TR/04/93 June 1993

CUTTING PLANE METHODS FOR GENERAL INTEGER PROGRAMMING

Fatimah Abdul Hamid, G. Mitra
K. Darby-Dowman

ABSTRACT

Integer programming (IP) problems are difficult to solve due to the integer restrictions imposed on them. A technique for solving these problems is the cutting plane method. In this method, linear constraints are added to the associated linear programming (LP) problem until an integer optimal solution is found. These constraints cut off part of the LP solution space but do not eliminate any feasible integer solution. In this report algorithms for solving IP due to Gomory and to Dantzig are presented. Two other cutting plane approaches and two extensions to Gomory's algorithm are also discussed. Although these methods are mathematically elegant they are known to have slow convergence and an explosive storage requirement. As a result cutting planes are generally not computationally successful.

Contents

1 - Introduction 1

2 - Dual cutting plane algorithms 2

2.1 Gomory's fractional algorithm 2

2.2 Gomory's all-integer algorithm 9

2.3 Dantzig's cut 16

2.4 Gomory's mixed integer algorithm 17

3 - Other cutting plane approaches 19

3.1 Primal cutting plane algorithm 19

3.2 Primal-Dual cutting plane algorithm 20

4 - Extensions of cutting plane algorithm 21

5 - Conclusions 23

6 - References 25

1 - Introduction

The concept of adding linear inequality constraints (cutting planes) to the linear programming(LP) problemwas first introduced by Dantzig, Fulkerson and Johnson (1954) in their work on the traveling salesman problem. These constraints effectively cut off parts of the convex feasible region which do not contain any feasible integer points. Later Markowitz and Manne (1957) suggested a similar approach.

In1958, Gomory (1963a) provided the first cutting plane algorithm that systematically generated cuts that can be applied to integer programming (IP) problems. In 1960, he produced a second cutting plane algorithm for the IP problems which maintains all-integer tableaux (Gomory 1963b). Gomory (1966) then extended his first cutting plane algorithm to deal with mixed integer programming (MIP) problems. Another researcher in this area is Dantzig (1959) who proposed a cut that did not lead to a convergent algorithm. All these algorithms are classified as dual cutting plane algorithms as they maintained dual feasibility when applied to the optimal solution of the LP-relaxation of the IP problems.

Glover (1965) and Young (1968) introduced cutting plane algorithms which maintain linear problems that are primal feasible. Since primal feasible integer solutions are successively produced, the technique is referred to as a primal cutting plane algorithm. In the 1980's a hybrid dual and primal algorithm was proposed by Ghandforoush and Austin (1981) to solve all-integer IP. This algorithm is a modification of a procedure introduced by Glover (1967).

The basic idea of the cutting plane method is very simple. The value of the optimal solution to the LP-relaxation (i.e., the IP problem without the integer restrictions) is an upper bound to the value of the IP objective function. If this solution is integer feasible then it is the optimal solution to IP. If the optimal solution of the LP-relaxation is fractional, then we generate a valid inequality that would cut off part of the LP solution space, including the current optimal basic solution but without cutting off any feasible integer solution. This procedure is repeated by adding cuts to the current LP until an integer solution is obtained.

2 - Dual cutting plane algorithms

Dual cutting plane algorithms utilize the dual simplex method to maintain dual feasibility. There are four algorithms classified as dual, three due to Gomory and one due to Dantzig.

2.1 Gomory's fractional algorithm

The first cutting plane algorithm introduced by Gomory (1963a) to solve pure IP (PIP) problems is known as Gomory*'s fractional cuts* because all the nonzero coefficients of the generated cuts are less than one. It is sometimes referred to as *the method of integer form.* This is also the first cutting plane algorithm that was proven to be finitely convergent.

Consider the PIP problem:

$$
\begin{aligned}
&\textit{maximise} \quad && \sum_{j=1}^{n} c_j x_j \\
&\textit{subject to} \quad && \sum_{j=1}^{n} a_{ij} x_j \le b_i, \qquad i = -1,...,m \qquad (1) \\
& && x_j \ge 0 \quad \textit{and} \quad \textit{integer}, \quad j = 1,...n.
\end{aligned}
$$

where the data are required to be integral (constraints are cleared of fractions) to ensure that the value of objective function and the slack variables are integral for any integer solution. Gomory's fractional cuts can be derived as follows. Suppose that we have an optimal solution of the LP-relaxation which is noninteger. Consider the *pth* row of the noninteger solution. This can be expressed as

$$x_p = \bar{a}_{p0} - \sum_{jel} \bar{a}_{pj} x_i \tag{2}$$

where J is the set of nonbasic variables, which can be written as

$$x_p = [\bar{a}_{p0}] + f_{p0} - \sum_{juJ} [\bar{a}_{pj}] x_j - \sum_{jeJ} f_{pj} x_j \tag{3}$$

Where $\bar{a}_{pj} = [\bar{a}_{pj}] + f_{pj,} 0 \le f_{pj} < 1$ and [y] denotes the largest less then or equal to y. This can be rearrange as

$$x_p = [\bar{a}_{p0}] + \sum_{juJ} [\bar{a}_{pj}] x_j = f_{p0} - \sum_{jeJ} f_{pj} x_j \tag{4}$$

Given that $f_{pj\ge 0}$ and $x_j \ge 0, j \in J$,

$$\sum f_{pJ} x_j \ge 0 \tag{5}$$

Thus

$$f_{p0} - \sum f_{pJ} x_j \le f_{p0} \tag{6}$$

Since all the variables are required to be integer, it follows that the left hand side of the equation (4) will be integer, hence the right hand side must also be integer. Thus the left hand side of (6) is required to be integer and therefore

$$f_{p0} - \sum_{jeJ} f_{pJ} x_j \le 0 \tag{7}$$

and can be written in the form

$$s_p = -f_{p0} + \sum_{p0} f_{pJ} x_j \qquad (8)$$

where s_p is a nonnegative integer slack variable. This is Gomory's cut for solving PIP problems. Adding the new constraint (8) to the optimal tableau of the related LP problem results in a primal infeasible problem which can be solved using the dual simplex method.

The basic algorithm for Gomory's fractional cuts is as follows:

Step 1: Initialisation

Solve the LP-relaxation. If it is infeasible, so is the IP problem-terminate. Otherwise go to step 2.

Step 2: Optimality test

If the optimal solution to LP-relaxation is integer feasible then it is optimal to IP - terminate. Otherwise go to step 3.

Step 3: Cutting and pivoting

Choose a row r with $f_{r0} > 0$ and add to the bottom of the simplex tableau, the fractional cut or constraint (8).

Step 4: Reoptimization

Reoptimize the new LP using the dual simplex method. If the solution to the new LP problem is infeasible, the IP has no solution - terminate. If the new optimum is integer feasible, the IP is solved - terminate. Otherwise go to step 2.

It is a standard practice to drop each added inequality immediately after its slack variable re-enters the set of basic variables. That is, when a new inequality is introduced it is used as the pivot row so its slack becomes nonbasic. When this slack variable returns to the basic set, it and its defining row are omitted. The tableau contains exactly n nonbasic

variables always. Hence if this rule is followed, there will never more than *n* cuts at any stage of the algorithm.

For the algorithm to converge to the optimal solution after a finite number of iterations, it is necessary to ensure that some lower bound *M* is known for the value of the objective function x_0 Selection of source row is also important to support a finite algorithm. The intention when selecting the source row is to produce an inequality in which the ratio f_0 / f_j is as large as possible since the larger the value of the ratio, the stronger the cut. Geometrically, the ratio f_0 / f_J can be regarded as the intersection point of the cut (8) at its limit (with zero valued slack variable) with each x_j axis. A classical rule suggested by Salkin and Mathur (1989) is to generate the inequality from the row with the largest fractional component.

Jenkins and Peters (1987) listed a variety of Gomory's cuts which are derived according to the selection of the source row. In their report they recommended the cut known as the *Maximum ratio* to solve general IP problems. This is the Gomory's fractional cut with the source row as the row that has

$$\max_i \frac{\left(\bar{b}_j - [\bar{b}_j]\right)}{\sum_j \left(\bar{a}_{ij} - [\bar{a}_{ij}]\right)}. \qquad (9)$$

An illustration of Gomory's fractional algorithm:

Example 1:

$$\text{maximize } z = 3x_1 + 4x_2$$

$$\text{subject to}$$

$$2x_1 + 5x_2 \le 15$$

$$2x_1 + 2x_2 \le 5$$

$$x_i \ge 0 \text{ and integer}$$

$$\text{for } i = 1,2.$$

Solution:

The optimal solution of the LP relaxation is

$$x_1 = 3.92857 \quad x_2 = 1.42857, \quad Z_{LP} = 17.5.$$

In this example the selection of source row is done by choosing the row with the largest fraction f_{r0}. It takes three cuts to solve this IP problem. In the tableuax, $(\leftarrow)$ indicates the leaving variable and $(\downarrow)$, the entering variable. The double underlined element denotes the pivot element.

Table 1.1: The optimal tableau for the LP relaxation

		1	$-x_4$ ↓	$-x_3$
	x_0	35/2	½	1
source	x_1	55/14	5/14	1/7
	x_2	10/7	-1/7	1/7
←	S_1	-13/14	-5/14	-1/7

$S_1 = -13/14+5/14x_4+1/7x_3$ In terms of structural variables, $x_1 \leq 3$.

Table 1.2:

		1	$-S_4$ ↓	$-x_3$
	x_0	81/5	7/5	4/5
	x_1	3	1	0
Source	x_2	9/5	-2/5	1/5
	x_4	13/5	-14/5	2/5
←	S_2	-4/5	-3/5	-1/5

Table 1.3:

		1	$-S_2$	$-x_3$ ↓
	x_0	43/3	7/3	1/3
source	x_1	5/3	5/3	-1/3
	x_2	7/3	-2/3	1/3
	S_1	4/3	-5/3	1/3
←	S_3	-2/3	-2/3	-2/3

$S_3 = -2/3+2/3S_2x_4+2/3x_3$ In terms of structural variables, $x_{1+}x_2 \leq 6$.

Since s_1 enters the basis we can delete s_1 and the corresponding row from the tableau.

Table 1.4

	1	$-S_2$	$-S_3$
x_0	14	2	2
x_1	2	2	-2
x_2	2	-1	2
x_4	5	-6	-2
x_3	1	1	-3/2

This gives an optimal solution for IP since all the variables x_i for i=1,2 are integers i.e., x_1=2, x_2=2 and the objective function value is 14.

Graphically we can illustrate this example as follows:

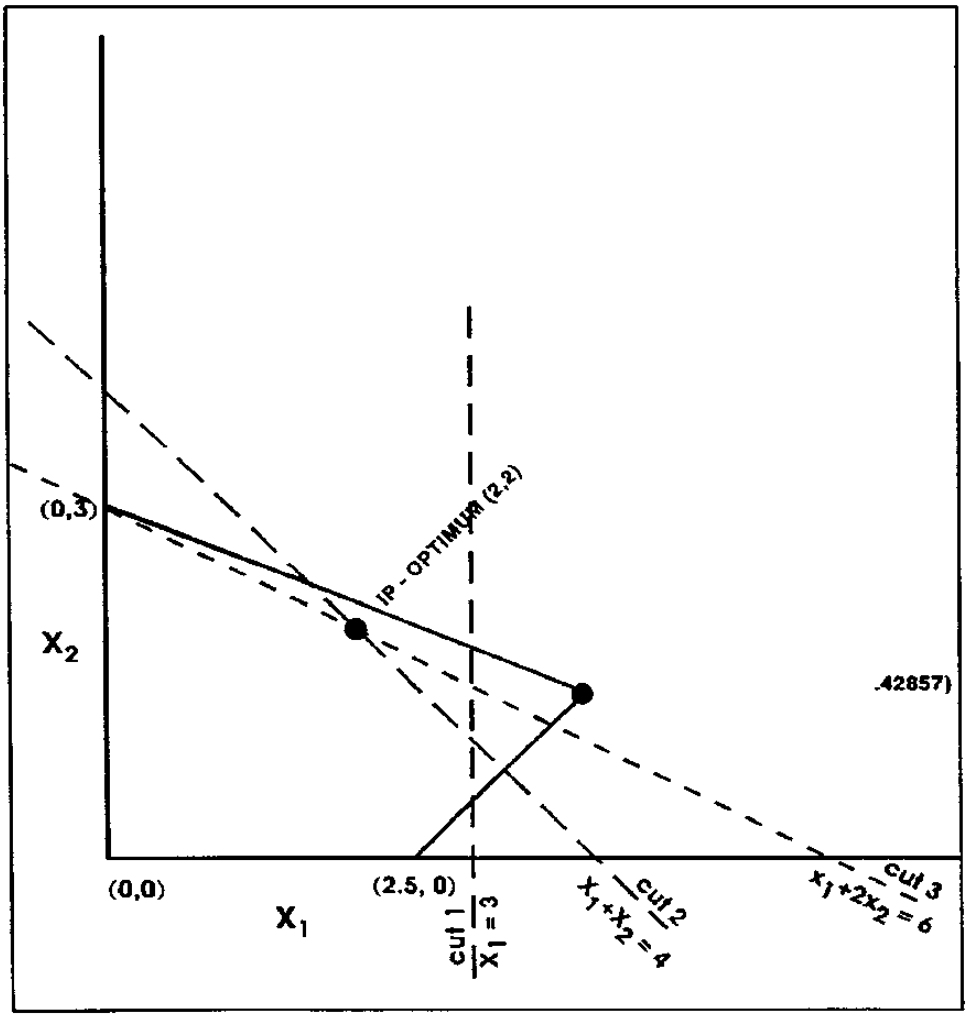

Figure 1

2.2 Gomory's all-integer algorithm

Gomory's dual fractional algorithm suffer from round-off error and has not been successful in practice for solving large problems. Gomory (1963b) developed a second cutting plane algorithm which is not susceptible to round-off errors. This algorithm is a direct extension of the classical dual simplex method and the only difference is that the pivot row in the all-integer algorithm is generated at each iteration and ensures a -1 pivot. Since the technique maintains all-integer tableaux, it is referred as dual all-integer algorithm.

The difference between this algorithm and the fractional algorithm is that the all-integer algorithm is applied to the initial tableau. Furthermore, there is no optimization, generation of constraints, reoptimization, etc. In this method, inequalities are generated at each iteration starting with the very first. Each of these constraints is used as the pivot row, and is constructed so that it has integral coefficients and the pivot is -1.

Consider the all-integer programming problem:

$$
\begin{aligned}
&\textit{maximise} \sum_{j=1}^{n} c_j x_j \\
&\textit{subject to} \sum_{j=1}^{n} a_{ij} x_j \le b_i, \qquad i = 1,...,m. \\
&x_j \ge 0 \ \textit{and integer}, \ j = 1,...,n. \\
&c_j, a_{ij}, b_i \quad \textit{integer for all } i, j
\end{aligned}
\tag{10}
$$

A general dual all-integer algorithm is as follows:

Step 1 :

Start with an all-integer simplex tableau which contains a dual feasible solution. Go to step 2.

Step 2:

Select a primal infeasible row $p\left(\text{i.e.}, \bar{a}_{p0} < 0, p \neq 0\right)$. If none exists, the tableau exhibits the optimal integer solution - terminate. Otherwise go to step 3.

Step 3:

Designate the pivot column q to be the lexicographically smallest among those having $\bar{a}_{pj} < 0$. If none exists $\left(\text{i.e.}, \bar{a}_{pj} \geq 0 \text{ for } j = 1,\ldots,n\right)$ there is no integer feasible solution - terminate. Go to step 4.

Step 4:

Derive an all-integer inequality for row p which is not satisfied at the current primal solution. (Its slack will be negative). It must also have a -1 coefficient in column q. Append it to the bottom of the tableau and label it the pivot row. Perform a dual simplex pivot operation and return to step 2.

Note that a vector A is *lexicographically positive* (denoted as $A > 0$) if its first nonzero element is positive. A vector A is said to be *lexicographically greater* than a vector B if A-$B > 0$. If $-A > 0$, A is called *lexicographically negative* (denoted as A-< 0) and if A-B•< 0, A is *lexicographically smaller* than B.

To begin the calculations, it is necessary to have all-integer dual feasible tableau with $a_{0j} > 0$ for all $j \in J$ where J is the set of nonbasic variables. If the initial solution is not dual feasible, that is, at least one $a_{0j} < 0$, then a redundant constraint of the form

$$s = p - \sum_{j \in J} x_j \qquad (11)$$

where p is a suitably large integer, is added.

Let the all-integer cut generated on the pth row be as follows:

$$s_p = \left[\frac{\bar{a}_{po}}{\lambda}\right] - \sum_{jeJ}\left[\frac{\bar{a}_{pj}}{\lambda}\right]x_j \qquad (12)$$

where s_p is a nonnegative integer slack variable and λ is a positive number found by the following rules:

Step 1 : select a column

With p as the generating row, le q be the lexicographically smallest column among those having $\bar{a}_{pj} < 0$ for ally $j \in J$.

Step 2: choose h_j

Let $h_q = 1$, and for every $j \geq 1(j \neq q)$ with $\bar{a}_{pj} < 0,$, let h_j be the largest integer satisfying

$$\left(\frac{1}{h_j}\right)p_j > p_q \qquad j \in J \qquad (13)$$

where P_J denotes the column vector of elements $\bar{a}_{ij}, i = \{0,1,2,...m\}$. The symbol '>' denotes lexicographically greater than.

Step 3: choose λ_j

For each $\bar{a}_{pj} < 0(j \geq 1),$ set

$$\lambda = \max_{jeJ}\left(\frac{\bar{a}_{pj}}{h_j}\right) \qquad (14)$$

Note that λ_j is not necessarily an integer.

Richard (1967) and Salkin and Mathur (1989) suggested a few rules for selecting an eligible row (the row with $\bar{a}_{r0} < 0, r \neq 0$) as the source row. Some of these include:

1. Select the first eligible row.

2. Select the first eligible row at the current iteration, the second eligible row at the next iteration, etc.. That is, select an eligible row by a cyclic process.

3. Randomly select an eligible row.

4. Select an eligible row which produces the lexicographically largest pivot column. This attempts to change the $\bar{a}_{r0}$ column as much as possible.

5. Select an eligible row containing the least number of negative elements. This attempts to avoid, or at least limit the degeneracy.

The disadvantage of this algorithm is the presence of dual-degenerate iterations in which it moves from one lattice point to another without improving the value of the objective function. It is also noted by Parker and Rardin (1988) that this dual all-integer algorithm is weaker than the fractional algorithm.

An illustration of all-integer algorithm:

Example 2:

$$
\begin{aligned}
\textit{maximise} \quad & x_0 = 4x_1 + 5x_2 + x_3 \\
\textit{subject to} \quad & 3x_1 + 2x_2 + 0x_3 \leq 10 \\
& x_1 + 4x_2 + 0x_3 \leq 11 \\
& 3x_1 + 3x_2 + x_3 \leq 13 \\
& x_j \geq 0 \textit{ and integer}
\end{aligned}
$$

Solution:

Table 2.1:

	1	$-X_1$	$-X_2$	$-X_3$
$-X_0$	10	-4	-5	-1
$-X_4$	10	3	2	0
$-X_5$	11	1	4	0
$-X_6$	13	3	3	1

The initial solution is not dual feasible, therefore a redundant constraint, $s = 15 - x_1 - x_2 - x_3$

Added

Table 2.2:

	1	$-X_1$	$-X_2$	$-X_3$
$-X_0$	0	-4	-5	-1
$\leftarrow S$	15	1	1	1
$-X_4$	10	3	2	0
$-X_5$	11	1	4	0
$-X_6$	13	3	3	1

By pivoting on this row and x_2 column we get a dual feasible initial solution.

Table 2.3:

		1	$-\overset{\downarrow}{X}$	$-S$	$-X_3$
	X_0	75	1	5	4
	X_2	15	1	1	1
	X_4	-20	1	-2	-2
Source	X_5	-49	-3	-4	-4
	X_6	-32	0	-3	-2
	$\leftarrow S_1$	-17	-1	-2	-2

P is x_5 and q is x_1

$h_{x1}=1, h_s=4$ and $h_{x3}=3$,

$h_{x3}=3$, $\lambda_s=1$ and $\lambda_{x3}=1\frac{1}{3}$,

$\lambda_{max}=3$

$s_1 = -17 + x_1 + 2s + 2x_3$

Table 2.4:

		1	$-S_1$	$-S$	$-\overset{\downarrow}{X}_3$
	X_0	58	1	3	2
	X_2	-2	1	-1	-1
Source	X_4	-37	1	-4	-4
	X_5	2	-3	2	2
	X_6	-32	0	-3	-2
	X_1	17	-1	2	2
	$\leftarrow S_2$	-10	0	-1	-1

P is X_4 and q is X_3
$h_{x3}=1$, $h_s=1$,
$\lambda_{x3=4}$, $\lambda_s=4\frac{1}{3}$
$\lambda_{max}=4$
$S_2=-10+0S_1+X_3$

Table 2.5:

		1	$-\overset{\downarrow}{S}_1$	$-S$	$-X_3$
	X_0	58	1	1	2
	X_2	8	1	0	-1
	X_4	3	1	0	-4
Source	X_5	-18	-3	0	2
	X_6	-12	0	-1	-2
	X_1	-3	-1	0	2
	X_3	10	0	1	-1
	$\leftarrow S_3$	-6	-1	0	0

P is X_5 and q is S_1
$h_{s1}=1$,
$\lambda_{max}=4$
$S_{3=}-6+S_1+0S+0S_2$

Table 2.6:

		1	$-S$	$-\overset{\downarrow}{S}$	$-X_3$
	X_0	32	1	1	2
	X_2	2	1	0	-1
	X_4	-3	1	0	-4
	X_5	0	-3	0	2
Source	X_6	-12	0	-1	-2
	X_1	-3	-1	0	2
	X_3	10	0	1	-1
	$\leftarrow S_4$	-6	0	-1	-1

P is X_5 and q is S
$h_{s1}=1$,$h_{s2}=1$
$\lambda_{max}=2$
$S_{4=}-6+0S_3+0S_3+S_2$

Table 2.7:

		1	$-S_3$	$-\overset{\downarrow}{S}_4$	$-S_2$
	X_0	26	1	1	1
	X_2	2	1	0	-1
	X_4	-3	1	0	-4
	X_5	0	-3	0	2
Source	X_6	-6	0	-1	-2
	X_1	3	-1	0	2
	X_3	4	0	1	-2
	S	6	0	-1	1
	$\leftarrow S_5$	-6	0	-1	-1

P is X_6 and q is S_4
$h_{s4}=1, h_{s2}=1$
$\lambda_{s4}=1, \lambda_{s2}=1$
$\lambda_{max}=1$
$S_5=-6+0S_3+S_4+S_2$

Table 2.8:

		1	$-S_3$	$-S_5$	$-\overset{\downarrow}{S}_2$
	X_0	20	1	1	0
	X_2	2	1	0	-1
Source	X_4	-3	1	0	-4
	X_5	0	-3	0	2
	X_6	0	0	-1	0
	X_1	3	-1	0	2
	X_3	-2	0	1	-3
	S_4	6	0	-1	1
	$\leftarrow S_6$	-1	0	0	-1

P is X_4 and q is S_2
$h_{s2}=1$
$\lambda_{s2}=4$
$\lambda_{max}=4$
$S_6=-1+0S_3+0S_5+S_2$

Table 2.9:

		1	$-\overset{\downarrow}{S}_3$	$-S_5$	$-S_2$
	X_0	20	1	1	0
	X_2	3	1	0	-1
Source	X_4	1	1	0	-4
	X_5	-2	-3	0	2
	X_6	0	0	-1	0
	X_1	1	-1	0	2
	X_3	1	0	1	-3
	S_2	1	0	0	-1
	$\leftarrow S_7$	-1	-1	0	0

P is X_4 and q is S_2
$h_{s3}=1$
$\lambda_{s3}=3$
$\lambda_{max}=3$
$S_7=-1+S_3+0S_5+0S_2$

Table 2.10:

	1	$-S_7$	$-S_5$	$-S_2$
X_0	19	1	1	0
X_2	2	1	0	-1
X_4	0	1	0	-4
X_5	1	-3	0	2
X_6	0	0	-1	0
X_1	2	-1	0	2
X_3	1	0	1	-3
S_3	1	-1	0	0

This tableau gives an IP optimal solution With $x_1=2$, $x_2=2$, $x_3=1$ And $x_0=19$.

(Whenever the slack variable of the added inequality becomes basic it and its corresponding row is omitted from the tableau.)

2.3 Dantzig's cut

Consider the PIP problem (1) where the data are required to be integral. Suppose an optimal solution to the LP-relaxation of the associated IP problem is given. If this solution is not integer feasible that is, if at least one of the basic variables is noninteger, then one or more of the nonbasic variables must be nonnegative in any IP optimal solution. Accordingly, the sum of the nonbasic variables must be at least unity in such a solution. Based on this argument Dantzig (1959) proposed using the cut

$$\sum_{j\varepsilon\varepsilon} x_j \geq 1 \qquad (15)$$

where J denotes the set of the current nonbasic variables.
Dantzig's cut follow the same algorithm as Gomory's fractional cut for solving PIP (with all constraints to be integral). Dantzig did not prove this method converges. However Gomory and Hoffman (1963) claimed that Dantzig cuts do not provide a finite algorithm.

Bowman and Nemhauser (1970) showed that a modified Dantzig cut yields a finite dual simplex algorithm for the PIP problem. Let p be the topmost row with x_p as a

noninteger basic variable. Then the modified stronger cut is as follows:

$$\sum_{j \varepsilon J^*} x_j \geq 1 \qquad (16)$$

where J^* = {j|j ∈ J and $\bar{a}_{pj}$ is noninteger}.

2.4 Gomory's mixed integer algorithm

Gomory's fractional cut algorithm was amended to deal with MIP problems. Garfinkel and Nemhauser (1972) asserted that to ensure that this algorithm is convergent it is required that the value of the objective function x_0 to be integer. If x_0 is not integer constrained then the algorithm may not always converge [see e.g., Garfinkel and Nemhauser (1972)].

Consider the MIP problem:

$$\begin{aligned} &\textit{maximize} \quad \sum_{j=1}^{n} c_j x_j \\ &\textit{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \qquad i = 1,...,m. \qquad (17) \\ &\qquad x_j \geq 0, \qquad j \in k \\ &\qquad x_j \leq 0, \qquad j \in k' \end{aligned}$$

where K and K' are the set of integer variables and the set of continuous variables respectively.

Let the cut generated from the *p*th row be as follows

$$s_p = -f_{po} + \sum_{j \in J} g_{pj} x_j \tag{18}$$

where

$$g_{pj} = \begin{cases} \bar{a}_{pj} & \text{if } \bar{a}_{pj} \geq 0 \text{ and } x_j \text{ is a continuous variable} \\ \dfrac{f_{p0}}{f_{p0} - 1} \bar{a}_{pj} & \text{if } \bar{a}_{pj} < 0 \text{ and } x_j \text{ is a continuous variable} \\ f_{pj} & \text{if } f_{pj} \leq f_{p0} \text{ and } x_j \text{ is an integer variable} \\ \dfrac{f_{p0}}{1 - f_{p0}} (1 - f_{pj}) & \text{if } f_{pj} > f_{fp0} \text{ and } x_j \text{ is an integer variable} \end{cases} \tag{19}$$

and $f_{pj} = \bar{a}_{pj} - [\bar{a}_{pj}]$ for j=0,l,...,n. The derivation of the expression g_{pj} can be found eg. in Salkin and Mathur (1989).

Since the slack variable s_p is not necessarily an integer linear combination of the original non-basic variables, it will not always take an integer value in every mixed integer solutions. Salkin and Mathur (1989) suggested that the slack variable and its corresponding row to be dropped after it re-enters the basis, as these constraints are not eligible to generate inequalities.

3 - Other cutting plane approaches

Two other cutting plane approaches for solving PIP problems are primal all-integer and primal-dual all-integer algorithm. The first finite primal algorithm was proposed by Young (1965) and the first hybrid algorithm was proposed by Glover (1967).

3.1 Primal cutting plane algorithm

The second all-integer cutting plane algorithm is the primal algorithm. It starts with a primal feasible but dual infeasible tableau. It maintains primal feasibility while moving towards dual feasibility and thus optimality. The cuts are generated in such a way that the pivot element is always "+1" thus maintaining integer tableaux.

The best known primal algorithm is the simplified primal algorithm due to Young (1968) which converges to optimality in a finite but possibly very large number of iterations. It uses the same cut form as Gomory's all-integer cut (12) except that $\bar{a}_{r0}>0$. A limitation of this algorithm is the possibility of long sequences of degenerate pivots during which the algorithm makes no progress towards the solution. In order to eliminate these long sequences Arnold and Bellmore (1974) modified an algorithm due to Glover. They examined the structure of these sequences so that both the termination of a sequence and the tableau at that point can be predicted. Then they proposed an algorithm that exploits this structure by performing the iterations of the sequence implicitly rather than explicitly.

It is known that primal algorithms are more likely to be slower than the Gomory's fractional algorithm (see eg., Nemhauser and Wolsey (1988)). Because of poor computational experience, little recent research on this approach has taken place.

3.2 Primal-Dual cutting plane algorithm

This approach is a hybrid, or primal-dual technique, in which the tableaux are permitted to be both primal-infeasible and dual-infeasible. Glover (1967) developed an algorithm called Pseudo-primal-dual to deal with solving an all-integer IP problem. This algorithm solves all-integer IP problem in two stages, systematically violating and restoring dual feasibility while maintaining an all-integer tableau.

It begins with a dual feasible tableau and a cut is generated from the row with the largest number of negative elements. The dual simplex method is then used to solve the new problem. At this stage, provided dual feasibility is not destroyed, the process is repeated until an optimal solution is reached. Otherwise dual feasibility is restored by a sequence of "pseudo-primal" pivot steps using the column that is lexicographically most negative when divided by the corresponding coefficient in the source row and the pivot row from the dual stage.

Ghandforoush and Austin (1981) proposed an algorithm called Constructive primal-dual algorithm (CPDA). This is a modification and improvement of Glover's method. This algorithm solves all-integer IP problems by starting primal-feasible and dual-feasible. CPDA avoids degenerate iterations by developing a cut which deliberately moves into the infeasible region and then attempts to return to primal feasiblity at a better solution point than the one for which it departed.

Later Ghandforoush (1983) proposed an improvement to CPDA by incoporating an advanced primal (feasible) start algorithm introduced by Austin and Hanna (1983). Its convergence to optimality in a finite number of iterations has not been proven. In 1985 Austin and Ghandforoush proposed another technique by the name Surrogate cutting plane algorithm (SCPA) for solving all-integer IP. At that time they found that (SCPA) was a promising approach for solving small problems. However, it appears that no computational experience of applying the method to large IP problems has been reported.

4 - Extensions of cutting plane algorithm

Since Gomory introduced his first cutting plane algorithm, many other cutting plane methods have been developed to improve the cut. Two such methods are the composite cut and the deep cut.

Martin's Accelerated Euclidean Algorithm

Gomory's fractional cuts algorithm is proven to be finite although convergence is often found to be very slow. To overcome this problem, Martin (1963) proposed an algorithm which is called Accelerated Euclidean Algorithm. He aimed to accelerate convergence towards optimality while maintaining dual feasiblity and primal feasiblity. This method is an extension of Gomory's fractional algorithm where a series of Gomory cuts were generated from the same row until the pivot element becomes integer. These cuts are referred to as *composite cuts.*

The basic technique consists of solving the LP-relaxation, and then introducing a composite cut and resolving the LP. The process repeated until an optimal LP tableau with an integer primal solution is found or reoptimization is not possible.

Transforming the optimal tableau to one with an integer primal solution is accomplished by a composite cut which can be done as follows:

Step 1:

Select a row *v* with a noninteger element. Generate a Gomory cut from *v* and determine a dual simplex pivot column *p.*

Step 2:

Update row *v* by pivoting on the element in the current Gomory cut in column *p.* If the pivot element becomes integer, go to step 4. Otherwise, go to step 3.

Step 3:

Derive a Gomory cut from the updated row *v*. Go to step 2.

Step 4:

Find the composite cut *r* which after one pivot on the element in its row in column *p* will convert the original row v to its form found in step 2.

Step 5:

Append *r* to the optimal LP simplex tableau and pivot on the element in its row in column *p.* If the new tableau has an integer solution, then the IP optimal solution has been found - terminate. Otherwise go to step 1. (Another composite cut is required.)

The composite cut is obtained by expressing the last of these cuts in terms of the previous cuts to get an expression in the form of nonbasic variables. Notice that when generating and applying the Gomory's cut, only the single generating row needs to be updated each time as the entire problem will be updated when the composite cut is added.

Deep cuts

In the early 70's a few researchers attempted to find better cuts which cut deeper into the convex hull of IP. Mitra et al (1970), based their algorithm on Gomory's fractional cut (8), by taking the fractional parts as ratios of I/D where I is the integral part and D is the modulus of the determinant. Consider a cut

$$d_0 \leq \sum_{p \in NB} d_p x_p \qquad (20)$$

which is a cut from a series of parallel cuts given by

$$d_0 + rD \leq \sum_{p \in NB} d_p x_p \qquad r = 0,1,2, \tag{21}$$

The larger the value of *r* the *deeper* the cut into the convex space of IP. To get a deep cut, it is sufficient to find the minimum value of *r* where x_p>0 and integer such that

$$\sum_{p \in NB} d_p x_p = d_0 + rD \tag{22}$$

Mitra et.al proposed an approach to find the minimum value of r which used the concept of solving diophantine equations referred as *Positive Diophantine* algorithm. Their aim was to locate lattice points on the finite parallel planes within the bounds

$$0 \leq x_p \leq \left[\frac{d_0 + rD}{d_p} \right]. \tag{23}$$

5 - Conclusions

In this report, a few algorithms for solving general IP are presented and some of them are discussed in detail. No comparison of their efficiency or superiority has been done because codes for cutting plane algorithm are not readily available.

The cutting plane approaches mentioned in this review are generally found to be unsuccessful computationally. Some of the reasons for this include:

(i) Early revised simplex codes had data structures which required matrix data to be specified in a column order format. Within this format, it was difficult to add cutting plane constraints.

(ii) The cutting planes discussed are known to have a high density of nonzero coefficients and thus lead to explosive storage requirement and destroy the sparsiry that is widely present in real life large scale LP and IP problems. The revised simplex method exploits this sparsity thus enabling large problem to be solved.
(iii) Cutting plane methods are genarally found to have slow convergence. Thus, many cuts are usually required to be added causing not only problems of solution time but also of problem size.

The main computational difficulty in the implementation of Gomory's fractional algorithm has come from numerical errors in computer arithmetic and not from the number of iterations. As simplex arithmetic proceeds, the least significant parts of computed values, that is, the fractional parts of coefficients in simplex tableaux, are naturally the most likely to include errors. In other words the algorithm suffers from round-off errors. To overcome this, all-integer algorithms were proposed. However, it is known (see e.g., Nemhauser and Wolsey (1988) and Parker and Rardin (1988)), that the all-integer algorithms (due to the all-integer restrictions imposed on this approach) are weaker than Gomory's fractional cut algorithm. Computational testing, although limited, has not in general been encouraging.

Only recently, the theory and application of stronger cuts such as those that define facets and faces of reasonable dimension have been developed. These cuts are derived by studying the facial structure of the related problems and are known to preserve sparsity and have moderate storage requirement. This approach of strong cutting plane methods will be discussed in a forthcoming report by Abdul Hamid et al. (1993).

6 - References

Abdul Hamid, F., G. Mitra, K. Darby-Dowman, L. Yarrow (1993). Polyhedral cutting plane methods for zero-one integer programming problems. *Department of Mathematics and Statistics Report , Brunel University.*

Arnold, L.R., M. Bellmore (1974). Iteration skipping in primal integer programming. *Operations Research* 22, 129-136.

Austin, L.M., M.E.Hanna (1983). A bounded dual (all-integer) Integer programming algorithm with an objective cut. *Naval Research Logistics Quarterly* 30, 271-281.

Austin, L.M., P. Ghandforoush (1985). A surrogate cutting plane algorithm for all-integer programming. *Computer and Operations Research* 12 (3), 241-250.

Bowman, V.J., G.L. Nemhauser (1970). A finiteness proof for modified Dantzig Cuts in integer programming. *Naval Logistics Quarterly* 17, 309-313.

Dantzig, G.B. (1959). Note on solving linear programs in integers. *Naval Research Logistics Quarterly* 6, 75-76.

Dantzig, G.B., D.R. Fulkerson, S.M. Johnson (1954). Solution of large scale traveling salesman problem. *Operations Research* 2 (4), 393-410.

Garfinkel, R.S., G.L. Nemhauser (1972).*Integer Programming,* John Wiley.

Ghandforoush, P., L.M. Austin (1981). A primal-dual cutting plane algorithm for all-integer programming. *Naval Research Logistics Quarterly* 28 (4), 559-566.

Ghandforoush, P. (1983). Accelerated primal-dual cutting plane algorithm for all-integer programming. *Computers and Operations Research* 10 (3), 249-254.

Glover, F. (1965). A new foundation for a simplified primal integer programming algorithm *Operations Research* 13 (6), 879-929.

Glover, F. (1967). A pseudo primal-dual integer programming algorithm. *Journal of Research of the National Bureau of Standards* 71B, 167-195.

Gomory, R.E. (1963a). An algorithm for integer solutions to linear programs. *Recent Advances in Mathematical Programming,* ed. Graves, R.L. and Wolfe, P. (Mcgraw Hill), 269-302.

Gomory, R.E. (1963b), An all-integer Integer programming algorithm. *Industrial Scheduling,* Muth, J.F. and Thompson, G.L. (Prentice Hall), 193-206.

Gomory, R.E. (1966). An algorithm for the mixed integer problem. RAND Report RM2597, July 1960, in M. Simmonard (ed) *Linear Programming,* (Prentice Hall), Englewood Cliffs, NJ.

Gomory, R.E., AJ. Hoffman (1963). On the convergence of an integer programming. *Naval Research Logistics Quarterly* 10 (2), 121-123.

Jenkins, L., and D. Peters (1987). A computation comparison of Gomory and knapsack cuts. *Computers and Operations Research* 14 (6), 449-456.

Martin, G.T. (1963). An accelerated euclidean algorithm for integer linear programming. *Recent Advances in Mathematical Programming,* ed. Graves, R.L. and Wolfe, P. (Mcgraw Hill), 311-317.

Mitra, G., D.B.C.Richards, K.Wolfenden (1970). An improved algorithm for the solution of integer programs by the solution of associated diophantine equations. *R. I. R. O.* -R.1, 47-66.

Markowitz, H., K.G.Manne (1957). On the solution of discrete programming problems. *Econometrica* 25, (1), 84-110.

Nemhauser, G.L., L.A. Wolsey (1988). Integer and combinatorial Optimization, John Wiley.

Nourie, F., R. Venta (1982). An upper bound on the number of cuts needed in Gomory's method of integer forms. *Operations Research Letters* 1, (4), 129-133.

Parker, R.G., R.L. Rardin (1988). Discrete Optimization, Academic Press.

Richards, D.B.C. (1967). Integer programming - Theory and Practice. Thesis for the university of London.

Salkin, H.M. (1975). *Integer Programming,* Addison-Wesley Publishing Company.

Salkin,H.M., Mathur K. (1989). *Foundations of Integer Programming,* Elsevier Science Publishing Company (North-Holland).

Young, R.D. (1968). A Simplified primal (all-integer) Integer programming. *Operations Research* 16 (4), 750-782.