Math 414  Lecture 18

Cutting Plane Method. Convert the problem to standard form with integer coefficients. Solve. If a primal variable is not integral, cut it off with a new integer-coefficient constraint. This 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 a constraint with only integers.
- Run the simplex method as usual and get a solution.
- If the primal variables are integral, stop. We’re done.
- If not, pick the constraint for the primal variable (ignore nonintegral slacks) with the largest decimal part (choose either one if two or more have the same 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 whose basic variables have identity columns.
- Apply the dual method to restore feasibility.
- Repeat the loop.

For integer problems duality theory fails. Omit dual variable calculations.

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

\[
\begin{array}{cccc}
1 & 0 & 2.5 & -0.5 & 0.5 \\
0 & 1 & 2 & 2 & 2 \\
0 & 0 & 9 & 9 & 9 \\
\end{array}
\]

To load the program ‘insert’, enter: load(‘414’) or copy the next line to the command SciLab command line.

\[
\text{insert}=[r,c]=\text{size}(a); a([r,r+1,:])=[\text{row; } a(r,:)] ; a(r,c+1)=1; a(:,[c;c+1])=a(:,[c+1;c]); \text{disp}(a)
\]

The floor of the nonintegral row \([2.5, -0.5, 0.5]\) is \([2, -1, 0]\).
To add row \([1, 0, 2, -1, 0]\) & slack column \([0; 0; 1; 0]\) enter:
\[
\text{row}=[1,0,2,-1,0]; \text{execstr(insert)}
\]

This gives

\[
\begin{array}{cccccc}
1 & 0 & 2.5 & -0.5 & 0 & 0.5 \\
0 & 1 & 2 & 2 & 0 & 2 \\
0 & 0 & 2 & -1 & 1 & 0 \\
0 & 0 & 9 & 9 & 0 & 9 \\
\end{array}
\]

The inserted row was “row = [2, -1, 0]”, not [2, -1, 1, 0], the “1” for the identity column was added automatically,

\[
\begin{align*}
\text{max } z &= y \\
\text{with } \begin{align*}
2x + 2y & \leq 3, \\
x, y & \in \mathbb{N}
\end{align*}
\end{align*}
\]

Initial tableau.

\[
\begin{array}{cccc}
x & y & r & b \\
\end{array}
\]

Final tableau. Select \(y\) to enter and pivot.

\[
\begin{array}{cccc}
x & y & r & b \\
\end{array}
\]

The solution is not integral. Cut it off by adding a new floor constraint.
The new constraint is: __________________
Adding a slack gives: __________________
New matrix.

\[
\begin{array}{cccccc}
x & y & r & s & b \\
\end{array}
\]

New tableau. Pivot to make \(y\) and \(s\) identity columns.

\[
\begin{array}{cccccc}
x & y & r & s & b \\
\end{array}
\]

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.

\[
\begin{array}{cccccc}
x & y & r & s & b \\
\end{array}
\]

Answer. max \(z = 1\) at \(y = 1, x = 0\).
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
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 of the desired type if all variables are of the desired type (in \( \mathbb{Z}, \mathbb{N} \), 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 a solution 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, there is no solution. Goto loop.

Get the branch and bound solutions geometrically.
\[
\begin{align*}
\text{max } z &= x + 4y + 3w \\
\text{with} & \quad \begin{array}{c}
1: x + y + 2w \leq 7 \\ 2: 7x - 2y + w \leq 9 \\ 3: 3x - y - 2w \geq 0
\end{array} \\
\text{where} & \quad \begin{array}{c}
x \in \mathbb{N} \\ y \in \{0,1\} \\ w \geq 0
\end{array}
\end{align*}
\]

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.

**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.