DISCUSSION PAPER No. 111

ON THE INTERRELATIONSHIPS BETWEEN GOMORY
AND DANTZIG CUTTING PLANES

By

Avinoam Perry

Revised June 1975

* Assistant Professor, Graduate School of Management, Northwestern University, Evanston, Illinois.
ABSTRACT

In this paper we developed a cutting plane formula and used it for solving the all integer program. The cut is based on properties of the Gomory mixed integer cut but, nevertheless, its associated slack is restricted to be integer. Also we showed that the new cut is uniformly deeper than the modified Dantzig cut, and under circumstances discussed in the paper it may be deeper than the Gomory mixed integer cut for the all integer program along one or more dimensions. Computational experience for this new cut is contrasted with other codes, and the results (as shown in this paper) are encouraging.
Consider the ILP

\[
\text{Max } Z = \sum_{j \in J} c_j x_j \\
\text{s.t. } \sum_{j \in J} b_{ij} x_j = b_{i0} \quad i = 1, 2, \ldots, m \\
x_j \geq 0 \text{ integer }, \quad j \in J
\]

The \(i\)th row corresponding to the LP solution of (1) is given by

\[
x_i = b_{i0} - \sum_{j \in J} b_{ij} t_j
\]

where \(t_j\) is used for signifying a nonbasic variable, and \(x_b\) is a basic variable.

The Comory Cut \([5]\) derived from (2) is given by

\[
\sum_{j \in J} b_j t_j \geq b_0
\]

where: \(b_j = b_{j0} - [b_j]\) and \(b_0 = b_{00} - [b_0]\) > 0

An alternative complementary cut is given by

\[
\sum_{j \in J} (1-b_j) t_j \geq 1 - b_0
\]

Convergence was proved in (3) and (4) in [5]. An intuitive cut derived by Dantzig \([3]\) is given by

\[
\sum_{j \in J} t_j \geq 1
\]

As noted by Comory and Hoffman \([7]\), an algorithm based on (5), does not converge in a finite number of steps.

A combination of (3) and (4) produce the modified Dantzig cut \([2]\) which is given by
\[ \sum_{j \in K} t_j \geq 1 \quad K \subseteq J \]

where \( K \) includes only those nonbasic variables for which \( b_j \neq 0 \). Note that (6) is at least as deep as (5). Bowman and Nemhauser [1] proved that algorithms applying (6) are finite.

Rubin and Graves [9] used (3) to strengthen (6) and derived what they call MDZ, which is given by

\[ \sum_{j \in K} t_j \geq 2 \]

This cut is realized under the condition that \( b_0 \neq b_j > 0 \) for all nonbasic variables in the corresponding source row.

Salkin [10] suggested that the Conory mixed integer cut [6] given by:

\[ \sum_{j \in R} b_j t_j + \sum_{j \in Q} \frac{b_0}{1-b_0} (1-b_j) t_j \geq b_0 \]

where \( R \cup Q = K \) and the assignment of any \( t_j \) to either \( R \) or \( Q \) is such that:

- if \( b_j > b_0 \), \( t_j \rightarrow Q \)
- if \( b_j \leq b_0 \), \( t_j \rightarrow R \)

is deeper than (3), (4), (5), and (6).

Garfinkel and Nemhauser [4] noted that in large problems it would appear that (8) is undesirable for ILP's because continuous slack variables are required.

Perry [8] tested 3 different versions of (8) on some 15 well known test problems and obtained encouraging results, but admitting difficulties in coding due to severe rounding error problems.

To facilitate the following discussion we simplify (8) as
\[ \sum_{j \in K} f_j t_j \geq f_0 \]

where \( t_j = \begin{cases} 
  b_j & \text{if } j \in R \\
  b_0 & \text{if } j \in Q \\
  \frac{b_0}{1-b_0} (1-b_j) & \text{if } j \in Q
\end{cases} \)

\[ f_0 = b_0 \]

Note that \( f_j \leq f_0 \) for all \( j \in R \) and the slack variable associated with (9) is continuous. It follows that (9) is always deeper than (or equivalent to) (3), (4), (5), and (6).

Let \( f_{mj} = \max_{j \in K} f_j \) for the \( ij \)th source row, then from (9) it follows that

\[ f_m \sum_{j \in K} t_j \geq f_0 \]

or

\[ \sum_{j \in K} t_j \geq \left( \frac{f_0}{f_m} \right)^0 \]

Since \( t_j \) is an integer and since \( f_m \leq f_0 \) (11) is equal to (6) if there is at least one \( b_j = b_0 \). (11) is equal to (7) if \( 1 < \frac{f_0}{f_m} \leq 2 \). (11) becomes

\[ \sum_{j \in K} t_j \geq M \text{ if } (M-1) < \frac{f_0}{f_m} \leq M \text{ where } M \text{ is an integer number.} \]

Note that if \( f_0/f_m \) is an integer (9) is consistently deeper than (11) but if \( f_0/f_m \) is noninteger than (11) is deeper than (9) along one or more dimensions. Also note that (11) assumes an integer slack while (9) assumes a continuous slack.

Taking into account the procedure for deriving (8) we can now use it for strengthening (11). (8) was derived by requiring

\[ \sum_{j \in R} b_j t_j \geq b_0 \]

\[ ^* \text{This notation is used to signify the smallest integer } \geq \frac{f_0}{f_m} \]
or

\[ (13) \quad \sum_{j \in Q} \frac{b_0}{1 - b_0} (1 - b_j) t_j \geq b_0 \]

let \( f_R = \max_{j \in R} b_j \) and \( f_Q = \max_{j \in Q} \frac{b_0}{1 - b_0} (1 - b_j) \), then from (12) and (13) respectively, it follows that either

\[ (14) \quad f_R \sum_{j \in R} t_j \geq b_0 \]

or

\[ (15) \quad f_Q \sum_{j \in Q} t_j \geq b_0 \]

From (14) and (15) we have

\[ (16) \quad \sum_{j \in R} t_j \geq \left( \frac{b_0}{f_R} \right) = M_1 = \text{integer} \]

or

\[ (17) \quad \sum_{j \in Q} t_j \geq \left( \frac{b_0}{f_Q} \right) = M_2 = \text{integer} \]

Combining (16) and (17) we have

\[ (18) \quad M_2 \sum_{j \in R} t_j + M_1 \sum_{j \in Q} t_j \geq M_1 \cdot M_2 \]

If \( M_1 \neq M_2 \) then (18) is deeper than (11) which is deeper than (6) and therefore an algorithm applying (18) converges in a finite number of steps. (18) preserves the integrality requirement of all the variables involved and is less sensitive to rounding errors than (8).

The problems used for testing purposes are those developed and reported by J. Haldi [5] to test the LIPI computer code. Further comparisons were made with respect to Truth and Woolsey's study in computational efficiency [9] who tested the LIPI, IPM3, and ILP2-1 codes. The results are presented in tables
I and II are self-explanatory. All times were computed from the first executed instruction of the program to the end of the minimum output needed to interpret the results. All times are given in seconds. The word "Iteration" refers to a single matrix pivot operation. MD and SGV2 were run on the CDC 6600 computer.

The first ten problems in the computational summary tables are Hald's fixed charge problems. They are followed by IBM integer programming test problems also in [5]. The results are summarized in the following tables:

### Table I
Fixed Charge Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Code</th>
<th>MD</th>
<th>Itr.</th>
<th>SGV2</th>
<th>Itr.</th>
<th>LIP1</th>
<th>Itr.</th>
<th>IPN-3</th>
<th>Itr.</th>
<th>ILP2-1</th>
<th>Itr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1.979</td>
<td>37</td>
<td>1.902</td>
<td>20</td>
<td>1.833</td>
<td>24</td>
<td>3.117</td>
<td>56</td>
<td>0.852</td>
<td>36</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.508</td>
<td>52</td>
<td>1.401</td>
<td>13</td>
<td>1.350</td>
<td>15</td>
<td>3.767</td>
<td>81</td>
<td>0.935</td>
<td>47</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>1.996</td>
<td>31</td>
<td>1.430</td>
<td>14</td>
<td>1.883</td>
<td>26</td>
<td>3.033</td>
<td>37</td>
<td>1.384</td>
<td>104</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1.001</td>
<td>10</td>
<td>0.266</td>
<td>6</td>
<td>1.483</td>
<td>18</td>
<td>4.100</td>
<td>91</td>
<td>0.674</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>3.765</td>
<td>48</td>
<td>2.414</td>
<td>16</td>
<td>9.012</td>
<td>156</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>3.708</td>
<td>45</td>
<td>2.819</td>
<td>24</td>
<td>7.507</td>
<td>123</td>
<td></td>
<td></td>
<td>+7000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>3.401</td>
<td>46</td>
<td>2.497</td>
<td>16</td>
<td>7.883</td>
<td>159</td>
<td></td>
<td></td>
<td>+7000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>3.322</td>
<td>45</td>
<td>2.310</td>
<td>14</td>
<td>6.417</td>
<td>126</td>
<td></td>
<td></td>
<td>+7000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>1.917</td>
<td>15</td>
<td>1.282</td>
<td>9</td>
<td>3.233</td>
<td>42</td>
<td>5.183</td>
<td>118</td>
<td>3.598</td>
<td>298</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>8.670</td>
<td>86</td>
<td></td>
<td>+5000</td>
<td></td>
<td>9.150</td>
<td>102</td>
<td>71.100</td>
<td>1396</td>
<td></td>
</tr>
</tbody>
</table>

MD is Applying the Cuts in (18) to reach an integer solution
SGV2 is utilizing the strengthened Gomory Mixed Integer Cut (8) Version 2 [8]
LIP1, IPN-3, ILP2-1 are codes tested by Trauth and Woolsey [11]

### Table II
Hald's IBM Problems

<table>
<thead>
<tr>
<th>Problem</th>
<th>Code</th>
<th>MD</th>
<th>Itr.</th>
<th>SGV2</th>
<th>Itr.</th>
<th>LIP1</th>
<th>Itr.</th>
<th>IPN-3</th>
<th>Itr.</th>
<th>ILP2-1</th>
<th>Itr.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
<td>Time</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2.095</td>
<td>12</td>
<td>1.512</td>
<td>8</td>
<td>1.866</td>
<td>11</td>
<td>2.300</td>
<td>8</td>
<td>1.010</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2.623</td>
<td>25</td>
<td>2.785</td>
<td>23</td>
<td>3.016</td>
<td>32</td>
<td>2.833</td>
<td>17</td>
<td>1.056</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2.300</td>
<td>41</td>
<td>2.617</td>
<td>41</td>
<td>2.866</td>
<td>53</td>
<td>2.533</td>
<td>72</td>
<td>0.705</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>12.988</td>
<td>85</td>
<td>8.882</td>
<td>40</td>
<td>11.666</td>
<td>73</td>
<td>9.933</td>
<td>74</td>
<td>3.492</td>
<td>61</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>81.650</td>
<td>402</td>
<td>31.566</td>
<td>149</td>
<td>66.683</td>
<td>351</td>
<td>51.600</td>
<td>1144</td>
<td>+7000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>302.795</td>
<td>663</td>
<td>357.002</td>
<td>841</td>
<td>675.100</td>
<td>953</td>
<td>633.313</td>
<td>6758</td>
<td>+7000</td>
<td></td>
</tr>
</tbody>
</table>
Acknowledgement

I wish to thank A. Tamir for his useful comments on an early draft of this paper.
References


