^* \in \boldsymbol{\Sigma}$ satisfies (15).

Now suppose that $\boldsymbol{\sigma}^* \in \boldsymbol{\Sigma}$ satisfies (15). Since the number of elements in $\boldsymbol{\Sigma}$ is finite, $d_* : \boldsymbol{\Sigma} \to R$ attains the minimum at some $\boldsymbol{\tau} \in \boldsymbol{\Sigma}$, which must satisfy the relation

$$f_{\tau(1)} \geq f_{\tau(2)} \geq \ \cdots \ \geq f_{\tau(m)} \tag{15'}$$

Since the relations (15) and (15)' imply that $f_{\sigma(i)} = f_{\tau(i)}$ for every $i \in \{1, 2, \ldots, m\}$, we obtain that $d_*(\boldsymbol{\sigma}) = d_*(\boldsymbol{\tau})$. Therefore $\boldsymbol{\sigma}^* \in \boldsymbol{\Sigma}$ is a minimizer of $d_*(\boldsymbol{\sigma})$ over all $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}$.

Assertion (ii) of the theorem follows from Lemma A.2.