CUTTING PLANE METHOD
Convert the problem to standard form with integer coefficients. Solve.
If a primal variable is not integral, cut it off with add a new integer-coefficient constraint.
Cutting it off, makes it nonfeasible. Use the dual method to restore feasibility.
Repeat until an optimal integer solution is found.

THE CUTTING PLANE ALGORITHM FOR INTEGER PROBLEMS

- If a constraint has fractions, multiply both sides by a common denominator to get an integer constraint.
- Run the simplex method; get a solution.

Loop:
- If the primal variables are integral, stop. We’re done.
- If not pick the constraint for the primal variable with the largest decimal part.
- Take the floor of its coefficients and its constant, and add a slack variable.
- Add this new constraint; pivot to make a tableau (the basic variables must have identity columns).
- Apply the dual method to restore feasibility.

Repeat the loop.
SciLab example. Let’s add a *floor* constraint which cuts off the nonintegral solution of the first row.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2.5</th>
<th>-0.5</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Load the program ‘insert’: `load(‘414’)` or copy the next line to the command line:

```plaintext
insert=’[r,c]=size(a);a([r,r+1],:)=[row;a(r,:)];a(r,c+1)=1;a(:,[c:c+1])=a(:,[c+1:c]);disp(a)’
```

The floor of the nonintegral row `[1, 0, 2.5, -0.5, 0.5]` is `[1, 0, 2, -1, 0]`. To add row `[1, 0, 2, -1, 0]` with its slack column `[0; 0; 1; 0]` enter:

```plaintext
row = [1, 0, 2, -1, 0]; execstr(insert).
```

This gives

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2.5</th>
<th>-0.5</th>
<th>0</th>
<th>0.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>9</td>
</tr>
</tbody>
</table>

The inserted row was row = `[1, 0, 2, -1, 0]`, not row = `[1, 0, 2, -1, 1, 0]`, the “1” for the new identity column is added automatically.
max \( z = y \)

with

\( a: 2x + 2y \leq 3, \)

\( x, y \in \mathbb{N} \)
\[ \text{max } z = y \]

with
\[ a: 2x + 2y \leq 3, \]
\[ x, y \in \mathbb{N} \]

Initial tableau.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

Final tableau.

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
max z = y
with
\( a: 2x + 2y \leq 3, \)
\( x, y \in \mathbb{N} \)

Initial tableau.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>r</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>r</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>z</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Final tableau.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>r</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>3/2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1/2</td>
<td>3/2</td>
</tr>
</tbody>
</table>

The solution is not integral.
Cut it off by adding a new floor constraint.
Final tableau.

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$r$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>3/2</td>
</tr>
<tr>
<td>$z$</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>3/2</td>
</tr>
</tbody>
</table>

The new constraint is: ??

Adding a slack gives:

New matrix.

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$r$</th>
<th>$s$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$z$</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
New matrix.

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$r$</th>
<th>$s$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$y$</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>0</td>
<td>3/2</td>
</tr>
<tr>
<td>$s$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$z$</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>3/2</td>
</tr>
</tbody>
</table>

New tableau. **Pivot to make $y$ and $s$ identity columns.**

<table>
<thead>
<tr>
<th></th>
<th>$x$</th>
<th>$y$</th>
<th>$r$</th>
<th>$s$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>$z$</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
The solution is not feasible. **Restore feasibility using the dual method.** In the row with the negative constant, pivot on the entry whose objective/(negative-coefficient) ratio is closest to 0.

**Final tableau.**

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>r</th>
<th>s</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>0</td>
<td>3/2</td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>0</td>
<td>-1/2</td>
<td>1</td>
<td>-1/2</td>
</tr>
<tr>
<td>z</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>3/2</td>
</tr>
</tbody>
</table>
New tableau.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>r</th>
<th>s</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>1/2</td>
<td>0</td>
<td>3/2</td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>0</td>
<td>-1/2</td>
<td>1</td>
<td>-1/2</td>
</tr>
<tr>
<td>z</td>
<td>1</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
<td>3/2</td>
</tr>
</tbody>
</table>

Final tableau.

<table>
<thead>
<tr>
<th></th>
<th>x</th>
<th>y</th>
<th>r</th>
<th>s</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>s</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>-2</td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Answer. \( \max z = 1 \) at \( y = 1, x = 0 \).
For integer problems duality theory fails. Omit dual variable calculations.

**Trees**

A tree consists of *nodes* connected by *edges*. There is a *root* node at the top. If two nodes are connected by an edge, the higher node is the *parent* and the lower one is the *child*. Nodes with no children are *terminal* nodes or *leaves*. In a *binary tree* nonterminal nodes have exactly two children. From each node, there is a unique *path* up to the root.
Branch and bound methods construct binary trees. Each edge is labeled with a constraint. Each node is labeled with an optimal simplex solution to the problem consisting of the original constraints plus those on the edges between the node and the root.

A node is

- A *solution has the desired type* if all variables are of the desired type (Z, N, or {0, 1} as required by the problem). These nodes are terminal and are circled.
- An *undesired type node* has some variable which is not of the desired type. These are be boxed. They eventually get children or get crossed-off.
- *Nonoptimal* if some other circled solution has a better objective value. These will be crossed-off.
- *Empty* if it has no feasible solutions. Cross these off.
**Branch and Bound Algorithm**

- Label the root node with the simplex solution to the original problem.
- If it is a solution of the desired type, circle it and stop. Otherwise box it.

**Loop:**
- Pick a boxed uncrossed-off terminal node.
- Select the first variable \( v \) whose value \( b \) isn’t of the desired type.
- Add two edges below the node labeled with
  
  (a) \( v = 0 \) and \( v = 1 \) if \( v \) is a 0-1 variable.
  
  or (b) \( v \leq [b] \) and \( v \geq [b]+1 \) if \( v \) is integral.
- Add a node below each such edge. Label it with the simplex solution to the problem with the original constraints plus those along the path to the root.
- For each such new node:
  - **Cross off** the node if it is empty or if it is less optimal than some circled solution.
  - **Box** the node if it is not of the desired type.
  - **Circle** the node if it is a solution of the desired type. In this case, cross off any terminal nodes with a less optimal objective value.
- If there are no boxed uncrossed-off terminal nodes, stop.
  
  All uncrossed-off terminal solutions are optimal.
- If all terminal nodes are empty, stop, there is no solution.
  
  Goto loop.
\[ \text{max } z = x + 4y + 3w \]
with
1: \( x + y + 2w \leq 7 \quad x \in N \)
2: \( 7x - 2y + w \leq 9 \quad y \in \{0,1\} \)
3: \( 3x - y - 2w \geq 0 \quad w \geq 0 \)

\[
\begin{array}{l}
z=22.75 \\
x=1.75 \\
y=5.25
\end{array}
\]
max $z = x + 4y + 3w$

with

1: $x + y + 2w \leq 7 \quad x \in N$
2: $7x - 2y + w \leq 9 \quad y \in \{0, 1\}$
3: $3x - y - 2w \geq 0 \quad w \geq 0$

\[
\begin{array}{c}
z=22.75 \\
x=1.75 \\
y=5.25
\end{array}
\]

$x \leq 1$

$z=13$

$x=1$

$y=3$

$x \geq 2$

$z=22$

$x=2$

$y=5$
\[ \text{max } z = x + 4y + 3w \]

with

1: \( x + y + 2w \leq 7 \) \( \quad x \in \mathbb{N} \)
2: \( 7x - 2y + w \leq 9 \) \( y \in \{0, 1\} \)
3: \( 3x - y - 2w \geq 0 \) \( w \geq 0 \)

\[ z = 22.75 \]
\[ x = 1.75 \]
\[ y = 5.25 \]

\[ x \leq 1 \]
\[ z = 13 \]
\[ x = 1 \]
\[ y = 3 \]

\[ x \geq 2 \]
\[ z = 22 \]
\[ x = 2 \]
\[ y = 5 \]
\[ \text{max } z = x + 4y + 3w \]

with

1: \[ x + y + 2w \leq 7 \quad x \in \mathbb{N} \]
2: \[ 7x - 2y + w \leq 9 \quad y \in \{0,1\} \]
3: \[ 3x - y - 2w \geq 0 \quad w \geq 0 \]
\[ \text{max } z = x + 4y + 3w \]

with

1: \( x + y + 2w \leq 7 \) \( x \in \mathbb{N} \)
2: \( 7x - 2y + w \leq 9 \) \( y \in \{0, 1\} \)
3: \( 3x - y - 2w \geq 0 \) \( w \geq 0 \)

\[ z = 22.75 \]
\[ x = 1.75 \]
\[ y = 5.25 \]
\[ \text{max } z = x + 4y + 3w \]

with

1: \( x + y + 2w \leq 7 \quad x \in \{0, 1\} \)
2: \( 7x - 2y + w \leq 9 \quad y \in \mathbb{N} \)
3: \( 3x - y - 2w \geq 0 \quad w \geq 0 \)

\[
\begin{array}{l}
z = 22.75 \\
x = 1.75 \\
y = 5.25
\end{array}
\]
\[
\text{max } z = x + 4y + 3w
\]
with

1: \(x + y + 2w \leq 7\) \hspace{1em} x \in \{0, 1\}

2: \(7x - 2y + w \leq 9\) \hspace{1em} y \in \mathbb{N}

3: \(3x - y - 2w \geq 0\) \hspace{1em} w \geq 0

\[
\begin{array}{c}
z = 22.75 \\
x = 1.75 \\
y = 5.25
\end{array}
\]

\[
\begin{array}{c}
z = 0 \\
x = 0 \\
y = 0
\end{array}
\]

\[
\begin{array}{c}
z = 0 \\
x = 1 \\
y = 3
\end{array}
\]
Get the branch and bound solutions geometrically.

\[
\text{max } z = x + y \\
\text{with}
\]

\[
r: x + y \leq 5 \\
s: -x + y \leq 0 \\
x, y \in \mathbb{N}
\]
Get the branch and bound solutions geometrically.

max $z = x+y$

with

$r$: $x + y \leq 5$

$s$: $-x + y \leq 0$

$x, y \in \mathbb{N}$
Use the branch-and-bound method for problems which have one or more 0-1 variables. For problems with only integer variables, the cutting-plane method is usually faster.

- \( \text{min } y \) with
  - \( r: 3x - 8y \leq 0 \)
  - \( s: 3x + 8y \geq 24 \)
  \( x, y \in \mathbb{N} \)

Start with \( x=4, y=1.5 \).
Breadth-first: Generate all nodes of one level before going to next level.
*Depth-first*: Follow down right-hand branch as far as possible, then backtrack up to next right-most node.