Available online at www.tjnsa.com
J. Nonlinear Sci. Appl. 9 (2016), 1153–1164
Research Article

A new branch and bound algorithm for integer quadratic programming problems

Xiaohua Ma, Yuelin Gao*, Xia Liu

Beifang University for Nationalities, Institute of Information and System Sciences, Yinchuan, 750021, China.

Communicated by Y. Yao

Abstract

For integer quadratic programming problems, a new branch and bound algorithm is proposed in this paper through a series of improvements on the traditional branch and bound algorithm, which can be used to solve integer quadratic programming problems effectively and efficiently. This algorithm employs a new linear relaxation and bound method and a rectangular deep bisection method. At the same time, a rectangular reduction strategy is used to improve the approximation degree and speed up the convergence of the algorithm. Numerical results show that the proposed algorithm is feasible and effective and has improved the existing relevant branch and bound algorithms.

Keywords: Integer quadratic programming, branch and bound, linear relaxation, rectangular deep bisection, rectangular reduction.
2010 MSC: 47H10, 54H25.

1. Introduction

Integer programming problems are optimization problems that minimize or maximize the objective function in the limitation of equality or inequality constraints and integer variables. More widely application, integer programming can be used to properly describe the decision problems on the management and effective use of resources in engineering technology, social science, finance, business administration and many other fields. With the development of science and technology and the urgent need for solving complex decision problems in the actual, the algorithm research on nonlinear integer programming problems has become one of the hot research topics in the field of operations research and optimization. Integer programming problems

*Corresponding author
Email addresses: mxh6464@163.com (Xiaohua Ma), gaoyuelin@263.net (Yuelin Gao), liuxia_929@163.com (Xia Liu)

Received 2015-09-04

that we usually discuss are linear and it is known that the general linear integer programming problem is NP- hard. From the mathematical programming section in appendix A6 of literature [13], we know that the problem is NP- hard if constraints are quadratic and the objective function is linear and it is undecidable if add an integer requirement to the last problem then the nonlinear integer programming problem is even more difficult. The deterministic methods for solving nonlinear integer programming problems are outer approximation methods [1, 2, 4, 6], cutting plane methods [5, 12, 16], decomposition algorithms [7, 15] and branch and bound methods [3, 8, 9, 10, 11, 14], in which branch and bound methods are more practical. However, as the integer programming problem is NP-hard problem, existing algorithms can only solve a particular form of integer programming problems and there are often many shortcomings, such as a slow convergence rate, large calculation quantity and poor efficiency. In view of this, as a preparation for seeking a common and effective algorithm of solving general nonlinear integer programming problems, this paper investigates the following special nonlinear integer programming problems − integer quadratic programming problems:

$$(IQP)\begin{cases} min & f(x)=\frac{1}{2}x^TQx+c^Tx \\ s.t. & x\in D=\{x\in R^n|Ax\le b, x\ge 0\}, \\ & x\in Z^n, \end{cases}$$

where $Q\in R^{n\times n}$ is a n-dimension symmetric matrix, $c\in R^n$, $A\in R^{m\times n}$, $b\in R^m$. Z^n is a set of integer vectors in R^n, and $D=\{x\in R^n|Ax\le b, x\ge 0\}$ is a non-empty and bounded set.

Since the optimal solution of the integer quadratic programming problem may not be able to meet KKT conditions, it is often more difficult to solve than the continuous optimization problem. Through a series of improvements on the traditional branch and bound method, this paper presents an algorithm which can be used to solve integer quadratic programming problems quickly and effectively.

This algorithm adopts a new linear relaxation and bound method, a rectangular deep bisection method and the rectangular reduction method in the literature [9]. The feasibility, effectiveness and superiority of the proposed new algorithm are explained by a lot of numerical experiments and the comparison with existing literatures. This paper is organized as follows. In section 2, the relaxation and bound method is given. In section 3, The subdivision and reduction of the rectangle are provided. The new branch and bound algorithm is described in detail in section 4, and the convergence of the algorithm is analyzed. Finally, a lot of numerical experiment results and the comparison with existing literatures turn out that the proposed new algorithm is feasible, effective and superior.

2. Relaxation and bound method

In this section, we construct the integer rectangle R first, which contains the feasible region of the original problem (IQP). For this purpose, calculating the following $2n$ linear programming problems first:

$$\begin{cases} min & h(x_j)=x_j, \\ s.t. & Ax\le b, \\ & x\ge o, \\ & x\in R^n. \end{cases} \quad \text{J}=1,2,...,n$$

$$\begin{cases} min & l(x_j)=x_j, \\ s.t. & Ax\le b, \\ & x\ge o, \\ & x\in R^n. \end{cases} \quad \text{J}=1,2,...,n$$

We obtain optimal solutions $h_j^*, j=1,2,...,n$ from (1) and optimal solutions $l_j^*, j=1,2,...,n$ from (2). Let

$$\underline{x}_j=\lceil h_j^*\rceil, j=1,2,...,n,$$

$$\overline{x}_j = \lfloor l_j^* \rfloor, j = 1, 2, ..., n.$$

Then we get the integer rectangle which contains the feasible region $F = D \cap Z^n$

$$R = [\underline{x}, \overline{x}] = \{x | \underline{x} \leq x \leq \overline{x}, \underline{x}, \overline{x} \in Z^n\},$$

where $\underline{x} = (\underline{x}_1, \underline{x}_2, ..., \underline{x}_n)^T$, $\overline{x} = (\overline{x}_1, \overline{x}_2, ..., \overline{x}_n)^T$ are integer vectors. So the original problem (IQP) is equivalent to the following problem (IQP)':

$$(IQP)' \begin{cases} min & f(x) = \frac{1}{2}x^T Qx + c^T x \\ s.t. & x \in D \cap R \cap Z^n. \end{cases}$$

Consider the following continuous relaxation programming problem (QP) of the problem (IQP)' :

$$(QP) \begin{cases} min & f(x) = \frac{1}{2}x^T Qx + c^T x \\ s.t. & x \in D \cap R. \end{cases}$$

The optimal value of the problem (QP) is a lower bound of the problem (IQP)' obviously. Therefore, it is an effective lower bound of the optimal value of the original problem (IQP).

Consider the lower bound estimation of the optimal value of the problem (QP) first. Introducing the variable $d = (d_1, d_2, ..., d_n)^T \in Z^n$, and it satisfies

$$d_j = min\{Q_j^T x | Ax \leq b, x \geq 0, \underline{x} \leq x \leq \overline{x}, x \in R^n\}, j = 1, 2, ..., n,$$

where Q_j represents the j-th row of the matrix Q. Because $x \geq 0$, so there is

$$f(x) = \frac{1}{2}x^T Qx + c^T x \geq (c + \frac{1}{2}d)^T x, \forall x \in D \cap R.$$

Then we get the following linear and relaxation programming problem of the problem (QP) on the rectangle R_k :

$$(LP(R_k)) \begin{cases} min & f(x) = (c + \frac{1}{2}d)^T x \\ s.t. & x \in D \cap R_k. \end{cases}$$

Solving the problem $(LP(R_k))$, we obtain the optimal value. It is a lower bound of the global optimal value $\nu(QP)$ of the problem (QP) on R_k, and it is also an effective lower bound of the global optimal value of the original problem (IQP) on R_k, i.e.

$$\nu((IQP)R_k) \leq \nu((QP)R_k) \leq \nu((LP)R_k).$$

Determining the upper bound is completed by feasible points of the problem (IQP) in the process of branching. Suppose x is the feasible point of the problem (IQP) and W represents the set of current feasible points of problems (IQP), which are obtained in the process of branch and bound. Let $x \to W$. γ is the upper bound of the current global optimal value of the problem (IQP). If $W = \emptyset$, let $\gamma = +\infty$; if $W \neq \emptyset$, let $\gamma = min\{f(x) : x \in W\}$ and find a current optimal solution $x^\gamma \in arg\ min\ W$. During the process of the branch and bound, all the feasible solutions $x \to W$ dynamically, and the upper bound is updated by adding new feasible point constantly.

3. The subdivision and reduction of the rectangle

3.1. The subdivision of the rectangle

Suppose that $R_k = [\underline{x}^k, \overline{x}^k] \subseteq R$ is the current sub-rectangle to be divided and (x^k, ν^k) denotes the optimal solution of the problem $(LP(R_k))$. Obviously, $x^k \in D \cap R_k$.

Assuming $X^k \in R_k$ but $x^k \notin Z^n$, do the following subdivision to $R_k = [\underline{x}^k, \overline{x}^k]$:

Step1 Compute $\omega = max\{(x_j^k - \underline{x}_j^k)(\overline{x}_j^k - x_j^k) : j = 1, 2, ..., n\}$,
if $\omega = 0$,
let $\overline{x}_\xi^k - \underline{x}_\xi^k = max\{\overline{x}_j^k - \underline{x}_j^k : j = 1, 2, ..., n\}$,
then $x_\xi^k = \dfrac{\overline{x}_\xi^k + \underline{x}_\xi^k}{2}$;
else ,
find the first $x_j^k \in arg\ maxW$ and mark $x_x^k = x_j^k$
end if;
mark $x^{'} = (\underline{x}_1^k, \underline{x}_2^k, ..., \underline{x}_{j-1}^k, \underline{x}_j^k, \underline{x}_{j+1}^k, ..., \underline{x}_n^k)$;

Step2 Divide the rectangle R_k into two sub-rectangles $R_{k1} = [\underline{x}^{k1}, \overline{x}^{k1}]$ and $R_{k2} = [\underline{x}^{k2}, \overline{x}^{k2}]$ by the line for $x^{'}$ and x^k. Dividing the edge $[\underline{x}_j^k, \overline{x}_j^k]$ into two parts and rounding them, we get $[\underline{x}_j^k, \lfloor \overline{x}_\xi^k \rfloor]$ and $[\lceil \underline{x}_\xi^k \rceil, \overline{x}_j^k]$. So the left lower vertex and the right upper vertex of the sub-rectangle $R_{k1} = [\underline{x}^{k1}, \overline{x}^{k1}]$ are:

$$\underline{x}^{k1} = (\underline{x}_1^{k1}, \underline{x}_2^{k1}, ..., \underline{x}_n^{k1})^T, \overline{x}^{k1} = (\overline{x}_1^{k1}, \overline{x}_2^{k1}, ..., \overline{x}_{j-1}^{k1}, \lfloor \overline{x}_\xi^k \rfloor, \overline{x}_{j+1}^{k1}, ..., \overline{x}_n^{k1})^T;$$

the left lower vertex and the right upper vertex of the sub-rectangle $R_{k2} = [\underline{x}^{k2}, \overline{x}^{k2}]$ are:

$$\underline{x}^{k2} = (\underline{x}_1^{k2}, \underline{x}_2^{k2}, ..., \underline{x}_{j-1}^{k2}, \lceil \underline{x}_j^k \rceil, \underline{x}_{j+1}^{k2}, ..., \underline{x}_n^{k2}), \overline{x}^{k2} = (\overline{x}_1^{k2}, \overline{x}_2^{k2}, ..., \overline{x}_n^{k1})^T.$$

3.2. The reduction of the rectangle

All linear and inequality constraints of the problem (IQP) can be expanded into the form $\sum_{j=1}^n a_{ij}x_j, (i = 1, 2, ..., m)$. Using the rectangular reduction technique of the literature [9], let $I^k := \{1, 2, ..., m\}$.

For $i = 1, 2, ..., m$, do begin
compute $rU_i := \sum_{j=1}^n max\{a_{ij}\underline{x}_j^k, a_{ij}\overline{x}_j^k$, and $rL_i := \sum_{j=1}^n min\{a_{ij}\underline{x}_j^k, a_{ij}\overline{x}_j^k$;
if $rL_i > b_i$
then stop. Problem (IQP) does not have feasible solutions on R_k. (R_k is removed);
else if $rU_i \le b_i$,
then $I^k := I^k - \{i\}$. The $i - th$ linear and inequality constraint of problem (IQP) is removed;
else
for $j = 1, 2, ..., n$, do
if $a_{ij} > 0$,
then $\overline{x}_j^k := min\{\overline{x}_j^k, \dfrac{b_i - rL_i + min\{a_{ij}\underline{x}_j^k, a_{ij}\overline{x}_j^k\}}{a_{ij}}\}$;
if $\overline{x}_j^k \in Z$,
$\overline{x}_j^k := \overline{x}_j^k$,
else
$\overline{x}_j^k := \lceil \overline{x}_j^k \rceil$
end if;
else if $a_{ij} < 0$
then $\underline{x}_j^k := max\{\underline{x}_j^k, \dfrac{b_i - rU_i + max\{a_{ij}\underline{x}_j^k, a_{ij}\overline{x}_j^k\}}{a_{ij}}\}$ if $\underline{x}_j^k \in Z$,
$\underline{x}_j^k := \underline{x}_j^k$,
else
$\underline{x}_j^k := \lfloor \underline{x}_j^k \rfloor$
end if;
end if;
end do;
end if;
end do.

For convenience, the new integer rectangle generated by this new algorithm still denoted by R_k, which is a subset of the original integer rectangle.

4. The new branch and bound algorithm

When the iteration proceed in step k, the feasible region of the problem (IQP) is denoted by F, R_k represents the integer rectangle which will be divided soon, the set of all current feasible points is denoted by W, T represents the set of remained rectangles after pruning, x^k, $\nu(R_k)$ are the optimal solution and the optimal value of problem $(LP(R_k))$ on R_k respectively. $\nu = min\{\nu(R_k) : R_k \in T$ and $x^\nu \in arg\ min\nu$ represent the current global optimal solution and global optimal value of the problem $(LP(R_k))$ respectively, ν is also a lower bound of the global optimal value of the problem (IQP), R_ν is the integer rectangle which correspond to ν, the upper bound of the global optimal value of the problem (IQP) is denoted by γ. If $W = \emptyset$, let $\gamma = +\infty$; if $W \neq \emptyset$, let $\gamma = min\{f(x) : x \in W\}$ and find a current optimal solution $x^\gamma \in arg\ minW$.

Step1(initialization)

Construct a n-dimension rectangle $R = [\underline{x}, \overline{x}]$ which contains F and let $W = \{\underline{x}, \overline{x}\} \cap F$. Solving the linear programming problem $(LP(R))$, we get the optimal value and the optimal solution, which is denoted by ν and x_ν respectively, where the rectangle to which x_ν belongs is denoted by R_ν, ν is a lower bound of the global optimal value of the problem (IQP). If $x_\nu \in F$ let $W = W \cup \{x_\nu\}$, $\gamma = min\{f(x) : x \in W\}$. If $W = \emptyset$, let $\gamma = +\infty$; if $W \neq \emptyset$, let $\gamma = min\{f(x) : x \in W\}$ and find a current optimal solution $x^\gamma \in arg\ minW$. Let $T = \{R\}, k = 1$.

Step2(termination criterion)

If $\nu = \gamma$, then stop the calculation and output the global optimal solution X_γ and the global optimal value $f(x_\gamma)$ of the problem (IQP), otherwise, go to the next step.

Step3(selection rule)

Select the rectangle R_k in T, which ν corresponds to, i.e. $\nu = \nu(R_k)$;

Step4(subdivision method)

If $x^k \in Z^n$, delete $R_k = [\underline{x}^k, \overline{x}^k]$ and keep the optimal solution x^k, otherwise, divide the rectangle R_k into two sub-rectangles R_{k1} and R_{k2} by the method in 3.1, and $intR_{k1} \cap intR_{k2=f}$, $W = W \cup \{\bigcup_{i=1}^{2}(\{\underline{x}^{ki}, \overline{x}^{ki}\} \cap F)\}$;

Step5(reduction method)

Perform the following reduction to the sub-rectangles after dividing by the technique in 3.2. For convenience, the new rectangles after reducing still will be denoted by $R_{ki}, i \subset \Gamma$, where Γ is the index set of the reduced rectangle:

Step5.1 if $\underline{x}^{k1} = \overline{x}^{k1}$
delete R_{k1} and reserve the optimal solution x^{k1};
else
reduce R_{k1} by the reduction technique in 3.2
if R_{k1} is deleted
$T = T \setminus \{R_k\}$, turn to **Step5.2**;
else
$T = T \setminus \{R_k\} \cup \{R_{k1}\}$
solving the linear relaxation programming problem $(LP(R_k))$ we can get the optimal value ν_{k1} and the optimal solution x^{k1}
if $x^{k1} \in Z^n$
$W = W \cup \{\{\underline{x}^{k1}, \overline{x}^{k1}, x^{k1}\} \cap F\}$;
else
$W = W \cup \{\{\underline{x}^{k1}, \overline{x}^{k1}\} \cap F\}$;
endif
endif
endif

Step5.2 if $\underline{x}^{k2} = \overline{x}^{k2}$
delete R_{k2} and reserve the optimal solution x^{k2};
else
reduce R_{k2} by the reduction technique in 3.2
if R_{k2} is deleted, turn to **Step6**;
else $T = T \cup \{R_{k2}\}$;
solving the linear relaxation programming problem $(LP(R_{k2}))$ we can get the optimal value ν_{k2} and the optimal solution x^{k2}
if $x^{k2} \in Z^n$
$W = W \cup \{\{\underline{x}^{k2}, \overline{x}^{k2}, x^{k2}\} \cap F\}$;
else
$W = W \cup \{\{\underline{x}^{k2}, \overline{x}^{k2}\} \cap F\}$;
endif
endif
endif

Step6(determine the upper bound)
If $W = \emptyset$, then $\gamma = \gamma$;
If $W \neq \emptyset$, then $\gamma = min\{f(x) : x \in W\}$ and find a current optimal solution $x^{\gamma} \in arg\ minW$.

Step7(pruning rule)
Let $T = T \setminus \{R : \nu(R) \geq \gamma, R \in T\}$.

Step8(determine the lower bound)

$$\nu = \begin{cases} \gamma, & T = \emptyset, \\ min\{\nu(R) : R \in T\}, & T = \emptyset. \end{cases}$$

Let $k \neg k + 1$. Turn to **Step2**

5. Numerical experiments

For the new algorithm in this paper and the method in literature [9], we apply MATLAB7.8.0 for programming. When Q is positive definite, negative definite and indefinite, we use symmetric matrices Q of different conditions and sizes to test respectively. All tests run on the computer with Intel (R) Core (TM) i5-3570K, CPU@3.40GHz, 4.00GB RAM, 32-bit operating system. Firstly, we explain the feasibility and effectiveness of the new algorithm through a simple example.

Example 5.1.

$$(IQP) \begin{cases} min & f(x) = \frac{1}{2}x^T Q x + c^T x \\ s.t. & Ax \leq b, \\ & 0 \leq x \leq 60, x \in Z^n, \end{cases}$$

where $Q = \begin{bmatrix} 2 & 4 & 6 \\ 4 & 4 & 10 \\ 6 & 10 & 6 \end{bmatrix}, c = \begin{bmatrix} 2 \\ 4 \\ 9 \end{bmatrix}, A = \begin{bmatrix} 2 & -5 & 3 \\ -4 & -1 & -3 \\ 0 & -1 & -1 \\ -14 & -2 & -14 \end{bmatrix}, b = \begin{bmatrix} 4 \\ -3 \\ -1 \\ -4 \end{bmatrix}$.

The initial rectangle of the original problem is $R_1 = \begin{bmatrix} 0 & 60 \\ 0 & 60 \\ 0 & 60 \end{bmatrix}$. Using the new proposed algorithm in this paper to solve the linear relaxation programming problem $(LP(R_1))$ of the original problem (IQP) first, we obtain the optimal solution $x^1 = [0.5000; 1.0000; 0.0000]$ and the optimal value 9.2500 then the lower bound and the upper bound of the original problem on the initial rectangle are 9.2500 and 94500 respectively, and the corresponding optimal solution is $x^* = [60; 60; 60]$. Updating the feasible solution set W constantly

during the process of calculating the lower bound and subdividing and reducing the rectangle, so as to update the upper bound. Select the rectangle $R_1 = \begin{bmatrix} 0 & 60 \\ 0 & 60 \\ 0 & 60 \end{bmatrix}$ with the minimum lower bound to divide. Divide the rectangle R_1 into two sub-rectangles $R_{11} = \begin{bmatrix} 0 & 60 \\ 0 & 1 \\ 0 & 60 \end{bmatrix}$ and $R_{12} = \begin{bmatrix} 0 & 60 \\ 1 & 60 \\ 0 & 60 \end{bmatrix}$ by the method in 3.1 and reduce it by the technique in 3.2, then the new rectangle after reducing is $R_2 = R_{12} = \begin{bmatrix} 0 & 60 \\ 1 & 60 \\ 0 & 60 \end{bmatrix}$. Solving the linear relaxation programming problem $(LP(R_2))$ on the rectangle R_2, we obtain the optimal value 9.2500 so the lower bound of the original problem is not updated, while the current upper bound is updated to 2736 and the corresponding optimal solution is $x^* = [0; 30; 0]$. Divide the rectangle R_2 into $R_{21} = \begin{bmatrix} 0 & 60 \\ 1 & 60 \\ 0 & 60 \end{bmatrix}$ and $R_{22} = \begin{bmatrix} 1 & 60 \\ 1 & 60 \\ 0 & 60 \end{bmatrix}$ then reduce them, the new rectangle after reducing is $R_3 = R_{22} = \begin{bmatrix} 1 & 60 \\ 1 & 60 \\ 0 & 60 \end{bmatrix}$. Solve the linear relaxation programming problem $(LP(R_3))$ and get the optimal value 9.2500, while the current upper bound is 13 and optimal solution is $x^* = [1; 1; 0]$. Subdividing and reducing R_3 constantly, its sub-rectangles are all deleted, so we can know that the global optimal value of the original problem (IQP) is 13 and the global optimal solution is $x^* = [1; 1; 0]$.

The two main factors that affect the difficulty of solving integer quadratic programming problems are the condition number of the matrix Q in the objective function and the form of the coefficient matrix A in constraints. In order to make the test general and persuasive, we randomly generate matrices Q of specified dimensions and different conditions and vectors of the first degree of the objective function in MATLAB. The specific forms of constraints are selected from Table 5.

To better illustrate the advantages of the new algorithm in this paper for solving general integer quadratic programming problems, we compare the new algorithm with the branch and bound method in [9], the specific results are listed in Table 1 to 4 respectively. In tables, the column of constraint No. represents the corresponding number of constraints in the Table 5. The column of the condition number of the matrix Q indicates the norm condition number of the matrix Q. Integer objective value represents the optimal value of the objective function when algorithm terminates. Alg.1 and Alg.2 denote the new branch and bound algorithm in this paper and the branch and bound method in literature [9] respectively. In order to investigate the effect when Q is positive definite, negative definite and indefinite on the algorithm respectively, Tables 1 to 3 give the test results for matrices Q with different constraints, sizes and conditions. From tables, we know that the test time when the matrix Q is indefinite is relatively longer than that when Q is positive definite and negative definite. In addition, in order to investigate the influence of the condition number of the matrix Q on the algorithm, we give test results of 100-dimension problems with same constraints but different condition numbers of matrices Q in Table 4. From the table, we know that the time which the algorithm needs to solve the problem increases with the condition number of the matrix Q, which indicates that the condition number of the matrix Q is indeed a major factor that affects the effectiveness of the algorithm.

Results in tables show that the calculation time of the new algorithm in this paper are all much less than that of the algorithm in [9] with the increase of the dimension and the calculation results are better, which indicates that the proposed algorithm improves the computational efficiency, so the algorithm in this paper is effective and feasible. From results of Table 1 to 4, although the iteration and the integer objective value of our new algorithm may exceed that in the algorithm of [9], for large-scale integer quadratic programming problems, running times of the new algorithm is far less than that in [9], so the algorithm in this paper is superior to the algorithm in the literature [9], while the algorithm in [9] is superior than the normal branch

and bound algorithm.

Generally speaking, with the increase of determine variables of the problem and the effective constraints, the time of solving the problem also increases. With the increase of the condition number of the matrix Q, the complexity and the time of solving the problem all increase. Tests which we use in this paper are all based on the literature [9], but from the calculation time of the proposed algorithm and the algorithm in [9], the former algorithm is far less than the latter one.

Table 1: when Q is positive definite, results of solving integer quadratic programming problems with different sizes

test No.	dimension	Constraint No.	Condition number of the matrix Q	Time (s)		Integer objective value		Iteration	
				Alg.1	Alg.2	Alg.1	Alg.2	Alg.1	Alg.2
1	10	1	305	1.136	2.675	352.5	200	5	7
2	30	2	3823	10.133	21.756	76112.5	73336.5	13	15
3	50	3	244269	10.642	39.278	1535261	1845614	7	9
4	50	4	244269	10.161	26.103	4930	4959	7	9
5	70	5	13114	52.829	63.653	27737.5	24445	25	17
6	70	6	13114	13.279	41.628	6804	6923.5	5	9
7	100	7	225175	20.861	17.709	10589	14400	5	3
8	100	8	225175	212.380	222.497	42018.5	38655	57	35
9	100	9	225175	55.056	87.462	10515	9850.5	13	11

Table 2: when Q is negative definite, results of solving integer quadratic programming problems with different sizes

test No.	dimension	Constraint No.	Condition number of the matrix Q	Time (s)		Integer objective value		Iteration	
				Alg.1	Alg.2	Alg.1	Alg.2	Alg.1	Alg.2
1	10	1	132	1.127	1.883	212.5	233.5	5	5
2	30	2	4514	8.558	12.810	75210	64662	11	9
3	50	3	4979	6.916	20.003	1187750	4837869	5	7
4	50	4	4979	3.684	18.734	4950	4264	3	7
5	70	5	8099	57.491	69.244	28939	26874	27	17
6	70	6	8099	13.701	12.953	7665	10380	5	3
7	100	7	577740	17.790	21.310	10602	15887.5	5	3
8	100	8	577740	205.962	238.168	42150.5	37818	59	35
9	100	9	577740	62.875	86.562	11152.5	9717.5	15	11

Table 3: when Q is indefinite, results of solving integer quadratic programming problems with different sizes

test No.	dimension	Constraint No.	Condition number of the matrix Q	Time (s)		Integer objective value		Iteration	
				Alg.1	Alg.2	Alg.1	Alg.2	Alg.1	Alg.2
1	10	1	204	1.908	3.275	255.5	274.5	9	9
2	30	2	8102	8.936	12.138	68010	57283	11	9
3	50	3	56641.8	7.843	22.749	16002009	2132540	5	7
4	50	4	56641.8	9.976	24.945	5221.5	5519	7	9
5	70	5	144119	49.891	64.527	28265.5	26016.5	23	17
6	70	6	144119	12.084	22.993	9441	7374	5	5
7	100	7	27341	19.982	21.463	9747	14430	5	3
8	100	8	27341	211.563	223.472	42277	38045.5	61	35
9	100	9	27341	50.597	88.102	9861.5	9225.5	13	11

Table 4: results of solving integer quadratic programming problems with same sizes and constraints but different conditions

test No.	dimension	Constraint No.	Condition number of the matrix Q	Time (s)		Integer objective value		Iteration	
				Alg.1	Alg.2	Alg.1	Alg.2	Alg.1	Alg.2
1	100	9	21645	52.133	84.947	10660	9489.5	13	11
2	100	9	63552	53.611	90.163	9990	9638.5	13	11
3	100	9	151295	65.511	73.871	10146.5	9709	15	9
4	100	9	397950	47.072	90.799	10049	9720.5	11	11
5	100	9	750783	45.727	109.414	10593	9468.5	11	13

Table 5: [9] constraints be used in different tests

Constraints No.	The specific form of constraints
1	$x_1 + x_2 + 2x_3 + 8x_4 + x_5 \leq 100$ $x_2 + 3x_4 + 15x_6 + x_9 \leq 200$ $x_3 + x_5 + 13x_6 + 10x_1 + x_9 \leq 300$ $x_4 + 20x_5 + 8x_7 + x_{10} \leq 300$ $\sum_{i=1}^{10} x_i \geq 15, 0 \leq x_i \leq 60, x_i \in Z, i = 1, 2, ..., 10$
2	$x_1 + 5x_6 + x_7 + 8x_8 + 2x_9 + x_{10} - 2x_{11} + 5x_{13} + 3x_{22} - x_{28} \leq 500$ $x_2 + 8x_{11} + 4x_{13} - x_{14} + 3x_{19} + 2x_{20} + x_{21} + 2x_{22} + x_{23} + x_{24} \leq 500$ $x_3 + 5x_7 + x_8 + 3x_{12} + 2x_{14} + x_{18} + 3x_{22} + 8x_{25} + 3x_{27} + x_{30} \leq 600$ $x_4 + x_9 + 2x_{13} + 5x_{14} + 3x_{15} + 2x_{16} + x_{17} + x_{26} \leq 400$ $x_5 + 2x_6 + x_{11} + 4x_{17} + 5x_{20} + 3x_{21} + 4x_{23} + x_{28} + x_{29} \leq 400$ $\sum_{i=1}^{30} x_i \geq 150, 0 \leq x_i \leq 60, x_i \in Z, i = 1, 2, ..., 30$

Table 6: example of table

3	$x_1 + x_9 + x_10 + 4x_{15} - 5x_{19} + 3x_{23} + x_{24} - x_{28} + 2x_{29} + x_{31} \leq 1000$ $x_2 - x_9 + x_{11} + 4x_{13} + 2x_{19} + 2x_{20} + x_{21} + 3x_{22} + x_{23} + x_{24} \leq 1000$ $2x_3 + x_{10} - 2x_{11} + 3x_{12} + x_{14} + 5x_{17} + 3x_{22} + 2x_{25} + 3x_{27} + x_{30} \leq 1000$ $x_4 + x_9 + 2x_{13} + 5x_{14} + 3x_{15} + x_{16} + x_{31} + x_{33} + x_{35} + x_{36} \leq 1000$ $4x_5 + x_{11} + x_{17} + 5x_{20} + x_{21} + 4x_{23} - x_{26} + 5x_{29} + 2x_{38} + x_{40} \leq 1000$ $x_6 + 2x_{10} + 2x_{12} + 5x_{13} + 8x_{16} + x_{18} + 3x_{26} + 2x_{28} + 4x_{32} + x_{44} \leq 1000$ $2x_7 + 4x_9 + x_{12} + 2x_{15} + x_{17} + 3x_{18} + 5x_{31} + 6x_{34} + x_{37} + x_{45} \leq 1000$ $x_8 + 2x_{39} + x_{41} + x_{42} + 4x_{43} + x_{46} + x_{47} + x_{48} + x_{49} + x_{50} \leq 1000$ $\sum_{i=1}^{50} x_i \geq 500, 0 \leq x_i \leq 800, x_i \in Z, i = 1, 2, ..., 50$
4	$x_1 + 2x_9 - x_{10} + 4x_{15} + x_{19} + x_{23} + x_{24} - x_{28} + 2x_{29} + x_{31} \leq 300$ $x_2 + x_{12} - x_{13} + 2x_{19} + x_{20} + x_{21} + 3x_{22} + x_{23} + x_{24} + x_{29} \leq 300$ $x_3 + x_{12} + x_{14} + 3x_{17} - 2x_{21} + x_{22} + 2x_{25} + 3x_{27} + x_{30} + x_{31} \leq 300$ $x_4 + x_{14} + 3x_{15} + x_{16} + 2x_{19} + x_{31} + 2x_{33} + x_{35} + x_{36} + 5x_{49} \leq 300$ $x_5 + 2x_{17} + x_{20} + x_{21} + 4x_{23} - x_{26} + x_{29} + 2x_{38} + x_{40} + x_{41} \leq 300$ $x_6 + x_{10} + 8x_{12} + x_{16} + x_{18} - x_{26} + 2x_{28} + 4x_{32} + x_{43} + x_{44} \leq 300$ $x_7 - x_{15} + x_{17} + x_{22} + 4x_{32} + 6x_{34} + 3x_{43} + x_{45} + x_{47} + x_{49} \leq 300$ $x_8 + 4x_{18} + x_{31} + x_{37} - 2x_{39} + x_{41} + x_{42} + 3x_{46} + x_{48} + x_{50} \leq 300$ $\sum_{i=1}^{50} x_i \geq 30, 0 \leq x_i \leq 50, x_i \in Z, i = 1, 2, ..., 50$
5	$\sum_{i=1}^{70} x_i = 500, \sum_{i=1}^{70} x_i \geq 60, 0 \leq x_i \leq 10, x_i \in Z, i = 1, 2, ..., 70$
6	$x_1 + 3x_{12} + x_{10} + 2x_{19} + x_{30} - 5x_{31} + x_{34} + x_{37} - x_{45} + x_{65} \leq 500$ $x_2 + x_{19} + 2x_{20} + 5x_{29} + x_{32} + x_{37} + x_{42} - x_{45} - x_{46} + 4x_{64} \leq 500$ $x_3 + 5x_{13} + 2x_{16} + x_{28} + x_{39} + x_{43} - x_{41} - x_{58} + 2x_{59} + 3x_{63} \leq 500$ $x_4 + x_{11} + x_{26} + x_{33} + 2x_{34} - x_{39} - 2x_{40} + x_{57} + 3x_{61} + 5x_{62} \leq 500$ $x_5 + x_{18} + x_{27} + 2x_{28} - x_{29} - x_{30} + x_{38} + x_{39} + 5x_{47} + x_{66} \leq 500$ $x_6 + x_{14} + 2x_{23} + 5x_{34} + x_{36} + x_{37} + x_{38} - x_{55} - x_{56} + 4x_{67} \leq 500$ $x_7 + 2x_{15} - x_{22} - x_{35} + 7x_{44} + x_{46} + x_{47} + x_{48} + x_{50} + x_{54} \leq 500$ $x_8 - x_{16} - 2x_{24} + 6x_{28} + 2x_{30} + x_{35} + x_{37} + x_{53} + x_{58} + 2x_{69} \leq 500$ $x_9 - x_{17} + 7x_{21} + x_{36} + 5x_{37} + x_{45} + x_{49} + x_{52} + x_{60} - 7x_{68} \leq 500$ $x_{10} + 6x_{20} + x_{25} + x_{27} + x_{39} + x_{40} + x_{50} + x_{51} + x_{54} + x_{70} \leq 500$ $\sum_{i=1}^{70} x_i \geq 30, 0 \leq x_i \leq 50, x_i \in Z, i = 1, 2, ..., 70$

Table 7: example of table

7	$x_1 + x_{11} + 3x_{21} + x_{36} + x_{48} + x_{58} + x_{66} + 2x_{71} + x_{81} + 3x_{99} \leq 550$ $x_2 + 4x_{12} + x_{22} + x_{38} + x_{44} - x_{57} + x_{65} + x_{77} + x_{79} + x_{98} \leq 550$ $x_3 + x_{12} + x_{23} + 5x_{31} + x_{47} + 2x_{53} + x_{70} + x_{72} + 2x_{82} + x_{91} \leq 550$ $x_4 + x_{14} + x_{24} + x_{33} + 2x_{43} - x_{59} - 2x_{64} + x_{73} + 3x_{76} + 5x_{100} \leq 550$ $x_5 + x_{15} + x_{27} + 2x_{39} + x_{42} - x_{52} + x_{63} + x_{78} + x_{80} + 3x_{97} \leq 550$ $x_6 + x_{16} + x_{30} + x_{32} - x_{41} + x_{45} + 3x_{54} + x_{69} + x_{74} + x_{92} \leq 550$ $x_7 + 2x_{17} + x_{26} + x_{40} + x_{45} - 2x_{51} + x_{62} + x_{77} + x_{83} + 7x_{96} \leq 550$ $x_8 + x_{18} + x_{25} + x_{34} + x_{46} - x_{55} + 5x_{61} + x_{75} + x_{79} + x_{93} \leq 550$ $x_9 - x_{19} + x_{29} + 4x_{35} + 5x_{49} + x_{60} + x_{67} + x_{80} - 3x_{84} + x_{95} \leq 550$ $x_{10} + x_{20} - x_{23} + x_{37} + 3x_{50} + x_{56} + x_{68} + 4x_{76} + x_{78} + x_{94} \leq 550$ $x_{11} + x_{23} + 3x_{31} + x_{37} + x_{39} + x_{54} + x_{64} + 2x_{67} + x_{79} + x_{90} \leq 550$ $x_{12} + x_{36} + 2x_{40} + x_{41} + 7x_{55} + x_{56} + x_{68} + x_{74} + x_{79} + x_{90} \leq 550$ $x_{13} + x_{18} + x_{25} + 5x_{42} + x_{51} - x_{57} + x_{58} + x_{71} + x_{72} + x_{87} \leq 550$ $x_{14} + x_{27} + x_{45} - 3x_{47} + x_{49} + x_{59} + x_{60} + x_{75} + 3x_{79} + 3x_{88} \leq 550$ $x_{15} + x_{31} + x_{40} + x_{50} + x_{51} + x_{62} + x_{63} + x_{81} + x_{85} + x_{89} \leq 550$ $\sum_{i=1}^{100} x_i \geq 30, 0 \leq x_i \leq 60, x_i \in Z, i = 1, 2, ..., 100$
8	$\sum_{i=1}^{100} x_i = 300, \sum_{i=1}^{100} x_i \geq 60, 0 \leq x_i \leq 4, x_i \in Z, i = 1, 2, ..., 100$
9	$x_1 + x_{11} + 3x_{21} + x_{36} + x_{48} + x_{58} - 7x_{66} + 2x_{71} + x_{81} + 3x_{99} \leq 60$ $x_2 + 4x_{12} + x_{22} - 3x_{38} + x_{44} - x_{57} + x_{65} + x_{77} + x_{79} + x_{98} \leq 60$ $x_3 + x_{12} + x_{23} + 5x_{31} + x_{47} + 2x_{53} + x_{70} + x_{72} + 2x_{82} + x_{91} \leq 60$ $x_4 + x_{14} - 3x_{24} + x_{33} + 2x_{43} + x_{59} + x_{64} + 4x_{73} + 3x_{76} + 5x_{100} \leq 60$ $x_5 + x_{15} + x_{27} + 2x_{39} + x_{42} - x_{52} + x_{63} + x_{78} + x_{80} + 3x_{97} \leq 60$ $x_6 + x_{16} + x_{30} + x_{32} - x_{41} + x_{45} + 3x_{54} + x_{69} + x_{74} + x_{92} \leq 60$ $x_7 + x_{17} + x_{26} + x_{40} + x_{45} - 2x_{51} + x_{62} + x_{77} + x_{83} + 7x_{96} \leq 60$ $x_8 + x_{18} + x_{25} + x_{34} + x_{46} - x_{55} + 5x_{61} + x_{75} + x_{79} + x_{93} \leq 60$ $x_9 - x_{19} + x_{29} + 4x_{35} + 5x_{49} + x_{60} + x_{67} + x_{80} - 3x_{84} + x_{95} \leq 60$ $x_{10} + x_{20} - x_{23} + x_{37} + 3x_{50} + x_{56} + x_{68} + 4x_{76} + x_{78} + x_{94} \leq 60$ $x_{11} + x_{23} - 2x_{31} + x_{37} + x_{39} + x_{54} + x_{64} + 2x_{67} + x_{79} + x_{90} \leq 60$ $x_{12} + x_{36} + 2x_{40} + x_{41} + 7x_{55} + x_{56} + x_{68} + x_{74} + x_{79} + x_{90} \leq 60$ $x_{13} + x_{18} + x_{25} + 5x_{42} + x_{51} - x_{57} + x_{58} + x_{71} + x_{72} + x_{87} \leq 60$ $x_{14} + x_{27} + x_{45} - 3x_{47} + x_{49} + x_{59} + x_{60} + x_{75} + 3x_{79} + 3x_{88} \leq 60$ $x_{15} + 4x_{31} + x_{40} + x_{50} + x_{51} + x_{62} + x_{63} + x_{81} + x_{85} + x_{89} \leq 60$ $\sum_{i=1}^{10} x_i \geq 30, 0 \leq x_i \leq 10, x_i \in Z, i = 1, 2, ..., 100$

6. Conclusion

For the preparation to seek a common and effective algorithm for solving general nonlinear integer programming problems, this paper comes up with an algorithm for solving integer quadratic programming problems quickly and effectively by using a new linear relaxation and bound method, a rectangular deep bisection method and a rectangular reduction technology to improve the traditional branch and bound algorithm. Numerical results show that the proposed algorithm is feasible and effective. Although the iteration and the integer objective value of the new algorithm in this paper may exceed those of the algorithm in [9] sometimes, for large-scale integer quadratic programming problems, the running time of our new algorithm is far less than that in [9] and our new algorithm have a better calculation results. The algorithm in this paper is superior to the branch and bound algorithm in the literature [9], which indicates that the proposed algorithm improves the calculation efficiency and is more obvious to large-scale integer quadratic programming problems.

Acknowledgements

This work was supported by the National Natural Science Foundation of P. R. China (11161001) and the Foundations of research projects of State Ethnic Affairs Commission of P.R. China (14BFZ003) and the Foundations of research projects of Beifang University of Nationalities (2015KJ10).

References

[1] M. L. Bergamini, I. Grossmann, N. Seenna, P. Aguirre, *An improved piecewise oute-approximation algorithm for the global optimization of MINLP models involving concave and bilinear terms*, Comput. Chem. Eng., **32** (2008), 477–493. 1

[2] C. Buchheim, L. Trieu, *Quadratic outer approximation for convex integer programming with box constraints*, Exp. Algorithms, **7933** (2013), 224–235. 1

[3] Z. P. Chen, F. Xi, *A new branch-and-bound algorithm for solving large complex integer convex quadratic programs*, Math. Numer. Sin., **26** (2004), 445–458. 1

[4] M. A. Duran, I. E. Grossmann, *An outer-approximation algorithm for a class of mixed-integer nonlinear programs*, Math. Program., **36** (1986), 307–339. 1

[5] V. P. Eronena, M. M. Mäkeläa, T. Westerlundb, *Extended cutting plane method for a class of nonsmooth non-convex MINLP problems*, Optim., **64** (2015), 641–661. 1

[6] R. Fletcher, S. Leyffer, *Solving mixed intger nonlinear programs by outer approximation*, Math. program., **66** (1994), 327–349. 1

[7] O. E. Flippo, A. H. Rinnoy Kan, *A note on benders decomposion in mixed-integer quadratic programming*, Oper. Res. Lett., **9** (1990), 81–83. 1

[8] Y. L. Gao, Y. J. Wang, X. W. Du, *A two-level relaxed bound method for a nonlinear class of 0-1 knapsack problems*, Intelligent Information, Manage. Syst. Technol., **3** (2005), 461–470. 1

[9] Y. L. Gao, F. Wei, *A branch-and-bound reduced method for a class of non-negative integer quadratic programming problems*, math. Numer. sin., **33** (2011), 233–248. 1, 3.2, 5, 5, 5, 6

[10] T. Kuno, *Solving a Class of Multiplicative programs with 0-1 Knapsack Constraints*, J. Optim. Theory Appl., **103** (1999), 121–135. 1

[11] R. Misener, C. A. Floudas, *GloMIQO: Global mixed-integer quadratic optimizer*, J. Global Optim., **57** (2013), 3–50. 1

[12] I. Nowak, S. Vigerske., *LaGO: a (heuristic) branch and cut algorithm for nonconvex MINLPs.*, CEJOR Cent. Eur. J. Oper. Res., **16** (2008), 127–138. 1

[13] A. Schrijver, *Theory of linear and integer programming*, John Wiley and Sons, Chichester, (1986). 1

[14] R. A. Stubbs, S. Mehrotea, *A branch-and-cut method for 0-1mixed convex programming*, Math. Program., **86** (1999), 512–532. 1

[15] X. L. Sun, J. L. Li, H. Z. Luo, *Convex relaxation and Lagrangian decomposition for indefinite quadratic integer programming*, Optim., **59** (2010), 627–641. 1

[16] T. Westerlund, F. Pettersson, *An extended cutting plane method for solving convex MINLP problems*, Comput. Chem. Eng., **19** (1995), 131–136. 1