Successive Convex Relaxation Approach to Bilevel Quadratic Optimization Problems

Akiko Takeda and Masakazu Kojima
Department of Mathematical and Computing Science
Tokyo Institute of Technology

August 4, 1999

Abstract. The quadratic bilevel programming problem is an instance of a quadratic hierarchical decision process where the lower level constraint set is dependent on decisions taken at the upper level. By replacing the inner problem by its corresponding KKT optimality conditions, the problem is transformed to a single yet non-convex quadratic program, due to the complementarity condition. In this paper we adopt the successive convex relaxation approach proposed by Kojima and Tunçel for computing a convex relaxation of a nonconvex feasible region. By further exploiting the special structure of the bilevel problem, we establish new techniques which enable the efficient implementation of the proposed algorithm. The performance of these techniques is tested in a comparison with other procedures using a number of test problems of quadratic bilevel programming.

1 Introduction.

Bilevel programming (abbreviated by BP) belongs to a class of nonconvex global optimization problems. It arises where decisions are made in a two-level hierarchical order and each decision maker has no direct control or influence upon the decision of the other, but actions taken by one decision maker affect returns from the other. Such problems can be formulated as two levels of nested mathematical programs as follows:

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{x}} & F(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x})) \\ \text{subject to} & \boldsymbol{y}(\boldsymbol{x}) \in \operatorname{argmin}\limits_{\boldsymbol{y}}\{G(\boldsymbol{x},\boldsymbol{y}) : \boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) \le 0\}, \\ & \boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x})) \le 0, \\ & \boldsymbol{w} = \begin{pmatrix} \boldsymbol{x} \\ \boldsymbol{y}(\boldsymbol{x}) \end{pmatrix} \in G_0, \end{array}\right\} \tag{1}$$

where $\boldsymbol{g}(\boldsymbol{x},\boldsymbol{y}) : R^n \to R^{m_1}$, $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x})) : R^n \to R^{m_2}$, $G(\boldsymbol{x},\boldsymbol{y}) : R^n \to R$, $F(\boldsymbol{x},\boldsymbol{y}(\boldsymbol{x})) : R^n \to R$, and G_0 denotes a nonempty compact polyhedral subset of R^n. Given an action $\boldsymbol{x}$ in the upper level, the lower level decision maker returns a minimizer $\boldsymbol{y}(\boldsymbol{x})$ of $G(\boldsymbol{x},\boldsymbol{y})$ subject to the

constraints $\boldsymbol{g}(\boldsymbol{x}, \boldsymbol{y}) \leq \boldsymbol{0}$ to the upper level. As a whole, the upper level decision maker needs to maximize his objective function $F(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x}))$ subject to the constraints $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) \leq \boldsymbol{0}$ and $(\boldsymbol{x}, \boldsymbol{y}(\boldsymbol{x})) \in G_0$.

Applications of BPs are numerous; for example, (i) hierarchical decision making policy problems in mixed economies, where policy makers at the top level influence the decisions of private individuals and companies, (ii) network facility location with delivered price competition, (iii) the Portland Aluminium Smelter in Victoria, Australia [12], aiming to maximize the aluminium production while minimizing the main cost associated with the production. See Vicente and Calamai [19] for a recent comprehensive review of the literature.

We call a BP lower-convex if the lower level objective function $G(\boldsymbol{x}, \boldsymbol{y})$ and constraint functions $g_i(\boldsymbol{x}, \boldsymbol{y})$ $(i = 1, 2, \ldots, m_1)$ are all convex in $\boldsymbol{y}$ for each fixed value of $\boldsymbol{x}$. Among the BPs, lower-convex BPs have received most of the attention in the literature. The advantage of dealing with lower-convex BPs is that under an appropriate constraint qualification, the lower level problem can be replaced by its Karush-Kuhn-Tucker (KKT) optimality condition to obtain an equivalent (one-level) mathematical program. There are three important classes of lower-convex BPs, namely:

(i) linear BPs, where all functions involved are affine,

(ii) linear-quadratic BPs, where $G(\boldsymbol{x}, \boldsymbol{y})$ is convex quadratic and all remaining functions are affine,

(iii) quadratic BPs, which differ from linear-quadratic BPs in that $F(\boldsymbol{x}, \boldsymbol{y})$ can also be a quadratic function.

In this paper, we propose new algorithms, based on the successive convex relaxation method [7, 8, 9, 17], for a more general class of BPs than the class (iii) by allowing that some of $g_i(\boldsymbol{x}, \boldsymbol{y})$ $(i = 1, 2, \ldots, m_1)$ are convex quadratic functions and some of $f_j(\boldsymbol{x}, \boldsymbol{y})$ $(j = 1, 2, \ldots, m_2)$ are (not necessarily convex) quadratic functions. We could further weaken the assumption of quadratic functions to a wide class of nonlinear functions according to the technique proposed by [6], but for simplicity of discussions, we will restrict ourselves to BQOPs, bilevel quadratic optimization problems where all functions involved in BPs are linear or quadratic. In these cases, the application of the KKT optimality condition to the lower level problem results in a one-level nonconvex QOP (quadratic optimization problem) including complementarity constraints. We further transform the QOP into a bounded constraint QOP having a linear objective function in order to utilize the successive convex relaxation method.

There are many variants of the successive convex relaxation method for a general QOP having a linear objective function $\boldsymbol{c}^T\boldsymbol{x}$ to be maximized over a compact feasible region F represented by linear or quadratic functions. The original method was proposed by Kojima and Tunçel [8]. It can be seen as an extension of the lift-and-project procedure [10] with the use of the SDP (semidefinite programming) relaxation or the SILP (semi-infinite linear programming) relaxation for 0-1 integer programs to general QOPs. (The latter relaxation is also called the RLT (reformulation-linearization technique) in the literature [14, 15]). The method generates a sequence of compact convex relaxations of F converging to the convex

hull of F. This original method is conceptual in the sense that we need to solve "*infinitely many* convex programs (linear programs or semidefinite programs) having *infinitely many* inequality constraints" at each iteration. See also the paper [7] for complexity analysis of the conceptual successive relaxation method.

In their succeeding paper [9], Kojima and Tunçel proposed two types of techniques which discretize and localize (along the objective direction $\boldsymbol{c}$) "*infinitely many* convex programs having *infinitely many* inequality constraints" in order to resolve the difficulty of implementing their original conceptual method of [8]. Given an accuracy $\epsilon > 0$, their discretized-localized variant attains an upper bound for the optimal value within ϵ in a finite number of iterations.

More recently, Takeda, Dai, Fukuda and Kojima [17] presented practical variants of the successive convex relaxation method by slimming down the discretized-localized variant of [9] to overcome a rapid explosion in the number of convex programs to be solved at each iteration. Although the methods of [17] no more enjoy the approximate optimality within a prescribed accuracy, the numerical results reported in the paper [17] look promising. In this paper, we adapt the discretized-localized SILP relaxation variant of [17] to BQOPs taking full advantage of a special structure of BQOPs, *i.e.,* the complementarity constraints induced from the KKT optimality condition on the lower level problem.

The paper is organized as follows. In the next section, we formally define a BQOP, and present its reformulation via the KKT optimality condition. Section 3 is devoted to a brief description of the successive convex relaxation method. In Section 4, we explain two types of techniques exploiting the special structure of the BQOP, and illustrate the techniques using a small example. In Section 5, we report some numerical results to see the effectiveness of our method. Finally, we conclude the paper by giving some discussions and possible future work in Section 6.

2 Bilevel Quadratic Optimization Problem.

Consider a bilevel quadratic optimization problem (BQOP) having a linear objective function:

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{u}} & \boldsymbol{c}_1^T\boldsymbol{v} + \boldsymbol{c}_2^T\boldsymbol{u} \\ \text{subject to} & \boldsymbol{v} \in \arg\min\limits_{\boldsymbol{v}} \{g_0(\boldsymbol{v},\boldsymbol{u}) : g_i(\boldsymbol{v},\boldsymbol{u}) \le 0 \ (i=1,\ldots,m_1)\}, \\ & f_j(\boldsymbol{v},\boldsymbol{u}) \le 0 \ (j=1,\ldots,m_2), \quad \boldsymbol{w} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \in G_0. \end{array}\right\} \tag{2}$$

where

$$\begin{array}{rcl} \boldsymbol{v} \in R^{n_1} & : & \text{the lower level variable vector,} \\ \boldsymbol{u} \in R^{n_2} & : & \text{the upper level variable vector,} \\ n & \equiv & n_1 + n_2, \\ G_0 & : & \text{a nonempty compact polyhedral subset of } R^n, \\ \boldsymbol{c}_1 & \in & R^{n_1},\ \boldsymbol{c}_2 \in R^{n_2}, \\ g_i(\cdot,\cdot), f_j(\cdot,\cdot) \in \mathcal{Q} & : & \text{a quadratic function on } R^n \end{array}$$

$$
\begin{aligned}
& (i = 0, 1, \ldots, m_1,\ j = 1, 2, \ldots, m_2), \\
\boldsymbol{g}(\cdot, \cdot) &= \begin{pmatrix} g_1(\cdot, \cdot) \\ \vdots \\ g_{m_1}(\cdot, \cdot) \end{pmatrix} : R^n \to R^{m_1}, \\
\boldsymbol{f}(\cdot, \cdot) &= \begin{pmatrix} f_1(\cdot, \cdot) \\ \vdots \\ f_{m_2}(\cdot, \cdot) \end{pmatrix} : R^n \to R^{m_2}.
\end{aligned}
$$

If a given BQOP has a quadratic objective function in the upper level, we could transform the problem into the form (2) above by adding a constraint "the objective function$-t = 0$" and replacing the objective function by t. We impose the following condition on the BQOP (2).

Condition 2.1.

(i) The quadratic function $g_i(\cdot, \boldsymbol{u}),\ i = 0, \ldots, m_1$, is convex on $R^{n_1} \times \{\boldsymbol{u}\}$ for fixed $\boldsymbol{u} \in R^{n_2}$.

(ii) If

$$
\begin{aligned}
& \sum_{i=1}^{m_1} \lambda_i \nabla_v g_i(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{0},\ g_i(\boldsymbol{v}, \boldsymbol{u}) \le 0,\ \lambda_i g_i(\boldsymbol{v}, \boldsymbol{u}) = 0,\ 0 \le \lambda_i\ (i = 1, \ldots, m_1), \\
& f_j(\boldsymbol{v}, \boldsymbol{u}) \le 0\ (j = 1, \ldots, m_2),\ \boldsymbol{w} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \in G_0,
\end{aligned}
$$

then $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_{m_1})^T = \boldsymbol{0}$ (A strong version of the Mangasarian Fromovitz constraint qualification).

Applying the Karush-Kuhn-Tucker (KKT) optimality condition to the lower level convex problem the BQOP (2) under Condition 2.1, we can reduce the constraint

$$
\boldsymbol{v} \in \arg\min_{\boldsymbol{v}} \{g_0(\boldsymbol{v}, \boldsymbol{u}) : g_i(\boldsymbol{v}, \boldsymbol{u}) \le 0\ (i = 1, \ldots, m_1)\}
$$

to the constraint

$$
\begin{aligned}
& \nabla_v g_0(\boldsymbol{v}, \boldsymbol{u}) + \sum_{i=1}^{m_1} \lambda_i \nabla_v g_i(\boldsymbol{v}, \boldsymbol{u}) = \boldsymbol{0}, \\
& g_i(\boldsymbol{v}, \boldsymbol{u}) \le 0,\ 0 \le \lambda_i,\ \lambda_i g_i(\boldsymbol{v}, \boldsymbol{u}) = 0\ (i = 1, \ldots, m_1),
\end{aligned}
$$

where λ_i denotes the Lagrange multiplier of the ith constraint $g_i(\boldsymbol{v}, \boldsymbol{u}) \le 0\ (i = 1, \ldots, m_1)$. Thus, introducing slack variables $s_i \ge 0\ (i = 1, \ldots, m_1)$ for the quadratic inequality constraints $g_i(\boldsymbol{v}, \boldsymbol{u}) \le 0\ (i = 1, \ldots, m_1)$, we can rewrite the BQOP (2) as a QOP (quadratic

optimization problem)

$$\left.\begin{array}{ll} \underset{\boldsymbol{\lambda},\boldsymbol{s},\boldsymbol{w}}{\max} & \boldsymbol{c}_1^T\boldsymbol{v}+\boldsymbol{c}_2^T\boldsymbol{u} \\ \text{subject to} & \nabla_v g_0(\boldsymbol{v},\boldsymbol{u})+\sum_{i=1}^{m_1}\lambda_i\nabla_v g_i(\boldsymbol{v},\boldsymbol{u})=\boldsymbol{0}, \\ & g_i(\boldsymbol{v},\boldsymbol{u})+s_i=0,\ 0\le s_i\le \bar{s}_i\ (i=1,\ldots,m_1), \\ & \lambda_i s_i=0,\ 0\le\lambda_i\ (i=1,\ldots,m_1), \\ & f_j(\boldsymbol{v},\boldsymbol{u})\le 0\ (j=1,\ldots,m_2),\quad \boldsymbol{w}=\begin{pmatrix}\boldsymbol{v}\\ \boldsymbol{u}\end{pmatrix}\in G_0. \end{array}\right\} \tag{3}$$

Here $\bar{s}_i$ denotes a positive number such that

$$\bar{s}_i \ge \max\{s_i : g_i(\boldsymbol{v},\boldsymbol{u})+s_i=0,\ s_i\ge 0,\ \boldsymbol{w}\in G_0\} \tag{4}$$

$(i=1,2,\ldots,m_1)$. Since G_0 is a compact polyhedral set, such a finite positive number $\bar{s}_i$ always exists. If some inequality constraint $g_i(\boldsymbol{v},\boldsymbol{u})\le 0$ $(i\in\{1,\ldots,m_1\})$ is linear, it is not necessary to introduce the slack variable s_i since the complementarity constraint $\lambda_i g_i(\boldsymbol{v},\boldsymbol{u})=0$ itself is a quadratic equality constraint. We also know from (ii) of Condition 2.1 that the Lagrange multipliers λ_i $(i=1,2,\ldots,m_1)$ involved in the QOP (3) are bounded. So we assume to know a sufficiently large number M which bounds the Lagrange multipliers. This is necessary because the successive SILP relaxation method which we will present in the next section requires an initial compact polyhedral relaxation C_0 of the feasible region of the problem to be solved. We must say, however, that the tightness of upper bounds for the Lagrange multipliers deeply affects the performance of the successive convex relaxation method. In Section 4, we present an additional technique to confine the Lagrange multipliers into a bounded convex set without setting *a priori* explicit bound M for them.

To meet the formulation for the successive SILP relaxation, we now rewrite the QOP (3) as

$$\text{maximize } \boldsymbol{c}^T\boldsymbol{x} \text{ subject to } \boldsymbol{x}\in F, \tag{5}$$

where

$$\begin{array}{rcl} \boldsymbol{x} & = & \begin{pmatrix}\boldsymbol{\lambda}\\ \boldsymbol{s}\\ \boldsymbol{w}\end{pmatrix}\in R^{2m_1+n},\ \boldsymbol{c}=\begin{pmatrix}\boldsymbol{0}\\ \boldsymbol{0}\\ \bar{\boldsymbol{c}}\end{pmatrix}\in R^{2m_1+n},\ \bar{\boldsymbol{c}}=\begin{pmatrix}\boldsymbol{c}_1\\ \boldsymbol{c}_2\end{pmatrix}\in R^n, \\ F & = & \{\boldsymbol{x}\in C_0 : p(\boldsymbol{x})\le 0\ \ (\forall p(\cdot)\in\mathcal{P}_F)\}, \\ \mathcal{P}_F & = & \left\{\begin{array}{l} q_i(\cdot)\ (i=1,\ldots,m_1+n_1+m_1+m_2), \\ -q_j(\cdot)\ (j=1,\ldots,m_1+n_1+m_1)\end{array}\right\}, \end{array}$$

$$\boldsymbol{q}(\boldsymbol{x}) = \begin{pmatrix} \lambda_1 s_1 \\ \vdots \\ \lambda_{m_1} s_{m_1} \\ \nabla_v g_0(\boldsymbol{v},\boldsymbol{u}) + \sum_{i=1}^{m_1} \lambda_i \nabla_v g_i(\boldsymbol{v},\boldsymbol{u}) \\ g_1(\boldsymbol{v},\boldsymbol{u}) + s_1 \\ \vdots \\ g_{m_1}(\boldsymbol{v},\boldsymbol{u}) + s_{m_1} \\ f_1(\boldsymbol{v},\boldsymbol{u}) \\ \vdots \\ f_{m_2}(\boldsymbol{v},\boldsymbol{u}) \end{pmatrix} \quad \text{for } \forall \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{s} \\ \boldsymbol{w} \end{pmatrix} \in R^{2m_1+n},$$

$$C_0 = \left\{ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\lambda} \\ \boldsymbol{s} \\ \boldsymbol{w} \end{pmatrix} \in R^{2m_1+n} : \begin{array}{l} 0 \le \lambda_i \le M \ (i = 1, \ldots, m_1), \\ 0 \le s_i \le \bar{s}_i \ (i = 1, \ldots, m_1), \\ \boldsymbol{w} \in G_0 \end{array} \right\}.$$

Note that $\mathcal{P}_F$ is a set of quadratic functions on R^{2m_1+n}. In particular it includes nonconvex quadratic functions induced from the complementarity constraints.

3 Successive Semi-Infinite Linear Programming Relaxation Method.

The successive convex relaxation method which we will present in this section is based a discretized-localized SILP (semi-infinite linear programming) relaxation variant given in [17]. Numerical results on this variant were also reported in the paper [17]. See [7, 8, 9] for more general variants with the use of the SDP and the SILP relaxations. We are concerned with a general QOP (quadratic optimization problem):

$$\text{maximize } \boldsymbol{c}^T \boldsymbol{x} \text{ subject to } \boldsymbol{x} \in F, \tag{6}$$

where

$$\begin{aligned} \boldsymbol{c} &\in R^n, \\ F &= \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \le 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F)\}, \\ \mathcal{P}_F &: \text{a set of finitely many quadratic functions on } R^n, \\ C_0 &: \text{a nonempty compact polyhedral subset of } R^n. \end{aligned}$$

To describe the successive SILP relaxation method for the QOP (6), we introduce the following notation:

$$\begin{aligned} \mathcal{S}^n &: \text{the set of } n \times n \text{ symmetric matrices}, \\ qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) &\equiv \gamma + 2\boldsymbol{q}^T \boldsymbol{x} + \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}, \ \forall \boldsymbol{x} \in R^n, \end{aligned}$$

$\boldsymbol{Q} \bullet \boldsymbol{X}$ $\equiv$ the (trace) inner product of two symmetric matrices $\boldsymbol{Q}$ and $\boldsymbol{X}$, i.e., $\boldsymbol{Q} \bullet \boldsymbol{X} \equiv \sum_i \sum_j Q_{ij} X_{ij}$,

$\overline{D}$ $=$ $\{\boldsymbol{d} \in R^n : \|\boldsymbol{d}\| = 1\}$ (the set of unit vectors in R^n),

$\boldsymbol{e}_i$: the ith unit coordinate vector.

Successive SILP relaxation method described below generates a sequence of convex relaxations $C_k \subseteq C_0$ $(k = 1, 2, \ldots,)$ of F. At each iteration, we first construct a set $\mathcal{P}_k = \{qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q})\}$ of "rank-2 quadratic functions" each of which induces a valid inequality for the kth iterate C_k:

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

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

$$F = \{\boldsymbol{x} \in C_0 : qf(\boldsymbol{x}; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \leq 0 \quad (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}_F \cup \mathcal{P}_k)\}. \tag{7}$$

We then apply the SILP relaxation (also called the RLT (reformulation-linearization technique) in the literature [14, 15]) to the set F with the representation in (7), which results in the next iterate

$$C_{k+1} = \left\{ \boldsymbol{x} \in C_0 : \begin{array}{c} \exists \boldsymbol{X} \in \mathcal{S}^n \text{such that} \\ \gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \\ (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}_F \cup \mathcal{P}_k) \end{array} \right\}. \tag{8}$$

By definition, C_{k+1} is a convex subset of C_0 since it is the intersection of C_0 with the $\boldsymbol{x}$-space projection of $(\boldsymbol{x}, \boldsymbol{X}) \subset R^{n+n^2}$ represented by the linear inequalities

$$\gamma + 2\boldsymbol{q}^T\boldsymbol{x} + \boldsymbol{Q} \bullet \boldsymbol{X} \leq 0 \quad (\forall qf(\cdot; \gamma, \boldsymbol{q}, \boldsymbol{Q}) \in \mathcal{P}_F \cup \mathcal{P}_k).$$

To see $F \subseteq C_{k+1}$, take $\boldsymbol{X} = \boldsymbol{x}\boldsymbol{x}^T$ for each $\boldsymbol{x} \in F$.

Algorithm 3.1. (Successive SILP relaxation method)

Step 0: Let $D_0 = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, -\boldsymbol{e}_1, \ldots, -\boldsymbol{e}_n\} \subset \overline{D}$. Compute

$$\alpha_0(\boldsymbol{d}) = \max\{\boldsymbol{d}^T\boldsymbol{x} : \boldsymbol{x} \in C_0\} \ (\forall \boldsymbol{d} \in D_0).$$

Let $C_1 = C_0$ and $k = 1$.

Step 1: If $C_k = \emptyset$ then stop. Compute an upper bound ζ_k of the maximum objective value of QOP (6) by $\zeta_k = \max\{\boldsymbol{c}^T\boldsymbol{x} :\in C_k\}$.

Step 2: Choose a set D_k of vectors in $\overline{D}$. Compute

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

Step 3: Let

$$\begin{aligned}\mathcal{P}_k &= \{\boldsymbol{d}^T\boldsymbol{x} - \alpha_0(\boldsymbol{d}) : \boldsymbol{d} \in D_0\} \\ &\quad \bigcup\{-(\boldsymbol{d}^T\boldsymbol{x} - \alpha_0(\boldsymbol{d}))(\boldsymbol{d}'^T\boldsymbol{x} - \alpha_k(\boldsymbol{d}')) : \boldsymbol{d} \in D_0,\ \boldsymbol{d}' \in D_k\}\end{aligned}$$

Step 4: Define C_{k+1} by (8).

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

Algorithm 3.1 generates a sequence of convex relaxations $C_k \subseteq C_0$ $(k = 1, 2, \dots)$ of F and a sequence of real numbers ζ_k $(k = 1, 2, \dots)\}$ satisfying

$$\begin{aligned}&C_0 \supseteq C_k \supseteq C_{k+1} \supseteq F \ (k = 1, 2, \dots), \\ &\zeta_k \geq \zeta_{k+1} \geq \zeta^* \equiv \sup\{\boldsymbol{c}^T\boldsymbol{x} : \boldsymbol{x} \in F\} \ (k = 1, 2, \dots).\end{aligned}$$

If in addition we take $D_k = \overline{D}$ $(k = 1, 2, \dots)$, then C_k and ζ_k converge to the convex hull of F and ζ^*, respectively. See [6, 8, 9] for more details on the convergence of C_k and ζ_k.

To implement Algorithm 3.1, it is necessary to choose a finite set of directions for D_k at each iteration. As in the paper [17], we take a set of $2n + 1$ vectors

$$D(\theta) \equiv \{\boldsymbol{c},\ \boldsymbol{b}_i(\theta),\ \bar{\boldsymbol{b}}_i(\theta) \ \ (i \in \{1, 2, \dots, n\})\} \tag{9}$$

for D_k, where

$$\begin{aligned}\boldsymbol{b}_i(\theta) &\equiv (\boldsymbol{c}\cos\theta + \boldsymbol{e}_i\sin\theta)/\|\boldsymbol{c}\cos\theta + \boldsymbol{e}_i\sin\theta\|, \\ \bar{\boldsymbol{b}}_i(\theta) &\equiv (\boldsymbol{c}\cos\theta - \boldsymbol{e}_i\sin\theta)/\|\boldsymbol{c}\cos\theta - \boldsymbol{e}_i\sin\theta\|,\end{aligned}$$

and $\theta \in (0, \pi/2]$ denotes a parameter. We will explain in Section 5 how we dynamically decrease θ from $\theta_0 = \pi/2$ to a prescribed positive value as the iteration proceeds. When we take $\theta = \pi/2$ during some iterations of Algorithm 3.1, the vectors $\boldsymbol{b}_i(\theta)$ and $\bar{\boldsymbol{b}}_i(\theta)$ in $D(\theta)$ turn out to be the unit vectors $\boldsymbol{e}_i$ and $-\boldsymbol{e}_i$, and the values $\alpha_k(\boldsymbol{e}_i)$ and $\alpha_k(-\boldsymbol{e}_i)$ give upper and lower bounds for the variable x_i, respectively $(i = 1, 2, \dots, n)$. In this case, the set $\{-(\boldsymbol{d}^T\boldsymbol{x} - \alpha_0(\boldsymbol{d}))(\boldsymbol{d}'^T\boldsymbol{x} - \alpha_k(\boldsymbol{d}')) : \boldsymbol{d} \in D_0,\ \boldsymbol{d}' \in D_1 = D(\pi/2)\}$ constructed in Step 3 of Algorithm 3.1 contains all rank-2 quadratic functions induced from the pairwise products of lower and upper bound constraints for variables x_i $(i = 1, 2, \dots, n)$. These constraints correspond to underestimators and overestimators for quadratic terms $x_ix_j,\ i, j \in \{1, 2, \dots, n\}$ which have been often used in many branch-and-bound methods (for instance, [13]). As $\theta \to 0$, we also see that both $\boldsymbol{b}_i(\theta)$ and $\bar{\boldsymbol{b}}_i(\theta)$ approach to the objective direction $\boldsymbol{c}$. See [17] for more details.

4 Additional Techniques for Bilinear Quadratic Optimization Problems.

In this section, we present two types of techniques to enhance the efficiency of Algorithm 3.1; one technique is a reformulation of the QOP (3) into an equivalent scaled problem with explicitly bounds for the Lagrange multipliers, and the other is a technique to tighten bounds for complementary pairs of variables.

4.1 Scaling Lagrange Multipliers.

In order to apply Algorithm 3.1, we need to give an initial compact polyhedral relaxation C_0 of the feasible region F. Among the variable vectors $\boldsymbol{\lambda}$, $\boldsymbol{w}$ and $\boldsymbol{s}$, $\boldsymbol{w}$ is confined into the compact polyhedral set G_0 and $\boldsymbol{s}$ into the compact polyhedral set $\Pi_{i=1}^{m_1}[0, \bar{s}_i]$. Here $\bar{s}_i$ denotes a finite positive number given in (4). We have also assumed to know a positive number M which bounds the Lagrange multipliers λ_i $(i = 1, 2, \ldots, m_1)$ in Section 2. Such a positive number certainly exists in view of (ii) of Condition 2.1. It is usually difficult, however, to estimate (tight) upper bounds for the Lagrange multipliers from the QOP formulation (3) of the BQOP (2).

To avoid such a difficulty, we artificially restrict the range of values for Lagrange multipliers by introducing a scaling variable α into the QOP (3):

$$\left.\begin{array}{lll} \max\limits_{\boldsymbol{\mu},\boldsymbol{s},\boldsymbol{w},\alpha} & \boldsymbol{c}_1^T\boldsymbol{v} + \boldsymbol{c}_2^T\boldsymbol{u} & \\ \text{subject to} & \alpha\nabla_v g_0(\boldsymbol{v},\boldsymbol{u}) + \sum\limits_{i=1}^{m_1}\mu_i\nabla_v g_i(\boldsymbol{v},\boldsymbol{u}) = \boldsymbol{0}, & \\ & g_i(\boldsymbol{v},\boldsymbol{u}) + s_i = 0,\ 0 \le s_i \le \bar{s}_i\ (i = 1,\ldots,m_1), & \\ & \mu_i s_i = 0,\ 0 \le \mu_i \le 1\ (i = 1,\ldots,m_1), & \\ & \alpha + \sum\limits_{i=1}^{m_1}\mu_i = 1,\ 0 \le \alpha \le 1, & \\ & f_i(\boldsymbol{v},\boldsymbol{u}) \le 0\ (i = 1,\ldots,m_2),\ \boldsymbol{w} = \begin{pmatrix} \boldsymbol{v} \\ \boldsymbol{u} \end{pmatrix} \in G_0. & \end{array}\right\} \tag{10}$$

Lemma 4.1. *The QOP (3) is equivalent to the scaled QOP (10) in the sense that* $(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w}) \in R^{2m_1+n}$ *is a feasible solution of the QOP (3) if and only if* $(\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{w}, \alpha) \in R^{2m_1+n+1}$ *is a feasible solution of the scaled QOP (10) with* $\boldsymbol{\mu} = \alpha\boldsymbol{\lambda}$ *and some* $\alpha > 0$.

Proof: Suppose that $(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w}) \in R^{2m_1+n}$ is a feasible solution of the QOP (3). Let

$$\alpha = \frac{1}{1 + \sum\limits_{i=1}^{m_1}\lambda_i} > 0 \quad \text{and} \quad \boldsymbol{\mu} \ = \ \alpha\boldsymbol{\lambda}.$$

The above equalities derive $\alpha\,(1 + \sum_{i=1}^{m_1}\lambda_i) = \alpha + \sum_{i=1}^{m_1}\mu_i = 1$, which corresponds to one of the constraints in (10). Clearly, $\boldsymbol{\mu}$ and α defined above satisfy the other constraints of (10), and hence, $(\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{w}, \alpha) \in R^{2m_1+n+1}$ is a feasible solution of the scaled BQOP (10). Now suppose that $(\boldsymbol{\mu}, \boldsymbol{s}, \boldsymbol{w}, \alpha)$ is a feasible solution of the scaled QOP (10). If $\alpha = 0$, then the constraints $\alpha + \sum_{i=1}^{m_1}\mu_i = 1$ and $\alpha\nabla_v g_0(\boldsymbol{v},\boldsymbol{u}) + \sum_{i=1}^{m_1}\mu_i\nabla_v g_i(\boldsymbol{v},\boldsymbol{u}) = \boldsymbol{0}$ of (10) contradict to (ii) of Condition 2.1. Hence $\alpha > 0$. Let $\boldsymbol{\lambda} = \boldsymbol{\mu}/\alpha$. Then $(\boldsymbol{\lambda}, \boldsymbol{s}, \boldsymbol{w})$ turns out to be a feasible solution of the QOP (3). ∎

We rewrite the scaled QOP (10) as

$$\text{maximize } \boldsymbol{c}^T\boldsymbol{x} \text{ subject to } \boldsymbol{x} \in F, \tag{11}$$

where

$$
\begin{aligned}
\boldsymbol{x} &= \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{s} \\ \boldsymbol{w} \\ \alpha \end{pmatrix} \in R^{2m_1+n+1},\ \boldsymbol{c} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{0} \\ \bar{\boldsymbol{c}} \\ 0 \end{pmatrix} \in R^{2m_1+n+1},\ \bar{\boldsymbol{c}} = \begin{pmatrix} \boldsymbol{c}_1 \\ \boldsymbol{c}_2 \end{pmatrix} \in R^n, \\
F &= \{\boldsymbol{x} \in C_0 : p(\boldsymbol{x}) \le 0 \ \ (\forall p(\cdot) \in \mathcal{P}_F)\}, \\
\mathcal{P}_F &= \left\{ \begin{array}{l} q_i(\cdot)\ (i = 1, \ldots, m_1 + n_1 + m_1 + m_2), \\ -q_j(\cdot)\ (j = 1, \ldots, m_1 + n_1 + m_1) \end{array} \right\}, \\
\boldsymbol{q}(\boldsymbol{x}) &= \begin{pmatrix} \mu_1 s_1 \\ \vdots \\ \mu_{m_1} s_{m_1} \\ \alpha \nabla_v g_0(\boldsymbol{v}, \boldsymbol{u}) + \sum_{i=1}^{m_1} \mu_i \nabla_v g_i(\boldsymbol{v}, \boldsymbol{u}) \\ g_1(\boldsymbol{v}, \boldsymbol{u}) + s_1 \\ \vdots \\ g_{m_1}(\boldsymbol{v}, \boldsymbol{u}) + s_{m_1} \\ f_1(\boldsymbol{v}, \boldsymbol{u}) \\ \vdots \\ f_{m_2}(\boldsymbol{v}, \boldsymbol{u}) \end{pmatrix} \quad \text{for } \forall \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{s} \\ \boldsymbol{w} \\ \alpha \end{pmatrix} \in R^{2m_1+n+1}, \\
C_0 &= \left\{ \boldsymbol{x} = \begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{s} \\ \boldsymbol{w} \\ \alpha \end{pmatrix} \in R^{2m_1+n+1} : \begin{array}{l} 0 \le \mu_i \le 1\ (i = 1, \ldots, m_1), \\ \alpha + \sum_{i=1}^{m_1} \mu_i = 1,\ 0 \le \alpha \le 1, \\ 0 \le s_i \le \bar{s}_i\ (i = 1, \ldots, m_1), \\ \boldsymbol{w} \in G_0 \end{array} \right\}.
\end{aligned}
$$

Now C_0 is a compact polyhedral set, so that we can apply Algorithm 3.1 to the problem (11).

4.2 Tightening Upper Bounds of Complementary Pairs of Variables.

If $\boldsymbol{x}$ is a feasible solution of the QOPs (5) or (11), then it must satisfy the complementarity condition:

$$x_i \ge 0,\ x_{m_1+i} \ge 0 \ \text{ and } x_i x_{m_1+i} = 0, \quad i = 1, 2 \ldots m_1. \tag{12}$$

Exploiting the complementarity condition, we can tighten upper bounds on some of the nonnegative variables x_i $(i = 1, 2, \ldots, 2m_i)$. In fact, if the set $\{\boldsymbol{x} \in C_k : x_{m_1+i} = 0\}$ is empty, we can conclude that $x_{m_1+i} > 0$ and $x_i = 0$ for every $\boldsymbol{x} \in F$. Otherwise

$$x_i \le \max\{\boldsymbol{e}_i^T \boldsymbol{x} : \boldsymbol{x} \in C_k,\ x_{m_1+i} = 0\} \ \text{ for every } \boldsymbol{x} \in F.$$

Therefore, in Step 0 and Step 2 with $D_k = D(\pi/2)$ of Algorithm 3.1, we can replace $\alpha_k(\boldsymbol{e}_i)$ by

$$\alpha'_k(\boldsymbol{e}_i) = \left\{ \begin{array}{ll} 0 & \text{if } \{\boldsymbol{x} \in C_k : x_{m_1+i} = 0\} = \emptyset, \\ \max\{\boldsymbol{e}_i^T \boldsymbol{x} : \boldsymbol{x} \in C_k,\ x_{m_1+i} = 0\} & \text{otherwise,} \end{array} \right.$$

and similarly $\alpha_k(\boldsymbol{e}_{m_1+i})$ by

$$\alpha'_k(\boldsymbol{e}_{m_1+i}) = \begin{cases} 0 & \text{if } \{\boldsymbol{x} \in C_k : x_i = 0\} = \emptyset, \\ \max\{\boldsymbol{e}_{m_1+i}^T \boldsymbol{x} : \boldsymbol{x} \in C_k,\ x_i = 0\} & \text{otherwise.} \end{cases}$$

It should be noted that $\alpha'_k(\boldsymbol{e}_i) \leq \alpha(\boldsymbol{e}_i)$ and $\alpha'_k(\boldsymbol{e}_{m_1+i}) \leq \alpha_k(\boldsymbol{e}_{m_1+i})$ in general. If the strict inequality holds above, then we can strengthen the SILP relaxation in Step 4 of Algorithms 3.1. We call Algorithm 3.1 combined with this technique as the modified Algorithms 3.1.

4.3 An Illustrating Example.

In this section, we present an example to highlight the main idea of this paper. This example of bilevel quadratic program is taken from Shimizu and Aiyoshi [16].

$$\left.\begin{array}{ll} \max\limits_{\boldsymbol{x}} & F(x,y) = -x^2 - (y-10)^2 \\ \text{subject to} & 0 \leq x \leq 15,\ -x + y \leq 0, \\ & \min\limits_{\boldsymbol{y}} \quad f(x,y) = (x + 2y - 30)^2 \\ & \text{subject to} \quad x + y \leq 20,\ 0 \leq y \leq 20. \end{array}\right\} \tag{13}$$

Applying the KKT optimality conditions to the lower level problem and introducing a scaling variable α, we transform the problem (13) into

$$\left.\begin{array}{ll} \max & t \\ \text{subject to} & \\ & \left.\begin{array}{l} x^2 + (y-10)^2 + t = 0, \\ -x + y \leq 0,\ x + y \leq 20 \end{array}\right\} \text{(feasibility)} \\ & \alpha(4x + 8y - 120) + \mu_1 + \mu_2 - \mu_3 = 0 \ \text{(stationarity)}, \\ & \left.\begin{array}{l} \mu_1(20 - x - y) = 0, \\ \mu_2(20 - y) = 0,\ \ \mu_3 y = 0, \end{array}\right\} \text{(complementarity)} \\ & \left.\begin{array}{l} \alpha + \sum\limits_{i=1}^{3} \mu_i = 1,\ \ 0 \leq \alpha \leq 1, \\ 0 \leq \mu_i \leq 1\ (i = 1, \cdots, 3), \\ 0 \leq x \leq 15,\ 0 \leq y \leq 20. \end{array}\right\} \text{(bounds)} \end{array}\right\} \tag{14}$$

In this example (13), the upper level problem has a quadratic objective function, so that we have replaced it by a new single variable t to make a linear objective function.

In the scaled QOP formulation (14), the scaling technique presented in Section 4.1 has derived the quadratic equality constraint

$$\alpha(4x + 8y - 120) + \mu_1 + \mu_2 - \mu_3 = 0,$$

while if we assume a sufficiently large number M as upper bounds for the Lagrange multipliers $\lambda_i, (i = 1, \ldots, 3)$ of the lower level problem of (13), we have the linear equality constraint

$$4x + 8y - 120 + \lambda_1 + \lambda_2 - \lambda_3 = 0.$$

Therefore, the scaling technique has created an additional quadratic equality constraint which may worsen the quality of approximation of the maximum objective value although the technique is important because M chosen is not guaranteed to bound the Lagrange multipliers λ_i $(i = 1, \ldots, 3)$.

Now we apply the technique proposed in Section 4.2 to the complementarity constraints $\mu_1(20 - x - y) = 0, \mu_2(20 - y) = 0$ and $\mu_3 y = 0$. For simplicity of notation, we use a variable itself instead of the corresponding unit coordinate vector $\boldsymbol{e}_i$ below; for example, $\alpha_k(y)$ stands for $\alpha_k(\boldsymbol{e}_i)$ where $\boldsymbol{e}_i$ denotes the unit coordinate vector corresponding to the y axis. When a set $D_k \subset \overline{D}$ of direction vectors at the kth iterate is equal to $D(\pi/2) = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, -\boldsymbol{e}_1, \ldots, -\boldsymbol{e}_n\}$, we compute

$$\begin{aligned} \alpha'_k(y) &= \max\{y : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ \mu_3 = 0\}, \\ \alpha'_k(-y) &= \max\{-y : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ \mu_2 = 0\}, \\ \alpha'_k(\mu_1) &= \max\{\mu_1 : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ x + y = 20\}, \end{aligned}$$

instead of $\alpha_k(y), \alpha_k(-y)$ and $\alpha_k(\mu_1)$. Also, if $\alpha'_k(y) < 20$ or $\alpha'_k(-y) < 0$ holds, we can set $\mu_2 = 0$ or $\mu_3 = 0$ in the succeeding iterations, respectively. Regarding to the upper bounds for μ_2 and μ_3, we also obtain tighter values by

$$\begin{aligned} \alpha'_k(\mu_2) &= \max\{\mu_2 : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ y = 20,\ \mu_3 = 0\}, \\ \alpha'_k(\mu_3) &= \max\{\mu_3 : (x, y, \boldsymbol{\mu}, \alpha, t) \in C_k,\ y = 0,\ \mu_2 = 0\}. \end{aligned}$$

5 Some Numerical Results.

In this section, we describe our implementation of Algorithm 3.1, and report some encouraging preliminary numerical results. The program was coded in C++ language and run on a DEC Alpha Workstation (600 MHz with 1GB of memory). We used CPLEX Version 6.0 as LP solver to compute $\alpha_k(\boldsymbol{d})$ in Steps 0 and Step 2 of Algorithm 3.1.

5.1 Some Implementation Details.

We start Algorithm 3.1 by constructing a set $D_1 = D(\theta)$ with $\theta = \pi/2$ of direction vectors according to the definition (9). If the decrease in the upper bound ζ_k for the maximum objective value becomes slow in some iteration, we reduce the value θ to replace $D(\theta)$. Otherwise, we use the same set of direction vectors as that of the previous iteration. Throughout the computational experiments, we use the following replacement rule:

Let $\ell = 0, \theta_0 = \pi/2, K = 3$ and $\{\sigma_j\}_{j=0}^{K}$ be a decreasing sequence such that $\{1, \frac{8}{9}, \frac{1}{2}, \frac{1}{4}\}$. If the upper bound ζ_k generated at the kth iteration remains to satisfy $\dfrac{\zeta_{k-1} - \zeta_k}{\max\{|\zeta_k|,\ 1.0\}} \geq 0.001 \times \sigma_\ell$, then set $\theta_k = \theta_{k-1}$. Else if $\ell < K$, then set $\ell = \ell + 1$ and $\theta_k = \sigma_\ell \theta_0$, which implies the replacement of $D(\theta_k)$. Otherwise stop the iteration.

For the comparison, we implemented Algorithm 3.1 with another typical algorithm related with the lift-and-project procedure for quadratic programs; the RLT (reformulation-linearization technique) proposed by [14, 15]. The QOP formulations (3) and (10) of BQOPs usually have some linear constraints such as bound constraints in addition to quadratic ones. Following the idea of the RLT, we generate quadratic constraints through the the products of pairwise linear constraints. Together with the original quadratic constraints, those new ones are added as input data for the algorithm. We call the expanded input data as "DataRLT" while the original data as "DataOrg". We compare the following five cases:

SSILP1	:	Algorithm 3.1 with input DataOrg;
SSILP1+RLT	:	Algorithm 3.1 with input DataRLT;
SSILP2	:	the modified Algorithm 3.1 (Section 4.2) with input DataOrg;
SSILP2+RLT	:	the modified Algorithm 3.1 (Section 4.2) with input DataRLT;
RLT	:	the LP relaxation for input DataRLT;

with respect to the following items:

$f_{\rm up}$	:	the solution value found by each algorithm,
R.error	:	the relative error of a solution, *i.e.*, $\dfrac{\lvert f_{\rm up} - f_{\rm opt}\rvert}{\max\{\lvert f_{\rm opt}\rvert, 1.0\}}$, where $f_{\rm opt}$ is the global optimum value;
cpu	:	cpu time in second;
iter.	:	number of iterations (Steps 1- 5) the algorithm repeated.

5.2 Test Problems.

Table 1: The test problems

Problem	Source	n_1	n_2	m_1	m_2	$f_{\rm opt}$
bard1	[4]	1	1	5	2	-17.00
bard2	–	1	1	5	2	68.78
bard3	[4]	2	2	5	4	14.36
shimizu1	[16]	1	1	4	4	-100.00
shimizu2	[16]	2	2	5	4	-225.00
aiyoshi1	[2]	2	2	7	5	-60.00
aiyoshi2	[1]	4	4	14	10	6600.00

We evaluate the five cases listed above using a set of test problems chosen from the literature. Table 1 shows some data of the test problems. Problem "bard1" has linear constraints and upper and lower level convex quadratic objective functions. Thus, the QOP formulation (3) of "bard1" has convex constraints except for complementarity constraints. We create a

nonconvex BQOP, which we call "bard2", from Problem "bard1" by multiplying -1 to the upper level convex objective function.

Tables 2 and 3 present numerical results on the test problems when they are reformulated into the QOP (3) and the scaled QOP (10), respectively. Figures 1 and 2 show how the upper bound ζ_k for the maximum objective value of Problem "shimizu1" decreases as the iteration proceeds. Problem "shimizu1" of Figure 1 takes the QOP formulation (3) and that of Figure 2 the scaled QOP formulation (10). The lines "SSILP1+RLT" and "SSILP2+RLT" in Figure 2 designate the similar performance of achieving the global optimum value of Problem "shimizu1" at the 1st iteration, though the initial upper bounds are different.

Our experiments were conducted in order to see how the following three factors affect the behavior of the algorithms: (i) the scaling technique for Lagrange multipliers (Section 4.1); (ii) the technique for tightening the bound constraints (Section 4.2); (iii) the effect of the RLT.

(i) In the scaled QOP formulation (10), we need no deductive upper bounds for Lagrange multipliers, while in the QOP formulation (3) we assumed that $\lambda_i \leq 1000$, $i = 1, \ldots, m_2$ by taking $M = 1000$ for all the test problems. The scaling technique works effectively in several instances such as Problems "bard1", "bard2" and "shimizu1" in which better upper bounds for the maximum objective values were attained with less computing time. As we have discussed in Section 4.2 and 4.3, however, the scaling technique generates new quadratic equality constraints, which influence the performance of Algorithm 3.1 applied to the scaled QOP formulation (10). In Problems "shimizu2" and "aiyoshi1" of Tables 2 and 3, we observe that the scaled QOP formulation (10) makes the quality of the upper bound worse.

(ii) Comparing the SSILP1 and SSILP2 (or, SSILP1+RLT and SSILP2+RLT) cases in Tables 2 and 3, we see the effect of tightening the bounds for some variables. Especially, Problem "aiyoshi1" in Table 2 shows the fast convergence to a better upper bound for the maximum objective value due to the tight bound constraints, and also, Problem "shimizu2" in Table 3 shows a significant improvement.

(iii) While the sole use of the RLT generates rough upper bounds, the technique enhances the efficiency of Algorithm 3.1 as Tables 2, 3 and Figures 1, 2 show. Although the combined method (SSILP2+RLT) required the greatest computational effort, it achieved the tightest upper bound with less computing time in several instances, due to its fast convergence.

6 Conclusion.

In this paper, we have shown two equivalent reformulations of a BQOP (bilevel quadratic optimization problem) into one-level QOPs (quadratic optimization problem), and presented a successive SILP relaxation method to the transformed one-level QOPs. An exploitation of the special structure of the transformed one-level QOPs accelerates the method and generates tighter upper bounds for maximum objective values. The numerical results on a

Table 2: DataOrg of the QOP (3)

Problem	SSILP1				SSILP1+RLT				RLT		
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu
bard1	-3.07	8.2e-1	4.6	46	-14.92	1.2e-1	4.4	30	-5.00	7.1e-1	0.0
bard2	100.96	4.7e-1	0.9	16	68.86	1.2e-3	0.8	7	75.78	1.0e-1	0.0
bard3	19.99	3.9e-1	2.8	11	14.54	1.2e-2	3.5	14	33.64	1.3e+0	0.0
shimizu1	-95.37	4.6e-2	0.3	14	-96.45	3.6e-2	0.3	11	0.00	1.0e+0	0.0
shimizu2	-224.13	3.9e-3	0.8	11	-225.00	6.1e-15	0.8	6	-125.00	4.4e-1	0.0
aiyoshi1	-55.41	7.7e-2	21.6	53	-59.69	5.2e-3	6.9	11	-41.70	3.1e-1	0.0
aiyoshi2	6786.19	2.8e-2	38.1	8	6625.64	3.9e-3	140.0	7	7200.00	9.1e-2	0.3
Problem	SSILP2				SSILP2+RLT						
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.			
bard1	-3.07	8.2e-1	4.2	46	-14.92	1.2e-1	4.2	30			
bard2	100.91	4.7e-1	0.8	15	68.86	1.2e-3	0.8	7			
bard3	18.90	3.2e-1	5.5	21	14.57	1.5e-2	2.9	12			
shimizu1	-95.75	4.3e-2	0.2	11	-96.74	3.3e-2	0.2	10			
shimizu2	-225.00	5.3e-15	0.4	6	-225.00	3.8e-16	0.7	6			
aiyoshi1	-59.82	3.0e-3	2.3	9	-60.00	1.2e-15	1.4	6			
aiyoshi2	6647.18	7.2e-3	20.3	6	6614.68	2.2e-3	100.2	6			

Table 3: DataOrg of the scaled QOP (10)

Problem	SSILP1				SSILP1+RLT				RLT		
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu
bard1	-14.75	1.3e-1	6.3	40	-16.40	3.6e-2	4.6	26	-3.00	8.2e-1	0.0
bard2	72.01	4.7e-2	1.8	24	68.78	3.0e-9	0.8	8	94.67	3.8e-1	0.0
bard3	14.80	3.1e-2	4.2	16	14.57	1.4e-2	3.9	13	33.64	1.3e+0	0.0
shimizu1	-98.39	1.6e-2	0.4	12	-99.36	6.4e-3	0.2	6	0.00	1.0e+0	0.0
shimizu2	-99.54	5.6e-1	1.9	11	-122.71	4.6e-1	6.6	24	-25.00	8.9e-1	0.0
aiyoshi1	-59.43	9.5e-3	29.6	26	-55.90	6.8e-2	29.8	31	-31.00	4.8e-1	0.0
aiyoshi2	6841.73	3.7e-2	19.5	8	6769.46	2.6e-2	48.1	6	7200.00	9.1e-2	0.1
Problem	SSILP2				SSILP2+RLT						
	$f_{\rm up}$	R.error	cpu	iter.	$f_{\rm up}$	R.error	cpu	iter.			
bard1	-14.79	1.3e-1	6.1	40	-16.42	3.4e-2	4.2	23			
bard2	72.35	5.2e-2	1.7	23	68.78	3.0e-9	0.8	8			
bard3	14.80	3.0e-2	4.0	15	14.56	1.4e-2	3.5	13			
shimizu1	-100.00	1.4e-15	0.2	6	-100.00	2.7e-15	0.3	6			
shimizu2	-168.58	2.5e-1	3.3	30	-210.25	6.6e-2	1.5	10			
aiyoshi1	-59.34	1.1e-2	25.3	26	-56.07	6.6e-2	22.0	32			
aiyoshi2	6841.72	3.7e-2	18.5	8	6768.79	2.6e-2	46.4	6			

Figure 1: Problem "shimizu1 (QOP)"

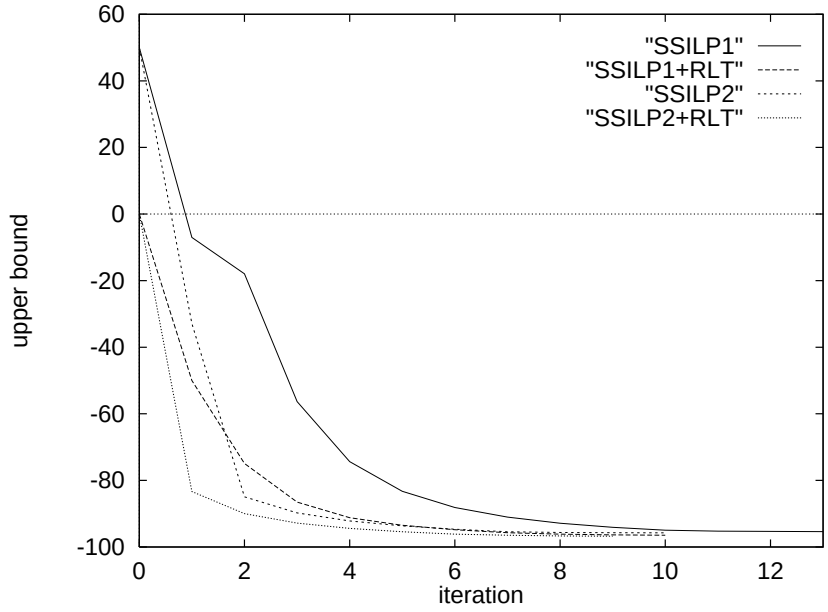
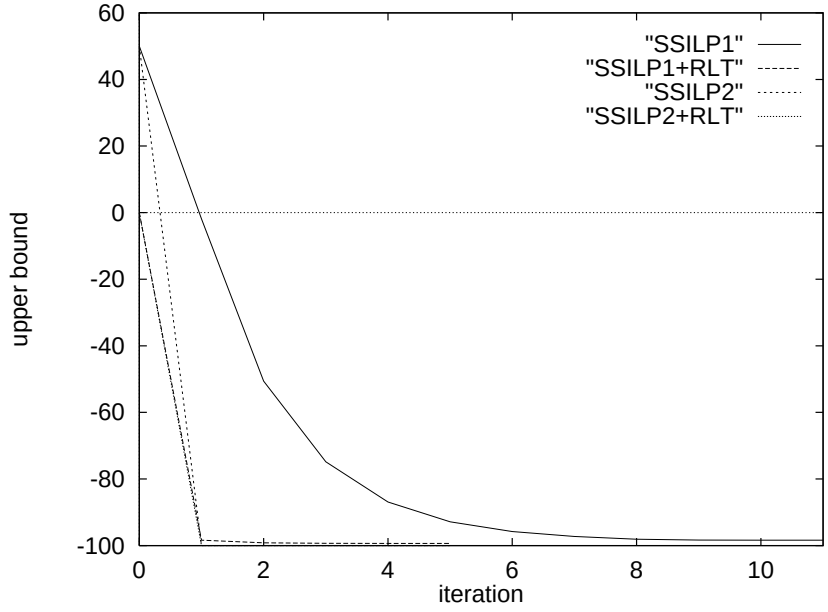


Figure 2: Problem "shimizu1 (Scaled QOP)"



set of test problems have highlighted the effectiveness of the new methods (SSILP2 and SSILP2+RLT), so that the results are satisfactory as a preliminary computational experiments.

Several issues are still left for further study. Our method consumes relatively much computing time in order to achieve tighter upper bounds for maximum objective values. It can be observed from Figures 1 and 2 that minor improvements in upper bounds for the maximum objective value require considerable computing time in later iterations. Hence, if we stopped Algorithms 3.1 within first several iterations, we would easily obtain rough but better upper bounds than existing techniques such as the RLT. Further extension of the research could be to incorporate the successive convex relaxation method into the branch-and-bound method for solving difficult BQOPs including several nonconvex quadratic constraints.

Acknowledgment: The authors are grateful to Levent Tunçel who suggested us to use the technique presented in Section 4.2 for tightening bounds of a complementary pair of variables.

References

[1] E. Aiyoshi and K. Shimizu, "Hierarchical decentralized systems and its new solution by a barrier method," *IEEE Transactions on Systems, Man and Cybernetics* **SMC-11** (1981) 444–449.

[2] E. Aiyoshi and K. Shimizu, "A solution method for the static constrained Stackelberg problem via penalty method," *IEEE Transactions on Automatic Control* **AC-29** (1984) 1111–1114.

[3] E. Balas, S. Ceria and G. Cornuéjols, "A lift-and-project cutting plane algorithm for mixed 0-1 programs," *Mathematical Programming* **58** (1993) 295–323.

[4] J. F. Bard, "Convex two-level optimization," *Mathematical Programming* **40** (1988) 15–27.

[5] P. H. Calamai and L. N. Vicente, "Generating quadratic bilevel programming problems," *ACM Transactions on Mathematical Software* **20** (1994) 103–122.

[6] M. Kojima, T. Matsumoto and M. Shida, "'Moderate Nonconvexity = Convexity + Quadratic Concavity" Technical Report B-348, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, March 1999, Revised April 1999.

[7] M. Kojima and A. Takeda, "Complexity analysis of successive convex relaxation methods for nonconvex sets," Technical Report B-350, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, April 1999. Revised July 1999.

[8] M. Kojima and L. Tunçel, "Cones of matrices and successive convex relaxations of nonconvex sets," Technical Report B-338, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, March 1998. Also issued as CORR 98-6, Dept. of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

[9] M. Kojima and L. Tunçel, "Discretization and localization in successive convex relaxation methods for nonconvex quadratic optimization problems," Technical Report B-341, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, July 1998. Also issued as CORR 98-34, Dept. of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada.

[10] L. Lovász and A. Schrijver, "Cones of matrices and set functions and 0-1 optimization," *SIAM J. on Optimization* **1** (1991) 166–190.

[11] R. Horst and H. Tuy, *Global Optimization, Second, Revised Edition*, (Springer-Verlag, Berlin, 1992).

[12] M. G. Nicholls, "The application of non-linear bilevel programming to the aluminium industry," *Journal of Global Optimization* **8** (1996) 245–261.

[13] N. V. Sahinidis, (1998), "BARON : Branch and reduce optimization navigator, User's manual ver. 3.0", Department of Chemical Engineering, University of Illinois at Urbana-Champaign.

[14] H. D. Sherali and A. Alameddine, (1992), "A new reformulation-linearization technique for bilinear programming problems," *Journal of Global Optimization* **2**, 379–410.

[15] H. D. Sherali and C. H. Tuncbilek, "A reformulation-convexification approach for solving nonconvex quadratic programming problems, " *Journal of Global Optimization* **7** (1995) 1–31.

[16] K. Shimizu and E. Aiyoshi, "A new computational method for Stackelberg and min-max problems by use of a penalty method," *IEEE Transactions on Automatic Control* **AC-26** (1981) 460–466.

[17] A. Takeda, Y. Dai, M. Fukuda, and M. Kojima, "Towards the Implementation of Successive Convex Relaxation Method for Nonconvex Quadratic Optimization Problems," Technical Report B-347, Dept. of Mathematical and Computing Sciences, Tokyo Institute of Technology, Meguro, Tokyo, Japan, March1999.

[18] H. Tuy, *Convex analysis and Global Optimization*, (Kluwer, Dordrecht, 1998).

[19] L. N. Vicente and P. H. Calamai, "Bilevel and multilevel programming : A bibliography review," *Journal of Global Optimization* **5** (1994) 291–306.

[20] V. Visweswaran, C. A. Floudas, M. G. Ierapetritou and E. N. Pistikopoulos, (1996), "A decomposition-based global optimization approach for solving bilevel linear and quadratic programs," in C. A. Floudas eds., *State of the Art in Global Optimization*, Kluwer, Dordrecht, 139-162.