

International Journal of LOGISTICS and SCM Systems
Offprint

Volume2 Number1 Nov., 2007

Parallel Solver for Semidefinite Programming

Makoto Yamashita, Katsuki Fujisawa and Kazuhide Nakata

Official Journal of IFLS:
The International Federation of LOGISTICS and SCM Systems
The Asia Pacific Federation of LOGISTICS and SCM Systems

Parallel Solver for Semidefinite Programming

Makoto Yamashita^{*A}, Katsuki Fujisawa^{*B} and Kazuhide Nakata^{*C}
^{*A} Kanagawa University, e-mail: Makoto Yamashita is.kanagawa-u.ac.jp
^{*B} Chuo University, e-mail: fujisawa indsys.chuo-u.ac.jp
^{*C} Tokyo Institute of Technology, e-mail: knakata me.titech.ac.jp

Abstract

A location problem to find the minimum-cost assignment of given plants to candidate locations is a typical combinatorial optimization problem in logistics. When the number of plants increases, however, the computational cost of the location problem grows up extremely. If we enumerated all possibilities, the cost would be proportional to the factorial of the number of plants. In general, meta-heuristic techniques, for example, Genetic Algorithm, are employed to find an approximate solution. Although meta-heuristic techniques are very useful, they cannot assure the accuracy of the solution, because they search only feasible solutions, hence their objective values are always only upper bounds of the optimal value. To assure the accuracy, the lower bound is necessary.

Semidefinite Programming relaxation method has attracted attention of researchers of combinatorial optimization. Semidefinite Programming (SDP) is a Linear Programming extended to the space of symmetric matrices with the constraints of positive semidefiniteness. SDP provides efficient computational schemes for various applications, such as control theory and quantum chemistry.

By ignoring difficult conditions like a rank condition of a variable matrix, many combinatorial optimization problems can be converted to standard SDP form. The location problem can also be one SDP application via Quadratic Assignment Problem.

In particular, the optimal value of the SDP relaxation is an excellent lower bound of the original problem. Therefore, we can detect the region where the optimal value of the original problem exists by integration with some meta-heuristic techniques studied in combinatorial optimization field.

In this paper, we focus on SDPs generated by an SDP relaxation method for the location problem. From the theoretical view, the computational cost to solve an SDP is shrunk to polynomial time. However, when we examine the location problems with many plants, the computational cost of the generated SDP obviously cannot be remain small. Therefore, the computational resource of single processor becomes insufficient for large-scale SDPs.

To solve extremely large SDPs, we have developed an SDP solver on a parallel computation environment.

We show the performance of the parallel SDP solver is promising to solve large SDPs from the location problem in a short time. Furthermore, the parallel solver enables us to extremely large SDPs which we could not handle on single processor computers.

Keywords: Semidefinite Programming, Location Problem, Quadratic Assignment Problem, Parallel Computation

1. Introduction

The concern with Semidefinite Programming (SDP) has been growing in the field of Mathematical Optimization. SDP supplies efficient and effective computational schemes to wide-range applications including control theory and combinatorial optimization. For other applications, see [11, 12, 13]. In recent years, the range of the applications covers quantum chemistry, quantum information and polynomial optimization and is extending. Over these twenty years, a considerable number of researches have been conducted on theoretical analysis and have implemented software packages. The representative packages among them are SDPA[14], SDPT3[10], SeDuMi[9].

From the viewpoints of logistics, the computational efficiency of an SDP relaxation method for NP-hard combinatorial optimization problems is worthy of notes. In particular, Quadratic Assignment Problem (QAP) introduced by [5] arising from a location problem is a typical NP-hard problem and the SDP relaxation method can compute efficiently a prominent lower bound of its optimal value.

The size of SDPs we are challenging in these years, however, often exceeds the capacity of single-processor computers due to the lack of memory space and the very long computational time. For example, when we solve an SDP generated from the location problem with N plants, the variables of the SDP become $(N^2 + 1) \times (N^2 + 1)$ symmetric matrices, and the number of constraints rapidly grows up in at least $O(N^3)$ to assure the

accuracy. In addition, SDPs arising from quantum chemistry usually requires more computational resources.

On the other hand, recent developments in parallel computation provide us the possibility to solve such extremely large mathematical optimization problem. Throughout these years, basing on SDPA, we have implemented parallel SDP solver, SDPARA (SemiDefinite Programming Algorithm paRAllel version) [15]. By replacing two computational bottlenecks by their parallel implementation, SDPARA gets great performance of parallel computation out and attains remarkable reduction in the computational time. Furthermore, the distributed memory space attached to parallel computers further increases the size of the solvable SDP.

The paper is organized as follows. Section 2 discusses an SDP relaxation method to reduce the location problem to the standard SDP form. In Section 3, we present algorithmic framework of Primal-Dual Interior-Point Methods (PDIPM), which are well-designed methods for SDP from both theoretical and practical viewpoints, and parallel schemes of PDIPM implemented in SDPARA. Section 4 gives some numerical results of SDPARA and discusses its performance. In Section 5, we conclude this paper and discuss further directions

Before introducing an SDP relaxation method for QAP, let us briefly review mathematical notations and the definitions of SDP. The remarkable feature of SDP is that SDP is an extension of Linear Programming to the space of symmetric matrices. We use S^n and $R^{m \times n}$ to denote the space of $n \times n$ symmetric matrices and the space of $m \times n$ matrices, respectively. The inner-product $X \bullet Y$ for $X, Y \in S^n$ is Hilbert-Schmit inner-product,

$$i.e., X \bullet Y = \sum_{i=1}^n \sum_{j=1}^n X_{ij} Y_{ij}. The inner-product$$

between $X, Y \in R^{n \times n}$ is also computed by

$$X \bullet Y = \sum_{i=1}^n \sum_{j=1}^n X_{ij} Y_{ij}. The notation$$

$X \succeq O$ ($X \succ O$) for $X \in S^n$ denotes X is positive semidefinite (positive definite), respectively.

The standard SDP form is defined by primal form P or dual form D .

$$\begin{aligned} P: \min \quad & C \bullet X \\ \text{s.t.} \quad & A_k \bullet X = b_k \quad (k=1,2,..,m) \quad (1) \\ & X \succeq O \end{aligned}$$

$$\begin{aligned} D: \max \quad & \sum_{k=1}^m b_k z_k \\ \text{s.t.} \quad & \sum_{k=1}^m A_k z_k + Y_k = C \\ & Y \succeq O \end{aligned}$$

The input data are $A_1, A_2, \dots, A_m, C \in S^n$ and $b_1, \dots, b_m \in R$, while the variable in P is $X \in S^n$ and those in D are $(Y, z) \in (S^n, R^{k \times 1})$. We call (X, Y, z) feasible if X and (Y, z) satisfy constraints of P and D , simultaneously. In addition to the feasibility, when $X \succ O$ and $Y \succ O$ are satisfied, (X, Y, z) is called an interior-point.

Throughout this paper, we call the set of primal and dual forms SDP simply. It is well known that their optimal values coincide with each other precisely under mild conditions.

2. Reduction of Location Problem to SemiDefinite Programming

In this section, we introduce an approach to reduce a location problem into an SDP via QAP. Further details can be found at [7, 17]. The most famous SDP relaxation method for NP-hard combinatorial optimization problem was studied in [2] for Max-Cut Problem. They ignored the difficult constraint, the rank-one condition of the variable matrix. Even though removing the constraint, they achieve an approximate lower bound of the max-cut problem with sufficient accuracy. The essential approach to reduce a location problem to an SDP here is also ignoring the rank-one condition.

Let us consider a location problem with N plants and N locations proposed for the plants. The transportation volume from i th plants to j th is f_{ij} units and the transportation cost from k th location to l location for each unit is g_{kl} . We assume the volume and the cost are symmetric, i.e., $f_{ij} = f_{ji}$ and $g_{kl} = g_{lk}$. In addition, h_{ik} denotes the cost to build i th plant at k th location. If we build i th plant at k th location, we set $w_{ik} = 1$, otherwise $w_{ik} = 0$.

The objective of the location problem we consider now is to minimize the total cost as below.

$$\begin{aligned}
& \sum_{i=1}^N \sum_{j=1}^N \sum_{k=1}^N \sum_{l=1}^N f_{ij} g_{kl} w_{ik} w_{jl} \\
\min : & \quad + \sum_{i=1}^N \sum_{k=1}^N h_{ik} w_{ik} \\
\text{s.t.} & \quad \sum_{k=1}^N w_{ik} = 1 (i=1, \dots, N) \\
& \quad \sum_{i=1}^N w_{ik} = 1 (k=1, \dots, N) \\
& \quad w_{ik} \in \{0, 1\} \\
& \quad (i=1, \dots, N, k=1, \dots, N)
\end{aligned}$$

By introducing the matrices $F, G \in S^n$ and $H, W \in R^{N \times 1}$ whose (i, j) th elements are $f_{ij}, g_{ij}, h_{ij}, w_{ij}$, respectively, we rewrite the location problem in the QAP form.

QAP

$$\begin{aligned}
\min : & \quad (FWG) \bullet W + H \bullet W \\
& \quad W^T W = I \\
\text{s.t.} & \quad WW^T = I \\
& \quad w_{ij}^2 = w_{ij} \quad (i=1, \dots, N, j=1, \dots, N)
\end{aligned}$$

A point to emphasize in the above form is that the feasible region of W corresponds to the set of permutation matrices of order N .

We define some notation to lift the QAP to S^{N^2+1} space. Stacking all columns of $W \in R^{N \times N}$, we define the vector $\text{vec}(W) \in R^{N^2 \times 1}$,

$$\text{vec}(W) = \begin{pmatrix} w_{11} \\ w_{21} \\ \vdots \\ w_{N1} \\ w_{12} \\ w_{22} \\ \vdots \\ w_{N2} \\ \vdots \\ w_{1N} \\ w_{2N} \\ \vdots \\ w_{NN} \end{pmatrix}$$

In addition, for matrices $A, B \in R^{N \times N}$, we use the tensor product $A \otimes B \in R^{N^2 \times N^2}$, that is, $((i-1) \times N + k, (j-1) \times N + l)$ th element of $A \otimes B$ corresponds to $a_{ij} b_{kl}$. The vector $e_i \in R^{N \times 1}$ denotes a vector whose i th element is 1 and others are 0.

We convert input data to some matrices of S^{N^2+1} ,

$$A_{ij}^1 = \begin{pmatrix} \frac{1}{2}(e_i e_j^T + e_j e_i^T) \otimes I & 0 \\ 0 & 0 \end{pmatrix},$$

$$A_{ij}^2 = \begin{pmatrix} \frac{1}{2} I \otimes (e_i e_j^T + e_j e_i^T) & 0 \\ 0 & 0 \end{pmatrix},$$

$$A_{ij}^3 = \begin{pmatrix} (e_j \otimes e_i)(e_j \otimes e_i)^T & -\frac{1}{2}(e_j \otimes e_i) \\ -\frac{1}{2}(e_j \otimes e_i)^T & 0 \end{pmatrix}$$

for $i=1, \dots, N, j=1, \dots, N$ and

$$A^4 = \begin{pmatrix} O & 0 \\ 0^T & 1 \end{pmatrix},$$

$$C = \begin{pmatrix} G \otimes F & \frac{1}{2} \text{vec}(H) \\ \frac{1}{2} \text{vec}(H)^T & 0 \end{pmatrix}.$$

These definitions enable us to have an equivalent QAP in the large matrix form, QAP-1.

QAP-1

$$\begin{aligned}
\min : & \quad C \bullet X \\
& \quad A_{ij}^1 \bullet X = \delta_{ij} (i=1, \dots, N, j=1, \dots, N) \\
& \quad A_{ij}^2 \bullet X = \delta_{ij} (i=1, \dots, N, j=1, \dots, N) \\
\text{s.t.} & \quad A_{ij}^3 \bullet X = 0 (i=1, \dots, N, j=1, \dots, N) \\
& \quad A^4 \bullet X = 1 \\
& \quad X \succeq O \\
& \quad \text{rank}(X) = 1
\end{aligned}$$

The equivalence of the above QAP and QAP-1 can be understood by the component form of the variable matrix $X \in S^{N^2+1}$,

$$X = \begin{pmatrix} \text{vec}(W) \text{vec}(W)^T & \text{vec}(W) \\ \text{vec}(W)^T & X_{N^2+1, N^2+1} \end{pmatrix}.$$

The most difficult constraint in QAP-1, which involves NP-hardness, is the rank-one condition for the variable matrix X .

Ignoring the rank-one constraint, we finally obtain an SDP relaxation for the QAP.

QAP-SDP

$$\begin{aligned}
\min : & \quad C \bullet X \\
& \quad A_{ij}^1 \bullet X = \delta_{ij} (i=1, \dots, N, j=1, \dots, N) \\
& \quad A_{ij}^2 \bullet X = \delta_{ij} (i=1, \dots, N, j=1, \dots, N) \\
\text{s.t.} & \quad A_{ij}^3 \bullet X = 0 (i=1, \dots, N, j=1, \dots, N) \\
& \quad A^4 \bullet X = 1 \\
& \quad X \succeq O
\end{aligned}$$

Of course, removing the rank-one constraint may let the optimal value of QAP-SDP be strictly lower than that of QAP. In general, to tighten the gap, some additional linear constraints are added. Effective additional constraints are studied in [8]. They show that the gap obtained by the refined SDPs is less than

10% of the optimal values of QAP for most problems. As a result, we can use the SDP relaxation method to obtain excellent lower bounds of the location problems.

3. Interior-Point Method and Parallel Computation

The numbers of methods to solve SDPs have been proposed during the last two decades. Primal-Dual Interior-Point Method (PDIPM)[3,4,6] is an outstanding method among them. In particular, polynomial time computational complexity and sophisticated computer software packages have accelerated PDIPM researches. We review its algorithmic framework and computational bottlenecks on single-processor, then discuss the replacements of the bottlenecks by their parallel version implemented in SDPARA[15].

The most remarkable feature of PDIPM is that it solves P and D in (1) simultaneously, keeping positive definiteness of the variable matrices X and Y . An optimal solution of (1) is characterized by the KKT condition.

$$KKT : \begin{cases} A_k \bullet X = b_k (k = 1, 2, \dots, m) \\ \sum_{k=1}^m A_k z_k + Y = C \\ XY = O \\ X \succeq O, Y \succeq O \end{cases}$$

The central path defined below is a key aspect to understand the PDIPM framework.

CentralPath :

$$\left\{ \begin{array}{l} \mu > 0 \\ A_k \bullet X_\mu = b_k (k = 1, \dots, m) \\ (X_\mu, Y_\mu, z_\mu) : \sum_{k=1}^m A_k (z_\mu)_k + Y_\mu = C \\ X_\mu Y_\mu = \mu I \\ X_\mu \succeq O, Y_\mu \succeq O \end{array} \right.$$

The central path consists of a smooth curve and it converges to a point which satisfies the KKT condition when $\mu \rightarrow 0$. In addition, the equation

$$X_\mu \bullet Y_\mu = n\mu \text{ holds on the central path.}$$

The algorithmic framework of PDIPM is to trace the central path numerically decreasing μ toward 0 as follows.

Framework of Primal-Dual Interior-Point Method:

1. We start from an initial point (X^0, Y^0, z^0) with $X \succ O, Y \succ O$ and set $p = 0$.
2. We compute a search direction $(\Delta X, \Delta Y, \Delta z)$ towards a point on the central path $(X_{\beta\mu^p}, Y_{\beta\mu^p}, z_{\beta\mu^p})$, where $0 < \beta < 1$ and $\mu^p = \frac{X^p \bullet Y^p}{n}$, by a modified Newton method ignoring the conditions $X \succeq O$ and $Y \succeq O$.
3. To keep positive definiteness, we compute a step length $\alpha = \max\{\alpha \in [0,1] : X^p + \alpha\Delta X \succeq O, Y^p + \alpha\Delta Y \succeq O\}$ and set $(X^{p+1}, Y^{p+1}, z^{p+1}) \leftarrow (X^p, Y^p, z^p) + \gamma\alpha(\Delta X, \Delta Y, \Delta z)$, where $0 < \gamma < 1$.
4. We set $p \leftarrow p + 1$. We return to step 2 until μ^p gets enough close to 0.

The PDIPM is implemented in the software package, SDPA[14], developed by SDPA Project¹. When we apply SDPA to SDPs from various applications on a single processor, however, most computation time is devoted to compute the search direction $(\Delta X, \Delta Y, \Delta z)$.

The computation can be reduced to solve the so-called *Schur complement equation*,

$$B\Delta z = r.$$

The coefficient matrix B is called *Schur complement matrix*, and its elements are evaluated by $B_{ij} = (XA_i Y^{-1}) \bullet A_j (i = 1, \dots, m, j = 1, \dots, m)$ (2)

Throughout all the iterations of the PDIPM, B is always positive definite. Therefore, we apply the Cholesky factorization to B to solve the Schur complement equation efficiently. However, the evaluation of the elements of B and its Cholesky factorization occupy more than 95% computational time on single processor. We call these computational bottlenecks as ELEMENTS and CHOLESKY. Actually, when we execute SDPA to QAP-SDP with $N = 20$ (the number of plants) on single-processor (Athlon 3800+, 1GB memory space), the computational time for ELEMENTS is 24% and CHOLESKY is 74%. Therefore, to shorten these two bottlenecks is necessary to solve QAP-SDP in the short time. In particular, when N becomes larger, the effect of the parallel computation is prominent, since the occupation by the bottlenecks is magnified.

The key point to evaluate the elements of B on a parallel environment is that each row can be

¹<http://homepage.mac.com/klabitech/sdpa-hompage>

computed completely independent from other rows. Let U be the number of available processors. We assign the processors to each row in a cyclic manner. Therefore, the set R_u of rows of B evaluated by u th processor is

$$R_u = \{i : 1 \leq i \leq m, (i-1)\%U = u-1\} \quad (u = 1, \dots, U),$$

where $a\%b$ is the remainder of a divided by b .

We call the assignment of the processors by R_u as *row-wise distribution*. Fig 1 is an example of the row-wise distribution for 7×7 Shcur complement matrix with 4 processors. The row-wise distribution appears to be very simple. However, all evaluations can be done without any network communications. This fact is very important to draw the performance of parallel computation out.

After the evaluation, we apply the Cholesky factorization to B . We adopt the parallel Cholesky

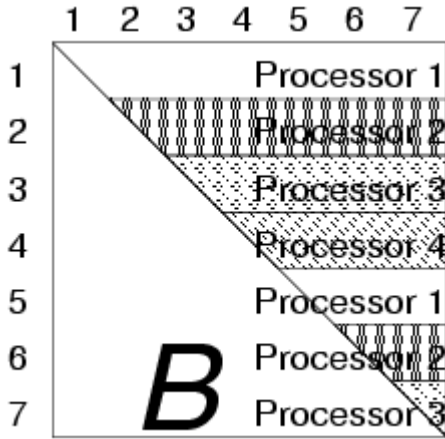


Fig 1: The row-wise distribution

factorization provided by ScaLAPACK². Since ScaLAPACK assumes the matrix to be factorized is distributed in the style of Two-Dimensional Block-Cyclic Distribution (TD-BCD), we redistribute the matrix on B from the row-wise distribution to TD-BCD on the distributed memory space using network communications. Fig 2 shows the TD-BCD of the same case as Fig1. For example, (4,2) element is stored on 3rd processor, while (2,5) element is stored on 1st processor. The cost of network communications for the redistribution, of course, increases the computational time. However, the cost can be neglected because the parallel Cholesky factorization on TD-BCD is much faster than on the row-wise distribution.

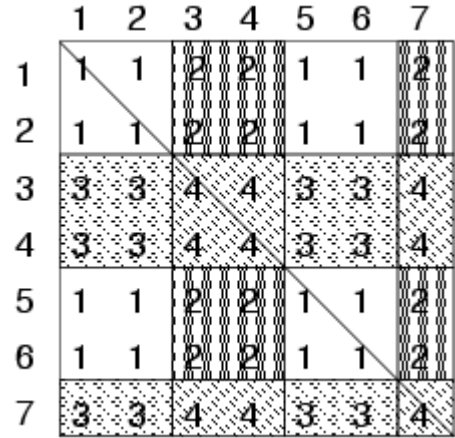


Fig 2: Two-Dimensional Block-Cyclic Distribution

Name	N	m	n
QAP-SDP12	12	1753	145
QAP-SDP16	16	4129	257
QAP-SDP20	20	8041	401
QAP-SDP24	24	13873	577
QAP-SDP26	26	17629	677
QAP-SDP28	28	22009	785
QAP-SDP30	30	27061	901
QAP-SDP32	32	32833	1025

Table 1: The size of QAP-SDP problems

²<http://www.netlib.org/scalapack/>

Name	# of Processors	1	2		4		8	
		time	time	scal	time	scal	time	scal
QAP-SDP12	Total	11.32	6.07	1.86	3.94	2.87	3.55	3.19
	ELEMENTS	1.97	0.96	2.05	0.94	4.05	0.29	6.90
	CHOLESKY	7.53	2.78	2.70	1.57	4.81	1.23	6.11
QAP-SDP16	Total	133.30	55.29	2.41	33.31	4.00	29.83	4.47
	ELEMENTS	13.01	6.29	2.07	3.19	4.07	2.18	5.96
	CHOLESKY	107.67	34.86	3.09	17.91	6.01	14.25	7.55
QAP-SDP20	Total	768.52	275.24	2.79	146.55	5.24	132.87	5.28
	ELEMENTS	57.25	28.29	2.02	13.62	4.20	14.16	4.04
	CHOLESKY	668.42	197.87	3.38	96.25	6.94	77.76	8.60
QAP-SDP24	Total	4386.42	1396.62	3.14	906.33	4.84	660.08	6.65
	ELEMENTS	309.32	144.70	2.14	69.9	4.43	78.47	3.94
	CHOLESKY	3944.74	1110.07	3.55	716.94	5.50	442.57	8.91

Table 2: The computational time for mild-size QAP-SDP (The time unit is second)

Name	# of Processors	2		4		8		16	
		time	scal	time	scal	time	scal	time	scal
QAP-SDP26	Total	3020.1	2	1840.7	3.28	1153.9	5.23	624.7	9.67
	ELEMENTS	274.9	2	112.7	4.87	60.7	9.05	30.4	18.07
	CHOLESKY	2356.9	2	1446.5	3.25	900.8	5.23	511.0	9.22
QAP-SDP28	Total	5195.4	2	3404.0	3.05	2161.9	4.80	1158.9	8.97
	ELEMENTS	443.0	2	196.7	4.50	105.6	8.39	55.9	15.83
	CHOLESKY	4184.1	2	2804.6	2.98	1738.4	4.81	972.4	8.61
QAP-SDP30	Total	x	x	6163.7	4	3731.5	6.61	2186.7	11.27
	ELEMENTS	x	x	329.1	4	190.6	6.91	120.8	10.90
	CHOLESKY	x	x	5207.2	4	3071.8	6.78	1810.5	11.50
QAP-SDP32	Total	x	x	x	x	6341.7	8	3916.9	12.95
	ELEMENTS	x	x	x	x	327.4	8	225.5	11.61
	CHOLESKY	x	x	x	x	5295.4	8	3198.7	13.24

Table 3: The computational time for large-size QAP-SDP (The time unit is second)

4. Numerical Results

This section shows numerical results of SDPARA applied to QAP-SDP on parallel computational environment, PC-cluster composed of 4 computers. The processor on each system is Xeon E5345 2.33GHz, which has 4 cores. In the paper, the number of processors is counted by the number of cores which participate the computation. In addition, the attached memory space to each computer is 16GB. The total computational power of the PC-cluster is 240GFLOPS.

Table 1 gives the size of problems. The column N indicates the number of plants of the location problem. What we would like to note here is that the size of SDP is roughly estimated by two factors, m (the number of equality constraints) and n (the dimension of variable matrices). Due to the additional linear constraints to tighten the gap between the SDP relaxation problem and QAP, the size of the

reinforced QAP-SDP (including additional constraints for higher accuracy) for N plants is $m = N^3 + 2N + 1$ and $n = N^2 + 1$.

Table 2 shows the computational time required for QAP-SDP by SDPARA. The row '# of Processors' indicates the number of available processors. The rows 'Total', 'ELEMENTS', 'CHOLESKY' are total time, the computational time for ELEMENTS, the computational time for CHOLESKY, respectively. The column 'scal' is the scalability, that is, how much faster SDPARA on multiple-processors can solve compared with on single-processor.

The SDPs chosen in Table 2 are mild size so that we can solve them on single processor; otherwise it would be difficult to estimate the scalability. In addition, in Table 2, we let the iteration number on multiple processors be the same on single-processor for the performance evaluation based on the scalability.

The numerical results in Table 2 prove the high scalability of the SDPARA. In particular, the

scalability sometime exceeds the number of processors. This phenomenon is induced by the fact that the memory space handled by each processor shrinks as more processors participate the computation; hence the performance of CPU cache is considerably enhanced. The phenomenon is prominent for the parallel Cholesky factorization.

The row-wise distribution for ELEMENTS works very well for general large SDPs. However, the occupation of ELEMENTS of these SDPs might not be large enough to obtain the best performance of the row-wise distribution. Numerical results for such extremely large-scale SDPs can be found in [16], even though their scalability based on single processor is not available since they cannot be stored on single processor.

The result of how large SDP can be solved by SDPARA is Table 3. The SDPs are so large that they also cannot be solved by single-processor computer. In Table 3, 'x' mark means SDPARA cannot solve the problem due to the memory lack. In addition, the base of the scalability is changed from single processor to the smallest number of the solvable processors.

Table 3 shows SDPARA enables us to handle these sizes of SDPs, while other SDP solvers cannot handle them. This is the remarkable advantage of SDPARA over other solvers.

Another point we want to note at the last of the section is that the scalability becomes higher for the larger problems. Therefore, we can conclude that the performance of SDPARA suits much for extremely-large scale SDPs.

5. Conclusions and Further Discussions

Throughout the paper, we introduce an SDP relaxation method for a location problem. The numerical results show that the parallel solver SDPARA is powerful tool to solve such SDPs in the short time, in particular for extremely large scale SDPs.

SDP covers not only QAP-SDP but also various practical applications. Numerical results for other SDP applications, for example, quantum chemistry and control theory, can be found in [1,15,16].

Furthermore, we are convinced that more problems in logistics fields can be converted into the standard SDP form. The SDP relaxation method will help us to solve many logistics problem in SDP framework.

Acknowledgment

The author M. Y. was partially supported by the Grant-in-Aids for Scientific Research from the Japanese Ministry of Education, Culture, Sports, Science and Technology, No. 18710141.

References

- [1] M. Fukuda, B. J. Braams, M. Nakata, M. L. Overton, J. K. Percus, M. Yamashita and Z. Zhao, "Large-scale semidefinite programs in electronic structure calculation", *Mathematical Programming*, Vol.109, pp.553-580, 2007
- [2] M. X. Goemans and .D. P. Williamson, "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming", *Journal of Association for Computing Machinery*, Vol.42, No.6, pp.1115—1145, 1995
- [3] C. Helmberg, F. Rendl, R. J. Vanderbei and H. Wolkowicz,, "An interior-point method for semidefinite programming", *SIAM Journal on Optimization*, Vol.6, pp.342—361, 1996
- [4] M. Kojima, S. Shindoh and H. Hara, "Interior-point methods for the monotone semidefinite linear complementarity problems", *SIAM Journal on Optimization*, Vol.7, pp.86—125, 1994.
- [5] T. C. Koopmans and M. Beckmann, "Assignment problems and the location of economic actives", *Journal of the Econometric Society*, Vol.25, No.1, pp.53—76, 1957
- [6] Y. E. Nesterov and M. J. Todd, "Primal-Dual Interior-Point Methods for Self-Scaled Cones", *SIAM Journal on Optimization*, Vol.8, pp.324—364, 1994
- [7] S. Poljak, F. Rendl and H. Wolkowicz, "A Recipe for Semidefinite Relaxation for (0,1)-Quadratic Programming", *Technical Report CORR 94/7*, University of Waterloo, 1994
- [8] F. Rendl and R. Sotirov, "Bounds for the quadratic assignment problem using the bundle method", *Mathematical Programming*, Vol.109, No.2-3, pp.505—524, 2007
- [9] J. F. Strum, "SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones", *Optimization Methods and Software*, Vol.11 & 12, pp.625—653, 1999
- [10] M. J. Todd, K. C. Toh and R. H. Tütüncü, "SDPT3 — a MATLAB software package for semidefinite programming, version 1.3", *Optimization Methods and Software*, Vol.11 & 12, pp.545—581, 1999
- [11] M. J. Todd, "Semidefinite optimization", *Acta Numerica*, Vol.10, pp.515—560, 2001
- [12] L. Vandenberghe and S. Boyd, "Positive-Definite Programming." In: J. R. Birge and K. G. Murty (Ed.), *Mathematical Programming: State of the Art 1994*, The University of Michigan, pp.276—308, 1994
- [13] H. Wolkowicz, R. Saigal and L. Vandenberghe, "Handbook of Semidefinite Programming, Theory, Algorithms, and Applications", Kluwer

- Academic Publishers, Massachusetts, 2000
- [14] M. Yamashita, K. Fujisawa and M. Kojima, "Implementation and Evaluation of SDPA6.0 (SemiDefinite Programming Algorithm 6.0)", *Optimization Methods and Software*, Vol.18, pp.491—505 (2003).
- [15] M. Yamashita, K. Fujisawa and M. Kojima, "SDPARA: SemiDefinite Programming Algorithm paRAllel version", *Parallel Computing*, Vol.29, pp.1053—1067, 2003
- [16] M. Yamashita, K. Fujisawa, M. Fukuda, M. Kojima and K. Nakata, "Parallel Primal-Dual Interior-Point Methods for SemiDefinite Programs", In E. G. Talbi (Ed.), *Parallel Combinatorial Optimization*, pp.211--238, Wiley, New Jersey, 2006
- [17] Q. Zhao, S. E. Karisch, F. Rendl and H. Wolkowicz, "Semidefinite Programming Relaxations for the Quadratic Assignment Problem", *Journal of Combinatorial Optimization*, Vol.2, pp.71—109, 1998

Makoto Yamashita is an assistant professor of Kanagawa University. He received his Doctor degree in Science from Tokyo Institute of Technology in 2004. His Research interests are in area of mathematical programming and parallel computations. He is a member of The Operations Research Society of Japan and Japan Industrial



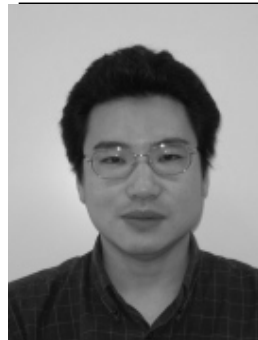
Management Association.

Katsuki Fujisawa is an associate professor of Chuo University. He received his Doctor degree in Science from Tokyo Institute of Technology in 1998. His Research interests are in area of mathematical programming and high-performance computing. He is a member of The Operations Research Society of Japan and The



Institute for Operations Research and The Management Sciences.

Kazuhide Nakata is an assistant professor of Tokyo Institute of Technology. He received his Doctor degree in Science from Tokyo Institute of Technology in 2002. His Research interests are in area of mathematical programming and numerical analysis. He is a member of The Operations Research Society of Japan and The



Japan Society for Industrial and Applied Mathematics.