A branch and cut algorithm for solving the linear and quadratic integer programming problems *

Yan Zizong† Fei Pusheng‡ Wan Zhongping§

Abstract

This paper first presents an improve cutting plane method for solving the linear programming problems, based on the primal simplex method with the current equivalent facet technique, in which the increment of objection function is allowed as a pivot variable to decide the search step size. We obtain a strong valid inequality from the objective increment model so as to find a better integer feasible solution in the discrete integral equivalent face. Then we propose a branch and cut for the quadratic integer programs. A example shows that it is superior to other approaches to solving IQP problems.

Keywords. Integer programming, Quadratic programs, Valid inequality, Cutting plane, Dual gap

AMS subject classifications. 90C10, 90C11

1 Introduction

Integer programming refers to mathematical programming with discrete variables in the objective function and constraints. The use of integer programming problems is a natural approach of formulating problems where it is necessary to simultaneously optimize the system structure.

Integer programming is modelling a variety of vary important problems in various applications, including the process industry and the financial, engineering, management science and operations research sectors. It includes problems in process flow sheets, portfolio selection, batch processing in chemical engineering, and optimal deign of gas or water transmission networks. Other areas of interest include the automobile, aircraft, and VLSI manufacturing areas. The needs in such diverse areas

*Supported by the National Natural Science Foundation of China (70371032) and the Doctor Educational Foundation of the Ministry of Education (20020486035), P.R.C.

†School of Mathematics and Statistics, Wuhan university, and School of Information and Mathematics, Yangtze university, Jingzhou, Hubei, China(zzyan@jznu.net).

‡School of Mathematics and Statistics, Wuhan university,China(pshfei@whu.edu.cn).

§School of Mathematics and Statistics, Wuhan university,China(zpwan-whu@126.com).

have motivated research and development in integer programming problems solver technology, particularly in algorithms for handling large-scale, highly combinatorial and highly nonlinear problems.

In this paper we consider global optimization problems of the following form

$$\begin{array}{ll} \min & f(x) = \frac{1}{2}x^T Q x + c^T x \\ s.t. & Ax = b, \quad x \in Z_+^n \end{array} \tag{1.1}$$

where $c \in Z^n$, $b \in Z^m$, $Q \in Z^{n \times n}$ is a symmetric positive semidefinite matrix and $A \in Z^{m \times n}$ is a full rank matrix. We denote by

$$P = \{x \in R^n | Ax = b,\ x \geq 0\}$$

the polyhedron of the linear programming (LP) relaxation

$$\begin{array}{ll} \min & \nabla f(x')^T x \\ s.t. & Ax = b, \quad x \in R_+^n \end{array} \tag{1.2}$$

where $\nabla f(x')$ denotes the gradient of f at x' that belongs to the discrete set

$$P_I = \{x \in Z^n | Ax = b,\ x \geq 0\}$$

or the polyhedron P.

Branch and bound is general method which has been applied to various problems in combinatorial optimization and integer programming problems, including in linear and quadratic integer programming problems. Land and Doig [4] first introduced a branch and bound algorithm for the travelling salesman problem. Recently, the branch and bound idea has also been systematically developed to become an important tool for solving various integer programming problems [3].

Simplex-based branch and cut methods have been successful in the last few years and become very popular tools in recent years for solving integer and mixed-integer programs to optimality (see Nemhauser & Wolsey [8]; Padberg & Rinaldi [9]; Caprara & Fischetti [2]). Virtually all of the cutting plane algorithms in the modern literature are so-called dual-fractional algorithms, based on the dual simplex method, in which the main role of the cutting planes is iteratively tighten a linear relaxation of the original problem. These essentially have their roots in Gomory's classical method of integer forms [6, 7].

There are however alternative approaches for solving integer programs available that depend explicitly on the discrete nature of the set of feasible solutions. Perhaps most important is the use of the primal cutting plane methods. Whereas much effort has been devoted to improving the dual fractional framework, very little has been devoted to the dual integral and primal variants. Algorithms of this type, base on the primal simplex method, were devised by Ben-Israel and Charnes [1] and Young [11], and subsequently simplified by Glover [5] and Young [12]. One of the main goal of the present paper is to convince the reader that this approach, the primal one, is viable for solving (1.2).

The traditional cutting plane methods of 1950's and 1960's largely ignored these mathematical underpinnings. Another special purpose cutting plane methods are based on the polyhedral analysis of the problem formulation. Since it is difficult to capture the whole structure of the optimization problem in these inequalities, the polyhedral analysis is usually performed on simple mathematical structures that are embedded in the problem formulation. Current implementations of branch and cut algorithms heavily rely on the polyhedra theory for designing the cutting plane of such these algorithms. Rather they addressed the question of solving integer programming problems by ways of "valid inequality" and "cutting planes" in a fairly direct algorithmic way.

Definition 1.1. (i) An inequality $g^T x \geq g_0$ is a valid inequality for (1.1) if $g^T x \geq g_0$ for all $x \in P_I$.

(ii) A valid inequality $g^T x \geq g_0$ for (1.1) is cut (or cutting plane) for (1.1) if

$$P \cap \{x \in R^n | g^T x \geq g_0\} \subset P$$

where the containment is proper.

Our first objective in this paper is to contribute to the development of techniques that can be used to generate new classes of strong valid inequalities for (1.2). It is very important step in the design of this algorithm to define classes of strong valid inequalities for P_I. The success of the algorithm depends on the quality of the tightened formulation as approximation of the convex hull of P_I. We investigate how to obtain new classes of valid inequalities by inducting the objective increment model of the linear programming relaxation.

The second objective is to improve upper bounds for branch and bound to solve the integer programs that can be found so as to obtain the integer optimal solution for (1.1) quickly. However, these bounds are usually very and rarely give rise to any fathoming at a later stage once an integer feasible solution been obtained by the use of the modern primal cutting plane algorithm (in section 4). This paper considers QP problems and shows how improved upper bounds can be derived for the subproblem of (1.1).

The remainder of this paper is organized as follows. In section 2, we review the current equivalent simplex method. In section 3 and in section 4, we set up a new strong valid inequality rule and present an improved primal cutting plane algorithm, respectively. In section 5, we describe the heuristics that our branch and bound algorithm uses. In section 6, we give a detailed solution of a example by the use of our branch-and-cut algorithm. Further remark is given in the last section.

2 Background

As mentioned above, cutting plane and branch-and-bound methods work by setting up a a linear programming relaxation (1.2) of the integer programming problem (1.1) (at $Q = 0$), solving that relaxation, and then, if necessary, refining the relaxation

so that the solution to the relaxation gets closer to the solution to the integer programming problem. They can be considered as an extension of a branch-and-cut method for the solution of general convex integer quadratic programming problem (at $Q \neq 0$).

For the problem (1.2), Let us recall its dual form

$$\min b^T w \quad s.t.\ A^T w + r = \nabla f(x'), \quad r \geq 0 \tag{2.1}$$

and the main result in the theory of LP:

Theorem 2.1. *For problems* (1.2) *and* (2.1) *one of the following alternatives hold:*

(i) Both two problems are feasible and there exist $x^* \in P$ *and* $(w, r) \in D$ *such that* $\nabla f(x')^T x^* = b^T y^*$;

(ii) (1.2) *is infeasible and* (2.1) *is unbounded;*

(iii) (2.1) *is infeasible and* (1.2) *is unbounded;*

(iv) Both are infeasible.

An alternative way of writing the optimality condition in the Theorem (2.1) is by using the complementary slack condition

$$r^T x = 0 \tag{2.2}$$

It represents a strong combinatorial character of linear programming problems that is featured by pivot algorithms. The set of optimal solutions can be characterized as the set of solutions of both the system

$$Ax = b, \quad A^T w + r = \nabla f(x'), \quad x, r \geq 0 \tag{2.3}$$

and the complementary slackness condition (2.2).

Consider the objective increment model of (1.2)

$$\min\ x_0 \quad s.t.\ \bar{A}\bar{x} = \bar{b},\ \bar{x} \geq 0 \tag{2.4}$$

where x_0 denotes the objective increment of (1.2) and

$$\bar{A} = \begin{pmatrix} 1 & \nabla f(x')^T \\ 0 & A \end{pmatrix}, \quad \bar{b} = \begin{pmatrix} f(x') \\ b \end{pmatrix}, \quad \bar{x} = \begin{pmatrix} x_0 \\ x \end{pmatrix}$$

The notation $\bar{x}$ will be replaced by x without any confession. So do the notation $\bar{A}$ and $\bar{b}$. A pair of vector (x, r) is called feasibility if there is a vector $w \in R^m$ satisfying (2.3). Analogous, it is called optimal if it is feasible and satisfies the the complementary slackness condition (2.2). The value

$$\delta = r^T x$$

is called the duality gap. And it is always greater than zero for a pair of feasible solution (x, r) expect it is optimal. The idea of the current equivalent facet algorithm is to compel it to become zero.

Because of the primal feasible constrict of variables, we must find a suitable objective increment x_0 so that the inequalities

$$b_i - a_{i0}x_0 \geq 0, \quad i = 0, 1, \cdots, m \tag{2.5}$$

are satisfied at each pivot step. The set of solutions of this inequalities is an internal if it is not empty. If $a_{i0} > 0$ is satisfied for an index i, the associated component will remain feasible up to the critical value $\delta_i = \frac{b_i}{a_{i0}}$.

Definition 2.2. The value $\delta_i = \frac{b_i}{a_{i0}}$ is called a upper bound of the current solution x for an index i if a_{i0} is great than zero. In particular, $\delta_i = +\infty$ is called a upper bound of the current solution x provided $a_{i0} = 0$ and $b_i > 0$, and $\delta = -\infty$ is called an upper bound of the current solution x provided $a_{i0} = 0$ and $b_i < 0$. Sometimes it is called an upper bound simply.

If δ_i is an upper bound with $a_{ij} \geq 0, j = 0, 1, \cdots, n$, it is called a hard upper bound; otherwise, it is called a soft upper bound.

Definition 2.3. The constraint row

$$\sum_{j=0}^{n} a_{lj}x_j = b_l$$

is said to be a dual feasible constraint, if $a_{lj} \geq 0$ is satisfied for $j = 0, 1, \cdots, n$.

A upper bound reflects a duality gap between the primal problem (1.2) and the dual problem (2.1) if the coefficient of its objective increment x_0 is not equal to zero at the corresponding row for the problem (2.4). In fact, a dual feasible constraint implies that there is a dual feasible solution

$$r = (\frac{a_{l1}}{a_{l0}}, \frac{a_{l2}}{a_{l0}}, \cdots, \frac{a_{l,n}}{a_{l0}})^T$$

for (2.1) only if $a_{l0} > 0$. We extend it to the general case.

Next we give the terminal criteria.

Theorem 2.4. *If a basic solution is primal feasible for* (2.4)*, then it is optimal if and only if there is a hard upper bound zero.*

Proof: Let x be a primal feasible basic solution for (2.4) with a basic index decomposition (B, N). If there is a hard upper bound zero, such as $a_{l0} > 0$ and $a_{lj} \geq 0 (j = 1, 2, \cdots, n)$, then it implies x_j is equal to zero for each $j \in N$. It means that the reduced costs are not negative and we can not obtain a better feasible solution with a less objective value by pivoting. In fact, we get a duality gap zero for a pair of feasible solution (x, r). Then it is optimal. And vice versa. □

We now present give a detailed statement to the current equivalent facet simplex algorithm that was given in [13, 14]. Our basic algorithmic framework is the following, which starts with a basic feasible solution.

Algorithm 2.5. Current Equivalent Facet Algorithm

Step 1. Choose a row index

$$p = \arg\min\{\frac{b_i}{a_{i0}} | a_{i0} > 0\} \tag{2.6}$$

and set $\lambda = \frac{b_p}{a_{p0}}$.

Step 2. Update the right hand side coefficients $b \leftarrow b - \lambda A_0$ with the step size λ, where A_0 denotes '0-th' column of A.

Step 3. If a_{pj} is equal to or greater than zero for each $j = 1, 2, \cdots, n$, STOP.

Step 4. Choose a column index q with $a_{pq} < 0$ and pivot on a_{pq}. Go to Step 1.

An interesting thing here is that the position of the test row always changes in our algorithm. This approach is different from the classical simplex methods. In additional, we also keep the primal feasibility of (2.1). Of course, we might select the column index q by using a least index rule or other rules.

For the sake of clarity we now present a simple example of the algorithm (2.5) in operation.

Example 2.6. Consider the linear programs

$$\begin{array}{ll} \min & x_0 = -20x_1 - 30x_2 - 10x_3 \\ s.t. & 4x_1 + 2x_2 + x_3 \leq 23, \quad x_1, x_2, x_3 \geq 0 \end{array}$$

An obvious initial basic, primal feasible solution is obtained by making a slack variable x_4 basic, to give $x' = (0, 0, 0)^T$ and the corresponding initial simplex format is as follows:

$$\begin{array}{rrrrrr} x_0 & -20x_1 & -30x_2 & -20x_3 & = & 0 \\ x_4 & +4x_1 & +2x_2 & +x_3 & = & 23 \end{array}$$

The algorithm (2.5) dictates that x_1 should be basic in place of x_0. The x_0 row there becomes the pivot row, leading to the following format:

$$\begin{array}{rrrrrr} x_1 & -\frac{1}{20}x_0 & +\frac{3}{2}x_2 & +\frac{1}{2}x_3 & = & \frac{23}{4} \\ x_4 & +\frac{1}{5}x_0 & -4x_2 & -x_3 & = & 0 \end{array}$$

The new x' is $(\frac{23}{4}, 0, 0)^T$ and the profit has decreased from 0 to -115. Now we should make x_2 basic in place of x_4. The next form is therefore

$$\begin{array}{rrrrrr} x_1 & +\frac{1}{40}x_0 & +\frac{1}{8}x_3 & +\frac{3}{8}x_4 & = & 0 \\ x_2 & -\frac{1}{20}x_0 & +\frac{1}{4}x_3 & -\frac{1}{4}x_4 & = & \frac{23}{2} \end{array}$$

The new x' is $(0, \frac{23}{2}, 0)^T$ and the profit has decreased from 0 to -345=-230-115. Moreover it is dual feasible because of its hard upper bound zero at the x_1 row, and hence optimal.

3 Strong valid inequality

As mentioned above, it is very important for the cutting plane algorithm to obtain the tightened formulation as approximation of the convex hull of the lattice set P_I. In this section we propose a new type of strong valid inequalities according to Gomory's cut such that our algorithm can be executed effectively.

Let us focus on the solution of the linear programming problem (1.1) (at $Q = 0$) in this section. The notation x' denotes a current feasible and integral solution. We hope that x' can be replaced by another better feasible and integral solution x'' (with a less objective function value or $f(x'') < f(x')$). On the assumption of the current objective function value 0, it is possible to reach only when we find a feasible solution with a positive objective function value. If we refer x_0 as the increment of the objective function value to (2.1), it implies that x_0 is equal to or greater than zero all the time. Then we present a result of Gomory [6] that gives a complete characterization of the valid inequality.

Theorem 3.1. *For every $\mu \in R^{1+n}$ the inequality*

$$(\lfloor \mu A \rfloor - \mu A)x \leq \lfloor \mu b \rfloor - \mu b \tag{3.1}$$

is valid for (2.4). *Moveover, the slack variable in* (3.1) *is an integer variable.*

Each row could serves as the source of a cut or simply as a source row. We might refer the principal pivot row as a source one in the algorithm (2.5) although it is not only choice. Suppose that a source row is given as follow:

$$x_{j_p} + a_0 x_0 + \sum_{j \in N} a_j x_j = b_p \tag{3.2}$$

where we assume that a_0 is greater than zero. In the current equivalent facet simplex algorithm (2.5), we maintain the coefficient of x_0 at the principal pivot row is positive all the time, before pivot steps. So we hope that there is a less upper bound at the adding cut than at the source row so that its cut become a new principal pivot row.

Our cutting planes are defined as follows. While we can not find a better feasible and integer solution x'' by the choice of the positive integer step size below the critical value $\frac{b_p}{a_0}$, in view of the theorem (3.1) we may generate a valid inequality

$$-a'_{j_p} x_{j_p} - \sum_{j \neq j_p} a'_j x_j \leq -b'_p \tag{3.3}$$

with $\mu = \frac{l}{a_0}$ from the source row (3.2) (l being a suitable integer number), where

$$0 \leq b'_p = \mu b_p - \lfloor \mu b_p \rfloor < 1, \quad 0 \leq a'_j = \mu a_j - \lfloor \mu a_j \rfloor < 1, j = 1, 2, \cdots, n$$

In general, we might choose

$$l = \arg\max\{b'_p | b'_p = \frac{l b_p}{a_0} - \lfloor \frac{l b_p}{a_0} \rfloor, l \in Z\} \tag{3.4}$$

After adding a slack variable s, the x_{j_p} row cut (3.3) is also equivalent to

$$a'_{j_p}a_0x_0 + \sum_{j\neq j_p}(\lfloor \mu a_j \rfloor - \lfloor \mu \rfloor a_j)x_j + s = a'_{j_p}b_p - b'_p \tag{3.5}$$

Then the coefficient of x_0 is greater than zero so as a new upper bound

$$\delta' = \frac{a'_{j_p}b_p - b'_p}{a'_{j_p}a_0} \tag{3.6}$$

to be acquired. This bound is equal to or less than the old upper bound δ.

If b'_p is greater than zero, the strict inequality is satisfied. It results in the feasible region of the linear programming relaxation is compressed because of the decrease of the pivot step size. So the equation (3.5) is a strong valid inequality or cut. If b'_p is equal to zero, the upper bound does not change from the source row (3.2) to the equation (3.5) so that it is almost invalid.

Definition 3.2. A valid inequality (3.3) is degenerate for the source row (3.2) if its right side hand constant is not equal to zero; otherwise, it is called is non-degenerate.

Theorem 3.3. *On the assumption that the coefficient a_0 at the source row* (3.2) *is greater than zero, then there is a strong valid inequality* (3.3) *with a less upper bound at it than at the source row* (3.2) *only if we choose a suitable integer l (or μ) such that b'_p is greater than zero at the inequality* (3.3).

It is possible for us to acquire a non-degenerate strong valid inequality only if $\frac{b_p}{a_0}$ is not integral. In additional, we may repeat to generate another valid inequality by the same way only if we refer the equation (3.5) as a new source row. This new inequality is also called the second stage valid inequality of (3.2). Analogous, we may recursive define the higher stage valid inequality of (3.2).

Example 3.4. From the source row

$$x_3 + \frac{8}{3}x_0 + \frac{5}{3}x_4 = 4$$

we might obtain a first stage cut

$$\frac{5}{3}x_0 + \frac{2}{3}x_4 + s_1 = 2 \quad (\mu = -\frac{3}{8})$$

and a second stage cut

$$\frac{2}{3}x_0 - \frac{1}{3}x_4 + s_2 = 0 \quad (\mu = -\frac{3}{5})$$

Of course, we might obtain other second stage cuts or the higher stage cuts.

Theorem 3.5. *On the assumption of the theorem* (3.3), *the number of the highest stage non-degenerate valid inequality of a source row is finite.*

Proof: From $a_{j_p} = 1$ we know $a'_{j_p} = \frac{l}{a_0} - \lfloor \frac{l}{a_0} \rfloor$. Then the multiply $a'_{j_p} a_0 = l - \lfloor \frac{l}{a_0} \rfloor a_0$ implies that it has the same denominator as the coefficient a_0. In other wards, the numerator of the coefficient of x_0 at the cut is less than at the source row. This means this result correct.□

If $\frac{b_p}{a_0}$ is integral, we must look for another cut by a new approach. Without loss of generality, we assume that there is $a_q > 0$ such that $\frac{b_p}{a_q}$ is fractional. By the same way, we can also generate a cut to decrease the ratio $\frac{b_p}{a_q}$.

4 Modern primal cutting algorithm

Let us now begin an informal description of the major steps of our algorithm for solving the problem (1.1) at $Q = 0$. Starting with an integer feasible solution x' we want to detect an augmenting direction that is applicable at x' or provide a proof that x' is optimal. The goal of the primal method is iteratively improve the solution x' by moving from one vertex of $conv(P_I)$ to another, until no more improvement is possible and optimality is proven. If x' is dual feasible, it is optimal and the method stops for the linear programming relaxation. If not, then a non-basic variable with a negative reduced cost is selected to come into leave the basis. Then a primal simplex is made, leading to a new vector x''. Then a new integer feasible solution in the internal

$$(x', x''] = \{x | x = tx' + (1-t)x'', t \in (0, 1]\}$$

becomes the new x'. If on the other hand there is not any integer solution in the internal $(x', x'']$, then a cutting plane is generated which cuts off x''. Then another attempt is made to pivot from x', leading to a different x'', and so on. The method terminates when x' has proved to be dual feasible.

Instead of x' being dual feasible, we next give a weaker terminal criteria.

Theorem 4.1. *If a solution is primal feasible and integral, then it is optimal if and only if there is a hard upper bound between* $[0, 1)$.
Proof: Let x' denote a primal feasible and integral solution. A hard upper bound between $[0, 1)$ means that there is not another primal feasible and integral solution with a greater objective value than one of x'.□

Algorithm 4.2. Equivalent Facet Cut Algorithm

Step 1. Choose a row index p as the ratio rule (2.6).

Step 2. If there is an integer step size λ below the upper bound $\frac{b_p}{a_{p0}}$ such that all right side constants are integer, then we update the right hand side coefficients $b \leftarrow b - \lambda A_0$; otherwise go to step 4.

Step 3. If there is a hard bound less than 1, STOP; otherwise, go to Step 5.

Step 4. Generate a cutting plane (3.5) from the p-th source row and add it to the bottom of the simplex tableau. Let p denote the index of the adding row.

Step 5. Choose a column index q according to some rule and pivot on a_{pq}.

Step 6. Go to Step 1.

In general, an adding cut (3.5) could make a new principal master row in place of a old source row (3.1). So we always expect that the right hand side constant at (3.5) is as little as possible. We might look for the higher stage cut to achieve this goal. If the right hand side constant is zero, this cut (3.5) is effectively for our algorithm because it can result in the change of the search direction.

Because of the adding cut, we can not assure that the coefficient a_{p0} is greater than zero. If a_{p0} is greater than zero, we might choose a principal master pivot element $a_{pq} < 0$ according to a least index rule, etc., like the algorithm (2.5); otherwise, we must select a suitable pivot element $a_{pq} > 0$ so that all right side hand constants are positive.

Example 4.3. Consider the integer programming problem

$$\begin{array}{lrrrcl} \min & x_0 = & -x_1 & -x_2 & & \\ s.t. & & -4x_1 & +x_2 & \leq & 1 \\ & & 4x_1 & +x_2 & \leq & 5 \\ & & x_1, x_2 & \in Z_+ & & \end{array}$$

Note that the slacks x_3 and x_4 in the two constraints are integer because of the integrality of the data. Now we make x_1 basic in place of x_0, leading the following format:

$$\begin{array}{rrrrcl} x_1 & & -x_0 & +x_2 & = & 1 \\ & x_3 & -4x_0 & +5x_2 & = & 5 \\ & x_4 & +4x_0 & -3x_2 & = & 1 \end{array}$$

The new solution is $(1, 0)$ and the profit has decreased from 0 to -1. Then we make x_2 basic in place of x_4, and acquire the following form:

$$\begin{array}{rrrrcl} x_1 & & +\frac{1}{3}x_0 & +\frac{1}{3}x_4 & = & 1 \\ & x_3 & +\frac{8}{3}x_0 & +\frac{5}{3}x_4 & = & 4 \\ & x_2 & -\frac{4}{3}x_0 & -\frac{1}{3}x_4 & = & 1 \end{array}$$

The new solution is $(1, 1)$ and the profit has decreased from -1 to -2.

The x_3 row can generate a cut

$$\frac{1}{3}x_0 - \frac{2}{3}x_4 + s_1 = 0 \quad (\mu = \frac{3 \times 3}{8})$$

We add it to the simplex tableau and pivot x_4 in place of s_1

$$\begin{array}{rrrrcl} x_1 & & +\frac{1}{2}x_0 & +\frac{1}{2}s_1 & = & 1 \\ & x_3 & +\frac{7}{2}x_0 & +\frac{5}{2}s_4 & = & 4 \\ & x_2 & -\frac{1}{2}x_0 & -\frac{1}{2}s_1 & = & 1 \\ & x_4 & -\frac{1}{2}x_0 & -\frac{3}{2}s_1 & = & 0 \end{array}$$

The x_3 row can generate a cut

$$\frac{5}{2}x_0 + \frac{3}{2}x_4 + s_2 = 2 \quad (\mu = \frac{6 \times 2}{7})$$

This cut implies that there exists a hard upper bound to be less than one and an optimal integer solution $(1,1)^T$ is obtained.

In our algorithm, it is easy to improve a feasible and integral solution because of the use of the objective increment. Perhaps the main task of cuts is to prove whether the last integer feasible solution $(1,1)^T$ is optimal, such as the example (4.3). It is very different from the classical dual cutting plane.

From the theorem (4.1) we easily obtain the following result:

Theorem 4.4. *If the adding cuts are non-degeneracy and the degeneracy does not appear at any steps, the algorithm* (2.5) *converges in a finite number of steps.*

5 Branch and bound

As mentioned above, branch and bound method is a general framework for solving integer and combinatorial problems. The combinatorial part of the problem is solved by a tree search in which LP relaxations (1.2) of the QP problem (1.1) are solved. To avoid complete enumeration, it is possible for us to limit the tree search by using lower and upper bounds on the optimal objective value.

Different from the traditional branch and bound, our method starts with a better integer feasible solution x' and a better upper bound. A branch and bound approach to QP problem would examine the solution x' to the QP relaxation and then split the problem into three new problems, one where $x_j \geq x'_j + 1$, one where $x_j = x'_j$, and where $x_j \leq x'_j - 1$ if $x_j \geq 1$. The solution of (1.1) is contained in the feasible region of one of the three new subproblems and the process can be repeated. In general, we first solve the middle branching. This treatment is often very effectively because it decreases the scale of the origin problem so that the upper bound can be improved more quickly. Then we use classical binary search to solve the following subproblems.

The branch and bound algorithm continues to solve and generate new subproblems performing a tree search the nodes of the tree correspond to QP subproblems. If the solution to any of the QP subproblems that arise process is integer then that point solves the corresponding part of the integer program; If any of the linear programs is infeasible, then the corresponding part of the integer program is also infeasible. The value of the QP subproblem provides a lower bound on the value of the corresponding part of the integer program, and this bound can be used to prune the search space and guide the search. The following observation often considerably reduces the size of the branch and bound tree.

Lemma 5.1. *Let f be continuously differentiable on an open set containing a compact convex set $S \in R^n$, and let x^* be an optimal solution of the problem*

$$\min \ f(x), \quad s.t. \ x \in S \tag{5.1}$$

Then x^* is also optimal for the program

$$\min \ \nabla f(x^*)^T x, \quad s.t.\ x \in S \tag{5.2}$$

Proof: The first order necessary optimality condition $\nabla f(x^*)^T d \geq 0$ for all feasible direction d must hold in x^*. Since S is convex, $d = x - x^*$ is feasible direction for every $x \in S$, and hence $\nabla f(x^*)^T x \geq \nabla f(x^*)^T x^*$ for every $x \in S$. $\square$

The nodes obtained by branching are call *child nodes* and the node from which they are generated is called the *parent node*. A node which has been fully explored is referred to as being *fathomed*. A *pend node* is node which has been generated by branching but has not yet been solved. The branch and bound algorithm searches the tree until no pending nodes remain. It is very important for us how to choose the next pending node. This is resolved in favour of a depth-first-search of the tree with the additional feature that backtracking is done to the most promising node. The aim is quickly to find an integer feasible solution so that the resulting upper bound can be used to fathom nodes whose lower bounds exceed the new upper bound.

6 A example

It is instructive to consider applying the above procedure to the following example:

Example 6.1. Let

$$H = \begin{pmatrix} 6 & -2 & 1 \\ -2 & 2 & -1 \\ 1 & -1 & 5 \end{pmatrix}, \quad g = \begin{pmatrix} -20 \\ -30 \\ -10 \end{pmatrix}$$

and

$$4x_1 + 2x_2 + x_3 \leq 23, \ x_1, x_2, x_3 \in Z_+$$

Starting with the initial primal feasible and integral solution $x' = (0, 0, 0)$ and its linear direction

$$\nabla f(x') = Qx' + c = (-20, -30, -10)^T$$

we obtain the following linear programming relaxation

$$\begin{array}{ll} \min & x_0 = -20x_1 - 30x_2 - 10x_3 \\ s.t. & 4x_1 + 2x_2 + x_3 \leq 23, \quad x_1, x_2, x_3 \geq 0 \end{array}$$

Note that we have solved it at the example (2.6). At the first pivot step, we might choose one of five step sizes

$$20, 40, 60, 80 \ or \ 100$$

so that a new solution is integer number. We find that $f(x') = -33$ is minimum at $x' = (3, 0, 0)^T$ by comparing five different objective values of the problem (1.1). The new form is therefore:

$$\begin{array}{rrrrrrr} x_1 & & -\frac{1}{20}x_0 & +\frac{3}{2}x_2 & +\frac{1}{2}x_3 & = & 3 \\ & x_4 & +\frac{1}{5}x_0 & -4x_2 & -x_3 & = & 11 \end{array}$$

Following we are faced with two choices: one way is to generate a new linear programming relaxation (1.2) at the new solution $(3,0,0)^T$ and another way is to continue solving the origin linear programming relaxation. Here we select the posterior way and, from the x_4 row, generate a cut

$$-x_0 + x_2 + s_1 = 3$$

where the choice of $\mu = \frac{1}{a_{02}} = \frac{2}{3}$ assure that the coefficient of x_2 is positive. Let this cut be added the simplex tableau and pivot two steps, to yield the following form:

$$\begin{array}{rrrrrrr} x_4 & & +\frac{1}{15}x_0 & +\frac{8}{3}x_1 & +\frac{1}{3}x_3 & = & 1 \\ & x_2 & -\frac{1}{30}x_0 & +\frac{2}{3}x_1 & +\frac{1}{3}x_3 & = & 11 \end{array}$$

We choose, $x' = (0,11,0)^T$ with the maximum objective value $f(x') = -209$, one of ten feasible and integral solutions that their all components is zeros expect for x_2 ranging from 2 to 11. Note that the s_1 row is deleted because the variable s_1 becomes a basic variable at the later step. We continue to generate the x_2 cut

$$-\frac{29}{30}x_0 + \frac{1}{3}x_1 - \frac{1}{3}x_3 + s_2 = 5 \quad (\mu = \frac{3}{2})$$

Adding it to the simplex tableau and pivoting three steps, we have

$$\begin{array}{rrrrrrr} x_2 & & +\frac{1}{10}x_0 & -2x_1 & -x_4 & = & 10 \\ & x_3 & +\frac{1}{5}x_0 & +8x_1 & +3x_4 & = & 3 \end{array}$$

The new x' is $(0,10,3)$ and the profit $f(x')$ becomes $-\frac{475}{2}$. Note that the s_2 row is also deleted because the variable s_2 becomes a basic variable at the second step.

Next we repeat to construct the linear programming relaxation (1.2) at the new point x'. We must make a new current equivalent facet at x' in place of the old equivalent facet at the initial point. In order to pivot in x_0, we must choose a row with the right hand side constant zero. If all the right hand side constants are not equal to zero, we must generate a cut, such as

$$-\frac{4}{5}x_0 + 7x_1 + 2x_4 + s_4 = 2 \quad (\mu = -\frac{1}{8})$$

from the x_3 row. Because its right hand side constant 2 is not still equal to zero, from this cut we generate another cut

$$-\frac{1}{5}x_0 + x_1 + s_5 = 0 \quad (\mu = \frac{5}{4})$$

It is also a second stage cut of the x_8 resource row. We add it to the simplex tableau and pivot in x_0 place of s_5 (deleting the x_0 row), to yield the form:

$$\begin{array}{rrrrrrr} x_2 & & -\frac{5}{2}x_1 & -x_4 & -\frac{1}{2}s_5 & = & 10 \\ & x_3 & +9x_1 & +3x_4 & +s_5 & = & 3 \end{array}$$

At the point x', the linear direction

$$\nabla f(x') = Qx' + c = (-37, -11, -5)^T$$

can generate the corresponding current equivalent facet

$$x_0 - \frac{49}{2}x_1 + 2x_4 - \frac{3}{2}s_5 = 0$$

We add it to the simplex tableau and make s_5 basic in place of x_0. After deleting the s_5 row, we have

$$\begin{array}{rccccc} x_2 & -\frac{1}{3}x_0 & +\frac{17}{3}x_1 & -\frac{5}{3}x_4 & = & 10 \\ x_3 & +\frac{2}{3}x_0 & -\frac{22}{3}x_1 & +\frac{13}{3}x_4 & = & 3 \end{array}$$

From the x_3 row we obtain a cut

$$\frac{1}{3}x_0 - \frac{11}{3}x_1 + \frac{5}{3}x_4 + s_6 = 1 \quad (\mu = \frac{3}{2})$$

Adding it to the simplex tableau and pivoting in x_1, we get

$$\begin{array}{rccccc} x_2 & +\frac{2}{11}x_0 & +\frac{10}{11}x_4 & +\frac{17}{11}s_6 & = & 9 \\ x_3 & & +x_4 & -2s_6 & = & 1 \\ x_1 & -\frac{1}{11}x_0 & -\frac{5}{11}x_4 & -\frac{3}{11}s_6 & = & 1 \end{array}$$

The new x' is $(1, 9, 1)$ and the profit $f(x')$ is $-\frac{479}{2}$.

Finally we consider the use of the branch and bound algorithm starting with $x' = (1, 9, 1)^T$ and a upper bound $\bar{f} = -239.5$. Because of its linear direction

$$\nabla f(x') = (-31, -15, -13)^T$$

we might partition P based on, x_1, an important of the integer variables. This is done by introducing three subproblems obtained from (1.1) by adding following three simple bound sets

$$S^1 = \{x \in R^n | x_1 \leq 0\}, \quad S^2 = \{x \in R^n | x_1 = 1\}, \quad S^3 = \{x \in R^n | x_1 \geq 2\}$$

The solution of (1.1) is obtained in the feasible region of one of the three new subproblems.

Following we solve their nonlinear programming relaxations

$$\min f(x) \quad s.t.\ x \in P \cap S^i, \quad i = 1, 2, 3 \tag{6.1}$$

by any nonlinear optimization method, such as the active set method, etc. We list in turn three optimal solutions and their objective values

solution	*objective value*	*constraint*
$(0, 10.1154, 2.7692)^T$	-237.6731	$P \cap S^1$
$(1, 8.5769, 1.8462)^T$	-242.8269	$P \cap S^2$
$(2, 7.0385, 0.9231)^T$	-222.5192	$P \cap S^3$

Of course, we solve the middle branching and abandon other two branchings. Following the branch and bound is classical and we acquired two optimal solutions of the subproblems, $(1, 8, 3)^T$ with the objective value 237.5 and $(1, 9, 1)^T$ with the objective value 239.5. The postern is optimal to (1.1) obviously.

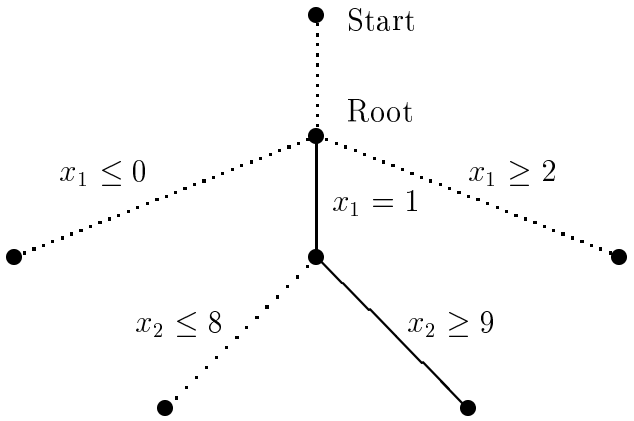

Fig1. The branch and bound path

7 Further remark

We have described a novel branch-and-cut algorithm for the quadratic integer programming problems with convex objective function and linear constraints. This first new feature of this algorithm is a method for obtaining a better feasible and integral solution of IQP problem from its linear programming relaxations. It is very critical to solve IQP problems by the branch and bound. The second new feature of this algorithm is an extraordinary branching procedure, which allows the algorithm to avoid a lot of complicated works of solving subproblems. In additional, we try to improve the effective of our algorithm by setting up a new cutting plane rule.

It seem useless to improve a feasible and integral solution by the use of the classical primal cutting plane algorithm, elegant though it is. While we performed with it to the linear programming relaxation, we have found that a huge number of cuts is generated, with coefficients growing in an exponential manner, so that the LP solver encounters precision problems and crashes, sometimes before even a single non-degenerate pivot has been made. This happens even for relatively small instances. Another deadly disadvantage is that we can not choose a suitable feasible and integral solution neatly like our algorithm. Perhaps we can not find any better feasible solution for IQP problems by the primal cutting plane algorithm.

It is inability to generate a feasible and integral solution by the use of the dual cutting algorithm. From the example (6.1) we might obtain the optimal solution $x^* = (0.7523, 8.5979, 2.0738)^T$ of the nonlinear programming relaxation. The gradient of f at x^* is

$$\nabla f(x^*) = Qx^* + c = (-31.3272, -15.6636, -7.8318)^T = -7.8318(4, 2, 1)^T$$

If we solve the linear programming relaxation (1.2) at x^* by the dual cutting plane algorithm, we can not acquire any useful information.

Perhaps our algorithm offers a good framework for solving IQP problems, or more complex $MINLP$ problems. The main advantage of our algorithm is that it

is convenience to find a better feasible and integral point by the solution of a series of linear programming relaxations. It will be our future work how to apply it to $MINLP$ problems.

References

[1] Ben-Israel A, Charnes A, (1962) On some problems of diophantine programming. Cahiers du Centre d'Études de Recherche Opérationelle 4:215-280.

[2] Capprara A, Fischetti M, (1997) Branch-and-cut algorithms. In Dell's Amico M, Maffioli F, Martello S(eds.) Annotated Bibliographices in Combinatorial Optimization. Wiley, New York, pp.45-64.

[3] R.Fletcher, (1995) Numberical experience with lower bound for MIQP branch and bound, SIAM Journal on optimization, 8(1998), 604-616.

[4] A.H. Land and A.G. Doig, (1960) An auomatic method for solving discrete programming problems, Econometrica, 28:494-520.

[5] Glover F, (1968)A new foundation for a simplified primal integer programming algorithm. Operations Research 16:727-740.

[6] Gomory R, (1958) Outline of an algorithm for integer solution programs. Bulletin of the AMS 64:275-278.

[7] Gomory R, (1963) An all integer programming algorithm. In Muth JF, Thompson GI(eds.) Industrial Scheduling, Prentice-Hall, New York.

[8] Nemhauser G.L, Wolsey LA (1988) Integer and combinationrial optimization. Wiley, New York.

[9] Padberg M.W, Rinalfi G(1991) A branch-and-cut algorithm for the resolution of large-scale symmetric travelling salesman problem. SIAM Review 33:60-100.

[10] Padberg M.W, Classical cuts for mixed-integer programming and branch-and-cut. math meth Oper Res 53:172-203(2001).

[11] Young RD (1965) A primal (all-integer) integer peogramming algorithm. Journal of Research of National Bureau of Standards 69B:213-250.

[12] Young RD (1968) A simplified primal (all-integer) integer programming algorithm. Operations Research 16:750-782.

[13] Yan Zizong, Fei Pusheng, The current equivalent facet algorithm for linear programming problems(In Chinese), http://www.paper.edu.cn, Mathematica Numerica Sinica, Vol.26, No.4(2004), 437-445.

[14] Yan Zizong, Fei Pusheng and Wang Xiaoli, The basic theory of the current equivalent facet algorithm for linear programming problems(In Chinese), Journal of Yangtze university (Natural Science), Vol.3, No.3(2004), 1-9.