Journal of Global Optimization **13:** 225–240, 1998.
© 1998 *Kluwer Academic Publishers. Printed in the Netherlands.*

Cutting Plane/Tabu Search Algorithms for Low Rank Concave Quadratic Programming Problems

HIROSHI KONNO[1], CHENGGANG GAO[1] and ICHIROH SAITOH[2]
[1]*Department of Industrial Engineering and Management, Tokyo Institute of Technology, Japan;*
[2]*Nomura Research Institute Ltd., Tokyo, Japan*

(Received 26 August 1996; accepted 13 November 1997)

Abstract. In this paper, we will propose an efficient heuristic algorithm for solving concave quadratic programming problems whose rank of the objective function is relatively small. This algorithm is a combination of Tuy's cutting plane to eliminate the feasible region and a kind of tabu-search method to find a 'good' vertex. We first generate a set of **V** of vertices and select one of these vertices as a starting point at each step, and apply tabu-search and Tuy's cutting plane algorithm where the list of tabu consists of those vertices eliminated by cutting planes and those newly generated vertices by cutting planes. When all vertices of the set **V** are eliminated, the algorithm is terminated. This algorithm need not converge to a global minimum, but it can work very well when the rank is relatively small (up to seven). The incumbent solutions are in fact globally optimal for all tested problems. We also propose an alternative algorithm by incorporating Rosen's hyperrectangle cut. This algorithm is more efficient than the combination of Tuy's cutting plane and tabu-search.

Key words: Low rank concave quadratic programming problem, Tuy's cutting plane, Rosen's cutting plane, Global optimization, Tabu-search, Heuristic algorithm

1. Introduction

Quadratic programming problems, namely the minimization of quadratic functions under linear constraints, have been under extensive research since the early days of mathematical programming. If the objective function is convex, then there are a number of efficient algorithms, such as several simplex type algorithms, interior point algorithms.

However, if the objective function is nonconvex, then the problem is *NP*-hard. Therefore, it has been considered that solving a large scale nonconvex quadratic programming problem is very difficult. In this paper, we will discuss concave quadratic programming problems which have applications in such areas as quadratic assignment problems, linear max- min problems, location-allocation problems and concave cost production- transportation problems.

Efforts for solving concave quadratic programming problems have a history of over 30 years. For a survey of this field, readers are referred to recent articles (Benson, 1994; Floudas and Visweswaran, 1995; Konno et al., 1996). Since an optimal solution exists among extreme points of a polytope, a number of extreme point enumeration methods have been proposed in early days (Murty, 1968; Cabot and Francis, 1970). However, these naive enumeration methods can be practical only when the size of the problem is very small.

Alternatively, a class of cutting plane algorithms have been proposed since 1970s based upon the pioneering work of Tuy (1964) for minimizing a general concave function on a polytope. For example, one of the authors (Konno and Saitoh, 1996) proposed a variant of Tuy's cutting plane algorithm by exploiting the quadratic property of the objective function. However, it turned out that the proportion of the feasible region eliminated by cutting planes diminishes as the dimension of the problem increases. Therefore, in practice, these cutting plane algorithms can solve problems of limited size.

In 1983, an alternative partitioning algorithm was proposed by Rosen (1983). This algorithm eliminates the largest hypercube inscribing a paraboloid corresponding to the incumbent solution from the feasible region and then applies a branch and bound algorithm to the residual feasible region by using successive underestimation method (Falk and Hoffman, 1976). This algorithm was later extended to a fairly large scale problem with a relatively few (concave) quadratic terms in the objective function. In fact, the computational results reported by Phillips and Rosen (1988) is very encouraging. The remarkable efficiency of this approach, to our understanding, is largely due to the low rank structure of the problem (Konno et al., 1996).

In this paper, we will propose an efficient heuristic algorithm which is a combination of (a) Tuy's cutting plane method to eliminate a portion of the feasible region and (b) a tabu-search method (Glover, 1989, 1990; Glover et al., 1993) to find a 'good' extreme point. Also we will incorporate (c) Rosen's hyperrectangle cut to eliminate the interior of the feasible region to generate even more efficient algorithms.

In Section 2, we will briefly describe Tuy's cutting plane and Rosen's cutting plane and show that these cuts tend to eliminate a larger portion of the feasible region when the rank of the objective function is small. The details of the tabu-search algorithm using these cuts will be presented in Section 3. There is no guarantee that these algorithms generate a globally optimal solution. However, as demonstrated in Section 4, these algorithms perform very well when the rank of the objective function is small.

2. Cutting planes

2.1. TUY'S CUTTING PLANE

Let $f(x)$ be a concave function defined on a polytope X. It is well known that a global minimum of f over X exists among extreme points (vertices) of X.

A cutting plane is a linear constraint which eliminates a locally optimal solution and yet does not eliminate a globally optimal solution. The most important and the

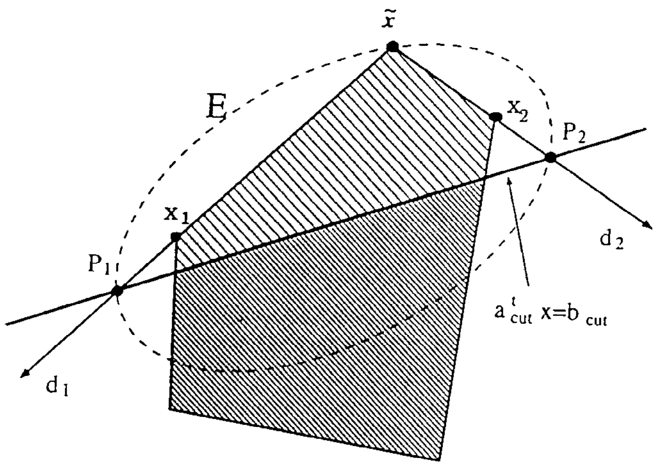

Figure 1. Tuy's cutting plane.

simplest among them is the one proposed by Tuy (1964). Figure 1 illustrates the basic idea of Tuy's cutting plane.

Let $\tilde{x}$ be a given locally optimal extreme point at which the value of the objective function is smaller than those at neighboring vertices. Also, let E be an ellipsoid associated with the contour of $f(\hat{x})$ where $\hat{x}$ is the incumbent solution. Since $f(x)$ is a concave function, we have $f(x) \geqslant f(\hat{x})$ for all $x \in E$. Therefore, we can ignore the region $X \cap E$ in the process of finding a global minimum of f over X. Along the edge d_i emanating from $\tilde{x}$, we look for a point P_i at which the objective function value is equal to the incumbent value $f(\hat{x})$. A linear constraint

$$a^t_{cut} x \leqslant b_{cut} \tag{2.1}$$

determined by $P_i (i = 1, \ldots, n)$ is the so-called Tuy's cutting plane.

Let us argue that Tuy's cut would be substantially deeper when the objective function $f(\cdot)$ is a low rank concave quadratic function. To see this, let us consider a canonical rank p concave quadratic programming problem:

$$\left|\begin{array}{ll} \text{minimize} & f(x) = c_0^T x - \dfrac{1}{2}\displaystyle\sum_{j=1}^{p}(c_j^T x)^2 \\ \text{subject to} & Ax \leqslant b \end{array}\right. \tag{2.2}$$

where $c_1, \ldots, c_p \in R^n$ are linearly independent set of vectors and $c_0 \in R^n$, $b \in R^m$, $A \in R^{m\times n}$ where $m \geqslant n$.

Let us note that a general rank p concave quadratic programming problem

$$\left|\begin{array}{ll} \text{minimize} & q^T y + \dfrac{1}{2} y^T Q y \\ \text{subject to} & A' y \leqslant b' \end{array}\right. \tag{2.3}$$

where Q is a symmetric negative semi-definite matrix whose rank is p can be converged to a canonical form.

Let $\tilde{x}$ be a non-degenerate locally optimal vertex. Associated with $\tilde{x}$ is a set of linearly independent row vectors $a_{i_l}(l = 1, \dots, n)$ of A such that $a_{i_l}\tilde{x} = b_{i_l}(l = 1, \dots, n)$.

Let

$$B = \begin{bmatrix} a_{i_1} \\ \vdots \\ a_{i_n} \end{bmatrix} \in R^{n\times n}, \qquad N = \begin{bmatrix} a_{i_{n+1}} \\ \vdots \\ a_{i_m} \end{bmatrix} \in R^{m-n)\times n} \tag{2.4}$$

and let b_B and b_N be, respectively the subvectors of b corresponding to B and N. By introducing a slack vector y corresponding to B, i.e.,

$$y = Bx - b_B \tag{2.5}$$

the feasible set of (2.2) can be represented in terms of y as follows:

$$Y = \{y \in R^n | \bar{N}y \leqslant \bar{b}_N, y \geqslant 0\} \tag{2.6}$$

where $\bar{N} = NB^{-1}$ and $\bar{b}_N = b_N - B^{-1}b_B$. Accordingly, we can represent the objective function $f(\cdot)$ in terms of y by using the relation (2.5) as follows:

$$\bar{f}(y) = f(\tilde{x}) + \sum_{l=1}^{n} \bar{c}_{0l}y_l - \frac{1}{2}\sum_{j=1}^{p}\left(\sum_{l=1}^{n} \bar{c}_{jl}y_l\right)^2 \tag{2.7}$$

Hence we have an alternative representation of the problem (2.2):

$$\left|\begin{array}{ll} \text{minimize} & \bar{f}(y) = f(\tilde{x}) + \sum_{l=1}^{n} \bar{c}_{0l}y_l - \frac{1}{2}\sum_{j=1}^{p}\left(\sum_{l=1}^{n} \bar{c}_{jl}y_l\right)^2 \\ \text{subject to} & \bar{N}y \leqslant \bar{b}_N,\ y \geqslant 0 \end{array}\right. \tag{2.8}$$

Since $\tilde{x}$ is a local minimum, $y_l = 0(l = 1, \dots, n)$ is a feasible solution of (2.8). Also $\bar{c}_{ol} \geqslant 0(l = 1, \dots, n)$.

Let $\bar{y}_l$ be the nonnegative solution of the quadratic equation

$$f(\tilde{x}) + \bar{c}_{0l}y_l - \frac{1}{2}\sum_{j=1}^{p}\bar{c}_{jl}^2 y_l^2 = f(\hat{x}) \tag{2.9}$$

where $\hat{x}$ is an incumbent solution. Then we have a valid cutting plane (Tuy's cutting plane)

$$\sum_{l=1}^{n} y_l/\bar{y}_l \geqslant 1. \tag{2.10}$$

Let us show that we usually have a deeper cut when p is substantially smaller than n. To see this, let $\bar{y}_l(r)$ be the nonnegative solution as the equation (2.9) when $p = r$. Then

$$y_l(r) = \left[\bar{c}_{0l} + \sqrt{\bar{c}_{0l}^2 + 2a_l(r)(\tilde{f} - \hat{f})}\right] / a_l(r)$$

where $\hat{f} = f(\hat{x})$, $\tilde{f} = f(\tilde{x})$ and $a_l(r) = \sum_{j=1}^{r} \bar{c}_{jl}^2$. Hence

$$\begin{aligned} \alpha_l(r) &\equiv \bar{y}_l(r)/y_l(n) \\ &= \frac{a_l(n)}{a_l(r)} \frac{\bar{c}_{0l} + \sqrt{\bar{c}_{0l}^2 + 2a_l(r)(\tilde{f} - \hat{f})}}{\bar{c}_{0l} + \sqrt{\bar{c}_{0l}^2 + 2a_l(n)(\tilde{f} - \hat{f})}} \\ &\geqslant \frac{a_l(n)}{a_l(r)} \sqrt{\frac{\bar{c}_{0l}^2 + 2a_l(r)(\tilde{f} - \hat{f})}{\bar{c}_{0l}^2 + 2a_l(n)(\tilde{f} - \hat{f})}} \\ &\geqslant \sqrt{\frac{a_l(n)}{a_l(r)}} \geqslant 1 \end{aligned}$$

by noting that $\bar{c}_{ol} \geqslant 0$, $a_l(n) \geqslant a_l(r)$ and $\tilde{f} - \hat{f} \geqslant 0$. As demonstrated by a series of numerical experiments using randomly generated problems, we usually have

$$\frac{a_l(n)}{a_l(r)} \approx \frac{n}{r}$$

as expected when $n \gg r$. This means that we have a substantially deeper cut when the rank p is small compared with n.

2.2. ROSEN'S CUTTING PLANE

Tuy's cutting plane eliminates a portion of the feasible region in the neighborhood of a locally optimal extreme point. Thus, it may be called a “boundary” cut. Rosen's cutting plane (1983), on the other hand, is an “interior” cut, which eliminates a hyperrectangle whose center is located at the global maximum of the objective function.

Let us consider the following rank p concave quadratic programming problem:

$$QP1 : \left| \begin{array}{ll} \text{minimize} & -\sum_{i=1}^{p} x_i^2 \\ \text{subject to} & Ax \leqslant b \end{array} \right. \tag{2.11}$$

where the feasible region is assumed to be compact.

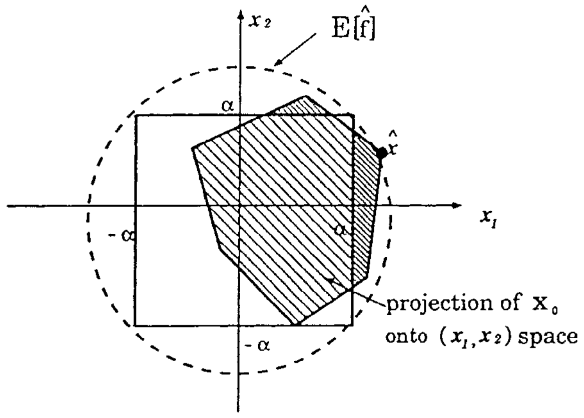

Figure 2. Rosen's cutting plane.

Let $\hat{x}$ and $\hat{f}$ be the incumbent solution and the associated function value, respectively. We consider the set

$$E[\hat{f}] = \{x \in R^n | f(x) \geqslant \hat{f}\} \tag{2.12}$$

which is a parallelepiped whose projection $E_p(\hat{f})$ into $(x_1, x_2, \dots, x_p)$ space is a sphere. Let S be the largest hypercube inscribing $E_p(\hat{f})$ denoted by

$$S = \{x | -\alpha \leqslant x_i \leqslant \alpha, \quad i = 1, 2, \dots, p\} \tag{2.13}$$

In view of the concavity of f, we have

$$\min\{f(x) | x \in X_0 \cap S\} \geqslant \hat{f} \tag{2.14}$$

where $X_0 = \{x \in R^n | Ax \leqslant b\}$. Therefore, we can partition the original problem $QP1$ into the following $2p$ subproblems.

$$QP^+[j] \left| \begin{array}{ll} \text{minimize} & -\sum_{i=1}^{p} x_i^2 \\ \text{subject to} & Ax \leqslant b \\ & x_j \geqslant \alpha, \quad j = 1, \dots, p \end{array} \right. \tag{2.15}$$

$$QP^-[j] \left| \begin{array}{ll} \text{minimize} & -\sum_{i=1}^{p} x_i^2 \\ \text{subject to} & Ax \leqslant b \\ & x_j \leqslant -\alpha, \quad j = 1, \dots, p \end{array} \right. \tag{2.16}$$

Figure 2 describes the essence of Rosen's cutting plane when rank $p = 2$. We see that a significant portion of the feasible region is eliminated.

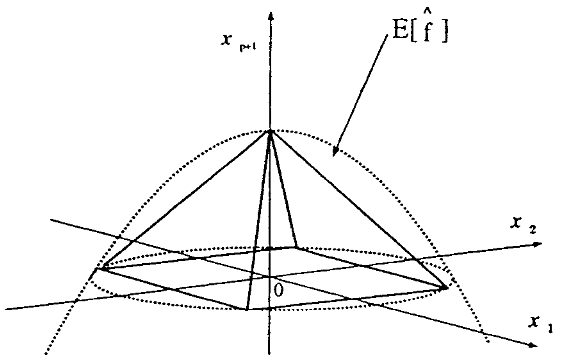

Figure 3. Generalized Rosen's cutting plane.

Rosen's cutting plane can be extended to a problem with a linear term in the objective function:

$$QP2: \left| \begin{array}{ll} \text{minimize} & -\sum_{i=1}^{p} x_i^2 - x_{p+1} \\ \text{subject to} & Ax \leqslant b \end{array} \right. \tag{2.17}$$

We will assume without loss of generality that

$$\min\{x_{p+1} | Ax \leqslant b\} = 0$$

Then $E[\hat{f}]$ is a paraboloid whose bottom surface is located at $x_{p+1} = 0$. As before, we can decompose the original problem $QP2$ into a number of subproblems by eliminating the largest hypertrapezoid. By an easy arithmetic, we find that the largest hypertrapezoid is in fact a htyporcone as depicted in Figure 3.

As a result, the problem $QP2$ is decomposed into $2p$ subproblems generated by adding a linear constraint corresponding to each one of the faces of the cone. The advantage of Rosen's cutting plane method is that a significant portion of the feasible region will be eliminated when the rank p of the problem is small compared to n.

3. Cutting plane/tabu-search algorithms

In this section, we will propose a heuristic algorithm for solving a concave quadratic programming problem:

$$\left| \begin{array}{ll} \text{minimize} & c^t x + \frac{1}{2} x^t Q_x \\ \text{subject to} & Ax \leqslant b \end{array} \right. \tag{3.1}$$

where $c \in R^n$, $Q \in R^{n\times n}$, $A \in R^{m\times n}$, $b \in R^m$. We assume that the rank p of the negative semi-definite matrix Q is small compared to n and that the feasible region

$$X_0 = \{x \in R^n | Ax \leqslant b\} \tag{3.2}$$

is bounded.

The underlying idea of the algorithm is described as follows. We first solve a series of linear programming problems

$$(LP)_k \left| \begin{array}{ll} \text{minimize} & (c^k)^t x \\ \text{subject to} & Ax \leqslant b \end{array} \right. \tag{3.3}$$

where c^k, $(k = 1, 2, \ldots, K)$ are given set of vectors in R^n. Let x^k be an optimal basic solution of $(LP)_k$ and let

$$\mathbf{V} = \{x^1, x^2, \ldots, x^K\} \tag{3.4}$$

We will choose a sequence of vectors c^k's in such a way that $\mathbf{V}$ is a set of points well scattered over X_0.

Let $x^0 \in \mathbf{V}$ be the starting point of a cutting plane/tabu-search method for locating a "good" extreme point solution of the problem (3.1), in which the tabu-list consists of those vertices

(a) which have been eliminated by Tuy's cuts added during the course of computation, and

(b) which are the set of extreme points of X_0 newly generated by Tuy's cuts which do not belong to V_0, the set of extreme points of X_0.

Those vertices satisfying condition (a) cannot be superior to the incumbent solution by the definition of Tuy's cut, while those vertices satisfying the condition (b) can be ignored in the search process because there is at least one optimal solution of (3.1) in V_0.

The general steps of our algorithm are given below.

ALGORITHM TT (Tuy's cutting plane/tabu-search algorithm):

Step 1. *Generate a number of vertices* $x^j \in V_0 (j = 1, \ldots, K)$ *by solving a series of linear programming problems: minimize* $\{(c^k)^t x | A_x \leqslant b\}$, $(k = 1, \ldots, K)$ *where* K *is some positive integer. Save these vertices to the set named* $\mathbf{V}$*;*

Step 2. *Let* $\hat{f} = \min\{f(x^k) | k = 1, \ldots, K\}$, $X_0 = \{x | Ax \leqslant b\}$, *tabu-list* $= \emptyset$, $k = 0$*;*

Step 3. *Choose a vertex* v *of the set* $\mathbf{V}$. *If* v *is tabooed, then* $\mathbf{V} := \mathbf{V} - \{v\}$. *If all vertices of* $\mathbf{V}$ *are tabooed, then go to Step 7;*

Step 4. *Generate a "good" vertex* $\tilde{x}$ *of* X_k *by a local search starting at* v *and traversing through non-tabooed neighboring vertices. If* $f(\tilde{x}) < \tilde{f}$, *then* $\hat{f} = f(\tilde{x})$, $\hat{x} = \tilde{x}$*;*

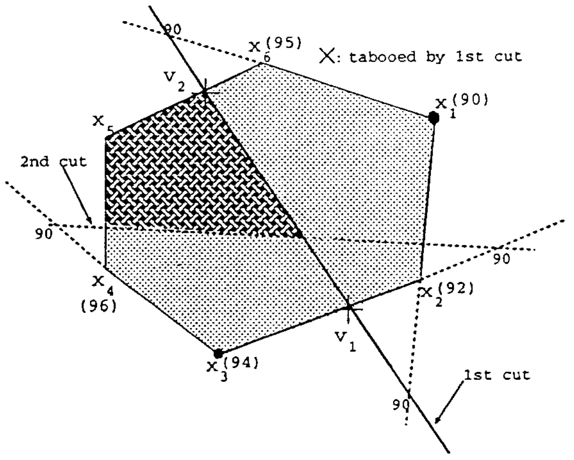

Figure 4. Tuy's cut/tabu-search algorithm.

Step 5. *Generate Tuy's cutting plane* $a_{cut}^t x \leqslant b_{cut}$.

Step 6. $X_{k+1} = X_k \cap \{x | a_{cut}^T x \leqslant b_{cut}\}$, $k = k + 1$. *Update the tabu-list and goto Step 3.*

Step 7. *Terminate the algorithm.*

Let us discuss Step 4 of this algorithm in more detail. In the first step, we apply tabu-search starting from an initial vertex selected from the set **V**, and calculate a locally optimal vertex by traversing the neighboring vertices by applying pivoting operations (Chvatál, 1983). At this vertex, we generate a Tuy's cutting plane. At the k-th step, we must check every vertex whether it is tabooed or not. Among the adjacent vertices which are not tabooed, we move to the vertex whose objective function value is the smallest. If this vertex is the smallest vertex among its non-tabooed neighbors, then we regard this vertex as "good" vertex and generate a Tuy's cut.

Figure 4 shows an example of this scheme in two-dimensional space. We first look for a "real" locally optimal vertex x_1 and generate the 1st Tuy's cutting plane. Then, we select one vertex x_3 from the set **V**. In this tabu-search, the vertex x_2 and new vertex v_1 are tabooed by the 1st cut, so we cannot move from x_3 to x_2. The vertex x_3 is considered as a "good" vertex, and we generate the 2nd cutting plane at this vertex.

As mentioned in Section 2, Tuy's cutting plane eliminates a part of the feasible region near the boundary. On the other hand, Rosen's cutting plane eliminates the interior of the feasible region. Therefore we will be able to generate an even more powerful algorithm by combining these two types of cutting planes. In this algorithm, we first eliminate the interior of the feasible region by Rosen's hyper-rectangle cut and partition the original problem into $2p$ subproblems, and then apply Tuy’s cutting plane/tabu-search method discussed in Section 3.1 to each subproblem.

When solving the problem (3.1), we may add generalized Rosen’s hyperrectangle cuts using the side-faces of the hypercone (Figure 3).

ALGORITHM TRT (Tuy–Rosen cutting plane/tabu-search algorithm):

Step 1. *Generate a number of vertices* $x^j \in V_0 (j = 1, \dots, K)$ *by solving a series of linear programming problems: minimize* $\{(c^k)^t x | Ax \leqslant b\}$, $(k = 1, \dots, K)$ *where* K *is some positive integer. Save these vertices to the set named* **V***;*

Step 2. *Let* $\hat{f} = \min\{f(x^k) | k = 1, \dots, K\}$, $X_0 = \{x | Ax \leqslant b\}$, *tabu-list* $= \emptyset$, $k = 0$*;*

Step 3. *Partition the feasible region* X_0 *into* $2p$ *subregions by Rosen’s hyperrectangle cut;*

Step 4. *Apply Algorithm TT to each subproblem whose feasible region is nonempty.*

4. Results of computational experiments

We tested Algorithm *TT* and Algorithm *TRT* using five different classes of low rank concave quadratic programming problems. These test problems are the ones which have been proved to be difficult for Tuy’s cutting plane algorithm and for Rosen–Tuy cutting plane algorithm without using tabu-search procedure (Konno and Saitoh, 1996).

We wanted to check whether a globally optimal solution has really been obtained by our heuristic algorithm. Therefore we had to limit the experiment to small scale problems because we have to enumerate all extreme points to identify a globally optimal solution

(A) PROBLEM DATA

Class I. m30n10

$$A \in R^{30\times 10}, \quad A_{ij} = \text{rand}[-5, 5], \quad b = e;$$

Class II. n10cube

$$A = \begin{bmatrix} I \\ -I \end{bmatrix} T^{-1} \in R^{20\times 10}, \quad T_{ij} = \text{rand}[-5, 5], \quad b = e;$$

Class III. n20cube

$$A = \begin{bmatrix} I \\ -I \end{bmatrix} T^{-1} \in R^{40\times 20}, \quad T_{ij} = \text{rand}[-5, 5], \quad b = e;$$

Class IV. n30cube

$$A = \begin{bmatrix} I \\ -I \end{bmatrix} T^{-1} \in R^{60\times 30}, \quad T_{ij} = \text{rand}[-5, 5], \quad b = e;$$

Class V. n50cube

$$A = \begin{bmatrix} I \\ -I \end{bmatrix} T^{-1} \in R^{100\times 50}, \quad b = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \quad b_1 = \text{rand}[-5, 5],$$

$$b_2 = b_1 + \text{rand}[2, 12];$$

where rand$[\alpha, \beta]$ means a random number uniformly distributed in the interval $[\alpha, \beta]$.

The linear constraints of n10cube, n20cube and n30cube are generated by distorting a hypercube

$$S = \{x \in R^n | -1 \leqslant x_i \leqslant 1, \quad i = 1, \dots, n\}$$

by applying a random affine transformation in such a way as to contain the origin in its interior as in Konno and Saitoh (1996). Therefore, the feasible region contains exactly 2^n vertices for which we can calculate the globally optimal solution by enumerating all vertices. On the other hand, the feasible region of n50cube are hyper-boxes:

$$X = \{x \in R^n | l_i \leqslant x_i \leqslant u_i, i = 1, \dots, n\}$$

Therefore, we can easily calculate the globally optimal solution.

(B) GENERATION OF THE EXTREME POINT SOLUTIONS

To start the algorithm, we have to generate a number of vertices of X_0, which are well scattered over X_0. How to choose K, the number of vertices and vectors $c^k (k = 1, \dots, K)$ are of significant importance for achieving good performance. One possible strategy would be to generate c^k's randomly. Unfortunately, however there is no definite criteria for these decisions unless we conduct a large scale systematic experiment.

Therefore we employed an admittedly ad-hoc strategy: Generate $2n^2$ vertices by solving the following linear programming problems.

$$LP^+[i] \left| \begin{array}{ll} \text{minimize} & x_i \\ \text{subject to} & Ax \leqslant b \end{array} \right. \qquad LP^-[i] \left| \begin{array}{ll} \text{minimize} & x_i \\ \text{subject to} & Ax \leqslant b \end{array} \right.$$

$$LP^+[ij] \left| \begin{array}{ll} \text{minimize} & x_i + x_j \\ \text{subject to} & Ax \leqslant b \end{array} \right. \qquad LP^-[ij] \left| \begin{array}{ll} \text{minimize} & x_i + x_j \\ \text{subject to} & Ax \leqslant b \end{array} \right.$$

Table 1. Number of problems (of 20) were solved

	QP1		QP2	
rank	Algorithm TT	Algorithm TRT	Algorithm TT	Algorithm TRT
m30n10				
3	(20,0)	(20,0)	(20,0)	(20,0)
4	(20,0)	(20,0)	(20,0)	(20,0)
5	(20,0)	(20,0)	(20,0)	(20,0)
6	(20,0)	(20,0)	(20,0)	(20,0)
7	(20,0)	(20,0)	(20,0)	(20,0)
8	(20,0)	(20,0)	(20,0)	(20,0)
9	(20,0)	(20,0)	(20,0)	(20,0)
n10cube				
3	(20,0)	(20,0)	(20,0)	(20,0)
4	(20,0)	(20,0)	(20,0)	(20,0)
5	(20,0)	(20,0)	(20,0)	(20,0)
6	(20,0)	(20,0)	(20,0)	(20,0)
7	(20,0)	(20,0)	(20,0)	(20,0)
8	(20,0)	(20,0)	(20,0)	(20,0)
9	(19,0)	(20,0)	(19,0)	(20,0)
n20cube				
3	(20,0)	(20,0)	(20,0)	(20,0)
4	(20,0)	(20,0)	(20,0)	(20,0)
5	(20,0)	(20,0)	(20,0)	(20,0)
6	(20,0)	(20,0)	(20,0)	(20,0)
7	(20,0)	(20,0)	(20,0)	(20,0)
8	(18,1)	(20,0)	(17,1)	(20,0)
9	(18,1)	(20,0)	(17,2)	(20,0)
n30cube				
3	(20,0)	(20,0)	(18,0)	(20,0)
4	(14,6)	(20,0)	(13,6)	(20,0)
5	(3,17)	(20,0)	(2,17)	(20,0)
6	(0,20)	(20,0)	(0,20)	(20,0)
7	(0,20)	(20,0)	(0,20)	(20,0)
n50cube				
3	(20,0)	(20,0)	(20,0)	(20,0)
4	(20,0)	(20,0)	(20,0)	(20,0)
5	(20,0)	(20,0)	(20,0)	(20,0)
6	(20,0)	(20,0)	(20,0)	(20,0)
7	(20,0)	(20,0)	(20,0)	(20,0)

Table 2. Average CPU time (sec)

	QP1		QP2	
rank	Algorithm TT	Algorithm TRT	Algorithm TT	Algorithm TRT
m30n10				
3	3.50 (0.559)	3.20 (0.392)	3.54 (0.658)	3.33 (0.537)
4	3.60 (0.626)	3.40 (0.594)	3.52 (0.603)	3.64 (0.836)
5	3.68 (0.673)	3.64 (0.769)	3.82 (0.834)	4.10 (1.369)
6	3.76 (0.615)	3.90 (0.935)	3.76 (0.661)	4.11 (1.082)
7	3.84 (0.733)	4.39 (1.259)	3.84 (0.721)	4.59 (1.595)
8	3.88 (0.736)	4.63 (1.458)	3.92 (0.793)	5.07 (2.031)
9	3.97 (0.822)	4.87 (1.772)	4.03 (0.761)	5.24 (2.228)
n10cube				
3	1.08 (0.109)	1.15 (0.098)	1.09 (0.097)	1.13 (0.095)
4	1.13 (0.098)	1.31 (0.134)	1.16 (0.093)	1.34 (0.154)
5	1.18 (0.099)	1.55 (0.185)	1.16 (0.097)	1.56 (0.203)
6	1.19 (0.109)	1.69 (0.241)	1.19 (0.106)	1.67 (0.216)
7	1.26 (0.126)	1.93 (0.295)	1.25 (0.135)	1.95 (0.307)
8	1.32 (0.117)	2.17 (0.238)	1.26 (0.101)	2.30 (0.280)
9	1.29 (0.115)	2.48 (0.392)	1.30 (0.128)	2.43 (0.365)
n20cube				
3	43.23 (2.826)	41.09 (2.713)	42.88 (2.486)	40.64 (2.332)
4	48.12 (3.542)	46.48 (3.499)	47.69 (3.596)	46.85 (3.305)
5	54.51 (6.023)	55.47 (4.536)	52.60 (4.536)	54.50 (3.573)
6	59.90 (5.960)	68.39 (6.415)	58.96 (5.558)	66.49 (5.348)
7	67.52 (8.373)	86.22 (9.162)	66.03 (7.459)	84.61 (9.717)
8	74.62 (10.37)	105.86 (12.88)	71.22 (8.780)	103.13 (13.12)
9	75.19 (10.61)	121.69 (17.92)	74.10 (10.59)	118.43 (17.42)
n30cube				
3	420.22 (27.06)	361.81 (14.14)	423.81 (32.31)	360.05 (11.94)
4	536.06 (43.57)	430.47 (23.18)	533.56 (44.81)	431.97 (22.97)
5	605.59 (42.82)	550.48 (55.17)	612.90 (44.36)	543.17 (50.24)
6	654.07 (48.14)	685.34 (93.59)	651.37 (47.30)	698.06 (90.20)
7	695.78 (41.82)	896.23 (123.5)	686.69 (43.49)	893.95 (119.4)
n50cube				
3	90.74 (14.55)	54.78 (11.88)	97.25 (14.39)	59.22 (19.91)
4	106.75 (20.61)	60.90 (20.75)	109.95 (23.19)	61.27 (17.52)
5	127.46 (28.05)	65.60 (13.84)	131.29 (33.63)	68.24 (13.29)
6	151.64 (38.37)	73.67 (21.61)	155.37 (42.67)	74.03 (20.90)
7	161.63 (34.04)	80.41 (20.84)	162.56 (40.32)	81.77 (19.95)

Table 3. Average number of cuts added

	QP1		QP2	
rank	Algorithm TT	Algorithm TRT	Algorithm TT	Algorithm TRT
m30n10				
3	16.7 (9.85)	8.3 (5.69)	14.4 (7.95)	9.3 (6.52)
4	18.7 (9.42)	12.9 (8.44)	17.2 (9.23)	15.2 (11.15)
5	21.2 (10.95)	18.7 (13.15)	21.7 (11.58)	23.6 (18.78)
6	23.0 (10.77)	23.6 (15.65)	21.5 (10.63)	26.3 (17.33)
7	25.2 (13.52)	32.5 (21.67)	23.2 (11.77)	33.2 (22.49)
n10cube				
3	10.8 (3.97)	10.8 (4.08)	10.9 (4.14)	10.9 (4.28)
4	13.8 (3.88)	18.9 (5.21)	14.2 (3.69)	18.8 (5.12)
5	15.7 (4.05)	28.4 (8.33)	15.6 (4.25)	28.6 (8.30)
6	17.8 (4.53)	36.9 (10.9)	17.8 (4.46)	36.3 (10.2)
7	19.0 (4.61)	47.4 (12.2)	19.3 (4.84)	47.2 (12.0)
n20cube				
3	40.0 (10.62)	22.9 (11.88)	38.5 (8.92)	22.6 (7.37)
4	60.9 (13.19)	44.1 (20.75)	59.2 (12.29)	44.2 (11.3)
5	83.2 (18.24)	75.6 (13.84)	78.8 (15.90)	73.8 (14.7)
6	103.1 (17.01)	114.2 (21.61)	100.0 (17.23)	112.7 (13.4)
7	128.2 (23.34)	169.3 (20.84)	123.0 (20.55)	164.2 (28.6)
n30cube				
3	94.05 (22.75)	37.4 (11.2)	94.75 (23.96)	35.2 (8.86)
4	174.65 (22.89)	88.9 (14.1)	173.98 (23.51)	89.3 (13.8)
5	197.1 (9.20)	159.3 (31.4)	197.8 (9.37)	158.6 (30.7)
6	$\geqslant$ 200 (—)	241.1 (44.1)	$\geqslant$ 200 (—)	238.6 (41.1)
7	$\geqslant$ 200 (—)	327.7 (66.5)	$\geqslant$ 200 (—)	338.5 (54.1)
n50cube				
3	90.7 (14.55)	48.6 (20.3)	97.3 (14.39)	50.9 (23.4)
4	106.8 (20.61)	55.4 (23.7)	106.8 (20.61)	55.5 (21.9)
5	127.4 (28.05)	69.8 (17.0)	131.3 (33.63)	73.8 (16.6)
6	151.6 (38.37)	84.2 (27.3)	155.4 (42.67)	83.7 (27.2)
7	161.0 (34.04)	99.7 (29.4)	162.6 (40.32)	98.7 (28.1)

where $i, j = 1, 2, \ldots, n, i \neq j$.

(C) COMPUTATIONAL RESULTS

The first number in the bracket of Table 1 shows the number of problems out of 20 tested problems successfully solved by our algorithms. The second number in the bracket shows the number of problems in which all vertices of set **V** could not be eliminated after adding at most 200 cuts to each (sub)problem.

We see from these data that Algorithm TRT could successfully solve all tested problems. This is a remarkable improvement over Algorithm TT which in turn performs much better than the cutting plane method proposed by Konno and Saitoh (1996).

These results show that Algorithm TRT can serve as a very good heuristic approach for solving a large scale concave quadratic programming problem whose rank is relatively small.

Table 2 shows the average CPU time and its standard deviation for solving twenty test problems. Programs are coded in C language and run at MicroSPARC/85MHz.

We can see that for m30n10, n10cube, n20cube and n30cube, the average CPU time by Algorithm TRT is about the same as by Algorithm TT. When the rank p is greater than 4, then Algorithm TT becomes a little faster, but the average CPU time by Algorithm TRT is about one half of Algorithm TT for n50cube. This is due to the fact that two thirds of subproblems generated by Rosen's hyperrectangle cuts become infeasible.

Also, we observe from this that the computation time is an increasing function of the rank p. This is primarily due to the fact that Tuy's cut tends to become shallower as the rank p increases (Konno and Saitoh, 1996).

Table 3 shows the average number and standard deviation of Tuy's cuts adds before all vertices of **V** were eliminated.

5. Conclusions and the future direction of research

In this paper we proposed a heuristic algorithm for solving low rank concave quadratic programming problems, whose components are (i) Tuy's concavity cuts, (ii) Rosen's hyperrectangle cut and (iii) tabu-search. We demonstrated that this algorithm performs very well for small-to- medium scale problems.

The first reason of this excellent performance is that bothy Tuy's cut and Rosen's cut tend to eliminate a larger portion of the feasible region when the rank of the problem is small as discussed in Section 2.

The second reason is that we improved the effectiveness of local search by ignoring those vertices which cannot be a candidate of a globally optimal solution. In particular, we prevented the common phenomena of the cutting plane algorithm that it tends to get trapped in a small local region, by ignoring the newly generated vertices.

We believe that this approach would also work well for a class of low rank (not necessarily quadratic) concave programming problems. We are now applying this class of algorithms to the minimization of a low rank cubic concave function over a polytope resulting from a class of financial optimization, whose result will be reported subsequently.

References

Balas, E. and Burdet, C.A. (1979), Maximizing a convex quadratic function subject to linear constraints, *Management Science Research Report No. 299*. Carnegie-Mellon-University, Pittsburgh

Benson, H.P. (1994), Concave minimization: theory, applications and algorithms, in: R. Horst and P.M. Pardalos (eds.), *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht/Boston/London.

Cabot, A.V. and Francis, R.L. (1970), Solving nonconvex quadratic minimization problems by ranking extreme points, *Operations Research* 18: 82–86.

Chvatál, V. (1983), *Linear Programming*. Freeman and Co.

Falk, J.E. and Hoffman, K.L. (1976), A successive underestimating method for concave minimization problems, *Mathematics of Operations Research* 1: 251–259.

Floudas, C.A. and Visweswaran, V. (1995), Quadratic optimization, in *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht/Boston/London.

Glover, F. (1989), Tabu search 1, *ORSA J. Computing* 1: 190–206.

Glover, F. (1990), Tabu search 2, *ORSA J. Computing* 2: 4–32.

Glover, F., Taillard, E. and De Werra, D. (1993), A user's guide to search, *Annals of Operations Research* 41: 3–28.

Horst, R. and Pardalos, P.M. (1994), *Handbook of Global Optimization*. Kluwer Academic Publishers, Dordrecht/Boston/London.

Horst, R. and Tuy, H. (1993), *Global Optimization: Deterministic Approaches*, 2nd edn. Springer Verlag, Berlin/New York.

Konno, H., Thach, T.H. and Tuy, H. (1996), *Optimization on Low Rank Nonconvex Structures*. Kluwer Academic Publishers, Dordrecht/Boston/London.

Konno, H. and Saitoh, I. (1996), Cutting plane algorithm for solving low rank concave quadratic programming problems. Technical Report 96-6, Department of IE and Management, Tokyo Institute of Technology.

Konno, H. (1976), Maximization of a convex quadratic function under linear constraints, *Mathematical Programming* 11: 117–127.

Luenbeger, D. (1984), *Introduction to Linear and Nonlinear Programming*, 3rd edn. Addison-Wesley, Reading, MA.

Murty, K.G. (1968), Solving the fixed-charge problem by ranking extreme points, *Operations Research* 16: 268–279.

Pardalos, P.M. and Vavasis, S.A. (1991), Quadratic programming with one negative eigenvalue is NP-hard, *J. Global Optimization* 1: 15–22.

Phillips, A.T. and Rosen, J.B. (1988), A parallel algorithm for constrained concave quadratic global minimization, *Mathematical Programming* 42: 421–448.

Rockafellar, R.T. (1970), *Convex Analysis*. Princeton University Press, Princeton, NJ.

Rosen, J.B. (1983), Global minimization of a linearly constrained concave function by partition of feasible domain, *Mathematics of Operations Research* 8: 215–230.

Tuy, H. (1964), Concave programming under linear constraints, *Soviet Mathematics* 5: 1437–1440.