Journal of Global Optimization **24:** 237–260, 2002.

Parallel Implementation of Successive Convex Relaxation Methods for Quadratic Optimization Problems

AKIKO TAKEDA[1], KATSUKI FUJISAWA[2], YUSUKE FUKAYA[3] and MASAKAZU KOJIMA[4]

[1]*Toshiba Corporation, Kawasaki, Japan (akiko.takeda@toshiba.co.jp),* [2]*Department of Architecture, Kyoto University, Kyoto, Japan (fujisawa@is-mj.archi.kyoto-u.ac.jp),* [3]*NS Solutions Corporation, Tokyo, Japan (Yusuke.Fukaya@open.enicom.co.jp),* [4]*Department of Mathematical and Computing Sciences, Tokyo Institute of Technology, Tokyo, Japan (kojima@is.titech.ac.jp)*

Abstract. As computing resources continue to improve, global solutions for larger size quadrically constrained optimization problems become more achievable. In this paper, we focus on larger size problems and get accurate bounds for optimal values of such problems with the successive use of SDP relaxations on a parallel computing system called Ninf (Network-based Information Library for high performance computing).

Key words: Nonconvex quadratic program; SDP relaxation; Lift-and-project LP relaxation; Lift-and-project procedure; Parallel computation; Global computing

1. Introduction

Quadratic optimization is known as one of the most important areas of nonlinear programming. In addition to its numerous applications in engineering, a quadratic optimization problem (abbreviated by QOP) covers various important nonconvex mathematical programs such as 0–1 linear and quadratic integer programs, linear complementarity problems, bilevel quadratic programs, linear and quadratic fractional programs, and so on. The general class of QOPs can be expressed in the following form:

$$\text{(QOP)} \left| \begin{array}{ll} \max & \boldsymbol{c}^T\boldsymbol{x} \\ \text{s.t.} & \gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} \leqslant 0 \quad (\ell = 1, \ldots, m), \end{array} \right. \tag{1}$$

where $\boldsymbol{c} \in R^n$, $\boldsymbol{q}_\ell \in R^n$ and $\boldsymbol{Q}_\ell \in R^{n\times n}$ ($\ell = 1, \ldots, m$). When a given QOP has a quadratic objective function such as $\gamma_0 + 2\boldsymbol{q}_0^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_0\boldsymbol{x}$, we can transform it into QOP (1) by replacing the quadratic objective function with a new variable t and adding $\gamma_0 + 2\boldsymbol{q}_0^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_0\boldsymbol{x} = t$ to the set of constraints. Therefore, (1) is a general form for quadratically constrained quadratic programs. Pardalos and Vavasis [16] showed that even the simplest quadratic program

$$\min\{-x_1^2 + \boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b},\ \boldsymbol{x} \geqslant \boldsymbol{0}\}$$

is an NP-hard problem. Additional quadratic constraints complicate the problem significantly.

As one solution approach to QOP (1), Kojima and Tunçel [9] established a theoretical framework of two types of successive convex relaxation methods (abbreviated by SCRMs); one is based on the lift-and-project LP (linear programming) relaxation and the other based on the SDP (semidefinite programming) relaxation. We can regard SCRMs as extensions of the lift-and-project procedure, which was proposed independently by Lovász and Schrijver [11] and Sherali and Adams [20] for 0–1 integer programs, to QOP (1).

We denote the feasible region of QOP (1) by F, that is,

$$F \equiv \{\boldsymbol{x} \in R^n : \gamma_\ell + 2\boldsymbol{q}_\ell^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q}_\ell \boldsymbol{x} \leqslant 0 \quad (\ell = 1, \ldots, m)\}.$$

Here we suppose that F is bounded and is included in a known compact convex set C_0, i.e., $F \subseteq C_0$. Starting from the initial convex relaxation C_0 of F, an SCRM successively constructs tighter convex relaxations C_k $(k = 1, 2, \ldots)$ of F with the successive use of the lift-and-project LP or SDP relaxation. Therefore, maximizing the linear objective function $\boldsymbol{c}^T\boldsymbol{x}$ of QOP (1) over C_k $(k = 0, 1, 2, \ldots)$, we successively obtain better upper bounds $\{\zeta_k \ (k = 0, 1, 2, \ldots)\}$ for the maximal objective function value of QOP (1). While the SCRMs proposed by [9] enjoy the global convergence property that $\cap_{k=0}^{+\infty} C_k =$ the convex hull of F, they involve an infinite number of semi-infinite LPs or SDPs to generate a new convex relaxation C_k of F. To resolve this difficulty, Kojima and Tunçel [10] proposed implementable versions of SCRMs by bringing two new techniques, 'discretization' and 'localization', into their theoretical framework. Their techniques allow us to solve finitely many LPs or SDPs having finitely many inequality constraints, so that the discretized-localized versions are implementable on a computer. However, they are still impractical because as a more accurate upper bound is required for the maximal objective function value of QOP (1), not only the number of LPs or SDPs to be solved but also their sizes explode quite rapidly. More recently, Takeda et al. [24] presented practical SCRMs, which overcame such a rapid explosion by further slimming down the discretized-localized SCRMs. Although these practical methods are no more guaranteed to achieve an upper bound with a prescribed accuracy, the numerical results reported in the paper [24] are promising.

In this paper, we propose parallel versions of practical SCRMs. For constituting C_k $(k = 1, 2, \ldots)$, SCRMs generate a large number of LPs or SDPs at each iteration. Our parallel implementation of SCRMs process those multiple LPs or SDPs simultaneously using multiple processors. To enhance the effect of parallel computing, we reduce the work of a client machine, and also decrease communication between processors as much as possible. We implement a highly parallel algorithm on a client-server based parallel computing system called Ninf (Network-based Information Library for high performance computing) [18,19]. Moreover, our parallel implementation of SCRMs adopt new construction for C_k $(k = 1, 2, \ldots)$ so that the number of constraints of each LP or SDP is considerably decreased. As a

result, we can deal with some larger size QOPs, which existing SCRMs cannot process.

This paper consists of five sections. In Section 2, we introduce basic discretized-localized SCRMs with the use of the lift-and-project LP and SDP relaxations, and present a serial implementation of SCRMs. In Section 3, we present new variants of discretized-localized SCRMs suitable for parallel computing, and show a parallel implementation of the new discretized-localized SCRMs. In Section 4, we report its numerical results implemented on Ninf. In Section 5, we give some concluding remarks.

2. Successive convex relaxation methods

We will overview a basic discretized-localized SCRM according to the recent paper [24], which discussed some implementation details of discretized-localized SCRMs and gave preliminary numerical results. We introduce a standard serial algorithm of discretized-localized SCRMs in Section 2.1, and we present some basic properties on the algorithm in Section 2.2.

2.1. PRELIMINARIES

The previous works [8–10, 24, 25] of SCRMs handled general quadratic optimization problems (abbreviated by QOPs) with the following form:

$$\max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in F\}, \tag{2}$$

where $F \equiv \{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leqslant 0 \ (\forall qf(\cdot\,; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathscr{P}_F)\}$, $C_0 \equiv$ a given compact convex set including F; we assume that C_0 is represented in terms of linear inequalities when we are concerned with SCRMs using the lift-and-project LP relaxation, while we assume that C_0 is represented in terms of linear matrix inequalities when we are concerned with SCRMs using the SDP relaxation, $qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \equiv \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}$ $(\forall \boldsymbol{x} \in R^n)$, $\mathscr{P}_F \equiv \{\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x} : \ell = 1, \ldots, m\}$.

To describe a basic discretized-localized SCRM, we introduce the following notation: $\mathscr{S}^n$, the set of $n \times n$ symmetric matrices; $\mathscr{S}^n_+$, the set of $n \times n$ positive semidefinite symmetric matrices; $\boldsymbol{Q} \cdot \boldsymbol{X} \equiv$ the inner product of two symmetric matrices $\boldsymbol{Q}$ and $\boldsymbol{X}$, i.e., $\boldsymbol{Q} \cdot \boldsymbol{X} \equiv \Sigma_i \Sigma_j Q_{ij}X_{ij}$; $\bar{D} = \{\boldsymbol{d} \in R^n : \|\boldsymbol{d}\| = 1\}$ (the set of unit vectors in R^n); $D_0 \subset \bar{D}$, a finite set of unit direction vectors; $\boldsymbol{e}_i$, the ith unit coordinate vector $(i = 1, \ldots, n)$.

For $\forall \boldsymbol{d}_0 \in D_0$, $\forall \boldsymbol{d} \in \bar{D}$, $\forall \boldsymbol{x} \in R^n$ and $\forall$ compact convex subset C of R^n, we define

$$\left.\begin{aligned}
&\alpha(C, \boldsymbol{d}) \equiv \sup\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C\}\,,\\
&\ell sf(\boldsymbol{x}; C, \boldsymbol{d}) \equiv \boldsymbol{d}^T\boldsymbol{x} - \alpha(C, \boldsymbol{d})\,,\\
&r2sf(\boldsymbol{x}; C, \boldsymbol{d}_0, \boldsymbol{d}) \equiv -(\boldsymbol{d}l_0^T\boldsymbol{x} - \alpha(C_0, \boldsymbol{d}_0))(\boldsymbol{d}^T\boldsymbol{x} - \alpha(C, \boldsymbol{d}))\,.
\end{aligned}\right\} \tag{3}$$

We call $\ell sf(\cdot; C, \boldsymbol{d})$ *a linear supporting function for* C *in a direction* $\boldsymbol{d} \in \bar{D}$, $r2sf(\cdot; C, \boldsymbol{d}_0, \boldsymbol{d})$ *a rank-2 (quadratic) supporting function for* C *in a pair of direcions* $\boldsymbol{d}_0 \in D_0$ *and* $\boldsymbol{d} \in \bar{D}$. Let

$$\left.\begin{aligned}
&\mathscr{P}^L(C, D) \equiv \{\ell sf(\cdot; C, \boldsymbol{d}) : \boldsymbol{d} \in D\}, \quad (\forall D \subseteq \bar{D})\,,\\
&\mathscr{P}^2(C, D_0, D) \equiv \{r2sf(\cdot; C, \boldsymbol{d}_0, \boldsymbol{d}) : \boldsymbol{d}_0 \in D_0,\ \boldsymbol{d} \in D\} \quad (\forall D \subseteq \bar{D})\,.
\end{aligned}\right\} \tag{4}$$

Now we summarize a basic discretized-localized SCRM (Algorithm 2.1 below) proposed by [10, 24]. At each iteration (say, the kth iteration) of the basic SCRM, we choose a finite direction-set $D_k \subseteq \bar{D}$, compute $\alpha(C_k, \boldsymbol{d})$ for $\forall \boldsymbol{d} \in D_k$, and construct a function-set $\mathscr{P}^2(C_k, D_0, D_k)$ using $\alpha(C_0, \boldsymbol{d}_0)$ $(\forall \boldsymbol{d}_0 \in D_0)$ and $\alpha(C_k, \boldsymbol{d})$ $(\forall \boldsymbol{d} \in D_k)$. Note that $\mathscr{P}_k = \mathscr{P}^2(C_k, D_0, D_k) \cup \mathscr{P}^L(C_0, D_0)$ induces valid inequalities for the kth iterate C_k. That is, any function $qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q})$ of $\mathscr{P}_k$ satisfies

$$qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leqslant 0 \quad \text{for every } \boldsymbol{x} \in C_k\,.$$

Since C_k was chosen to include F at the previous iteration, each $qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leqslant 0$ serves as a (redundant) valid inequality for F; hence F is represented as

$$F = \{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leqslant 0 \quad (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathscr{P}_F \cup \mathscr{P}_k)\}\,. \tag{5}$$

We then apply the lift-and-project LP or SDP relaxation to the region F with the representation of (5), and obtain the region

$$\begin{aligned}
\hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_k) &= \left\{\boldsymbol{x} \in C_0 : \begin{array}{l}\exists \boldsymbol{X} \in \mathscr{S}^n \text{ such that}\\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \cdot \boldsymbol{X} \leqslant 0\ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathscr{P}_F \cup \mathscr{P}_k)\end{array}\right\}\\
&= \text{a lift-and-project LP relaxation of } F\\
&\quad \text{with the use of the representation } \mathscr{P}_F \cup \mathscr{P}_k
\end{aligned} \tag{6}$$

or

$$\begin{aligned}
\hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_k) &= \left\{\boldsymbol{x} \in C_0 : \begin{array}{l}\exists \boldsymbol{X} \in \mathscr{S}^n \text{ such that } \begin{pmatrix}1 & \boldsymbol{x}^T\\ \boldsymbol{x} & \boldsymbol{X}\end{pmatrix} \in \mathscr{S}_+^{1+n}\,,\\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \cdot \boldsymbol{X} \leqslant 0\ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathscr{P}_F \cup \mathscr{P}_k)\end{array}\right\},\\
&= \text{an SDP relaxation of } F \text{ with the use of the representation}\\
&\quad \mathscr{P}_F \cup \mathscr{P}_k\,.
\end{aligned} \tag{7}$$

Each region corresponds to the $(k+1)$th iterate C_{k+1}. By definition, it is clear that C_{k+1} is a convex subset of C_0 and that $F \subseteq C_{k+1}$.

ALGORITHM 2.1. (Serial implementation of a basic discretized-localized SCRM)
Step 0: Let $D_0 \subseteq \bar{D}$. Compute

$$\alpha(C_0, \boldsymbol{d}_0) = \max\{\boldsymbol{d}_0^T\boldsymbol{x} : \boldsymbol{x} \in C_0\} \quad (\forall \boldsymbol{d}_0 \in D_0)\,,$$

and construct $\mathscr{P}^L(C_0, D_0)$ according to (3) and (4). Let $C_1 = C_0$ and $k = 1$.

Step 1: Compute an upper bound ζ_k of the maximum objective function value of QOP (2) by $\zeta_k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$. If ζ_k satisfies some termination criteria, then stop.

Step 2: Choose a finite direction-set $D_k \subseteq \bar{D}$. Compute

$$\alpha(C_k, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \quad (\forall \boldsymbol{d} \in D_k)\,.$$

Step 3: Construct a set $\mathscr{P}_k = \mathscr{P}^2(C_k, D_0, D_k) \cup \mathscr{P}^L(C_0, D_0)$ according to (3) and (4).

Step 4: Let $C_{k+1} = \hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_k)$ or $C_{k+1} = \hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_k)$.

Step 5: Let $k = k + 1$, and go to Step 1.

Note that the problem $\max\{\boldsymbol{d}_0^T\boldsymbol{x} : \boldsymbol{x} \in C_0\}$ to be solved in Step 0 is either an LP or an SDP since we have assumed that C_0 is represented in terms of either linear inequalities or linear matrix inequalities. Also, in Step 2, we solve LPs over the polyhedral feasible region $C_k = \hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_{k-1})$ described in terms of linear inequalities or SDPs over the convex feasible region $C_k = \hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_{k-1})$ described in terms of linear matrix inequalities to obtain $\alpha(C_k, \boldsymbol{d})$ for $\forall \boldsymbol{d} \in D_k$ ($k = 1, 2, \ldots$). We will give a termination criteria used in our numerical experiments in Section 4.2. Algorithm 2.1 above lacks a description for the direction-sets D_k ($k = 0, 1, 2, \ldots$). In previous works [9, 10, 24, 25], SCRMs commonly utilized D_0 such as

$$D_0 = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, -\boldsymbol{e}_1, \ldots, -\boldsymbol{e}_n\}\,, \tag{8}$$

and adopted various kinds of direction-sets $D_k \subseteq \bar{D}$ ($k = 1, 2, \ldots$).

Algorithm 2.1 with the above choice of D_0 and $D_k \subseteq \bar{D}$ ($k = 1, 2, \ldots$) generates a sequence of convex sets $C_k \subseteq C_0$ ($k = 1, 2, \ldots$) and a sequence of real numbers ζ_k ($k = 1, 2, \ldots$) satisfying

$$C_0 \supseteq C_k \supseteq C_{k+1} \supseteq F \quad (k = 1, 2, \ldots)\,,$$
$$\zeta_k \geqslant \zeta_{k+1} \geqslant \zeta^* \equiv \sup\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in F\} \quad (k = 1, 2, \ldots)\,.$$

If we took $D_k = \bar{D}$ ($k = 1, 2, \ldots$), C_k ($k = 1, 2, \ldots$) would converge to the convex hull of F and ζ_k ($k = 1, 2, \ldots$) to an optimal value ζ^* of QOP (2) as $k \to \infty$. However, such SCRMs with $D_k = \bar{D}$ ($k = 1, 2, \ldots$) are conceptual and not implementable, because they involve an infinite number of LPs or SDPs to be solved at each iteration. See [9] for more details of conceptual SCRMs.

To implement Algorithm 2.1 on a computer, it is necessary to choose a finite set of directions for D_k at the kth ($k = 1, 2, ., .$) iteration. In the remainder of the paper, we take

$$D_k \equiv \{\boldsymbol{b}_{i+}(\theta_k), \boldsymbol{b}_{i-}(\theta_k), \quad i = 1, 2, \ldots, n\} \tag{9}$$

with a parameter $\theta_k \in (0, \pi/2]$ according to the paper [24]. Here,

$$\boldsymbol{b}_{i+}(\theta) = \frac{\boldsymbol{c}\cos\theta + \boldsymbol{e}_i \sin\theta}{\nu_{i+}(\theta)}, \qquad \boldsymbol{b}_{i-}(\theta) = \frac{\boldsymbol{c}\cos\theta - \boldsymbol{e}_i \sin\theta}{\nu_{i-}(\theta)}$$
$$\nu_{i+}(\theta) = \|\boldsymbol{c}\cos\theta + \boldsymbol{e}_i \sin\theta\|, \qquad \nu_{i-}(\theta) = \|\boldsymbol{c}\cos\theta - \boldsymbol{e}_i \sin\theta\|.$$

Then $\mathscr{P}^2(C_k, D_0, D_k)$ consists of $4n^2$ quadratic functions such that

$$\mathscr{P}^2(C_k, D_0, D_k) = \left\{ \begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta_k)),\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta_k)) \\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta_k)),\ r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta_k)) \\ i = 1, \ldots, n,\ j = 1, \ldots, n \end{array} \right\} \tag{10}$$

See (3) and (4). We will call Algorithm 2.1 *DLSLP* if it takes the finite direction-sets D_k $(k = 0, 1, 2, \ldots)$ introduced above and the lift-and-project LP relaxation $\hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_k)$ in Step 4, while we call Algorithm 2.1 *DLSSDP* if it takes the finite direction-sets D_k $(k = 0, 1, 2, \ldots)$ above and the SDP relaxation $\hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_k)$ for C_{k+1} in Step 4.

We choose $\theta_1 = \pi/2$ at the first iteration of Algorithm 2.1. In this case, the vectors $\boldsymbol{b}_{i+}(\theta_1)$ and $\boldsymbol{b}_{i-}(\theta_1)$ of D_1 turn out to be the ith unit coordinate vector $\boldsymbol{e}_i$ and its minus $-\boldsymbol{e}_i$, respectively. Then, the values $\alpha(C_1, \boldsymbol{e}_i)$ and $-\alpha(C_1, -\boldsymbol{e}_i)$ correspond upper and lower bounds for the variable x_i, respectively. Hence the set $\mathscr{P}^2(C_1, D_0, D_1)$ constructed in Step 3 of Algorithm 2.1 contains all rank-2 quadratic functions induced from the pairwise products of lower and upper bounding constraints for variables x_i $(i = 1, 2, \ldots, n)$. These constraints correspond to underestimators and overestimators of quadratic terms $x_i x_j$ $(i, j = 1, 2, \ldots, n)$, which were introduced in [13] and have been used as lower (or upper) bounding techniques of some branch-and-bound methods (for instances, see [17,28]). We also see that both $\boldsymbol{b}_{i+}(\theta)$ and $\boldsymbol{b}_{i-}(\theta)$ $(i = 1, \ldots, n)$ approach to the objective direction $\boldsymbol{c}$ as $\theta \to 0$.

2.2. SOME PROPERTIES OF SCRMs

Two important key words in this section are 'linearized' and 'convexified'. To explain these words, we will utilize Lemma 2.2, Examples 2.3 and 2.4 below.

We write the set $\mathscr{Q}_+$ of convex quadratic functions on R^n and the set $\mathscr{L}$ of linear functions on R^n as

$$\mathscr{Q}_+ \equiv \{qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) : \gamma \in R,\ \boldsymbol{q} \in R^n,\ \boldsymbol{Q} \in \mathscr{S}^n_+\},$$
$$\mathscr{L} \equiv \{qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) : \gamma \in R,\ \boldsymbol{q} \in R^n,\ \boldsymbol{Q} \in \boldsymbol{O}\},$$

respectively. Let c.cone($\mathscr{P}$) denote the convex cone generated by a set $\mathscr{P}$ of quadratic functions;

$$\text{c.cone}(\mathscr{P}) \equiv \left\{ \sum_{i=1}^{\ell} \lambda_i p_i(\cdot) : \lambda_i \geqslant 0,\ p_i(\cdot) \in \mathscr{P}\ (i = 1, 2, \ldots, \ell),\ \ell \geqslant 0 \right\}.$$

LEMMA 2.2.[1] *(Theorem 2.4 and Corollary 2.5 of Kojima and Tunçel* [9]*)*
(i) $\hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_k) \subset \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leqslant 0 \ (\forall p(\cdot) \in \text{c.cone}(\mathscr{P}_F \cup \mathscr{P}_k) \cap \mathscr{L})\}$,
(ii) $\hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_k) \subset \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \leqslant 0 \ (\forall p(\cdot) \in \text{c.cone}(\mathscr{P}_F \cup \mathscr{P}_k) \cap \mathscr{Q}_+)\}$.

EXAMPLE 2.3 (Figure 1). Let

$$C_0 \equiv \{(x_1, x_2) : 0 \leqslant x_1 \leqslant 1,\ 0 \leqslant x_2 \leqslant 1\},$$

$$\mathscr{P}_F \equiv \left\{-x_1, x_1 - 1, -x_2, x_2 - 1, -(x_1 - 1)^2 - (x_2 - 1)^2 + \left(\frac{3}{4}\right)^2\right\},$$

$$F \equiv \{(x_1, x_2) \in C_0 : qf((x_1, x_2); \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leqslant 0 \ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathscr{P}_F)\}$$

$$= \left\{(x_1, x_2) : 0 \leqslant x_1 \leqslant 1,\ 0 \leqslant x_2 \leqslant 1,\ -(x_1 - 1)^2 - (x_2 - 1)^2 + \left(\frac{3}{4}\right)^2 \leqslant 0\right\}.$$

The shaded area of Figure 1 illustrates the feasible region F. Take $\theta_1 = \pi/2$ at the first iteration of Algorithm 2.1. Then Step 3 of Algorithm 2.1 constructs a finite set $\mathscr{P}^2(C_1, D_0, D_1)$ of quadratic functions including $x_1^2 - x_1$ and $x_2^2 - x_2$. The addition of $x_1^2 - x_1 \leqslant 0$ and $x_2^2 - x_2 \leqslant 0$ to the nonconvex quadratic inequality constraint

$$-(x_1 - 1)^2 - (x_2 - 1)^2 + \left(\frac{3}{4}\right)^2 \leqslant 0 \tag{11}$$

involved in the description of F removes the nonconvex quadratic terms $-x_1^2$ and $-x_2^2$ of (11), and generates a linear inequality

$$x_1 + x_2 - \frac{23}{16} \leqslant 0. \tag{12}$$

Since

$$x_1 + x_2 - \frac{23}{16} \in \text{c.cone}(\mathscr{P}_F \cup \mathscr{P}_1) \cap \mathscr{L},$$

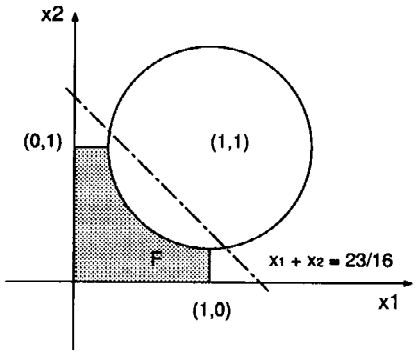

Figure 1. Feasible region F of Example 2.3.

[1] Kojima and Tunçel [9] presented a stronger result in which the equalities hold in (i) and (ii) below. Their proof for the $\supset$ part is incomplete but the $\subset$ remains valid. See also Fujie and Kojima [5].

we see by Lemma 2.2 that any point $\boldsymbol{x}$ of a lift-and-project LP relaxation $C_2 = \hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_1)$ satisfies (12). As we see in Figure 1, the linear inequality (12) cuts off the initial convex relaxation C_0 of the feasible region F effectively. We may regard the inequality (12) as a linear relaxation of the nonconvex quadratic constraint (11) with the help of two quadratic functions $x_1^2 - x_1$ and $x_2^2 - x_2$ in $\mathscr{P}^2(C_1, D_0, D_1)$. We say that the nonconvex quadratic function $-(x_1 - 1)^2 - (x_2 - 1)^2 + (\frac{3}{4})^2$ in $\mathscr{P}_F$ is linearized with the help of quadratic functions in $\mathscr{P}^2(C_1, D_0, D_1)$. The set c.cone$(\mathscr{P}_F \cup \mathscr{P}_1) \cap \mathscr{L}$ consists of all linearizations of quadratic functions $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) \in \mathscr{P}_F$ with the help of quadratic functions in $\mathscr{P}^2(C_1, D_0, D_1)$.

EXAMPLE 2.4 (Figure 2). Let

$$C_0 \equiv \{(x_1, x_2) : 0 \leqslant x_1 \leqslant 1,\ 0 \leqslant x_2 \leqslant 1\},$$

$$\mathscr{P}_F \equiv \left\{ -x_1,\ x_1 - 1,\ -x_2,\ x_2 - 1,\ x_1^2 - x_2^2 - \frac{1}{4} \right\},$$

$$F \equiv \left\{ (x_1, x_2) \in C_0 : x_1^2 - x_2^2 - \frac{1}{4} \leqslant 0 \right\}$$

$$= \left\{ (x_1, x_2) : 0 \leqslant x_1 \leqslant 1,\ 0 \leqslant x_2 \leqslant 1,\ x_1^2 - x_2^2 - \frac{1}{4} \leqslant 0 \right\}.$$

As in the previous example, we take $\theta_1 = \pi/2$ and construct a finite set $\mathscr{P}^2(C_1, D_0, D_1)$ including the quadratic function $x_2^2 - x_2$ in Step 3 of Algorithm 2.1. Now we can remove the nonconvex quadratic term $-x_2^2$ of the function $x_1^2 - x_2^2 - \frac{1}{4}$ in $\mathscr{P}_F$ by adding the quadratic function $x_2^2 - x_2$ to generate a convex quadratic function $x_1^2 - x_2 - \frac{1}{4}$ in c.cone$(\mathscr{P}_F \cup \mathscr{P}_1)$. Hence we know that

$$C_2 = \hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_1) \subset \left\{ (x_1, x_2) \in C_0 : x_1^2 - x_2 - \frac{1}{4} \leqslant 0 \right\}.$$

Thus the quadratic function $x_1^2 - x_2^2 - \frac{1}{4}$ in $\mathscr{P}_F$ is convexified with the help of the quadratic function $x_2^2 - x_2$ in $\mathscr{P}^2(C_1, D_0, D_1)$. The set c.cone$(\mathscr{P}_F \cup \mathscr{P}_1) \cap \mathscr{Q}_+$

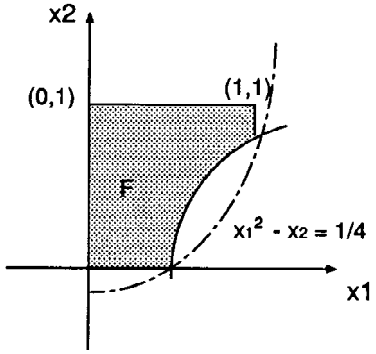

Figure 2. Feasible region F of Example 2.4.

consists of all convexifications of quadratic functions $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) \in \mathscr{P}_F$ with the help of quadratic functions in $\mathscr{P}^2(C_1, D_0, D_1)$.

Generally, to linearize (or convexify) each quadratic function $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell)$ of $\mathscr{P}_F$, the DLSLP and DLSSDP utilize an 'atomic rank-1' quadratic function with only one quadratic term $x_i x_j$ in c.cone($\mathscr{P}^2(C_k, D_0, D_k)$), which is constructed as a nonnegative combination of two functions $r2sf(\cdot; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta))$ and $r2sf(\cdot; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta))$ in $\mathscr{P}^2(C_k, D_0, D_k)$, such that

$$\left.\begin{aligned} p_{ij}(\boldsymbol{x}) &\equiv \frac{\nu_{i+}(\theta)}{2\sin\theta} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta)) + \frac{\nu_{i-}(\theta)}{2\sin\theta} r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta)) \\ &= \boldsymbol{x}^T \boldsymbol{e}_i \boldsymbol{e}_j^T \boldsymbol{x} + (\boldsymbol{a}_{ij}^T \boldsymbol{x} + \beta_{ij}) . \end{aligned}\right\} \tag{13}$$

Here

$$\left.\begin{aligned} \boldsymbol{a}_{ij} &= \frac{1}{2\tan\theta}\{\alpha(C_0, -\boldsymbol{e}_j) + \alpha(C_0, \boldsymbol{e}_j)\}\boldsymbol{c} \\ &\quad + \frac{1}{2}\{\alpha(C_0, -\boldsymbol{e}_j) - \alpha(C_0, \boldsymbol{e}_j)\}\boldsymbol{e}_i \\ &\quad + \frac{1}{2\sin\theta}\{\nu_{i-}(\theta)\alpha(C_k, \boldsymbol{b}_{i-}(\theta)) - \nu_{i+}(\theta)\alpha(C_k, \boldsymbol{b}_{i+}(\theta))\}\boldsymbol{e}_j , \\ \beta_{ij} &= -\frac{1}{2\sin\theta}\{\nu_{i+}(\theta)\alpha(C_0, -\boldsymbol{e}_j)\alpha(C_k, \boldsymbol{b}_{i+}(\theta)) + \nu_{i-}(\theta)\alpha(C_0, \boldsymbol{e}_j)\alpha(C_k, \boldsymbol{b}_{i-}(\theta))\} . \end{aligned}\right\} \tag{14}$$

A nonnegative combination of some other two functions in $\mathscr{P}^2(C_k, D_0, D_k)$ leads to another 'atomic rank-1' quadratic functions with only one quadratic term $-x_i x_j$ as

$$\left.\begin{aligned} p'_{ij}(\boldsymbol{x}) &\equiv \frac{\nu_{i+}(\theta)}{2\sin\theta} r2sf(\boldsymbol{x}; C_k, \boldsymbol{e}_j, \boldsymbol{b}_{i+}(\theta)) + \frac{\nu_{i-}(\theta)}{2\sin\theta} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{e}_j, \boldsymbol{b}_{i-}(\theta)) \\ &= -\boldsymbol{x}^T \boldsymbol{e}_i \boldsymbol{e}_j^T \boldsymbol{x} + (\boldsymbol{a}'^T_{ij} \boldsymbol{x} + \beta'_{ij}) \quad (i, j = 1, 2, \ldots, n) . \end{aligned}\right\} \tag{15}$$

Here $\boldsymbol{a}'_{ij} \in R^n$ and $\beta'_{ij} \in R$ have similar representations to $\boldsymbol{a}_{ij} \in R^n$ and $\beta_{ij} \in R$ given in (14), respectively. We see by definition that $p_{ij}(\boldsymbol{x})$, $p'_{ij}(\boldsymbol{x}) \in$ c.cone($\mathscr{P}_k$) for $i, j = 1, \ldots, n$. Hence c.cone($\mathscr{P}_F \cup \mathscr{P}_k$) includes the following linear function $g_\ell(\boldsymbol{x})$.

$$\left.\begin{aligned} g_\ell(\boldsymbol{x}) &\equiv qf(\boldsymbol{x}; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) + \sum_{(i,j)} Q^{(i,j)}_{\ell+} p'_{ij}(\boldsymbol{x}) - \sum_{(i,j)} Q^{(i,j)}_{\ell-} p_{ij}(\boldsymbol{x}) \\ &= \left(\gamma_\ell + \sum_{(i,j)} Q^{(i,j)}_{\ell+}\beta'_{ij} - \sum_{(i,j)} Q^{(i,j)}_{\ell-}\beta_{ij}\right) + \left(2\boldsymbol{q}_\ell + \sum_{(i,j)} Q^{(i,j)}_{\ell+}\boldsymbol{a}'_{ij} - \sum_{(i,j)} Q^{(i,j)}_{\ell-}\boldsymbol{a}_{ij}\right)^T \boldsymbol{x} , \end{aligned}\right\} \tag{16}$$

where $Q^{(i,j)}_\ell$ denotes the (i, j)th element of the matrix $\boldsymbol{Q}_\ell$ and

$$Q^{(i,j)}_{\ell+} = \begin{cases} Q^{(i,j)}_\ell & \text{if } Q^{(i,j)}_\ell > 0 \\ 0 & \text{otherwise} , \end{cases} \qquad Q^{(i,j)}_{\ell-} = \begin{cases} Q^{(i,j)}_\ell & \text{if } Q^{(i,j)}_\ell < 0 \\ 0 & \text{otherwise} . \end{cases}$$

Thus we obtain $g_\ell(\cdot) \in$ c.cone($\mathscr{P}_F \cup \mathscr{P}_k$) $\cap\, \mathscr{L}$ with the help of quadratic functions in

$\mathscr{P}^2(C_1, D_0, D_1)$, and an associated linear valid inequality $g_\ell(\boldsymbol{x}) \leqslant 0$ for the feasible region F of QOP (2). In general, we can expect that c.cone$(\mathscr{P}_F \cup \mathscr{P}_k) \cap \mathscr{L}$ involves linearizations of the function $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) \in \mathscr{P}_F$ which induce more effective valid inequalities than $g_\ell(\boldsymbol{x}) \leqslant 0$ constructed above.

Similarly we derive a convexification of each quadratic function $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) \in \mathscr{P}_F$. In this case, we restrict ourselves to a nonconvex part of $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) \in \mathscr{P}_F$. Suppose that $\boldsymbol{Q}_\ell = \boldsymbol{Q}_\ell^+ + \boldsymbol{Q}_\ell^-$ with a positive semidefinite $\boldsymbol{Q}_\ell^+$ and a negative semidefinite $\boldsymbol{Q}_\ell^-$. Then we see that

$$qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell) = qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell^+) + qf(\cdot; 0, \boldsymbol{0}, \boldsymbol{Q}_\ell^-).$$

Now we can apply the same argument as above to construct a linearization $g_\ell^-(\cdot)$ of $qf(\cdot; 0, \boldsymbol{0}, \boldsymbol{Q}_\ell^-)$ with the help of 'atomic rank-1' quadratic functions $p_{ij}(\cdot)$ and $p'_{ij}(\cdot)$ $(i, j = 1, 2, \ldots, n)$. Finally we obtain $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell^+) + g_\ell^-(\cdot) \in$ c.cone$(\mathscr{P}_F \cup \mathscr{P}_k) \cap \mathscr{Q}_+$.

3. Parallel implementation of SCRMs

In Step 2 of Algorithm 2.1, $2n$ problems (LPs or SDPs) involving $4n^2$ additional quadratic constraints are generated for constructing $\mathscr{P}^2(C_k, D_0, D_k)$. It should be emphasized that these $2n$ problems are independent and they can be processed in parallel. In this section, we consider parallel computation of those problems. Although parallel computation is much help to reduce computational time drastically, $4n^2$ constraints of each problem become an obstacle when we solve a large size QOP (2). In Section 3.1, we design new SCRMs so that each LP or SDP has a fewer number of constraints than $4n^2$. Then, in Section 3.2, we will modify Algorithm 2.1 and present a parallel algorithm for a client-server based parallel computing system.

3.1. AN EFFECTIVE TECHNIQUE FOR REDUCING INEQUALITIES

In Step 2 of Algorithm 2.1, LPs or SDPs over a feasible region C_k are constructed. In order to reduce the number of constraints of each LP or SDP, we devise different constructions for D_0, D_k $(k = 1, 2, \ldots)$ and $\mathscr{P}^2(C_k, D_0, D_k)$.

We first introduce new notation. Let $\lambda_1^\ell, \ldots, \lambda_n^\ell$ denote the n eigenvalues of the coefficient matrix $\boldsymbol{Q}_\ell$ of the ℓth quadratic constraint of QOP (2), and let $\boldsymbol{\Lambda}_\ell$ denote a diagonal matrix $diag(\lambda_1^\ell, \ldots, \lambda_n^\ell)$. Then there exists a real orthogonal matrix $\boldsymbol{P}_\ell$ such that $\boldsymbol{P}_\ell^T \boldsymbol{Q}_\ell \boldsymbol{P}_\ell = \boldsymbol{\Lambda}_\ell$. Define the sets $I_+(\ell)$ and $I_+(\ell)$ $(\ell = 1, \ldots, m)$ as

$$\begin{aligned} I_+(\ell) &\equiv \text{the set of indices corresponding to positive diagonal} \\ &\qquad \text{elements of } \boldsymbol{\Lambda}_\ell, \text{ that is, } \lambda_i^\ell > 0 \text{ for } \forall i \in I_+(\ell), \\ I_-(\ell) &\equiv \text{the set of indices corresponding to negative diagonal} \\ &\qquad \text{elements of } \boldsymbol{\Lambda}_\ell, \text{ that is, } \lambda_j^\ell < 0 \text{ for } \forall j \in I_-(\ell). \end{aligned}$$

From the definition, we see that $I_+(\ell) \subseteq \{1, 2, \ldots, n\}$, $I_-(\ell) \subseteq \{1, 2, \ldots, n\}$ and $I_+(\ell) \cap I_-(\ell) = \emptyset$. Define new vectors with a parameter $\theta \in (0, \pi/2]$:

$$\boldsymbol{b}^{\ell}_{i+}(\theta) = \frac{\boldsymbol{c}\cos\theta + (\boldsymbol{P}_\ell \boldsymbol{e}_i)\sin\theta}{\nu^{\ell}_{i+}(\theta)}, \qquad \boldsymbol{b}^{\ell}_{i-}(\theta) = \frac{\boldsymbol{c}\cos\theta - (\boldsymbol{P}_\ell \boldsymbol{e}_i)\sin\theta}{\nu^{\ell}_{i-}(\theta)},$$

$$\nu^{\ell}_{i+}(\theta) = \|\boldsymbol{c}\cos\theta + (\boldsymbol{P}_\ell \boldsymbol{e}_i)\sin\theta\|, \qquad \nu^{\ell}_{i-}(\theta) = \|\boldsymbol{c}\cos\theta - (\boldsymbol{P}_\ell \boldsymbol{e}_i)\sin\theta\|.$$

Now we are ready to propose different constructions of D_0, D_k ($k = 1, 2, \ldots$) and $\mathscr{P}^2(C_k, D_0, D_k)$:

$$\left.\begin{aligned} &D_0^S \equiv \{\boldsymbol{P}_\ell \boldsymbol{e}_i, -\boldsymbol{P}_\ell \boldsymbol{e}_i,\ i \in I_-(\ell),\ \ell = 1, \ldots, m\},\\ &D_k^S \equiv \{\boldsymbol{b}^{\ell}_{i+}(\theta_k), \boldsymbol{b}^{\ell}_{i-}(\theta_k),\ i \in I_-(\ell),\ \ell = 1, \ldots, m\},\\ &\mathscr{P}^2_S(C_k, D_0^S, D_k^S) \equiv \left\{\begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^{\ell}_{i+}(\theta_k)),\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^{\ell}_{i-}(\theta_k))\\ i \in I_-(\ell),\ \ell = 1, \ldots, m \end{array}\right\} \end{aligned}\right\} \tag{17}$$

for the SDP relaxation, and

$$\left.\begin{aligned} D_0^L &\equiv D_0^S \cup \{\boldsymbol{P}_\ell \boldsymbol{e}_i, -\boldsymbol{P}_\ell \boldsymbol{e}_i,\ i \in I_+(\ell),\ \ell = 1, \ldots, m\}\\ &= \{\boldsymbol{P}_\ell \boldsymbol{e}_i, -\boldsymbol{P}_\ell \boldsymbol{e}_i,\ i \in I_-(\ell) \cup I_+(\ell),\ \ell = 1, \ldots, m\},\\ D_k^L &\equiv D_k^S \cup \{\boldsymbol{b}^{\ell}_{i+}(\theta_k), \boldsymbol{b}^{\ell}_{i-}(\theta_k),\ i \in I_+(\ell),\ \ell = 1, \ldots, m\}\\ &= \{\boldsymbol{b}^{\ell}_{i+}(\theta_k), \boldsymbol{b}^{\ell}_{i-}(\theta_k),\ i \in I_-(\ell) \cup I_+(\ell),\ \ell = 1, \ldots, m\},\\ \mathscr{P}^2_L(C_k, D_0^L, D_k^L) &\equiv \mathscr{P}^2_S(C_k, D_0^S, D_k^S) \cup \\ &\qquad \left\{\begin{array}{l} r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}^{\ell}_{j+}(\theta_k)), r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}^{\ell}_{j-}(\theta_k))\\ j \in I_+(\ell),\ \ell = 1, \ldots, m \end{array}\right\}\\ &= \left\{\begin{array}{l} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^{\ell}_{i+}(\theta_k)),\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}^{\ell}_{i-}(\theta_k))\\ r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}^{\ell}_{j+}(\theta_k)),\ r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}^{\ell}_{j-}(\theta_k))\\ i \in I_-(\ell),\ j \in I_+(\ell),\ \ell = 1, \ldots, m \end{array}\right\} \end{aligned}\right\} \tag{18}$$

for the lift-and project LP relaxation. We designate Algorithm 2.1 which takes $D_0 = D_0^S$, $D_k = D_k^S$, $\mathscr{P}^2(C_k, D_0, D_k) = \mathscr{P}^2_S(C_k, D_0^S, D_k^S)$ and the SDP relaxation $\hat{F}(C_0, \mathscr{P}_F \cup \mathscr{P}_k)$ for C_{k+1} in Step 4 as *DLSSDP-diag*, while we designate Algorithm 2.1 which takes $D_0 = D_0^L$, $D_k = D_k^L$, $\mathscr{P}^2(C_k, D_0, D_k) = \mathscr{P}^2_L(C_k, D_0^L, D_k^L)$ and the lift-and-project LP relaxation $\hat{F}^L(C_0, \mathscr{P}_F \cup \mathscr{P}_k)$ for C_{k+1} in Step 4 as *DLSLP-diag*.

We will show how each quadratic function $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell)$ of $\mathscr{P}_F$ is convexified in DLSSDP-diag with the help of quadratic functions of $\mathscr{P}^2_S(C_k, D_0^S, D_k^S)$. First note that each coefficient matrix $\boldsymbol{Q}_\ell$ of the quadratic constraints of QOP (2) can be expressed as $\boldsymbol{Q}_\ell = \boldsymbol{Q}^+_\ell + \boldsymbol{Q}^-_\ell$ using a positive semidefinite matrix $\boldsymbol{Q}^+_\ell$ and a negative semidefinite matrix $\boldsymbol{Q}^-_\ell$;

$$\boldsymbol{Q}^+_\ell \equiv \sum_{i \in I_+(\ell)} \lambda^{\ell}_i (\boldsymbol{P}_\ell \boldsymbol{e}_i)(\boldsymbol{P}_\ell \boldsymbol{e}_i)^T \quad \text{and} \quad \boldsymbol{Q}^-_\ell \equiv \sum_{i \in I_-(\ell)} \lambda^{\ell}_i (\boldsymbol{P}_\ell \boldsymbol{e}_i)(\boldsymbol{P}_\ell \boldsymbol{e}_i)^T.$$

As a nonnegatie combination of two functions of $\mathscr{P}_S^2(C_k, D_0^S, D_k^S)$, we have an 'atomic rank-1' quadratic function

$$\begin{aligned} p_i^\ell(\boldsymbol{x}) &\equiv \frac{\nu_{i+}^\ell(\theta_k)}{2\sin\theta_k} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}_{i+}^\ell(\theta_k)) + \frac{\nu_{i-}^\ell(\theta_k)}{2\sin\theta_k} r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_i, \boldsymbol{b}_{i-}^\ell(\theta_k)) \\ &= \boldsymbol{x}^T(\boldsymbol{P}_\ell \boldsymbol{e}_i)(\boldsymbol{P}_\ell \boldsymbol{e}_i)^T\boldsymbol{x} + (\boldsymbol{a}_i^\ell)^T\boldsymbol{x} + \beta_i^\ell \quad (i \in I_-(\ell)) \end{aligned}$$

for some $\boldsymbol{a}_i^\ell \in R^n$ and $\beta_i^\ell \in R$. Then, using these atomic rank-1 quadratic functions $p_i^\ell(\boldsymbol{x})$ $(i \in I_-(\ell))$, we can convexify the quadratic function $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell)$ such that

$$\left.\begin{aligned} h_\ell(\boldsymbol{x}) &= (\gamma_\ell + 2\boldsymbol{q}_\ell^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell\boldsymbol{x}) - \sum_{i \in I_-(\ell)} \lambda_i^\ell p_i^\ell(\boldsymbol{x}) \\ &= \left(\gamma_\ell - \sum_{i \in I_-(\ell)} \lambda_i^\ell \beta_i^\ell\right) + \left(2\boldsymbol{q}_\ell - \sum_{i \in I_-(\ell)} \lambda_i^\ell \boldsymbol{a}_i^\ell\right)^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}_\ell^+\boldsymbol{x}\,. \end{aligned}\right\} \tag{19}$$

Thus we have obtained $h_\ell(\cdot) \in \text{c.cone}(\mathscr{P}_F \cup \mathscr{P}_k) \cap \mathscr{Q}_+$, which induces a convex quadratic valid inequality $h_\ell(\boldsymbol{x}) \leqslant 0$ for the feasible region F of QOP (2).

To linearize the quadratic function $qf(\cdot; \gamma_\ell, \boldsymbol{q}_\ell, \boldsymbol{Q}_\ell)$ of $\mathscr{P}_F$, we further need quadratic functions in the difference set $\mathscr{P}_L^2(C_k, D_0^L, D_k^L) \backslash \mathscr{P}_S^2(C_k, D_0^S, D_k^S)$. As a nonnegative combination of two functions in this difference set, we have

$$\begin{aligned} p_j^\ell(\boldsymbol{x}) &\equiv \frac{\nu_{j+}^\ell(\theta_k)}{2\sin\theta_k} r2sf(\boldsymbol{x}; C_k, \boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}_{j+}^\ell(\theta_k)) + \frac{\nu_{j-}^\ell(\theta_k)}{2\sin\theta_k} r2sf(\boldsymbol{x}; C_k, -\boldsymbol{P}_\ell \boldsymbol{e}_j, \boldsymbol{b}_{j-}^\ell(\theta_k)) \\ &= -\boldsymbol{x}^T(\boldsymbol{P}_\ell \boldsymbol{e}_j)(\boldsymbol{P}_\ell \boldsymbol{e}_j)^T\boldsymbol{x} + (\boldsymbol{a}_j^\ell)^T\boldsymbol{x} + \beta_j^\ell \quad (j \in I_+(\ell)) \end{aligned}$$

for some $\boldsymbol{a}_j^\ell \in R^n$ and $\beta_j^\ell \in R$. Now we obtain a linear function in c.cone($\mathscr{P}_F \cup \mathscr{P}_L^2(C_k, D_0^L, D_k^L)$) such that

$$\left.\begin{aligned} \bar{h}_\ell(\boldsymbol{x}) &\equiv h_\ell(\boldsymbol{x}) + \sum_{j \in I_+(\ell)} \lambda_j^\ell p_j^\ell(\boldsymbol{x}) \\ &= \left(\gamma_\ell + \sum_{j \in I_+(\ell)} \lambda_j^\ell \beta_j^\ell - \sum_{i \in I_-(\ell)} \lambda_i^\ell \beta_i^\ell\right) + \left(2\boldsymbol{q}_\ell + \sum_{j \in I_+(\ell)} \lambda_j^\ell \boldsymbol{a}_j^\ell - \sum_{i \in I_-(\ell)} \lambda_i^\ell \boldsymbol{a}_i^\ell\right)^T\boldsymbol{x}\,, \end{aligned}\right\} \tag{20}$$

and an associated linear valid inequality $\bar{h}_\ell(\boldsymbol{x}) \leqslant 0$ for the feasible region F of QOP (2).

Table 1 shows the number of SDPs or LPs to be solved at every iteration and the

Table 1. Comparison among four SCRMs

Methods	#SDPs	#LPs	#Const.
DLSSDP	$2n$		$4n^2$
DLSSDP-diag	$2\sum_{\ell=1}^m \lvert I_-(\ell)\rvert$		$2\sum_{\ell=1}^m \lvert I_-(\ell)\rvert$
DLSLP		$2n$	$4n^2$
DLSLP-diag		$2\sum_{\ell=1}^m (\lvert I_+(\ell)\rvert + \lvert I_-(\ell)\rvert)$	$2\sum_{\ell=1}^m (\lvert I_+(\ell)\rvert + \lvert I_-(\ell)\rvert)$

number of constraints each problem has, comparing four variants of the SCRMs which we have discussed so far. $|P|$ denotes the number of elements contained in the set P. The entries of Table 1 are computed as

$$\#\text{SDPs} = |D_k| \quad \#\text{LPs} = |D_k| \quad \text{and} \quad \#\text{Const.} = |\mathscr{P}^2(C_k, D_0, D_k)| .$$

It should be noted that $\Sigma_{\ell=1}^{m} |I_-(\ell)| \leqslant \Sigma_{\ell=1}^{m} (|I_+(\ell)| + |I_-(\ell)|) \leqslant mn$. Hence, if $m \leqslant 2n$ holds between the number of constraints m and the number of variables n for QOP (2), '#Const.' of DLSSDP-diag (or DLSLP-diag) is smaller than that of DLSSDP (or DLSLP). In the test problems of QOP (2) of our numerical experiments reported in Section 4, the number of SDPs (or LPs) to be solved in DLSSDP-diag (or DLSLP-diag) is larger than that in DLSSDP (or DLSLP), while each problem generated by DLSSDP-diag (or DLSLP-diag) has much less constraints than those generated by DLSSDP (or DLSLP). We can confirm this fact in Tables 4 and 5 of Section 4.2.

3.2. A PARALLEL ALGORITHM

Algorithm 2.1 generates multiple LPs or SDPs at each iteration. Those LPs or SDPs can be processed simultaneously in parallel. Here we consider a parallel implementation of the algorithm on a client-server based parallel computing system as Figure 3 shows. We suppose that we have a computer system consisting of one client processor and V ($\leqslant |D_k|$) server processors; if we have more than $|D_k|$ processors available, we only use the first $|D_k|$. At the kth iteration of a new parallel Algorithm 3.1, the client processor allocate $|D_k|$ LPs or SDPs of the form

$$\alpha(C_k, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \quad (\forall \boldsymbol{d} \in D_k) ,$$

to V server processors. Thus, at the kth iteration, each server processor solves roughly $|D_k|/V$ problems in average. In the following parallel implementation of Algorithm 2.1, the work of the client processor and that of each server processor are

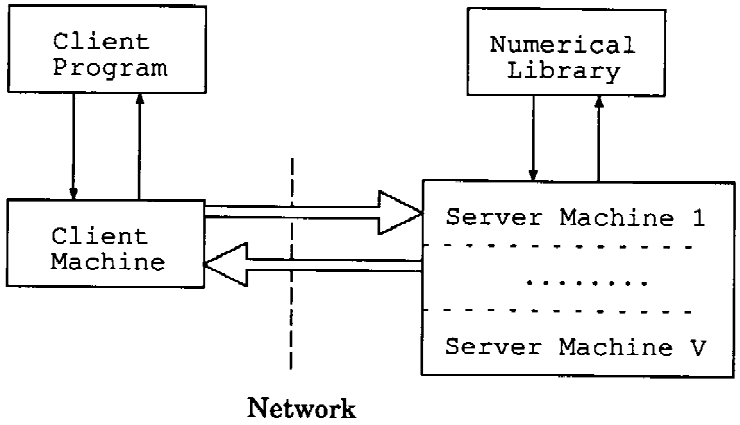

Figure 3. The client–server based parallel computing system.

described together, but each step is discriminated by the symbol (C) or (S); (C) stands for the former and (S) for the latter.

ALGORITHM 3.1. (Parallel implementation of discretized-localized SCRMs)

Step 0 (C): Define D_0 and D_1 with some value θ_1. Assign each $\boldsymbol{d} \in D_0 \cup D_1$ to an idle server processor among V server processors, and send data of $\boldsymbol{d}$ and C_0 to it.

Step 1 (S): Compute $\alpha(C_0, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_0\}$ for some $\boldsymbol{d} \in D_0 \cup D_1$ designated by the client processor. Return $\alpha(C_0, \boldsymbol{d})$ to the client processor.

Step 2 (C): Let $C_1 = C_0$ and $k = 1$.

Step 3 (C): Compute an upper bound ζ_k for the maximum objective function value of QOP (2) by $\zeta_k = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$. If ζ_k satisfies some termination criteria, then stop.

Step 4 (C): Choose a direction-set D_{k+1}. Assign each LP or SDP $\max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$ to an idle server processor; more specifically allocate each $\boldsymbol{d} \in D_{k+1}$ to an idle server processor, and send the data of $\boldsymbol{d}$, C_0, D_0, $\alpha(C_0, \boldsymbol{d}_0)$ $(\forall \boldsymbol{d}_0 \in D_0)$, D_k and $\alpha(C_k, \boldsymbol{d}_k)$ $(\forall \boldsymbol{d}_k \in D_k)$ to it.

Step 5 (S): Generate $\mathscr{P}_k = \mathscr{P}^2(C_k, D_0, D_k) \cup \mathscr{P}^L(C_0, D_0)$, and define C_{k+1}.

Step 6 (S): Compute $\alpha(C_{k+1}, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_{k+1}\}$. Return the value $\alpha(C_{k+1}, \boldsymbol{d})$ to the client processor.

Step 7 (C): Let $k = k + 1$ and go to Step 3 (C).

In our numerical experiments, we add the objective direction $\boldsymbol{c}$ to the set D_k and solve $\alpha(C_k, \boldsymbol{c}) = \max\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\}$ in some server processor. Then, in Step 3 (C), we find $\alpha(C_k, \boldsymbol{c})$ among $\alpha(C_k, \boldsymbol{d})$ $(\boldsymbol{d} \in D_k)$, and set $\zeta_k = \alpha(C_k, \boldsymbol{c})$. Therefore, the work of the client processor is only to assign each LP or SDP to one of the V server processors, to update a direction-set D_{k+1}, and to check whether ζ_k satisfies the termination criteria. The client processor avoids the computation for not only solving a bounding problem but constructing C_{k+1} $(k = 1, 2, \ldots)$.

REMARK 3.2. In Step 5 (S), the common set $\mathscr{P}_k$ is generated in each server processor. This redundant work is to reduce the communication time between the client processor and each server processor. From our preliminary numerical results, we found that sending all data of $\mathscr{P}_k$ from the client processor to each server processor took much communication time. Therefore, it is better to reduce the amount of data to be transmitted through the network as much as possible.

REMARK 3.3. If we have enough processors to handle $|D_0| \cup |D_k|$ problems in parallel, it is better to solve all $(|D_0| + |D_k|)$ problems;

$$\alpha(C_k, \boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_k\} \quad (\forall \boldsymbol{d} \in D_0 \cup D_k)$$

at every kth iteration, and construct C_{k+1} using $\alpha(C_k, \boldsymbol{d})$ instead of $\alpha(C_0, \boldsymbol{d})$ for

$\forall \boldsymbol{d} \in D_0$. Then we can obtain a tighter relaxation C_{k+1} of the nonconvex feasible region F of QOP (2).

4. Computational experiments

In this section, we present our four kinds of test problems, describe some implementation details on Algorithms 2.1 and 3.1, and report some encouraging numerical results.

4.1. TEST PROBLEMS

We summarize some basic characteristics of our test problems in Tables 2 and 3. They consist of four types of problems such as (a) 0–1 integer QOPs, (b) linearly constrained QOPs, (c) bilevel QOPs, and (d) fractional QOPs. We transform these four types of problems (a), (b), (c) and (d) into QOPs of the form (2). The type of each test problem is denoted in the second column of Tables 2 and 3. The columns n and m denote the number of variables and the number of constraints (not including box constraints) of the transformed QOP (2), respectively. The column '#QC' denotes the number of quadratic constraints among m constraints of QOP (2). The last column gives the number of local optima for some types of the test problems. We denote '?' for cases where the number of local optima is not available. We know optimal values for all test problems of Tables 2 and 3 in advance. We give more precise description for problems (a)–(d) below.

(a) **0-1IQOP** (0–1 integer QOP):

$$\min \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} \text{ subject to } \boldsymbol{x} \in \{0, 1\}^n .$$

Table 2. Small size test problems

Problem	Type	Source	n	m	#QC	#Local
01int20	0-1IQOP	[15]	21	21	21	?
01int30	0-1IQOP	[15]	31	31	31	?
LC30-36	LCQOP	[4]	31	46	1	36
LC30-162	LCQOP	[4]	31	46	1	162
LC40-36	LCQOP	[4]	41	61	1	6
LC30-72	LCQOP	[4]	41	61	1	72
LC50-1296	LCQOP	[4]	51	76	1	1296
BLevel3-6	BLQOP	[3]	19	25	10	4
BLevel8-3	BLQOP	[3]	21	22	10	4
Frac20-10	FQOP	–	21	12	1	?
Frac30-15	FQOP	–	31	17	1	?

Table 3. Large size test problems

Problem	Type	Source	n	m	#QC	#Local
01int50	0-1IQOP	[15]	51	51	51	?
01int55	0-1IQOP	[15]	56	56	56	?
01int60	0-1IQOP	[15]	61	61	61	?
LC60-72	LCQOP	[4]	61	91	1	72
LC70-72	LCQOP	[4]	71	106	1	144
LC80-144	LCQOP	[4]	81	121	1	144
BLevel30-4	BLQOP	[3]	47	29	13	8
BLevel40-4	BLQOP	[3]	57	29	13	8
Frac50-20	FQOP	–	51	22	1	?
Frac60-20	FQOP	–	61	22	1	?
Frac70-25	FQOP	–	71	27	1	?

We used the code of Pardalos and Rodgers [15] to generate coefficient matrices $\boldsymbol{Q}$ of the test problems.

(b) **LCQOP** (Linearly constrained QOP):

$$\min \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} \quad \text{subject to } \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b},$$

where $\boldsymbol{Q} \in R^{n\times n}$, $\boldsymbol{q} \in R^n$, $\boldsymbol{x} \in R^n$, $\boldsymbol{b} \in R^m$ and $\boldsymbol{A} \in R^{m\times n}$. We generate each test LCQOP by the code of Calamai et al. [4]. Their construction of LCQOP provides not only its optimal solution but the number of its local minima.

(c) **BLQOP** (Bilevel QOP):

$$\begin{aligned} &\min_{x} && \gamma + 2\boldsymbol{q}^T\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{Q}\boldsymbol{z} \\ &\text{subject to} \\ &\qquad \min_{y} && \boldsymbol{z}^T\boldsymbol{R}\boldsymbol{z} \\ &\qquad \text{subject to} && \boldsymbol{A}\boldsymbol{z} \leqslant \boldsymbol{b}, \quad \boldsymbol{z} = \begin{pmatrix}\boldsymbol{x}\\ \boldsymbol{y}\end{pmatrix}, \end{aligned}$$

where $\boldsymbol{x} \in R^p$, $\boldsymbol{y} \in R^q$, $\boldsymbol{z} \in R^n$ with $n = p + q$, $\boldsymbol{A} \in R^{m\times n}$, $\boldsymbol{Q} \in \mathcal{S}^n_+$ and $\boldsymbol{R} \in \mathcal{S}^n_+$. We generate each test BLQOP by the code of Calamai and L.N. Vincente [3]. See Takeda and Kojima [25] to know how to transform BLQOP into QOP (2).

(d) **FQOP** (Fractional QOP):

$$\begin{aligned} &\min && g(\boldsymbol{x}) = \frac{1/2\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x}}{\boldsymbol{q}^T\boldsymbol{x} - \gamma} \\ &\text{subject to} && \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b},\ \boldsymbol{q}^T\boldsymbol{x} \geqslant 3/2\gamma. \end{aligned}$$

Here $\boldsymbol{x} \in R^n$, $\boldsymbol{q} \in R^n$, $\boldsymbol{A} \in R^{m\times n}$, $\boldsymbol{Q}$: $n \times n$ positive definite matrix and $\gamma \equiv 1/2\boldsymbol{q}^T\boldsymbol{Q}^{-1}\boldsymbol{q}$. If we take a constant term $\boldsymbol{b} \in R^m$ so that $\boldsymbol{A}\boldsymbol{Q}^{-1}\boldsymbol{q} < \boldsymbol{b}$ holds, the

above FQOP has an optimal value 1. Indeed, note that FQOP is equivalent to the problem finding $\lambda^* \geqslant 0$ such that $\pi(\lambda^*) = 0$, where

$$\left.\begin{aligned} &\pi(\lambda) = \min\{1/2\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \lambda(\boldsymbol{q}^T\boldsymbol{x} - \gamma) : \boldsymbol{x} \in X\}, \\ &\text{where } X \equiv \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b},\ \boldsymbol{q}^T\boldsymbol{x} \geqslant 3/2\gamma\}. \end{aligned}\right\} \tag{21}$$

We see that the problem

$$\min\{1/2\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} - \boldsymbol{q}^T\boldsymbol{x} + \gamma\} \tag{22}$$

has an optimal solution $\boldsymbol{x}^* = \boldsymbol{Q}^{-1}\boldsymbol{q}$, since $\boldsymbol{Q}$ is a positive definite matrix. Then, the optimal solution $\boldsymbol{x}^*$ of (22) achieves $\pi(1) = 0$ for the problem (21), and hence, FQOP generated by this technique has the optimal value $\lambda^* = g(\boldsymbol{x}^*) = 1$.

4.2. NUMERICAL RESULTS

To execute Algorithms 2.1 and 3.1, it is necessary to clarify the issues (A) the value θ_k for D_k ($k = 1, 2, \ldots$), (B) termination criteria, and (C) SCRMs to be used.

(A) We start Algorithms 2.1 and 3.1 by constructing a direction-set D_1 with $\theta_1 = \pi/2$. If the decrease $|\zeta_k - \zeta_{k-1}|$ in bounds for the optimal value becomes small at the kth iteration of Algorithms 2.1 and 3.1, we choose some smaller value than θ_k for θ_{k+1}, and reconstruct the direction-set D_{k+1} using the updated θ_{k+1}. Otherwise, we use the same $\theta_{k+1} = \theta_k$ and the same direction-set $D_{k+1} = D_k$ for the next iteration. Throughout the computational experiments, we use the following rule for updating θ_k:

Let $\ell = 1$, $\theta_1 = \pi/2$, $K = 4$ and $\{\sigma_j\}_{j=1}^K$ be a decreasing sequence such that $\{1, \frac{8}{9}, \frac{4}{9}, \frac{2}{9}\}$. If a bound ζ_k generated at the kth iteration for the optimal value remains to satisfy

$$\frac{|\zeta_{k-1} - \zeta_k|}{\max\{|\zeta_k|, 1.0\}} \geqslant 1.0^{-3} \times \sigma_\ell,$$

then set $\theta_{k+1} = \theta_k$. Else if $\ell < K$, then set $\ell = \ell + 1$ and $\theta_{k+1} = \sigma_\ell\theta_1$, which implies an update of D_{k+1}.

(B) If $\ell = K$ and $|\zeta_{k-1} - \zeta_k|/\max\{|\zeta_k|, 1.0\} < 1.0^{-3} \times \sigma_K$, we terminate the algorithm with the best bound ζ_k for the optimal value.

(C) We choose two SCRMs, a serial implementation of DLSLP and a parallel implementation of DLSSDP-diag for comparison. The former is coincident with the practical SCRM presented in the paper [24]. We also tried to compare them with a serial implementation of DLSSDP to see the effectiveness of the inequality reducing technique of DLSSDP-diag described in Section 3.1. When θ_k becomes small, however, each SDP subproblem of DLSSDP contains so many similar constraints, induced from the pairwise products of linear supporting functions with similar directions in D_k (defined by (9)), that our SDP solver

SDPA [7] used in DLSSDP and DLSSDP-diag encounters serious numerical instabilities. Because of this reason, DLSSDP could not solve most of the test problems of Tables 2 and 3. Even when DLSSDP did not fail, it required tremendous cpu time. For example, DLSSDP could solve the problem '01int20', which is one of the smallest size problems listed in Table 2, but it required 40631 seconds, more than 10 hours. On the other hand, DLSSDP-diag spent 1070 seconds for the same problem as shown below in Table 6. Therefore we compare a parallel implementation of DLSSDP-diag only with a serial implementation of DLSLP in the remainder of this section.

The programs of Algorithms 2.1 and 3.1 were coded in ANSI C++ language. We used CPLEX Version 6.5 as an LP solver in DLSLP, and SDPA Version 5.0 [7] as an SDP solver in DLSSDP-diag. We implemented DLSSDP-diag on a parallel computing system called Ninf (Network-based Information Library for high performance computing) [18, 19]. The basic Ninf system supports client-server based computing, and provides a global network-wide computing infrastructure for high-performance numerical computation services. It intends not only to exploit high performance in global network parallel computing, but also to provide a simple programming interface similar to conventional function calls in existing languages.

Our experiments were conducted to see the following three factors: (i) comparison between DLSLP and DLSSDP-diag with respect to the number of problems (LPs or SDPs) generated at every iteration and the size of each problem; (ii) the accuracy of bounds obtained by DLSSDP-diag and DLSLP for optimal values; (iii) computational efficiency of parallel DLSSDP-diag using 1, 2, 4, 8, 16, 32, 64 and 128 server processors.

(i) Tables 4 and 5 show the number of LPs ($\#\text{LPs} = |D_k|$) generated at each iteration of DLSLP and the number of SDPs ($\#\text{SDPs} = |D_k|$) of DLSSDP-diag. Also, they show the number of constraints ($\#\text{Tot\_const.} = |\mathscr{P}_F| + |\mathscr{P}^2(C_k, D_0, D_k)| + |\mathscr{P}^L(C_0, D_0)|$ in each problem. We see from these tables that SDPs of DLSSDP-diag have much less constraints than LPs of DLSLP.

(ii) We summarize numerical results on a parallel implementation of DLSSDP-diag in Tables 6 and 7, and summarize those on a serial implementation of DLSLP in Table 8. We use the notation below in those tables: *r.Err*, the relative error of a solution, i.e., $r.Err = |f_{\text{up}} - f_{\text{opt}}|/\max\{|f_{\text{opt}}|, 1.0\}$; f_{opt}, the global optimal value of QOP (2); f_{up}, the best bound found by each algorithm for f_{opt}; $r.Err^1$, *r.Err* at the first iteration; *r.Err**, *r.Err* at the last iteration; iter., the number of iterations each algorithm repeated; R.time, the real time in second; C.time, the cpu time in second.

We ran DLSSDP-diag using 128 processors of 64 server machines and one processor of a client machine. We slightly modified Algorithm 3.1 according to what we have mentioned in Remark 3.3 so that the modified algorithm generates ($|D_0| + |D_k|$) SDPs at every iteration. Note that the number ($|D_0| + |D_k|$) is almost twice of $\#$SDPs described in Tables 4 and 5. Tables 6 and 7

Table 4. LPs and SDPs generated by DLSLP and DLSSDP-diag for small size test problems

Problem	DLSLP		DLSSDP-diag	
	#LPs	#Tot_const.	#SDPs	#Tot_const.
01int20	40	1621	81	122
01int30	60	3631	117	178
LC30-36	60	3646	61	137
LC30-162	60	3646	61	137
LC40-6	80	6461	81	182
LC40-72	80	6461	81	182
BLevel3-6	36	1321	55	98
BLevel8-3	40	1622	59	101
Frac20-10	40	1612	43	76
Frac30-15	60	3617	63	110

include not only solution information achieved by DLSSDP-diag but time information such as $C \Rightarrow S$ (the total transmitting time from the client processor to each server processor), exec.time (the total execution time on server processors), and $S \Rightarrow C$ (the total transmitting time from each server processor to the client processor). These time data were measured in real time. As we stated in Remark 3.2, a small amount of data are transmitted through a network

Table 5. LPs and SDPs generated by DLSLP and DLSSDP-diag for large size test problems

Problem	DLSLP		DLSSDP-diag	
	#LPs	#Tot_const.	#SDPs	#Tot_const.
01int50	100	10051	201	302
01int55	110	12156	221	332
01int60	120	14461	241	362
LC60-72	120	14491	121	272
LC70-72	140	19706	141	317
LC80-144	160	25721	161	362
BLevel30-4	92	8493	117	192
BLevel40-4	112	12573	137	222
Frac50-20	100	10022	103	175
Frac60-20	120	14422	123	205
Frac70-25	140	19627	143	240

Table 6. Numerical results of DLSSDP-diag on a PC cluster (small size test problems)

Problem	DLSSDP-diag				Time Info (s)		
	$r.Err^1$	$r.Err^*$	iter.	R.time (s)	C ⇒ S	exec.time	S ⇒ C
01int20	8.34	6.23	6	24	0.07	1070	0.06
01int30	6.20	2.98	6	58	0.11	5811	0.10
LC30-36	100.00	5.30	9	28	0.09	1319	0.09
LC30-162	100.00	27.42	18	55	0.18	2790	0.20
LC40-6	100.00	0.89	8	43	0.12	3292	0.13
LC40-72	100.00	4.14	9	52	0.14	3783	0.14
BLevel3-6	6.53	2.44	13	18	0.11	847	0.10
BLevel8-3	6.53	2.45	13	28	0.12	1114	0.14
Frac20-10	89.36	0.92	27	54	0.22	3166	0.18
Frac30-15	89.58	0.88	26	345	0.37	25913	0.35

in the parallel implementation of DLSSDP-diag, so that we can take little notice of transmitting time between client and server processors.

Table 8 presents our numerical results on a serial implementation of DLSLP. DLSLP cannot deal with the large size test QOPs of Table 5 due to the shortage of memory on our computational environment. Thus we show our numerical results of DLSLP restricted to the small size test QOPs.

Table 7. Numerical results of DLSSDP-diag on a PC cluster (large size test problems)

Problem	DLSSDP-diag				Time Info (s)		
	$r.Err^1$	$r.Err^*$	iter.	R.time (s)	C ⇒ S	exec.time	S ⇒ C
01int50	107.40	104.74	3	267	0.17	26127	0.12
01int55	100.15	75.37	5	905	0.31	86000	0.23
01int60	105.20	102.53	3	607	0.22	65560	0.15
LC60-72	100.00	3.13	8	171	0.22	12627	0.20
LC70-72	100.00	2.78	8	183	0.27	18601	0.24
LC80-144	123.70	2.94	8	406	0.44	35000	0.32
BLevel30-4	12.41	8.75	13	167	0.29	11326	0.27
BLevel40-4	12.40	9.08	12	230	0.33	192970	0.27
Frac50-20	89.50	0.94	26	3200	0.75	305002	0.78
Frac60-20	89.53	0.97	26	6318	1.30	734037	0.98
Frac70-25	89.37	1.50	25	14196	1.72	1483764	1.21

DLSSDP-diag is executed on the PC cluster consisting of one client machine and 64 server machines with 128 processors. Each server machine has two processors (CPU Pentium III 800 MHz) with 640 MB memory.

Table 8. Numerical results of DLSLP on a single processor (small size test problems)

Problem	DLSLP			
	$r.Err^1$	$r.Err^*$	iter.	C.time (s)
01int20	51.40	48.84	6	50.90
01int30	1.90	1.90	5	36.53
LC30-36	51.45	38.22	9	185.03
LC30-162	74.45	58.15	11	215.83
LC40-6	38.09	27.45	9	454.22
LC40-72	57.53	44.77	9	548.88
BLevel3-6	100.00	43.99	39	86.50
BLevel8-3	100.00	100.00	5	19.97
Frac20-10	100.00	100.00	5	18.58
Frac30-15	100.00	100.00	5	149.15

DLSLP is implemented on one processor of DEC Alpha Workstation (CPU 600 MHz, 1 GB memory).

From comparison between Table 6 and Table 8, we see that the bounds for optimal values obtained in DLSSDP-diag are more accurate than those in DLSLP in most cases. Especially for fractional QOPs, DLSSDP-diag improves bounds for optimal values significantly, compared with DLSLP. Therefore we expect DLSSDP-diag to be a practical bounding method for some difficult nonconvex QOPs, if enough processors for a parallel implementation are available. On the other hand, DLSLP has the merit that it attains a bound for the optimal value fast as Table 8 shows. We have no choice but to use different processors for numerical experiments of DLSSDP-diag and DLSLP due to the commercial license of the CPLEX software, so that comparison in computational time between these two methods would be ambiguous.

Table 9. Computational efficiency by increasing the number of processors

#Proc.	LC80-144		Frac50-20	
	R.time (s)	Ratio	R.time (s)	Ratio
1	33125	1.00	289259	1.00
2	16473	2.01	145980	1.98
4	8238	4.02	72343	3.99
8	4145	7.99	36272	7.97
16	2099	15.78	18595	15.56
32	1118	29.62	0424	30.69
64	624	53.08	4822	60.00
128	361	91.76	3200	90.39

DLSSDP-diag is executed on the PC cluster consisting of one client machine and 64 server machines with 128 processors. Each server machine has two processors (CPU Pentium III 800 MHz) with 640 MB memory.

(iii) Table 9 shows computational efficiency in proportion to the number of server processors. The 'Ratio' stands for R.time of ($\#$proc. = 1) divided by R.time of ($\#$proc. = k). If the ratio is sufficiently close to k ($=\#$proc.), we may regard Algorithm 3.1 as well paralleled. As Table 9 shows, the algorithm is well paralleled with relatively small number of $\#$proc., since the number of SDPs is sufficiently large in comparison with $\#$proc. so that such SDPs are allocated to each server processor in balance and the total computational time consumed by each server machine is almost the same. Therefore a good performance of parallel computation is attained.

5. Concluding remarks

We have proposed DLSSDP-diag, a new variant of discretized-localized successive SDP relaxation methods for QOP (2) which is suitable for a parallel implementation. The numerical results have demonstrated that DLSSDP-diag implemented in a client–server-based parallel computing system obtains better bounds for optimal values in most test problems than DLSLP, a serial variant of discretized-localized successive lift-and-project LP relaxation methods. The key feature of DLSSDP-diag is an effective construction of SDP relaxations based on the eigenvalue structure of the coefficient matrices $\boldsymbol{Q}_\ell$ ($\ell = 1, 2, \ldots, m$) of the constraint inequalities of QOP (2), which considerably reduces the number of constraints involved in the SDPs to be solved at each iteration. This makes DLSSDP-diag handle larger size test QOPs than ones DLSLP can attack. As the first parallel implementation of successive convex relaxation methods, our numerical experiment is satisfactory.

In general, the SDP relaxation is much more expensive than the LP relaxation although the former is known to attain better bounds for optimal values than the latter. Furthermore the number of relaxed SDPs to be solved at each iteration of DLSSDP-diag increases as the number of negative eigenvalues involved in the constraint coefficient matrices $\boldsymbol{Q}_\ell$ ($\ell = 1, 2, \ldots, m$) increases. Therefore a powerful parallel computing facility is inevitable to apply DLSSDP-diag to highly nonconvex large size QOPs. If computer environment develops further in future, DLSSDP-diag can be a practical bounding method for optimal values of such QOPs. At present, a practical compromise may be to use DLSSDP-diag in the branch-and-bound framework; we can terminate DLSSDP-diag within a few iteration to get a relatively good bound for the optimal value of a QOP, branch the QOP into multiple subproblems, and then apply DLSSDP-diag to each subproblem. The numerical results of the paper [24] show that the drastic decrease of the relative error occurs at an early stage of the execution of DLSLP.

Acknowledgment

The authors would like to thank Professor Satoshi Matsuoka of Tokyo Institute of

Technology. He offered them to use an advanced PC cluster in his laboratory for their research.

References

[1] Audet, C., Hansen, P., Jaumard, B. and Savard, G. (2000), A Branch and Cut Algorithm for Nonconvex Quadratically Constrained Quadratic Programming, *Mathematical Programming* 87, 131–152.
[2] Balas, E., Ceria, S. and Cornuéjols, G. (1993), A Lift-and-Project Cutting Plane Algorithm for Mixed 0-1 Programs, *Mathematical Programming* 58, 295–323.
[3] Calamai, P.H. and Vicente, L.N. (1994), Generating Quadratic Bilevel Programming Problems, *ACM Transactions on Mathematical Software* 20, 103–122.
[4] Calamai, P.H., Vincente, L.N. and Judice, J.J. (1993), A New Technique for Generating Quadratic Programming Test Problems, *Mathematical Programming* 61, 215–231.
[5] Fujie, T. and Kojima, M. (1997), Semidefinite Relaxation for Nonconvex Programs, *Journal of Global Optimization* 10, 367–380.
[6] Fujisawa, K., Kojima, M. and Nakata, K. (1997), Exploiting Sparsity in Primal-Dual Interior-Point Methods for Semidefinite Programming, *Mathematical Programming* 79, 235–253.
[7] Fujisawa, K., Kojima, M. and Nakata, K. (1999), SDPA (Semidefinite Programming Algorithm), User's Manual, Technical Report B-308, Tokyo Institute of Technology, Oh-Okayama, Meguro, Tokyo 152-8552, Japan.
[8] Kojima, M. and Takeda, A. (2001), Complexity Analysis of Successive Convex Relaxation Methods for Nonconvex Sets, *Mathematics of Operations Research* 26, 519–542.
[9] Kojima, M. and Tunçel, L. (2000), Cones of Matrices and Successive Convex Relaxations of Nonconvex Sets, *SIAM Journal on Optimization* 10, 750–778.
[10] Kojima, M. and Tunçel, L. (2000), Discretization and Localization in Successive Convex Relaxation Method for Nonconvex Quadratic Optimization Problems, *Mathematical Programming* 89, 79–111.
[11] Lovász, L. and Schrijver, A. (1991), Cones of Matrices and Set Functions and 0–1 Optimization, *SIAM Journal on Optimization* 1, 166–190.
[12] Horst, R. and Tuy, H. (1992), *Global Optimization, Second, Revised Edition*, Springer, Berlin.
[13] McCormick, G.P. (1976), Computability of Global Solutions to Factorable Nonconvex Programs: Part I—Convex Underestimating Problems, *Mathematical Programming* 10, 147–175.
[14] Nicholls, M.G. (1996), The Application of Non-Linear Bilevel Programming to the Aluminium Industry, *Journal of Global Optimization* 8, 245–261.
[15] Pardalos, P.M. and Rodgers, G. (1990), Computational Aspects of a Branch and Bound Algorithm for Quadratic 0-1 Programming, *Computing* 45, 131–144.
[16] Pardalos, P.M. and Vavasis, S.A. (1991), Quadratic Programming with One Negative Eigenvalue is NP-Hard, *Journal of Global Optimization* 1, 843–855.
[17] Ryoo, H.S. and Sahinidis, N.V. (1996), A Branch-and-Reduce Approach to Global Optimization, *Journal of Global Optimization* 8, 107–139.
[18] Sato, M., Nakada, H., Sekiguchi, S., Matsuoka, S., Nagashima, U. and Takagi, H. (1997), Ninf: A Network Based Information Library for a Global World-Wide Computing Infrastructure, in Sloot, P. and Hertzberger, B. (eds.), *High-Performance Computing and Networking, Lecture Notes in Computing Science*, Vol. 1225, Springer, Berlin.
[19] S. Sekiguchi, Sato, M., Nakada, H., Matsuoka, S. and Nagashima, U. (1996), Ninf: Network Based Information Library for Globally Performance Computing, in *Proc. of Parallel Object-Oriented Methods and Applications (POOMA'96).*

[20] Sherali, H.D. and Adams, W.P. (1990), A Hierarchy of Relaxations Between the Continuous and Convex Hull Representations for Zero-One Programming Problems, *SIAM Journal on Discrete Mathematics* 3, 411–430.
[21] Sherali, H.D. and Alameddine, A. (1992), A New Reformulation-Linearization Technique for Bilinear Programming Problems, *Journal of Global Optimization* 2, 379–410.
[22] Sherali, H.D. and Tuncbilek, C.H. (1995), A Reformulation–Convexification Approach for Solving Nonconvex Quadratic Programming Problems, *Journal of Global Optimization* 7, 1–31.
[23] Stubbs, R.A. and Mehrotra, S. (1997), A Branch-and-Cut Method for 0-1 Mixed Convex Programming, Technical Report 96-01, Dept. of IE/MS, Northwestern University, Evanston, IL 60208.
[24] Takeda, A., Dai, Y., Fukuda, M. and Kojima, M. (2000), Towards the Implementation of Successive Convex Relaxation Method for Nonconvex Quadratic Optimization Problems, in Pardalos, P.M. (ed.), *Approximation and Complexity in Numerical Optimization: Continuous and Discrete Problems*, Kluwer Academic Publishers, Dordrecht.
[25] Takeda, A. and Kojima, M. (2000), Successive Convex Relaxation Approach to Bilevel Quadratic Optimization Problems, in Ferris, M.C., Mangasarian, O.L. and Pang, J.S. (eds.), *Applications and Algorithms of Complementarity*, Kluwer Academic Publishers, Dordrecht.
[26] Vicente, L.N. and Calamai, P.H. (1994), Bilevel and Multilevel Programming: A Bibliography Review, *Journal of Global Optimization* 5, 291–306.
[27] Visweswaran, V., Floudas, C.A., Ierapetritou, M.G. and Pistikopoulos, E.N. (1996), A Decomposition-Based Global Optimization Approach for Solving Bilevel Linear and Quadratic Programs, in Floudas, C.A. and Pardalos, P.M. (eds.), *State of the Art in Global Optimization*, Kluwer Academic Publishers, Dordrecht.
[28] Voorhis, T.V. and Al-khayyal, F. (1996), Accelerating Convergence of Branch-and-Bound Algorithms for Quadrically Constrained Optimization Problems, in Floudas, C.A. and Pardalos, P.M. (1996), *State of the Art in Global Optimization*, Kluwer Academic Publishers, Dordrecht.