数理解析研究所講究録
1174 巻 2000 年 130-137

Solving Sparse Semidefinite Programs by Matrix Completion (part II)

東京大学 工学系研究科 中田 和秀 (Kazuhide Nakata)
京都大学 工学研究科 藤沢 克樹 (Katsuki Fujisawa)
東京工業大学 情報理工学研究科 福田 光浩 (Mituhiro Fukuda)
東京工業大学 情報理工学研究科 小島 政和 (Masakazu Kojima)
京都大学 数理解析研究所 室田 一雄 (Kazuo Murota)

This article succeds the previous article (Solving Sparse Semidefinite Programs by Matrix Completion (part I). In this article, we will propose a primal-dual interior-point method based on the sparse factorization formula which arises from positive definite matrix completion [3]. We call this method the *completion method.* Next we will present some numerical results which compare with the original method, the conversion method and the completion method.

1 Completion Method

One disadvantage of the conversion method in the previous article is an increase in the number of equality constraints. In this section, we propose a primal-dual interior-point method based on positive semidefinite matrix completion which we can apply directly to the original SDP without adding any equality constraints.

1.1 Search Direction

Various search directions [1, 4, 5, 6, 7, 8, 9, 10] have been proposed so far for primal-dual interior-point methods. Among others, we restrict ourselves to the HRVW/KSH/M search direction [4, 6, 7].

Let $(\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{z}})$ be a point obtained at the kth iteration of a primal-dual interior-point method using the HRVW/KSH/M search direction ($k \geq 1$) or given initially ($k = 0$). We assume that $\bar{\boldsymbol{X}} \in \mathcal{S}^n_{++}(F, ?)$ and $\bar{\boldsymbol{Y}} \in \mathcal{S}^n_{++}(E, 0)$.

In order to compute the HRVW/KSH/M search direction, we use the whole matrix values for both $\bar{\boldsymbol{X}} \in \mathcal{S}^n_{++}(F, ?)$ and $\bar{\boldsymbol{Y}} \in \mathcal{S}^n_{++}(E, 0)$, so that we need to make a positive definite matrix completion of $\bar{\boldsymbol{X}} \in \mathcal{S}^n_{++}(F, ?)$. Let $\hat{\boldsymbol{X}} \in \mathcal{S}^n_{++}$ be the maximum-determinant positive definite matrix completion of $\bar{\boldsymbol{X}} \in \mathcal{S}^n_{++}(F, ?)$. Then we compute the HRVW/KSH/M search direction $(d\boldsymbol{X}, d\boldsymbol{Y}, dz)$ by solving the system of linear equations

$$\left.\begin{array}{l} \boldsymbol{A}_p \bullet d\boldsymbol{X} = g_p \ (p = 1, 2, \ldots, m), \ d\boldsymbol{X} \in \mathcal{S}^n \\ \sum_{p=1}^{m} \boldsymbol{A}_p dz_p + d\boldsymbol{Y} = \boldsymbol{H}, \ d\boldsymbol{Y} \in \mathcal{S}^n(E, 0), \ dz \in \mathbb{R}^m, \\ \widetilde{d\boldsymbol{X}}\bar{\boldsymbol{Y}} + \hat{\boldsymbol{X}} d\boldsymbol{Y} = \boldsymbol{K}, \ d\boldsymbol{X} = (\widetilde{d\boldsymbol{X}} + \widetilde{d\boldsymbol{X}}^T)/2, \end{array}\right\} \quad (1)$$

where $g_p = b_p - \boldsymbol{A}_p \bullet \hat{\boldsymbol{X}} \in I\!R$ $(p = 1, 2, \ldots, m)$ (the primal residual), $\boldsymbol{H} = \boldsymbol{A}_0 - \sum_{p=1}^m \boldsymbol{A}_p \bar{z}_p - \bar{\boldsymbol{Y}} \in \mathcal{S}^n(E, 0)$ (the dual residual), $\boldsymbol{K} = \mu \boldsymbol{I} - \hat{\boldsymbol{X}}\bar{\boldsymbol{Y}}$ (an $n \times n$ constant matrix), and $\widetilde{d\boldsymbol{X}}$ denotes an $n \times n$ auxiliary matrix variable. The search direction parameter $\mu$ is usually chosen to be $\beta \hat{\boldsymbol{X}} \bullet \bar{\boldsymbol{Y}}/n$ for some $\beta \in [0, 1]$. We can reduce the system of linear equations (1) to

$$\left.\begin{array}{l} \boldsymbol{B} d\boldsymbol{z} = \boldsymbol{s}, \ d\boldsymbol{Y} = \boldsymbol{H} - \sum\limits_{p=1}^{m} \boldsymbol{A}_p dz_p, \\ \widetilde{d\boldsymbol{X}} = (\boldsymbol{K} - \hat{\boldsymbol{X}} d\boldsymbol{Y})\bar{\boldsymbol{Y}}^{-1}, \ d\boldsymbol{X} = (\widetilde{d\boldsymbol{X}} + \widetilde{d\boldsymbol{X}}^T)/2, \end{array}\right\} \tag{2}$$

where

$$\left.\begin{array}{l} B_{pq} = \text{Trace } \boldsymbol{A}_p \hat{\boldsymbol{X}} \boldsymbol{A}_q \bar{\boldsymbol{Y}}^{-1} \ (p = 1, 2, \ldots, m, \ q = 1, 2, \ldots, m), \\ s_p = g_p - \text{Trace } \boldsymbol{A}_p (\boldsymbol{K} - \hat{\boldsymbol{X}} \boldsymbol{H}) \bar{\boldsymbol{Y}}^{-1} \ (p = 1, 2, \ldots, m). \end{array}\right\}$$

Note that $\boldsymbol{B}$ is a positive definite symmetric matrix.

As we have seen in the previous article, The maximum-determinant positive definite matrix completion $\hat{\boldsymbol{X}}$ of $\bar{\boldsymbol{X}} \in \mathcal{S}^n_{++}(F, ?)$ can be expressed in terms of the sparse factorization formula

$$\hat{\boldsymbol{X}} = \boldsymbol{W}^{-T}\boldsymbol{W}^{-1}, \tag{3}$$

where $\boldsymbol{W}$ is a lower triangular matrix which enjoys the same sparsity structure as $\bar{\boldsymbol{X}}$. Also $\bar{\boldsymbol{Y}} \in \mathcal{S}^n_{++}(E, 0)$ is factorized as $\bar{\boldsymbol{Y}} = \boldsymbol{N}\boldsymbol{N}^T$ without any fill-in's except for entries in $F \backslash E$, where $\boldsymbol{N}$ is a lower triangular matrix. We can effectively utilize these factorizations of $\hat{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$ for the computation of the search direction $(d\boldsymbol{X}, d\boldsymbol{Y}, d\boldsymbol{z})$. In particular, the coefficients B_{pq} $(p = 1, 2, \ldots, m, \ q = 1, 2, \ldots, m)$ in the system (2) of linear equations is computed by

$$\begin{array}{rcll} B_{ij} & = & \sum\limits_{k=1}^{n} (\boldsymbol{W}^{-T}\boldsymbol{W}^{-1}\boldsymbol{e}_k)^T \boldsymbol{A}_j (\boldsymbol{N}^{-T}\boldsymbol{N}^{-1}[\boldsymbol{A}_i]_{*k}) & (i, j = 1, 2, \cdots, m), \\ s_i & = & b_i + \sum\limits_{k=1}^{n} (\boldsymbol{W}^{-T}\boldsymbol{W}^{-1}\boldsymbol{e}_k)^T \boldsymbol{H} (\boldsymbol{N}^{-T}\boldsymbol{N}^{-1}[\boldsymbol{A}_i]_{*k}). & \\ & & \quad - \mu \boldsymbol{e}_k^T \boldsymbol{N}^{-T}\boldsymbol{N}^{-1}[\boldsymbol{A}_i]_{*k} & (i = 1, 2, \cdots, m) \end{array}$$

Here $[\boldsymbol{A}_i]_{*k}$ denote the kth column of the matrix $\boldsymbol{A}_i$. Since we just store the sparse and lower triangular matrix $\boldsymbol{W}$ and $\boldsymbol{N}$ instead of $\bar{\boldsymbol{X}}$ and $\boldsymbol{Y}$, we can not use the current computational formulas. Instead, we compute $\boldsymbol{B}$, $\widetilde{d\boldsymbol{X}}$ and $\boldsymbol{s}$ based on matrix-vector multiplications. For instance, to determine the the product $\boldsymbol{h} = \boldsymbol{W}^{-1}\boldsymbol{v}$, we solve the linear system $\boldsymbol{W}\boldsymbol{h} = \boldsymbol{v}$, where $\boldsymbol{W}$ is lower triangular and sparse.

1.2 Step Size

As we will see below in the computation of the step length and the next iterate, we need the partial symmetric matrix with entries $[d\boldsymbol{X}]_{ij}$ specified in $F$, but not the whole search direction matrix $d\boldsymbol{X} \in \mathcal{S}^n$ in the primal space (hence the partial symmetric matrix with entries $[\widetilde{d\boldsymbol{X}}]_{ij}$ specified in $F$ but not the whole matrix $\widetilde{d\boldsymbol{X}}$). Hence it is possible to carry out all the matrix computation above using only partial matrices with entries specified in $F$. Therefore we can expect to save both CPU time and memory in our computation of the search direction. To clarify the distinction between the whole primal search direction matrix $d\boldsymbol{X} \in \mathcal{S}^n$ and the corresponding partial symmetric matrix with entries specified in F in the discussions below, we use the notation $d\hat{\boldsymbol{X}}$ for the former whole matrix in $\mathcal{S}^n$ and $d\bar{\boldsymbol{X}}$ for the latter partial symmetric matrix in $\mathcal{S}^n(F;?)$. Now, supposing that we have computed the HRVW/KSH/M search direction $(d\hat{\boldsymbol{X}}, d\boldsymbol{Y}, dz) \in \mathcal{S}^n \times \mathcal{S}^n(E,0) \times \mathbb{R}^m$, and we describe how to compute a step length $\alpha > 0$ and the next iterate $(\boldsymbol{X}', \boldsymbol{Y}', \boldsymbol{z}') \in \mathcal{S}^n \times \mathcal{S}^n(E,0) \times \mathbb{R}^m$. Usually we compute the maximum $\hat{\alpha}$ of α's satisfying

$$\hat{\boldsymbol{X}} + \alpha d\hat{\boldsymbol{X}} \in \mathcal{S}^n_+ \text{ and } \bar{\boldsymbol{Y}} + \alpha d\boldsymbol{Y} \in \mathcal{S}^n_+, \tag{4}$$

and let $(\boldsymbol{X}', \boldsymbol{Y}', \boldsymbol{z}') = (\hat{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{z}}) + \gamma\hat{\alpha}(d\hat{\boldsymbol{X}}, d\boldsymbol{Y}, dz)$ for some $\gamma \in (0,1)$. Then $\boldsymbol{X}' \in \mathcal{S}^n_{++}$ and $\boldsymbol{Y}' \in \mathcal{S}^n_{++}(E,0)$. The computation of $\hat{\alpha}$ is necessary to know how long we can take the step length along the search direction $(d\hat{\boldsymbol{X}}, d\boldsymbol{Y}, dz)$. The computation of $\hat{\alpha}$ is usually carried out by calculating the minimum eigenvalues of the matrices

$$\hat{\boldsymbol{M}}^{-1} d\hat{\boldsymbol{X}} \hat{\boldsymbol{M}}^{-T} \text{ and } \boldsymbol{N}^{-1} d\boldsymbol{Y} \boldsymbol{N}^{-T},$$

where $\hat{\boldsymbol{X}} = \hat{\boldsymbol{M}}\hat{\boldsymbol{M}}^T$ and $\bar{\boldsymbol{Y}} = \boldsymbol{N}\boldsymbol{N}^T$ denote the factorizations of $\hat{\boldsymbol{X}}$ and $\bar{\boldsymbol{Y}}$, respectively.

Instead of (4), we propose to employ

$$\bar{\boldsymbol{X}}_{C_rC_r} + \alpha d\bar{\boldsymbol{X}}_{C_rC_r} \in \mathcal{S}^{C_r}_+ \ (r = 1, 2, \ldots, \ell) \text{ and } \bar{\boldsymbol{Y}} + \alpha d\boldsymbol{Y} \in \mathcal{S}^n_+(E,0). \tag{5}$$

Recall that $\{C_r \subseteq V : r = 1, 2, \ldots, \ell\}$ denotes the family of maximal cliques of $G(V, F^\circ)$ and $\ell \leq n$. Let $\bar{\alpha}$ be the maximum of α's satisfying (5), and let

$$(\boldsymbol{X}', \boldsymbol{Y}', \boldsymbol{z}') = (\bar{\boldsymbol{X}}, \bar{\boldsymbol{Y}}, \bar{\boldsymbol{z}}) + \gamma\bar{\alpha}(d\bar{\boldsymbol{X}}, d\boldsymbol{Y}, dz) \in \mathcal{S}^n(F,?) \times \mathcal{S}^n_{++}(E,0) \times \mathbb{R}^m$$

for some $\gamma \in (0,1)$. $\boldsymbol{X}' \in \mathcal{S}^n(F,?)$ has a positive definite matrix completion, so that the point $(\boldsymbol{X}', \boldsymbol{Y}', \boldsymbol{z}') \in \mathcal{S}^n_{++}(F,?) \times \mathcal{S}^n_{++}(E,0) \times \mathbb{R}^m$ can be the next iterate. In this case, the computation of $\bar{\alpha}$ is reduced to the computation of the minimum eigenvalues of the matrices

$$\bar{\boldsymbol{M}}_r^{-1} d\bar{\boldsymbol{X}}_{C_rC_r} \bar{\boldsymbol{M}}_r^{-T} \ (r = 1, 2, \ldots, \ell) \text{ and } \boldsymbol{N}^{-1} d\boldsymbol{Y} \boldsymbol{N}^{-T},$$

where $\bar{\boldsymbol{X}}_{C_rC_r} = \bar{\boldsymbol{M}}_r\bar{\boldsymbol{M}}_r^T$ denotes a factorization of $\bar{\boldsymbol{X}}_{C_rC_r}$ $(r = 1, 2, \ldots, \ell)$. Thus the computation of the minimum eigenvalue of $\hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{X}}\hat{\boldsymbol{M}}^{-T}$ has been replaced by the computation of the minimum eigenvalues of ℓ smaller submatrices $\bar{\boldsymbol{M}}_r^{-1} d\bar{\boldsymbol{X}}_{C_rC_r} \bar{\boldsymbol{M}}_r^{-T}$ $(r = 1, 2, \ldots, \ell)$. On the other hand, when we compute the minimum eigenvalue of $\boldsymbol{N}^{-1} d\boldsymbol{Y} \boldsymbol{N}^{-T}$, we can compute it easily by Lanczos methods, because $\boldsymbol{N}$ and $d\boldsymbol{Y}$ are sparse and we can compute the multiplication between $\boldsymbol{N}^{-1} d\boldsymbol{Y} \boldsymbol{N}^{-T}$ and vector $\boldsymbol{v}$ using technic we mentioned.

2 Numerical Experiments

All numerical experiments in this section were executed on DEC Alpha Station (CPU Alpha 21164-600MHz with 1024MB). For the original method, we used the SDPA 5.0[2]. For the conversion method, we first converted the SDPs into block diagonal SDPs and solved them by the SDPA[2]. For the completion method, we used a new software called SDPA-C which we incorporated the sparse factorization formula.

2.1 Norm Minimization Problems

Let $\boldsymbol{F}_i \in \mathbb{R}^{q \times r}$ $(0 \leq i \leq t)$. The norm minimization problem is defined as:

$$\left.\begin{array}{ll} \text{minimize} & \left\| \boldsymbol{F}_0 + \sum_{i=1}^{t} \boldsymbol{F}_i z_i \right\| \\ \text{subject to} & z_i \in \mathbb{R} \ (1 \leq i \leq t). \end{array}\right\}$$

We can reduce this problem to an SDP:

$$\left.\begin{array}{ll} \text{maximize} & -z_{t+1} \\ \text{subject to} & \sum_{i=1}^{t} \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_i^T \\ \boldsymbol{F}_i & \boldsymbol{O} \end{pmatrix} z_i + \begin{pmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{I} \end{pmatrix} z_{t+1} + \begin{pmatrix} \boldsymbol{O} & \boldsymbol{F}_0^T \\ \boldsymbol{F}_0 & \boldsymbol{O} \end{pmatrix} \in \mathcal{S}_+^{q+r}. \end{array}\right\}$$

Table 1 shows the performance comparisons among the original method (SDPA), the conversion method applied before solving with SDPA, and the completion method. As q becomes larger, the aggregate sparsity pattern and extended sparsity pattern becomes denser.

Table 1: Numerical results on norm minimization problems

			original		conversion		completion	
q	n $(q+r)$	m $(t+1)$	CPU (s)	memory (MB)	CPU (s)	memory (MB)	CPU (s)	memory (MB)
1	1000	10	3492.0	321	4.7	11.3	100.0	5.18
2	1000	10	4263.0	321	9.7	16.0	157.7	5.98
5	1000	10	5387.0	321	28.4	26.9	378.5	10.2
10	1000	10	6941.4	321	86.4	42.6	696.1	17.8
20	1000	10	7233.3	321	326.6	84.1	2484.1	44.9
50	1000	10	7844.1	321	2195.3	192.0	—	—

In the case of norm minimization problems, the conversion method is better than the other methods.

2.2 Quadratic Programs with Box Constraints

Let $\boldsymbol{Q} \in \mathcal{S}^n$ and $\boldsymbol{q} \in \mathbb{R}^n$. The quadratic program with box constraints is defined as:

$$\left.\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{q}^T\boldsymbol{x} \\ \text{subject to} & -1 \leq x_i \leq 1 \ (i = 1, 2, \cdots, n). \end{array}\right\}$$

We have the following semidefinite programming relaxation of the above problem

$$\left.\begin{array}{ll} \text{minimize} & \frac{1}{2}\begin{pmatrix} 0 & q^T & 0^T \\ q & Q & O \\ 0 & O & O \end{pmatrix} \bullet X \\ \text{subject to} & \begin{pmatrix} 1 & 0^T & 0^T \\ 0 & O & O \\ 0 & O & O \end{pmatrix} \bullet X = 1, \\ & \begin{pmatrix} 0 & 0^T & 0^T \\ 0 & E_{ii} & O \\ 0 & O & E_{ii} \end{pmatrix} \bullet X = 1 \ \ (i = 1, 2, \cdots, n), \quad X \in \mathcal{S}^{2n+1}_+. \end{array}\right\}$$

Here E_{ii} denotes the $n \times n$ symmetric matrix with (i, i)th element 1 and all others 0.

Table 2 shows the performance comparisons of these methods applied to this particular class of problems. α denotes the sparsity of the objective function matrix. The entry "1001,1000" in the n column means that the primal matrix variable X has block matrices of sizes 1001×1001 and 1000×1000.

Table 2: Numerical results on relaxation of the quadratic programs with box constraints

			original		conversion		completion	
			CPU	memory	CPU	memory	CPU	memory
α	n	m	(s)	(MB)	(s)	(MB)	(s)	(MB)
0.0020	1001,1000	1001	3782.2	316	1153.8	147	758.0	18.1
0.0025	1001,1000	1001	3771.0	316	1681.6	183	877.2	22.2
0.0030	1001,1000	1001	3648.4	316	2179.4	225	1086.7	29.1
0.0035	1001,1000	1001	3634.6	316	2561.7	251	1417.5	37.4
0.0040	1001,1000	1001	3629.3	316	3104.8	277	2090.7	56.8

In the case of SDP relaxations of quadratic programs with box constraints, the completion method is better than the other methods.

2.3 Max-cut Problems over Lattice Graphs

Let $G = (V, E)$ be a lattice graph which is of size $k1 \times k2$ with a vertex set $V = \{1, 2, \ldots, n\}$, and an edge set $E = \{(i, j) : i, j \in V, \ i < j\}$. We assign a weight $C_{ij} = C_{ji}$ to each edge $(i, j) \in E$. The maximum cut problem is to find a partition (L, R) of V that maximizes the *cut* $c(L, R) = \sum_{i \in L, j \in R} C_{ij}$. Introducing a variable vector $u \in \mathbb{R}^n$, we can reformulate the problem as a nonconvex quadratic program:

$$\text{maximize} \quad \frac{1}{2}\sum_{i<j} C_{ij}(1 - u_i u_j) \ \text{subject to} \quad u_i^2 = 1 \ (1 \leq i \leq n). \ \Big\}$$

Here each feasible solution $u \in \mathbb{R}^n$ of this problem corresponds to a cut (L, R) with $L = \{i \in V : u_i = -1\}$ and $R = \{i \in V : u_i = 1\}$. If we define C to be the $n \times n$ symmetric matrix with elements $C_{ji} = C_{ij}$ $((i, j) \in E)$ and $C_{ij} = 0$ $((i, j) \notin E)$, and the $n \times n$ symmetric matrix $A_0 \in \mathcal{S}^n$

by $A_0 = \mathrm{diag}(Ce) - C$, where $e \in \mathbb{R}^n$ denotes the vector of ones and $\mathrm{diag}(Ce)$ the diagonal matrix of the vector $Ce \in \mathbb{R}^n$, we can obtain the following semidefinite programming relaxation of the maximum cut problem:

$$\left.\begin{array}{ll} \text{minimize} & -A_0 \bullet X \\ \text{subject to} & E_{ii} \bullet X = 1/4 \ (1 \le i \le n), \ X \in \mathcal{S}^n_+. \end{array}\right\}$$

Table 3 compares the three algorithms for this problem. As $k1$ becomes larger, the aggregate sparsity patterns remain sparse, but the extended sparsity pattern becomes denser.

Table 3: Numerical results on relaxation of the maximum cut problems

			original		conversion		completion	
			CPU	memory	CPU	memory	CPU	memory
$k_1 \times k_2$	n	m	(s)	(MB)	(s)	(MB)	(s)	(MB)
2 × 500	1000	1000	2984.2	315	133.0	53.0	183.7	15.5
4 × 250	1000	1000	2971.5	315	168.5	73.3	213.7	16.9
5 × 200	1000	1000	2815.1	315	184.7	83.6	233.1	17.8
8 × 125	1000	1000	2972.6	315	221.5	85.7	256.6	18.8
10 × 100	1000	1000	2835.9	315	263.2	96.8	320.3	19.3
20 × 50	1000	1000	3125.2	315	730.5	118.0	347.8	21.9
25 × 40	1000	1000	3118.8	315	804.9	180.0	337.7	21.8

For the semidefinite programming relaxation of maximum cut problems over lattice graphs, the conversion and the completion methods are better.

2.4 Semidefinite Programming Relaxation of Graph-partition Problem

Let $G = (V, E)$ be a lattice graph which is of size $k1 \times k2$ with a vertex set $V = \{1, 2, \dots, n\}$, and an edge set $E = \{(i,j) : i,j \in V, \ i < j\}$. We assign a weight $C_{ij} = C_{ji}$ to each edge $(i,j) \in E$. We assume that n is an even number. The graph partition problem is to find a uniform partition (L, R) of V, *i.e.*, a partition (L, R) of V with the same cardinality $|L| = |R| = n/2$, that minimizes the cut $c(L, R) = \sum_{i \in L, j \in R} C_{ij}$. This problem is formulated as a nonconvex quadratic program:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\sum_{i<j} C_{ij}(1 - u_i u_j) \\ \text{subject to} & (\sum_{i=1}^n u_i)^2 = 0, \ u_i^2 = 1 \ (1 \le i \le n). \end{array}$$

As in the maximum cut problem, we can derive a semidefinite programming relaxation of the graph partition problem:

$$\left.\begin{array}{ll} \text{minimize} & A_0 \bullet X \\ \text{subject to} & E_{ii} \bullet X = 1/4 \ (1 \le i \le n), \ E \bullet X = 0, \ X \in \mathcal{S}^n_+. \end{array}\right\} \qquad (6)$$

Here A_0 and E_{ii} $(1 \le i \le n)$ are the same matrices as in the previous section, and E denotes the $n \times n$ matrix with all elements 1. Table 4 compares the three algorithms for this problem. As $k1$

Table 4: Numerical results on relaxation of the graph partition problems

			original		conversion		completion	
			times	memory	times	memory	times	memory
$k_1 \times k_2$	n	m	(s)	(MB)	(s)	(MB)	(s)	(MB)
2×500	1000	1001	4065.7	315	221.6	53.1	439.9	18.0
4×250	1000	1001	4073.1	315	212.6	74.8	563.0	22.8
5×200	1000	1001	4065.3	315	511.7	125	667.7	26.8
8×125	1000	1001	2843.1	315	1341.2	206	806.8	25.4
10×100	1000	1001	4065.7	315	740.3	168	892.7	27.2
20×50	1000	1001	4064.5	315	2481.5	315	1323.0	31.6
25×40	1000	1001	3912.3	315	3918.0	315	1333.2	34.1

becomes larger, the aggregate sparsity patterns remain sparse, but the extended sparsity pattern becomes denser.

For the semidefinite programming relaxation of graph partition problems over lattice graphs, the conversion and the completion methods are better.

3 Concluding Remarks

In this article, we explained the mechanisms of the completion method. Next we presented some numerical results which compared the original method, the conversion method and the completion method. As a result, we confirmed the effectiveness of the conversion method and the completion method.

References

[1] F. Alizadeh, J.-P. A. Haeberly and M. L. Overton, *Primal-dual interior-point methods for semidefinite programming: convergence rates, stability and numerical results*, SIAM J. Optim., 8 (1998), pp. 746–768.

[2] 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, Japan, December 1995 (revised August 1996). Available at ftp://ftp.is.titech.ac.jp/pub/OpRes/software/SDPA.

[3] M. Fukuda, M. Kojima, K. Murota and K. Nakata, *Exploiting sparsity in semidefinite programming via matrix completion I: general framework*, SIAM J. Optim., to appear.

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

[5] M. Kojima, M. Shida and S. Shindoh, "Search directions in the SDP and the monotone SDLCP: generalization and inexact computation," *Mathematical Programming* **85** (1999) 51–80.

[6] M. Kojima, S. Shindoh and S. Hara, "Interior-point methods for the monotone semidefinite linear complementarity problems," *SIAM Journal on Optimization* **7** (1997) 86–125.

[7] R. D. C. MONTEIRO, *Primal-dual path-following algorithms for semidefinite programming*, SIAM J. Optim., 7 (1997), pp. 663–678.

[8] R.D.C. Monteiro and Y. Zhang, "A unified analysis for a class of path-following primal-dual interior-point algorithms for semidefinite programming," *Mathematical Programming* **81** (1998) 281–299.

[9] YU. E. NESTEROV AND M. J. TODD, *Primal-dual interior-point methods for self-scaled cones*, SIAM J. Optim., 8 (1998), pp. 324–364.

[10] 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, to appear in *SIAM Journal on Optimization.*