*Journal of Global Optimization* **13:** 417–432, 1998.
© 1998 *Kluwer Academic Publishers. Printed in the Netherlands.*

# A Simplicial Branch-and-Bound Method for Solving Nonconvex All-Quadratic Programs

ULRICH RABER
*Department of Mathematics, University of Trier, Trier, Germany*

(Accepted in final form 30 June 1998)

**Abstract.** In this paper we present an algorithm for solving nonconvex quadratically constrained quadratic programs (all-quadratic programs). The method is based on a simplicial branch-and-bound scheme involving mainly linear programming subproblems. Under the assumption that a feasible point of the all-quadratic program is known, the algorithm guarantees an $\varepsilon$-approximate optimal solution in a finite number of iterations. Computational experiments with an implementation of the procedure are reported on randomly generated test problems. The presented algorithm often outperforms a comparable rectangular branch-and-bound method.

**Key words:** Branch-and-bound, Global optimization, Indefinite quadratic optimization under indefinite quadratic constraints

## 1. Introduction

Quadratically constrained quadratic programs have a wide variety of applications. All bilinear optimization problems, for example pooling problems in petrochemistry [23], modularization of product sub-assemblies [17], and special classes of structured stochastic games [6], can be interpreted as quadratic problems. Boolean variables may also be represented by concave quadratic constraints: $x_i \in \{0, 1\} \Leftrightarrow x_i^2 - x_i \geq 0, x_i \in [0, 1]$.

The so-called *Packing Problem*, i.e., the problem of maximizing the minimum pairwise Euclidean distance of $n$ points, which are contained in the unit square, can be formulated as a global optimization problem with concave quadratic constraints: $\max\{t : t - \|x_i - x_j\|_2^2 \leq 0\,,\ 1 \leq i < j \leq n\,,\ x_i \in [0, 1]^2\,,\ i = 1, \ldots, n\}$. For applications of this special problem we refer to the book of Conway and Sloane [4]. A related class of global optimization problems are minmax location problems [14] which also lead to quadratic constraints.

Chance-constrained problems, which, for example, are involved in production planning or portfolio optimization [5, 14, 26], provide further applications of this type of optimization problem. Other applications include the fuel mixture problem encountered in the oil industry [15], and also placement and layout problems in integrated circuit design (see [2, 3] and references therein for further applications).

In this paper we will consider the general all-quadratic program ($QP$)

$$\begin{aligned}&\min x^T Q^0 x + (d^0)^T x\\ &x^T Q^i x + (d^i)^T x + c^i \leq 0, \quad i = 1, \dots, p\\ &Ax \leq b\end{aligned} \tag{1}$$

where $Q^i$ $(i = 0, \dots, p)$ are real $n \times n$ symmetric matrices, $d^i \in \mathbb{R}^n$ $(i = 0, \dots, p)$, $c^i \in \mathbb{R}$ $(i = 1, \dots, p)$, $A$ is a real $m \times n$ matrix and $b \in \mathbb{R}^m$. The set $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is assumed to be a nonempty polytope, i.e., $P$ is bounded. In the following we assume that for $i \in \{0, \dots, p\}$, real $n \times n$ symmetric matrices $C^i$ and $D^i$ are known with the properties that $C^i$ is a positive semidefinite matrix, $D^i$ is a negative semidefinite matrix and $Q^i = C^i + D^i$. There are different ways to construct such matrices, for example spectral decomposition (see, e.g. [13]). For brevity we define (using $c^0 = 0$)

$$\begin{aligned}&f^i(x) := x^T Q^i x + (d^i)^T x + c^i\ , \quad i = 0, \dots, p\\ &D := \{x \in P : f^i(x) \leq 0\ , i = 1, \dots, p\}\ .\end{aligned}$$

Note that in general the feasible region $D$ is nonconvex and possibly disconnected.

By using the fact that $f^i(x) = x^T C^i x + (d^i)^T x + c^i - x^T(-D)^i x, i = 0, \dots, p$ is a d.c. function, Problem (1) can be interpreted as a d.c. problem. Therefore one possible approach for solving (1) is to apply algorithms for solving general d.c. optimization problems. See Horst et al. [10, 12] and the relevant article in [9] for the framework of d.c. optimization and [15] for a d.c. algorithm for a special quadratically constrained optimization problem.

Another possible approach for solving (1) results from the fact that a nonconvex all-quadratic optimization problem can be transformed to a semidefinite programming problem (SDP) with an additional rank-one constraint (see [18]). Omitting the rank-one constraint leads to the widely explored (SDP) relaxation of Problem (1) (see [7, 21, 22] and references therein). Using this relaxation Ramana [18] presents a cutting plane technique for solving all-quadratic problems (see also [11] for an extension of this approach).

Most of the methods in the literature rely on the bilinearity of the quadratic functions. By substituting $y^i = Q^i x$, each function $f^i(x)$ can be interpreted as a bilinear function $f^i(x, y^i)$. Visweswaran and Floudas [24, 25] propose an algorithm for solving Problem (1) through a series of primal and relaxed dual problems. This method is designed to solve certain classes of nonconvex optimization problems [24], but as shown in [25], it is possible to enhance the computational performance of this algorithm in the quadratic case. The subproblems are considerably more tractable in this special case.

An approach for solving polynomial programming problems, i.e., an optimization problem with a polynomial objective function and polynomial constraints, and so especially for solving bilinear programs, was presented by Sherali and Tuncbilek [20]. Under the assumption that additional box constraints for the variables are known, they generate nonlinear implied constraints which are included in the original problem. Then they linearize the resulting problem by defining new variables, one for each distinct nonlinear term. By embedding this reformulation-linearization technique in a rectangular branch-and-bound scheme they get a convergent algorithm (see [19] for the reformulation-linearization technique in the bilinear case). In other words they combine a linear outer approximation of the feasible set with a branch-and-bound scheme for solving this global optimization problem.

In the algorithm of Al-Khayyal et al. [2, 3] for solving all-quadratic programs with additional box constraints, the authors also combine an outer approximation technique with a rectangular branch-and-bound scheme. First each quadratic function $f^i(x)$ is interpreted as a bilinear function $f^i(x, y^i)$. Each bilinear part $x_j y^i_j$ of $f^i(x, y^i)$ is then bounded from below by its convex envelope [1] and from above by the corresponding concave envelope. By substituting $Q^i x = y^i$ one obtains a linear subproblem with respect to the used rectangle, where this subproblem has $n+(p+1)n$ variables and $4pn+2n+p+m+2n$ constraints. Linear subproblems are then used to calculate the lower bounds in the rectangular branch-and-bound algorithm (for details, we refer to [2]).

In our algorithm, we use the same ideas as Al-Khayyal et al. mentioned above. By using simplices, instead of rectangles, as partitioning elements, we obtain linear subproblems with only $n$ variables and $p + m + n + 1$ constraints. As the computational results show, we thus often have a better performance with respect to run-time than the rectangular algorithm by Al-Khayyal et al. has, particularly in the cases where $p \geq n$.

In the next section, a linear relaxation of Problem (1) is derived with respect to a given $n$-simplex $S = [v_0, \dots, v_n]$, where $[v_0, \dots, v_n] := \{x \in \mathbb{R}^n : x = \sum_{i=0}^n \lambda_i x_i, \lambda_i \geq 0, \sum_{i=0}^n \lambda_i = 1\}$ denotes the convex hull of the set $\{v_0, \dots, v_n\}$. This relaxation is used to calculate a lower bound on the optimal value of Problem (1) which is necessary to develop a branch-and-bound algorithm. A simplicial branch-and-bound algorithm for solving Problem (1) is presented in Section 3. In Section 4, we prove that the algorithm will stop after a finite number of steps, if no feasible point exists. For the case $D \neq \emptyset$ any accumulation point of the sequence of points generated by the algorithm is an optimal solution. This is shown in the convergence theorem. Section 5 reports on results of computational comparism between our simplicial algorithm and the rectangular algorithm of Al-Khayyal et al. [2].

## 2. A linear programming relaxation over a simplex

In this section, we construct a linear programming relaxation of the Problem (1) with the additional constraint that $x$ must lie in an $n$-simplex $S$. Therefore, let $S = [v_0, \dots, v_n]$ be an $n$-simplex with $P \cap S \neq \emptyset$ and let $W_S$ be the real $n \times n$ matrix with columns $(v_i - v_0)$ $(i = 1, \dots, n)$. Then we can rewrite (1) as an all-quadratic program over the standard simplex $B = \{\lambda \in \mathbb{R}^n : e^T\lambda \leq 1, \lambda_i \geq 0$ $(i =$

$1, \dots, n)\}$ by substituting $x = v_0 + W_S\lambda$ in the following way

$$\begin{aligned}&\min\,(W_S\lambda)^T Q^0 W_S\lambda + (d_S^0)^T W_S\lambda + c_S^0\\&(W_S\lambda)^T Q^i W_S\lambda + (d_S^i)^T W_S\lambda + c_S^i \le 0, \quad i = 1, \dots, p\\&AW_S\lambda \le b - Av_0, \quad \lambda \in B\end{aligned} \tag{2}$$

with

$$\begin{aligned}d_S^i &= d^i + 2Q^i v_0, \quad i = 0, \dots, p\\c_S^i &= c^i + v_0^T Q^i v_0 + (d^i)^T v_0, \quad i = 0, \dots, p\ .\end{aligned}$$

Each quadratic function of Problem (2) can be split into a convex and a concave part. If we neglect the convex quadratic part and underestimate the concave part with its convex envelope, we will get a linear underestimator for each quadratic function.

Note that the convex envelope of a concave function $f$ over a simplex $S$ is the uniquely determined affine function which coincides with the function $f$ in each vertex of $S$ (compare with Horst et al. [10]).

So we have for each $\lambda \in B$ and $i \in \{0, \dots, p\}$

$$\begin{aligned}\bar{f}_S^i(\lambda) &:= (W_S\lambda)^T Q^i W_S\lambda + (d_S^i)^T W_S\lambda + c_S^i\\&= \underbrace{(W_S\lambda)^T C^i W_S\lambda}_{\ge 0} + \underbrace{(W_S\lambda)^T D^i W_S\lambda}_{\ge \varphi_S^i(\lambda)} + (d_S^i)^T W_S\lambda + c_S^i\\&\ge \varphi_S^i(\lambda) + (d_S^i)^T W_S\lambda + c_S^i =: \bar{l}_S^i(\lambda)\end{aligned}$$

where

$$\varphi_S^i(\lambda) := \sum_{j=1}^{n} \lambda_j (v_j - v_0)^T D^i (v_j - v_0)$$

is the convex envelope of $(W_S\lambda)^T D^i W_S\lambda$ over $S$.

By using the affine functions $\bar{l}_S^i$ $(i = 0, \dots, p)$, a linear programming relaxation of (2) is

$$\begin{aligned}&\min\ \bar{l}_S^0(\lambda)\\&\bar{l}_S^i(\lambda) \le 0, \quad i = 1, \dots, p\\&AW_S\lambda \le b - Av_0, \quad \lambda \in B\ .\end{aligned} \tag{3}$$

REMARK 1. If we do not omit the convex quadratic part in underestimating $\bar{f}_S^i$, we can get a convex relaxation of (2) in the same way.

Actually, Problem (3) is equivalent to the following problem

$$\begin{aligned}&\min\ l_S^0(x)\\&l_S^i(x) \le 0, \quad i = 1, \dots, p\\&Ax \le b, \quad x \in S\ ,\end{aligned} \tag{4}$$

where $l_S^i(x) := \sum_{j=1}^n (W_S^{-1}(x - v_0))_j (v_j - v_0)^T D^i (v_j - v_0) + (d_S^i)^T (x - v_0) + c_S^i$, $i = 0, \dots, p$, is the convex envelope of the quadratic concave function $f^i(x) - (x - v_0)^T C^i (x - v_0) = (x - v_0)^T D^i (x - v_0) + (d_S^i)^T (x - v_0) + c_S^i$.

REMARK 2. From an implementational point of view, Problem (3) is much easier to solve than Problem (4). While solving (3) instead of (4) we do not need to calculate the inverse of $W_S$, and the constraints descibing $B$ are explicitly given whereas $S$ is given by its vertices. Problem (4), i.e., a formulation of the linear relaxation of (1) in the $x$-space, is needed for the following theoretical analysis.

A necessary result for proving the convergence of our simplicial branch-and-bound algorithm is the following lemma.

LEMMA 1. *Let $\delta^2(S)$ denote the squared diameter of the simplex $S$, i.e., $\delta^2(S) = \max\{\|v_i - v_j\|_2^2, \ i, j \in \{0, \dots, n\}\}$, and $\rho(C^i)$ respectively $\rho(D^i)$ the spectral radius of $C^i$ respectively $D^i$ $(i = 0, \dots, p)$, i.e., $\rho(C^i) = \max\{|\lambda_j(C^i)|, j = 1, \dots, n\}$ ( $\lambda_j(C^i)$ denotes the $j$-th eigenvalue of $C^i$), then for each $i \in \{0, \dots p\}$ it holds*

$$\max_{x \in S} |f^i(x) - l_S^i(x)| \leq \delta^2(S)(\rho(C^i) + \rho(D^i)) . \tag{5}$$

*Proof.* Let $i \in \{0, \dots, p\}$ and $x \in S$ be fixed. Then there exists a uniquely defined $\lambda_x \in B$ with $f^i(x) = \bar{f}_S^i(\lambda_x)$ and $l_S^i(x) = \bar{l}_S^i(\lambda_x)$. By arguments similar to those we used for $(W_S\lambda)^T D^i W_S \lambda$ and $\varphi_S^i(\lambda)$, we know that $\psi_S^i(\lambda) := \sum_{j=1}^n \lambda_j (v_j - v_0)^T C^i (v_j - v_0)$ is a linear overestimator of $(W_S\lambda)^T C^i W_S \lambda$ over $B$. So there holds

$$\begin{aligned}
|f^i(x) - l_S^i(x)| &= \bar{f}_S^i(\lambda_x) - \bar{l}_S^i(\lambda_x) \\
&= \underbrace{(W_S\lambda_x)^T (C^i + D^i) W_S \lambda_x}_{\leq \sum_{j=1}^n \lambda_{xj} (v_j - v_0)^T C^i (v_j - v_0)} - \sum_{j=1}^n \lambda_{xj} (v_j - v_0)^T D^i (v_j - v_0) \\
&\leq \sum_{j=1}^n \lambda_{xj} (v_j - v_0)^T (C^i - D^i)(v_j - v_0) \\
&\leq \sum_{j=1}^n \lambda_{xj} \|v_j - v_0\|_2 \rho(C^i - D^i) \|v_j - v_0\|_2 \\
&\leq \delta^2(S) \rho(C^i - D^i) \underbrace{\sum_{j=1}^n \lambda_{xj}}_{=1} \\
&\leq \delta^2(S)(\rho(C^i) + \rho(D^i)) .
\end{aligned}$$

Note that for symmetric matrices, the spectral radius $\rho$ is a matrix norm which is compatible with the Euclidean vector norm. □

REMARK 3. If we construct the matrices $C^i$ and $D^i$ by spectral decomposition then it is possible to prove that $\rho(C^i - D^i) = \rho(Q^i)$ holds. So we can replace the righthand side of (5) by $\delta^2(S)\rho(Q^i)$.

## 3. A simplicial branch-and-bound algorithm

We now present a branch-and-bound scheme for solving Problem (1) (see Horst and Tuy [12] for the theory and framework of general branch-and-bound algorithms). As partition sets we use simplices and the branching procedure is based on the subdivision of a simplex $S$ into two simplices $S^1$ and $S^2$ by bisection (for the definition of simplex bisection see also [12]). This branching rule in connection with the result of Lemma 1 will ensure the convergence of the algorithm. To each simplex $S$, we assign a lower bound $\mu(S)$ for $f^0$ over $S$ by solving the linear program (4) with respect to $S$. Each generated feasible point $x \in D$ is used to get an upper bound for $f^0$ over $D$. The algorithm is as follows.

ALGORITHM 1

```
**Initialization**
    Determine a simplex $S_0$ with $S_0 \supset P$ .
    $FLP_{S_0} \leftarrow \{x \in S_0 \cap P : l^i_{S_0}(x) \leq 0,\ i = 1, \dots, p\}$
    **If** $FLP_{S_0} \neq \emptyset$ **Then**
        Solve the LP $\min_{x \in FLP} l^0_{S_0}(x)$ .
        Let $x_0$ be an optimal solution and $\mu(S_0)$ be the optimal value.
        $\mu^0 \leftarrow \mu(S_0)$ , $\mathcal{P} \leftarrow \{S_0\}$
        **If** $x_0 \in D$ **Then**
            $Q \leftarrow \{x_0\}$ , $\eta^0 \leftarrow f^0(x_0)$ , $x_f \leftarrow x_0$
        **Else**
            $Q \leftarrow \emptyset$ , $\eta^0 \leftarrow \infty$
        **EndIf**
        STOP $\leftarrow$ **False** , $k \leftarrow 0$
    **Else**
        STOP $\leftarrow$ **True** ($D = \emptyset$)
    **EndIf**

**While** STOP = **False Do**
    **If** $\mu^k = \eta^k$ **Then**
        STOP $\leftarrow$ **True** ($x_f$ is optimal solution of (1) )
    **Else**
        Bisect $S_k$ into two simplices $S^1_k$ and $S^2_k$.
        **For** $j = 1$ **To** 2
            $FLP_{S^j_k} \leftarrow \{x \in S^j_k \cap P : l^i_{S^j_k}(x) \leq 0,\ i = 1, \dots, p\}$
            **If** $FLP_{S_k^j} \neq \emptyset$ **Then**
                Solve the LP $\min_{x \in FLP_j} l^0_{S_k^j}(x)$ .
                Let $x_k^j$ be an optimal solution
                    and $\mu(S_k^j)$ be the optimal value.
                **If** $x_k^j \in D$ **Then**
                    $Q \leftarrow Q \cup \{x_k^j\}$
                **EndIf**
                $\mathcal{P} \leftarrow \mathcal{P} \cup \{S_k^j\}$
            **EndIf**
        **EndFor**
        $\mathcal{P} \leftarrow \mathcal{P} \setminus \{S_k\}$
        **If** $Q \neq \emptyset$ **Then**
            $\eta^{k+1} \leftarrow \min_{x \in Q} f^0(x)$, choose $x_f \in Q$ with $\eta^{k+1} = f^0(x_f)$ .
        **Else**
            $\eta^{k+1} \leftarrow \eta^k$
        **EndIf**
        $\mathcal{P} \leftarrow \mathcal{P} \setminus \{S \in \mathcal{P} : \mu(S) \geq \eta^{k+1}\}$
        **If** $\mathcal{P} \neq \emptyset$ **Then**
            $\mu^{k+1} \leftarrow \min_{S \in \mathcal{P}} \mu(S)$ , choose $S_{k+1} \in \mathcal{P}$ with $\mu^{k+1} = \mu(S_{k+1})$
                and $x_{k+1} \in S_{k+1} \cap P$ with $\mu(S_{k+1}) = l^0_{S_{k+1}}(x_{k+1})$ .
        **Else**
            **If** $Q \neq \emptyset$ **Then**
                $\mu^{k+1} \leftarrow \eta^{k+1}$
            **Else**
                STOP $\leftarrow$ **True** $(D = \emptyset)$
            **EndIf**
        **EndIf**
        $k \leftarrow k + 1$
    **EndIf**
**EndWhile**
```

REMARK 4.

- It is possible that after a finite number of steps the algorithm never detects a feasible point $x_f \in D$, i.e., $Q$ could always be empty.
- The bisection strategy has the property that for each infinite nested sequence $\{S_q\}_{q \in \mathbb{N}}$ of simplices

$$\delta^2(S_q) \to 0 \quad (q \to \infty) \tag{6}$$

(see Horst [8]). As the proof of the convergence theorem shows, this property is sufficient for the convergence of the algorithm. Therefore any branching rule for simplices with the property (6) leads to a convergent algorithm.

– In an implementation of the presented Algorithm 1 we do not solve Problem (4) directly. As said in Remark 2 it is better to use the equivalent Problem (3).

## 4. Convergence

It is obvious that, when finite, the algorithm will determine an optimal solution. For the infinite case, we will state and prove the following convergence theorem. At first, however, one additional lemma is needed to establish this theorem.

LEMMA 2. *The algorithm stops after a finite number of iterations if no feasible point for Problem (1) exists, i.e., if $D = \emptyset$.*

*Proof.* $F(x) := \max_{i=1,\dots,p} f^i(x)$ is a continuous function. So $F$ attains its minimum over the compact set $P$. It follows for $D = \emptyset$

$$\exists \delta > 0 \text{ with } \min_{x \in P} F(x) \geq \delta \,. \tag{7}$$

Assume now that $D = \emptyset$ and the algorithm generates an infinite sequence $\{S_k\}_{k \in \mathbb{N}}$ of simplices. Then there exists an infinite nested subsequence $\{S_{k_q}\}_{q \in \mathbb{N}}$ with the properties

$$FLP_{S_{k_q}} \neq \emptyset \quad \forall q \in \mathbb{N} \quad \text{and}$$

$$\delta^2(S_{k_q}) \to 0 \;\; (q \to \infty) \text{ (compare with (6))}.$$

Choose $0 < \bar{\delta} < \delta[1/\max_{i=1,\dots,p}(\rho(C^i) + \rho(D^i))]$, then there exists a $q_0 \in \mathbb{N}$ such that $\delta^2(S_{k_q}) \leq \bar{\delta}$ $(\forall q \geq q_0)$. So due to Lemma 1, it follows for $x \in FLP_{S_{k_q}}$ $(q \geq q_0)$ and $i \in \{1, \dots, p\}$

$$\begin{aligned} f^i(x) = f^i(x) - l^i_{S_{k_q}}(x) + \underbrace{l^i_{S_{k_q}}(x)}_{\leq 0} \; &\leq \; \delta^2(S_{k_q})(\rho(C^i) + \rho(D^i)) \\ &\leq \; \bar{\delta}(\rho(C^i) + \rho(D^i)) \; < \; \delta \,. \end{aligned}$$

Thus $F(x) < \delta$ holds which contradicts (7). □

The convergence of the algorithm can now be shown.

THEOREM 1. *If the algorithm generates an infinite sequence $\{x_k\}_{k \in \mathbb{N}}$, then every accumulation point $x^\star$ of this sequence is an optimal solution of Problem (1).*

*Proof.* Due to Lemma 2, we know that there exists an optimal solution $\bar{x}$ of Problem (1) with optimal value $\bar{f}^0$. Since we always choose the simplex $S_k$ where $\mu(S_k)$ is smallest, we have that $\mu(S_k)$ is a lower bound for $f^0(x)$ over $D$ and $\{\mu(S_k)\}_{k \in \mathbb{N}}$ is a nondecreasing sequence which is obviously bounded from above by $\bar{f}^0$.

Let $x^\star$ be an accumulation point of the sequence $\{x_k\}_{k \in \mathbb{N}}$ and let $\{x_{k_q}\}_{q \in \mathbb{N}}$ be a subsequence converging to $x^\star$. Without loss of generality, we assume that $\{S_{k_q}\}_{q\in\mathbb{N}}$ is a nested sequence of simplices. With respect to Lemma 1 in connection with the property (6), we know for $i \in \{0, \dots, p\}$

$$0 \le f^i(x_{k_q}) - l^i_{S_{k_q}}(x_{k_q}) \le \delta^2(S_{k_q})(\rho(C^i) + \rho(D^i)) \to 0 \quad (q \to \infty)\,.$$

Since $f^0(x)$ is a continuous function, we have

$$\bar f^0 \ge \mu(S_{k_q}) = l^0_{S_{k_q}}(x_{k_q}) \to f^0(x^\star) \quad (q \to \infty) \tag{8}$$

and by the same arguments for $i \in \{1, \dots, p\}$

$$0 \ge l^i_{S_{k_q}}(x_{k_q}) \to f^i(x^\star) \quad (q \to \infty)\,. \tag{9}$$

Thus we have proved that $x^\star$ is feasible, i.e., $f^i(x^\star) \le 0$ $(i = 1, \dots, p)$ (compare with (9)), and $f^0(x^\star) \le \bar f^0$ (compare with (8)). Therefore, $x^\star$ is an optimal solution of Problem (1). □

If we are satisfied with an approximate solution, we could replace the stopping criterion

```
    If $\eta^k = \mu^k$ Then STOP $\leftarrow$ True
```

by the following

```
If $Q \neq \emptyset$ Then
    If $\eta^k - \mu^k \le \varepsilon$ Then STOP $\leftarrow$ True                                   (10)
Else
    If $\delta^2(S_k) \max_{i=0,\dots,p} (\rho(C^i) + \rho(D^i)) \le \varepsilon$ Then STOP $\leftarrow$ True     (11)
EndIf
```

for some prespecified tolerance $\varepsilon$.

If the algorithm stops with (10), an $\varepsilon$-optimal solution has been found, i.e., a point $x_f \in D$ with $f^0(x_f) - \bar f^0 \le \varepsilon$, where $\bar f^0$ denotes the optimal value of Problem (1).

In the other case, the algorithm has not found a feasible point till iteration $k$. Due to Lemma 1, we only know that the point $x_k$ is $\varepsilon$-feasible, i.e., $f^i(x_k) \le \varepsilon$ , $i = 1, \dots, p$, and that $f^0(x_k) - \mu^k = f^0(x_k) - l^0_{S_k}(x_k) \le \varepsilon$. We do not know anything about the optimality of $x_k$.

Under the assumption that a feasible point $\bar x$ of Problem (1) is known, we can initialize $Q$ with the point $\bar x$. Then the algorithm surely stops after a finite number of steps with a feasible point $x_f \in D$ which is $\varepsilon$-optimal.

In the following section, we will demonstrate the better performance of the presented simplicial branch-and-bound algorithm in comparison with the performance of the rectangular version of Al-Khayyal et al. [2].

## 5. Computational results

The presented simplicial algorithm and the rectangular algorithm of Al-Khayyal et al. were encoded in C++ with management of partition sets by AVL-trees. To test and compare the computational performance of both algorithms, problems of varying sizes were randomly generated and solved. The test problems had the general form (1), where most of the parameters were integers randomly generated according to the following specifications.

First a dense matrix $\bar{A} \in \mathbb{R}^{2n\times n}$ was generated with entries between $-10$ and 10. Then the righthand side vector $\bar{b} \in \mathbb{R}^{2n}$ was constructed in a way which guaranteed that the polyhedron $\bar{P} = \{x \in \mathbb{R}^n : \bar{A}x \leq \bar{b}\}$ was not empty. To ensure the boundedness of $P$, we then intersected $\bar{P}$ with the simplex $S_n = [0, ne_1, \dots, ne_n]$ ($e_i$ denotes the $i$-th unit-vector), so that for the describing matrix $A$ of $P$, $A \in \mathbb{R}^{(3n+1)\times n}$ holds. Note that we iterate the construction of the polytopes $\bar{P}$ until a polytope with $\text{int}P = \text{int}(S_n \cap \bar{P}) \neq \emptyset$ was found. Because of the special construction of $b$, we could find a point $\bar{x} \in \text{int}P = \{x \in \mathbb{R}^n : Ax < b\}$.

In the next step, dense matrices $Q^i \in \mathbb{R}^{n\times n}$ and vectors $d^i \in \mathbb{R}^n$ ($i = 0, \dots, p$) were also randomly generated with entries between $-10$ and 10. The coefficients $c^i \in \mathbb{R}$ ($i = 1, \dots, p$) were chosen in a way to assure that $f^i(\bar{x}) \leq -\delta < 0$ ($i = 1, \dots, p$) holds for some prespecified value $\delta$.

Through this strategy, we obtained test problems with a nonempty feasible region $D$, even with $\text{int}D \neq \emptyset$. For solving these test problems with the two proposed algorithms, we had to construct a starting rectangle $R_0$ with $R_0 \supset P$, and a starting simplex $S_0$ with $S_0 \supset P$, respectively (refer to [10] or [16] for the construction of such initial sets). To avoid excessive run-time requirements we restricted our test problems to problems with the property that a starting simplex with a diameter smaller than 10 exists.

The implementation of both algorithms used to solve the described randomly generated problems closely follows the algorithms which were presented in [2] and earlier in this paper. As branching rule, we used bisection in both cases, i.e., each rectangle or simplex was partioned into two rectangles or two simplices by dividing the longest edge in its midpoint.

In the simplicial branch-and-bound algorithm, we added the following cheap test to decide whether $D \cap S = \emptyset$ holds for a given simplex $S = [v_0, \dots, v_n]$.

$$\max_{i=1,\dots,p} \min_{j=1,\dots,n} (v_j - v_0)^T D^i (v_j - v_0) + (d_S^i)^T (v_j - v_0) + c_S^i > 0 \\ \Rightarrow \quad D \cap S = \emptyset \tag{12}$$

(see [16] for details). If the left-hand side of (12) is satisfied, $S$ can be eliminated from the collection $\mathcal{P}$ of partition sets.

As noted in the previous section, to achieve an approximate solution with the simplicial algorithm, branching may be stopped if (10) or (11) is satisfied. A similar $\varepsilon$-convergence criterion is known for the rectangular algorithm (refer to [2]). We used $\varepsilon = 10^{-4}$ in the computational tests and each point $x_k$ which satisfied all quadratic constraints with a tolerance value $\delta = 10^{-6}$ was interpreted as feasible.

Fifty test problems were generated for each combination of $n \in \{2, \ldots, 8\}$ and $p \in \{1, \ldots, 2n\}$. With respect to the sparse structure of the linear subproblems in the rectangular algorithm (see [2] for details) we applied *MINOS 5.4* to solve them. We also implemented versions of both algorithms using the LP-subroutine *E04NFF* of the *NAG*-library, which does not exploit sparse structures of optimization problems. The subproblems of the simplicial algorithm in general have a dense structure and so it does not matter which LP-solving-routines are applied. Computational tests have shown that the simplicial algorithm is faster with the *NAG*-library than with *MINOS 5.4,* at least for small dimensions ($n \leq 6$). However the rectangular algorithm is much slower. In order to achieve comparable results, we present only the computational tests where *MINOS 5.4* is used for both algorithms.

REMARK 5.
– As noted in section 2, it is also possible to construct a simplicial branch-and-bound scheme with quadratic convex subproblems. We tested this version by using the *MINOS 5.4* convex solver. Even though the necessary number of iterations decreased, the run-time increased so much, that the version with linear subproblems is essentially faster.
– We also tried to solve the Packing-Problem with these general algorithms. Because of the high dimension and the big number of constraints, both algorithms have shown very bad performance. By exploiting the special structure of this problem, it is possible to develop a more efficient algorithm. This will be discussed in a future paper.

Tables 1 and 2 show some numerical results for the generated test problems run on a *SUN SPARCserver 1000* workstation. We use the abbreviations NoP S<R for the number of problems where the simplicial algorithm was faster than the rectangular one, AvgNoLP for the average number of LP's solved for each test problem with the simplicial algorithm (S) or the rectangular algorithm (R), STDLP for the standard deviation of the number of LP's, AvgTime for the average computing time in seconds necessary for solving a problem, Su for the average speedup between the simplicial and the rectangular version and STDTime for the standard deviation of the computing time.

The numerical results show that for $p \geq n$ the simplicial algorithm is nearly always faster with respect to average run-time than is the rectangular algorithm. Only for the combinations with $p \leq \max\{1, \lfloor \frac{n}{2} \rfloor - 1\}$ does the rectangular version need less time in more than 50% of the test examples. For fixed $n$ the relative performance of the presented algorithm improves with growing $p$ (compare with

*Table 1.* All test results for $n = 2, 3, 4$

| p | NoP | AvgNoLP | | STDLP | | AvgTime | | Su | STDTime | |
|---|---|---|---|---|---|---|---|---|---|---|
| | S<R | S | R | S | R | S | R | | S | R |
| $n = 2$ | | | | | | | | | | |
| 1 | 15 | 45.6 | 24.7 | 2.71 | 2.29 | 0.2 | 0.18 | 0.9 | 0 | 0.01 |
| 2 | 32 | 43.6 | 27.9 | 2.82 | 2.16 | 0.22 | 0.26 | 1.19 | 0.01 | 0.01 |
| 3 | 48 | 66.7 | 45.4 | 4.91 | 2.53 | 0.28 | 0.46 | 1.64 | 0.01 | 0.02 |
| 4 | 46 | 61.4 | 39.9 | 4.02 | 1.91 | 0.28 | 0.5 | 1.79 | 0.01 | 0.02 |
| $n = 3$ | | | | | | | | | | |
| 1 | 19 | 127.2 | 54.6 | 12 | 4 | 0.64 | 0.52 | 0.81 | 0.06 | 0.03 |
| 2 | 38 | 151.7 | 69.3 | 23.1 | 6.72 | 0.76 | 0.98 | 1.29 | 0.1 | 0.09 |
| 3 | 46 | 194.6 | 86.2 | 23.5 | 7.61 | 1.02 | 1.63 | 1.6 | 0.09 | 0.14 |
| 4 | 49 | 147.4 | 69.9 | 10.9 | 3.44 | 0.78 | 1.56 | 2 | 0.05 | 0.07 |
| 5 | 46 | 169.4 | 76.7 | 15.4 | 3.18 | 0.98 | 2.19 | 2.23 | 0.11 | 0.1 |
| 6 | 50 | 178.2 | 83.1 | 12.9 | 3.87 | 1.02 | 2.92 | 2.86 | 0.07 | 0.16 |
| $n = 4$ | | | | | | | | | | |
| 1 | 18 | 322.6 | 98.4 | 48.9 | 11.83 | 1.91 | 1.37 | 0.72 | 0.27 | 0.16 |
| 2 | 36 | 341.7 | 101.1 | 47.8 | 10.42 | 2.08 | 2.19 | 1.05 | 0.27 | 0.26 |
| 3 | 40 | 337.6 | 113.7 | 57.1 | 9.4 | 2.2 | 3.44 | 1.56 | 0.35 | 0.33 |
| 4 | 47 | 603.5 | 176.7 | 118.7 | 26.13 | 4.09 | 7.4 | 1.81 | 0.76 | 1.06 |
| 5 | 50 | 356.2 | 125.3 | 37.7 | 7.27 | 2.4 | 6.7 | 2.79 | 0.24 | 0.49 |
| 6 | 48 | 726.3 | 176.4 | 239 | 25.99 | 6.13 | 13.41 | 2.19 | 2.44 | 2.43 |
| 7 | 50 | 428.7 | 155.9 | 44.1 | 12.6 | 3.23 | 12.42 | 3.85 | 0.33 | 1.01 |
| 8 | 49 | 415.8 | 134.5 | 57.2 | 6.75 | 3.25 | 13.22 | 4.07 | 0.42 | 0.82 |

the speedup column in Tables 1 and 2). In the cases where $p = 2n$, it outperforms the rectangular version. It is then up to four times faster.

For solving the test problems, the simplicial algorithm needs many more LP's than the rectangular one and this rate increases with growing dimension. There is at least one reason for this effect. In constructing the LP-relaxation of the all-quadratic program in Section 2, we neglect the convex information of the transformed problem (2), whereas Al-Khayyal et al. do not. They use all available information to generate their lower bounds. Therefore it is not surprising that the lower bounds used in the rectangular version are better than those in our algorithm. But even though the number of LP's increased, the reduction in the complexity of each linear subproblem (smaller dimension and fewer constraints) led to a decrease in run-time.

Figures 1 and 2 illustrate the computational results. In Figure 1 the numbers of test problems where the simplicial algorithm was faster than the rectangular

*Table 2.* Some test results for $n = 5, 6, 7, 8$

| p | NoP | AvgNoLP | | STDLP | | AvgTime | | Su | STDTime | |
|---|---|---|---|---|---|---|---|---|---|---|
| | S<R | S | R | S | R | S | R | | S | R |
| $n = 5$ | | | | | | | | | | |
| 2 | 27 | 962 | 179 | 207.1 | 22.1 | 7.6 | 5.7 | 0.75 | 1.5 | 0.7 |
| 4 | 40 | 1099 | 221 | 210.2 | 30.3 | 9.9 | 14.1 | 1.42 | 1.8 | 2.1 |
| 6 | 48 | 1033 | 237 | 148.9 | 24.5 | 10.4 | 24.9 | 2.39 | 1.5 | 2.3 |
| 8 | 48 | 1327 | 275 | 194.1 | 31.7 | 14.6 | 39.5 | 2.71 | 2.2 | 4.1 |
| 10 | 50 | 850 | 218 | 114.4 | 12.4 | 10.4 | 42.4 | 4.08 | 1.4 | 2.4 |
| $n = 6$ | | | | | | | | | | |
| 4 | 35 | 5407 | 378 | 2114 | 48.1 | 59.3 | 37.6 | 0.63 | 21.3 | 4.3 |
| 8 | 46 | 4314 | 463 | 865.6 | 50.4 | 60.9 | 110.2 | 1.81 | 12.1 | 11.7 |
| 12 | 49 | 3594 | 451 | 561.9 | 36.6 | 60.1 | 189.1 | 3.15 | 9.2 | 16.1 |
| $n = 7$ | | | | | | | | | | |
| 4 | 26 | 12928 | 634 | 5565 | 136.8 | 180 | 86.8 | 0.48 | 77.9 | 17.8 |
| 8 | 39 | 9617 | 751 | 1665 | 85.2 | 170 | 252 | 1.48 | 28.8 | 27.7 |
| 12 | 49 | 8985 | 677 | 1652 | 76.3 | 198 | 417 | 2.11 | 37.2 | 41.9 |
| $n = 8$ | | | | | | | | | | |
| 4 | 26 | 14272 | 686 | 3687 | 56.3 | 262 | 137 | 0.52 | 68.8 | 11.4 |
| 8 | 35 | 33256 | 1326 | 8469 | 180.9 | 764 | 672 | 0.88 | 194.4 | 87.1 |
| 12 | 42 | 20537 | 1171 | 3378 | 112.3 | 594 | 1054 | 1.77 | 91.9 | 91.1 |
| 16 | 47 | 21862 | 1270 | 4016 | 149.8 | 726 | 1797 | 2.48 | 137.9 | 200 |

one are displayed in percent. Figure 2 shows the speedup coefficients for all tested combinations of the dimension $n$ and the number of quadratic constraints $p$. Although both graphics show that the relative performance of the rectangular version improves with growing dimension and small numbers of quadratic constraints, they emphasize that for higher numbers of quadratic constraints the performance of the simplicial approach is much better.

The numerical results also show that the effort for solving Problem (1) depends essentially on the dimension and the number of quadratic constraints. But an interesting result of our numerical tests is that the run-time of the simplicial algorithm is by far less sensitive to the number of quadratic constraints than the run-time of the rectangular one. For example, in the test problems with dimension 8, the average run-time of the simplicial algorithm grows by a factor of nearly 3, whereas the average run-time of the rectangular version grows by a factor of nearly 40.

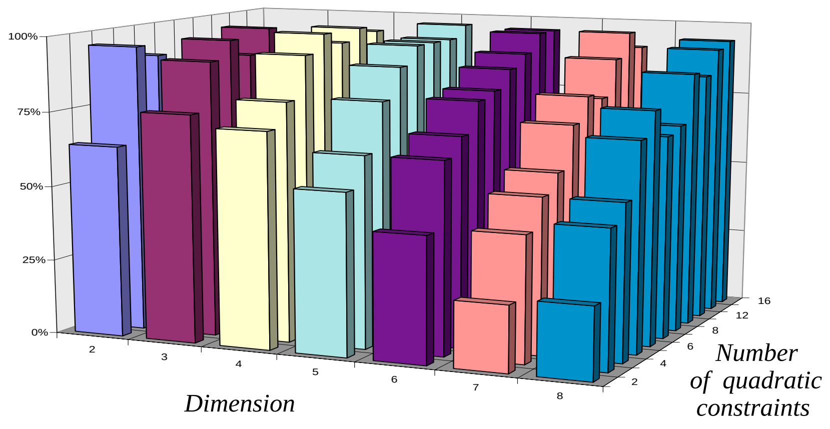

*Figure 1.* Simplicial algorithm faster than rectangular one.

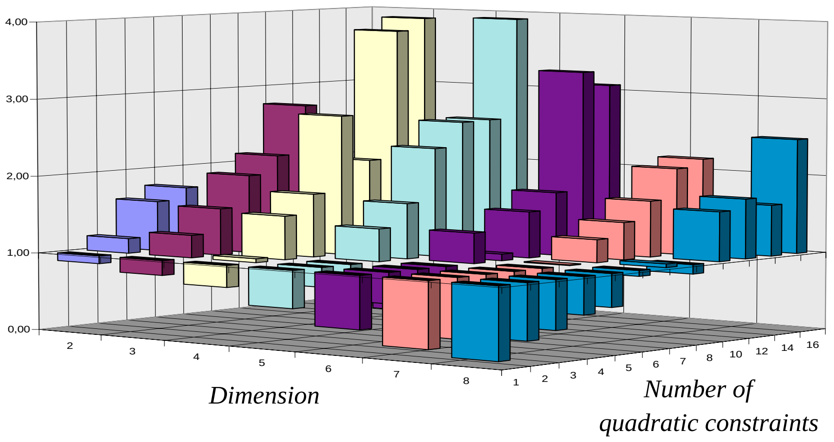

*Figure 2.* Speedup.

Another result of our computational tests is that for problems with dense structure and a dimension higher than 8, both algorithms do not seem to be attractive, because they require excessive run-time.

## 6. Conclusion

In this paper we have developed a convergent algorithm for solving nonconvex all-quadratic optimization problems. This algorithm can be regarded as a combination of an outer approximation (construction of LP-relaxations of Problem (1) with respect to a given $n$-simplex $S$) and a simplicial branch-and-bound scheme. Comprehensive computational results have shown that for small numbers of quadratic constraints (small with respect to the dimension), a comparable rectangular branch-and-bound scheme seems to be a faster approach, whereas for higher numbers of quadratic constraints the presented simplicial algorithm often outperforms this rectangular version. The performance of the suggested algorithm depends heavily on the upper bound (mimimal function value of all found feasible points), as is the case for most branch-and-bound algorithms. One way to improve the performance of this simplicial branch-and-bound algorithm could be the development of cheap strategies for detecting additional feasible points.

**Acknowledgments**

The author is indebted to two anonymous reviewers and the responsible associate editor for valuable comments and suggestions which help to improve the first version of this article.

## References

1. Al-Khayyal, Faiz A. and Falk, J.E. (1983), Jointly constrained biconvex programming, *Annals of Operations Research* 25: 169–180.
2. Al-Khayyal, Faiz A., Larsen, C. and Van Voorhis, T. (1995), A relaxation method for nonconvex quadratically constrained quadratic programs, *Journal of Global Optimization* 6: 215–230.
3. Al-Khayyal, Faiz A. and Van Voorhis, T. (1996), Accelerating convergence of branch-and-bound algorithms for quadratically constrained optimization problems, in C.A. Floudas (ed.), *State of the Art in Global Optimization: Computational Methods and Applications*, Kluwer Academic Publishers, Dordrecht/Boston/London.
4. Conway, J.H. and Sloane, N.J.A. (1993), *Sphere Packings, Lattices and Groups*, 2nd edn, Springer, New York.
5. Demands, E.V. and Tang, C.S. (1992), Linear control of a Markov production system, *Operations Research* 40: 259–278.
6. Filar, J.A. and Schultz, T.A. (1987), Bilinear programming and structured stochastic games, *Journal of Optimization Theory and Applications* 53: 85–104.
7. Fujic, T. and Kojima, M. (1997), Semidefinite programming relaxation for nonconvex quadratic programs, *Journal of Global Optimization* 10: 367–380.
8. Horst, R. (1997), On generalized bisection of $n$-simplices, *Mathematics of Computation* 66(218): 691–698.
9. Horst, R. and Pardalos, P.M. (1995), *Handbook of Global Optimization*, Kluwer Academic Publishers, Dordrecht/Boston/London.
10. Horst, R., Pardalos, P.M. and Thoai, N.V. (1995), *Introduction to Global Optimization*, Kluwer Academic Publishers, Dordrecht/Boston/London.

11. Horst, R. and Raber, U. (1998), Convergent outer approximation algorithms for solving unary programs, *Journal of Global Optimization* 13: 123–149.
12. Horst, R. and Tuy, H. (1996), *Global Optimization: Deterministic Approaches*, 3rd enlarged edn, Springer, Heidelberg.
13. Johnson, L.W., Riess, R.D. and Arnold, J.T. (1993), *Introduction to Linear Algebra*, 3rd edn, Addison-Wesley Publishing Company.
14. Phan-huy-Hao, E. (1982), Quadratically constrained quadratic programming: Some applications and a method for solution, *Zeitschrift für Operations Research* 26: 105–119.
15. Thai Quynh Phing, Pham Dinh Tao and Le Thi Hoai An, (1994), A method for solving d.c. programming problems, Application to fuel mixture nonconvex optimization problems, *Journal of Global Optimization* 6: 87–105.
16. Raber, U. (1996), *Global Optimization with Indefinite Quadratic Constraints*, Diploma Thesis (in German), University of Trier.
17. Rutenberg, D.P. and Shaftel, T.L. (1971), Product design: Sub-assemblies for multiple markets, *Managment Science* 18: B220–B231.
18. Ramana, M. (1993), *An Algorithmic Analysis of Multiquadratic and Semidefinite Programming Problems*, Ph.D. Thesis, The Johns Hopkins University, Baltimore.
19. Sherali, H.D. and Alameddine, A. (1992), A new reformulation-linearization technique for bilinear programming problems, *Journal of Global Optimization* 2: 379–410.
20. Sherali, H.D. and Tuncbilek, C.H. (1992), A global optimization algorithm for polynomial programming problems using a reformulation-linearization technique, *Journal of Global Optimization* 2: 101–112.
21. Shor, N.Z. (1987), Quadratic optimization problems, *Soviet J. Computer and Systems Sciences* 25: 1–11.
22. Shor, N.Z. (1998), *Nondifferentiable Optimization and Polynomial Problems*, Kluwer Academic Publishers, Dordrecht/Boston/London.
23. Visweswaran, V. and Floudas, C.A. (1990), A global optimization algorithm (GOP) for certain classes of nonconvex NLP's: II. Applications of theory and test problems, *Computers and Chemical Engineering* 14: 1417–1434.
24. Visweswaran, V. and Floudas, C.A. (1993), Primal-relaxed dual global optimization approach, *Journal of Optimization Theory and Applications* 78: 187–225.
25. Visweswaran, V. and Floudas, C.A. (1993), New properties and computational improvement of the GOP algorithm for problems with quadratic objective function and constraints, *Journal of Global Optimization* 3: 439–462
26. Weintraub, A. and Vera, J. (1991), A cutting plane approach for chance-constrained linear programs, *Operations Research* 39: 776–785.