Lecture 3

Branch-and-Bound Algorithm

MATH3220 Operations Research and Logistics
Jan. 13, 2015

Pan Li
The Chinese University of Hong Kong

Agenda

Is IP easy to solve?

- At the first glance, IP may be easy to solve since the number of feasible solution is smaller and is limited. But...
- If the feasible region is bounded, the number of feasible solution is finite. But, the number is simply too large (exponential growth). With n variables, there are 2^n solutions to be considered.
- The corner point is (generally) no longer feasible.

Overview

- Enumerating all solution is too slow for most problems.
- Branch and bound starts the same as enumerating, but it cuts out a lot of the enumeration whenever possible.
- Branch and bound is the starting point for all solution techniques for integer programming.

Capital Budgeting Example

investment budget = $14,000

Investment	1	2	3	4	5
Cash Required	$5,000	$4,000	$7,000	$3,000	$6,000
Present Value	$12,000	$11,000	$13,000	$8,000	$15,000

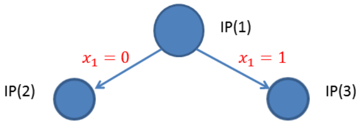
$$\begin{array}{ll} \max & 12x_1 + 11x_2 + 13x_3 + 8x_4 + 15x_5 \\ \text{s.t.} & 5x_1 + 4x_2 + 7x_3 + 3x_4 + 6x_5 \leq 14 \\ & x_j \in \{0, 1\} \text{ for each } j = 1 \text{ to } 5 \end{array}$$

Complete enumeration

- Systematically considers all possible values of the decision variables. - If there are n binary variables, there are 2^n different ways.
- Usual idea: iteratively break the problem into two.
  At the first iteration, we consider separately the case that $x_1 = 0$ and $x_1 = 1$.
- Each node of the tree represents the original problem plus additional constraints.

An Enumeration Tree

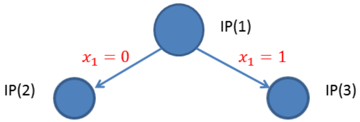
An Enumeration Tree



We refer to node 2 and 3 as the children of node 1 in the enumeration tree.

We refer to node 1 as the parent of node 2 and 3.

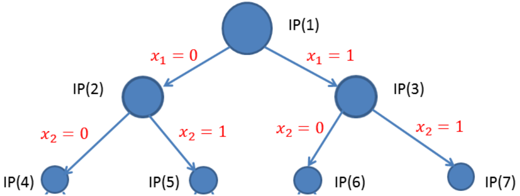
An Enumeration Tree



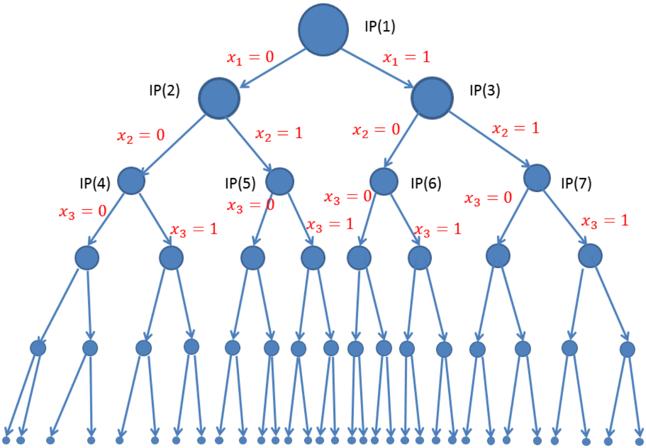
Which of the following is false?

1. IP(1) is the original integer program.
2. IP(3) is obtained from IP(1) by adding the constraint $x_1 = 1$.
3. It is possible that there is some solution that is feasible for both IP(2) and IP(3).

An Enumeration Tree



An Enumeration Tree



Number of leaves of the tree : 32. $\Rightarrow$ If there are n variables, the number of leaves is 2^n.

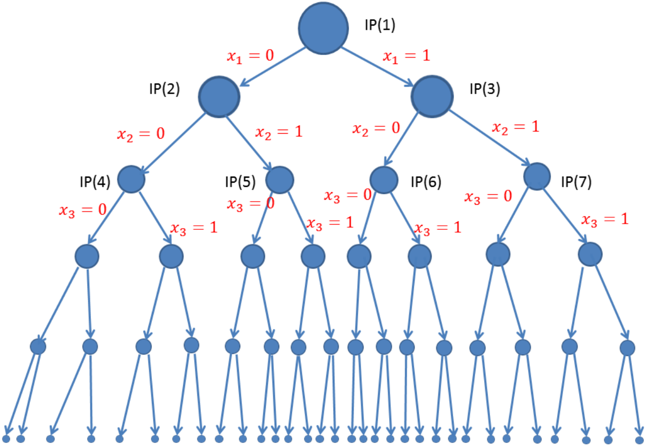
On complete enumeration

- Suppose that we could evaluate 1 billion solutions per second.
- Let $n =$ number of binary variables.
- Solution times
  - $n = 30$, 1 second
  - $n = 40$, 17 minutes
  - $n = 50$, 11.6 days
  - $n = 60$, 31 years
  - $n = 70$, 31,000 years

How to solve large size integer program faster?

Only a tiny fraction of the feasible solutions actually need to be examined.

Subtrees of an enumeration tree



If we can eliminate an entire subtree in one step, we can eliminate a fraction of all complete solutions at a single step.

Branch-and-Bound Algorithm

- Branch-and-bound algorithm are the most popular methods for solving integer programming problems.
- Basic Idea: Enumeration procedure can always find the optimal solution for any bounded IP problem. But it takes too much time. So, we consider the partial enumeration. That is, divide and conquer.
- They enumerate the entire solution space but only implicity; hence they are called implicit enumeration algorithms.
- Bounding, branching, and fathoming are the three components of Branch-and-bound technique.
- Running time grows exponentially with the problem size, but small to moderate size problems can be solved in reasonable time.

A simpler problem to work with

$$\begin{array}{rll} \text{Max} & 24x_1 + 2x_2 + 20x_3 + 4x_4 & \\ \text{s.t.} & 8x_1 + x_2 + 5x_3 + 4x_4 \leq 9 & \text{IP(1)} \\ & x_i \in \{0, 1\} \text{ for } i = 1 \text{ to } 4 & \end{array}$$

LP Relaxation

LP Relaxation: The LP obtained by omitting all integer or 0-1 constraints on variables is called the LP relaxation of IP.

IP:

$$\begin{aligned} \text{max (or min)} \quad & cx \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \text{ and integers} \end{aligned}$$

LP Relaxation:

$$\begin{aligned} \text{max (or min)} \quad & cx \\ \text{subject to} \quad & Ax \leq b \\ & x \geq 0 \end{aligned}$$

IP and LP Relaxation

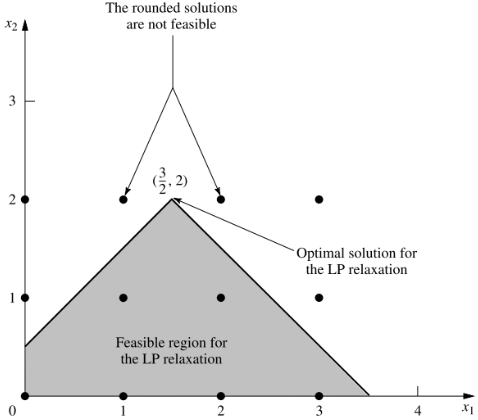
Since LP relaxation is less constrained than IP, the following are immediate:

- If IP is a minimization problem, the optimal objective value for LP relaxation is less than or equal to the optimal objective for IP.
- If IP is a maximization, the optimal objective value for LP relaxation is greater than or equal to that of IP.
- If LP relaxation is infeasible, then so is IP.
- If LP relaxation is optimized by integer variables, then that solution is feasible and optimal for IP.

So, solving LP relaxation does give some information: it gives a bound on the optimal value, and if we are lucky, may give the optimal solution to IP.

Why can't we solve the LP relaxation and round it up/down to get the required integer solution?

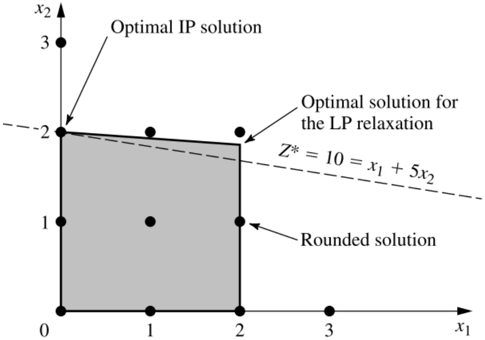
- Rounding does not guarantee the feasibility.



$$
\begin{array}{ll}
\text{Max} & z = x_2 \\
\text{s.t.} & -x_1 + x_2 \leq 0.5 \\
 & x_1 + x_2 \leq 3.5 \\
 & x_1, x_2 \geq 0 \text{ and} \\
 & \text{are integers}
\end{array}
$$

Why can't we solve the LP relaxation and round it up/down to get the required integer solution?

- Rounding does not guarantee the optimality.



$$
\begin{array}{ll}
\text{Max} & z = x_1 + 5x_2 \\
\text{s.t.} & x_1 + 10x_2 \leq 20 \\
& x_1 \leq 2 \\
& x_1, x_2 \geq 0 \text{ and} \\
& \text{are integers}
\end{array}
$$

Branch-and-Bound

The essential idea: search the enumeration tree, but at each node:

1. Solve the LP relaxation at the node
2. Eliminate the subtree (fathom it) if
   1. The solution is integer (there is no need to go further) or
   2. There is no feasible solution or
   3. The best solution in the subtree cannot be as good as the best available solution (the incumbent).

Bounding - For each problem or subproblem (will define later), we need to obtain a bound on how good its best feasible solution can be.

- Usually, the bound is obtained by solving the LP relaxation.

  LP relaxation of IP(1):

$$
\begin{array}{lll}
\text{Max} & 8x_1 + 11x_2 + 6x_3 + 4x_4 & \\
\text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 & \qquad \text{LP(1)} \\
 & 0 \leq x_i \leq 1 \text{ for } i = 1 \text{ to } 4 &
\end{array}
$$

Bounding - For each problem or subproblem (will define later), we need to obtain a bound on how good its best feasible solution can be.

- Usually, the bound is obtained by solving the LP relaxation.

  LP relaxation of IP(1):

$$\begin{array}{lll} \text{Max} & 8x_1 + 11x_2 + 6x_3 + 4x_4 & \\ \text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 & \text{LP(1)} \\ & 0 \leq x_i \leq 1 \text{ for } i = 1 \text{ to } 4 & \end{array}$$

- The LP relaxation of the knapsack problem can be solved using a "greedy algorithm".

  Think of the objective in terms of dollars, and consider the constraint as bound on the weight.

Solving the LP relaxation (LP(1))

$$\begin{array}{lll} \text{Max} & 8x_1 + 11x_2 + 6x_3 + 4x_4 & \\ \text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 & \text{LP(1)} \\ & 0 \leq x_i \leq 1 \text{ for } i = 1 \text{ to } 4 & \end{array}$$

item	1	2	3	4
unit value	\$1.6	\$1.571	\$1.5	\$1.333

Consider the unit value of the four items. Put items into the knapsack in decreasing order of unit value. What do you get?

Solving the LP relaxation (LP(1))

$$
\begin{array}{lll}
\text{Max} & 8x_1 + 11x_2 + 6x_3 + 4x_4 & \\
\text{s.t.} & 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 & \text{LP(1)} \\
 & 0 \leq x_i \leq 1 \text{ for } i = 1 \text{ to } 4 &
\end{array}
$$

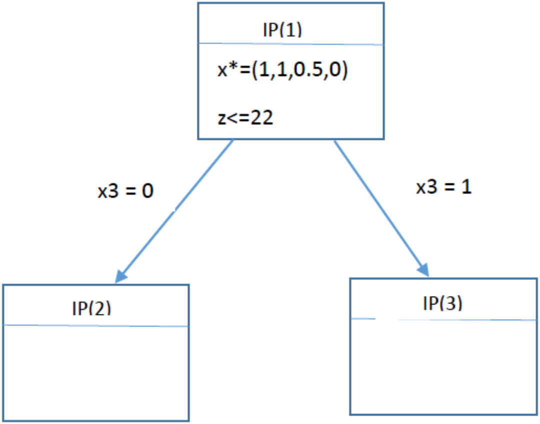
item	1	2	3	4
unit value	$1.6	$1.571	$1.5	$1.333

Consider the unit value of the four items. Put items into the knapsack in decreasing order of unit value. What do you get?

$\Rightarrow$ $(x_1, x_2, x_3, x_4) = (1, 1, 0.5, 0)$, with $z = 22$
$\Rightarrow$ No integer solution will have value lager than 22.

Branching - partitioning the entire set of feasible solutions into smaller and smaller subsets.

Set " No incumbent with objective value $z^* = -\infty$".
x_3 - branching variable



Note that any optimal solution to the overall problem must be feasible to one of the subproblems.

More on the incumbent

- The incumbent is the feasible solution for the IP. It is the best solution so far in the B&B search.
- As Branch-and-bound proceeds, new solutions will be evaluated. If a new solution is better than the current incumbent, it replaces the current incumbent.
- So, the incumbent is always the best solution seen so far.

Solving the LP relaxation (LP(2))

IP(2): $x_3 = 0$

$$\begin{array}{rll} \text{Max} & 8x_1 + 11x_2 + 6 \times 0 + 4x_4 & \\ \text{s.t.} & 5x_1 + 7x_2 + 4 \times 0 + 3x_4 \leq 14 & \text{LP(2)} \\ & 0 \leq x_i \leq 1 \text{ for } i = 1, 2, 4 & \end{array}$$

item	1	2	3	4
unit value	\$1.6	\$1.571	\$0	\$1.333

$\Rightarrow (x_1, x_2, x_3, x_4) = (1, 1, 0, \frac{2}{3})$, with $z = 21\frac{2}{3}$
$\Rightarrow$ No integer solution for this subproblem will have value lager than 21, but we don't have any feasible integer solution.

Solving the LP relaxation (LP(3))

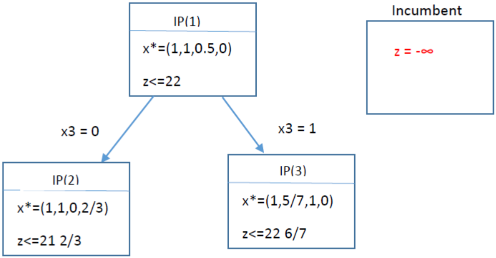
IP(3): $x_3 = 1$

$$\begin{array}{lll} \text{Max} & 8x_1 + 11x_2 + 6 \times 1 + 4x_4 & \\ \text{s.t.} & 5x_1 + 7x_2 + 4 \times 1 + 3x_4 \leq 14 & \text{LP(3)} \\ & 0 \leq x_i \leq 1 \text{ for } i = 1, 2, 4 & \end{array}$$

item	1	2	4
unit value	\$1.6	\$1.571	\$1.333

$\Rightarrow (x_1, x_2, x_3, x_4) = (1, \frac{5}{7}, 1, 0)$, with $z = 21\frac{6}{7}$
$\Rightarrow$ No integer solution for this subproblem will have value lager than 21, but we don't have any feasible integer solution.

Enumeration Tree

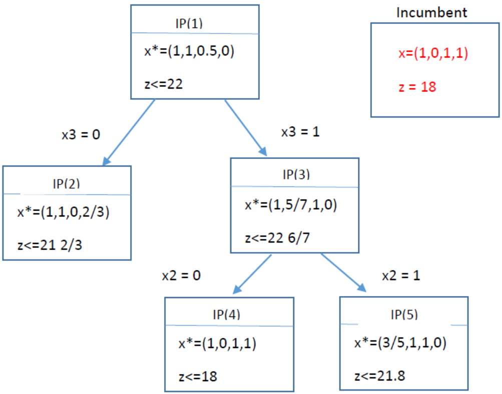


We do not have any feasible integer solution. So, we will take a subproblem and branch on one of its variables.
In general, we will choose the subproblem as follows:

- choose an active subproblem, which so far only means we have not chosen before, and
- choose the subproblem with the highes solution value (for maximization) (lowest for minimization).

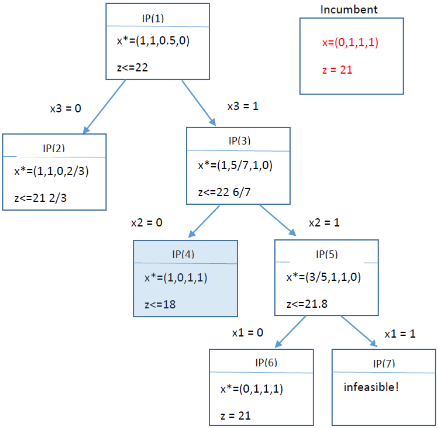
$\Rightarrow$ Choose IP(3) and branch on x_2.

Enumeration Tree



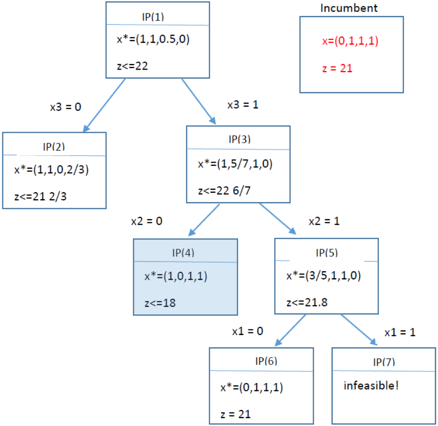
IP(4): Feasible integer solution with value 18. $\Rightarrow$ No further branching on subproblem 4 is needed.
$\Rightarrow$ Fathoming case 1: The optimal solution for its LP relaxation is *integer*.
$\Rightarrow$ If this solution is better than the incumbent, it becomes the new incumbent.

Enumeration Tree



IP(6): Feasible integer solution with value 21 (> 18).
$\Rightarrow$ No further branching on subproblem 6 is needed.
(Fathoming case 1).
$\Rightarrow$ Update the incumbent and its value.

Enumeration Tree

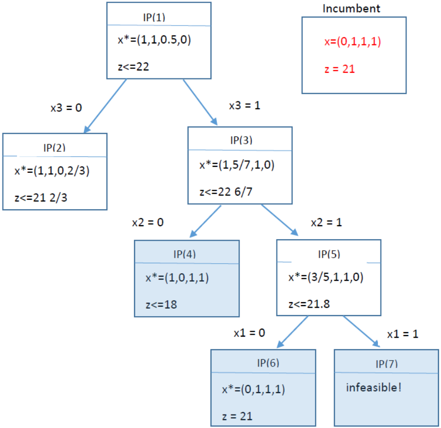


IP(7): Infeasible solution

⇒ No further branching on subproblem 7 is needed.

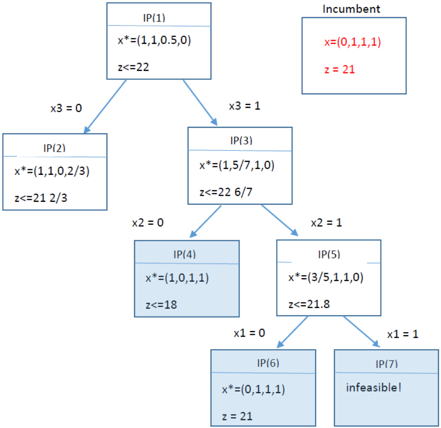
⇒ Fathoming case 2: The LP relaxation has no feasible solutions.

Enumeration Tree



Only one active subproblem left is subproblem 2 with its bound $= 21\frac{2}{3}$.
Since the optimal objective value of the original problem is integer, if the best value solution for a node is at most $21\frac{2}{3}$, then we know the best bound is at most 21. (Other bounds can also be rounded down.)

Enumeration Tree

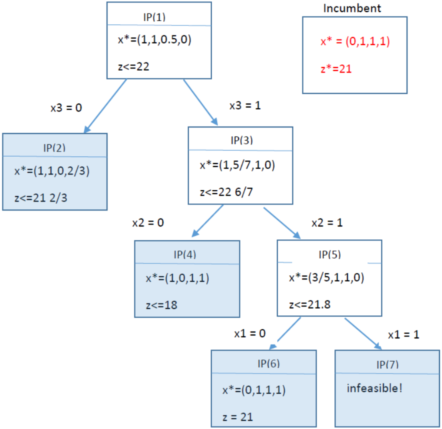


Only one active subproblem left is subproblem 2 with its bound = 21.

⇒ we fathom this subproblem.

⇒ Fathoming case 3: Subproblem's bound $\leq z$.

Enumeration Tree



There are no longer any active subproblems, so the optimal solution value is 21.

Branch-and-Bound Algorithm

- Branch-and-bound strategy:
  - Solve the linear relaxation of the problem.
    If the solution is integer, then we are done.
    Otherwise, create two new subproblems by branching on a fractional variable.
  - A node (subproblem) is not active when any of the following occurs:
    1. The node is being branched on;
    2. The solution is integral;
    3. The subproblem is infeasible;
    4. You can fathom the subproblem by a bounding argument.
  - Choose an active node and branch on a fractional variable.
    Repeat until there are no active subproblems.

A branch-and-bound algorithm for mixed integer programming

$$
\begin{aligned}
\text{Max} \quad & z = \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \le b_i, \ \text{for } i = 1, \dots, m \\
& x_j \ge 0, \ \text{for } j = 1, \dots, n \\
& x_j \text{ is integer}, \ \text{for } j = 1, \dots, I \quad (I \le n)
\end{aligned}
$$

Modifications:

- branching variables: only variables considered are the integer-restricted variables that have a noninteger value in the optimal solution for the LP relaxation of the current subproblem.
- values assigned to the branching variable for creating the new smaller subproblems:

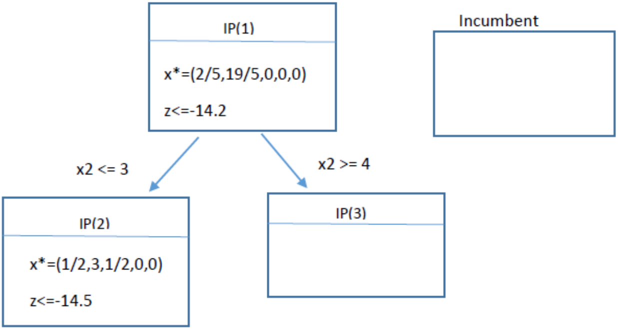
$$x_j \le \lfloor x_j^* \rfloor \text{ and } x_j \ge \lceil x_j^* \rceil$$

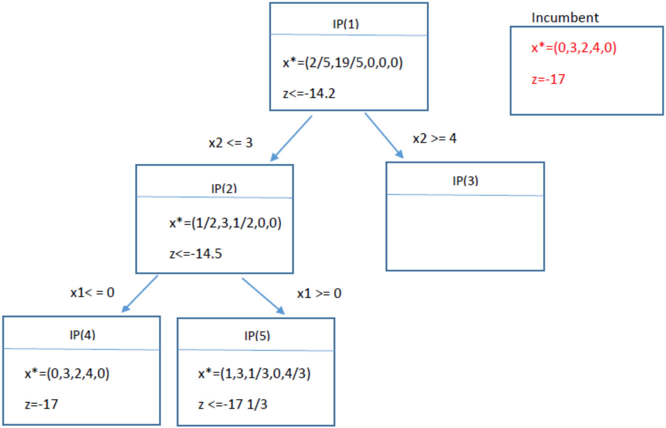
Example

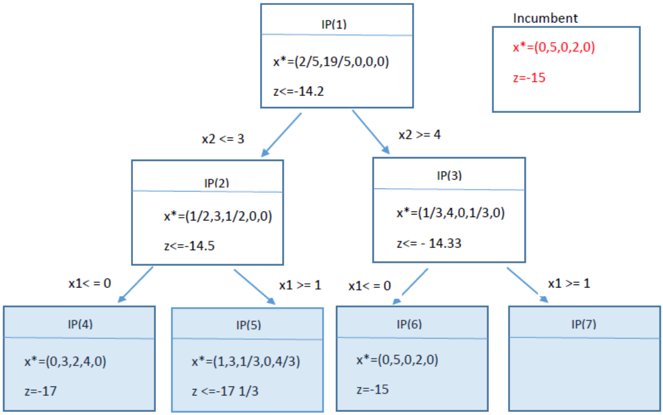
$$
\begin{array}{ll}
\max & z = -7x_1 - 3x_2 - 4x_3 \\
\text{s.t.} & \left\{\begin{array}{rcrcrcrcrcl}
x_1 & + & 2x_2 & + & 3x_3 & -x_4 & & = & 8, \\
3x_1 & + & x_2 & + & x_3 & & -x_5 & = & 5, \\
x_1, & & x_2, & & \cdots, & & x_5 & \geq & 0 \text{ and integer.}
\end{array}\right.
\end{array}
$$

IP(1)
x*=(2/5,19/5,0,0,0) z<=-71/5

Incumbent


IP(1)
x*=(2/5,19/5,0,0,0)
z<=-14.2
Incumbent
x2 <= 3
x2 >= 4
IP(2)
x*=(1/2,3,1/2,0,0)
z<=-14.5
IP(3)


IP(1)
x*=(2/5,19/5,0,0,0)
z<=-14.2
Incumbent
x*=(0,3,2,4,0)
z=-17
x2 <= 3
x2 >= 4
IP(2)
x*=(1/2,3,1/2,0,0)
z<=-14.5
IP(3)
x1< = 0
x1 >= 0
IP(4)
x*=(0,3,2,4,0)
z=-17
IP(5)
x*=(1,3,1/3,0,4/3)
z <=-17 1/3


IP(1)
x*=(2/5,19/5,0,0,0)
z<=-14.2
Incumbent
x*=(0,5,0,2,0)
z=-15
x2 <= 3
x2 >= 4
IP(2)
x*=(1/2,3,1/2,0,0)
z<=-14.5
IP(3)
x*=(1/3,4,0,1/3,0)
z<= - 14.33
x1< = 0
x1 >= 1
x1< = 0
x1 >= 1
IP(4)
x*=(0,3,2,4,0)
z=-17
IP(5)
x*=(1,3,1/3,0,4/3)
z <=-17 1/3
IP(6)
x*=(0,5,0,2,0)
z=-15
IP(7)

Lessons learned

- Branch-and-bound can speed up the search. - Only 7 nodes (LPs) out of 16 were evaluated.
- Branch-and-bound relies on eliminating subtrees, either because the IP at the node was solved, or else because the IP solution cannot possibly be optimum.
- Complete enumerations not possible (because the running time) if there are more than 100 variables. (Even 50 variables would take too long.)
- In practice, there are a lot of ways to make Branch-and-bound even faster.

How to Branch?

- We want to divide the current problem into two or more subproblems that are easier than the original. A commonly used branching method:

$$x_i \leq \lfloor x_i^* \rfloor, x_i \geq \lceil x_i^* \rceil$$

  where x_i^* is a fractional variable.

- Which variable to branch?
  A commonly used branching rule: Branch the most fractional variable.

- We would like to choose the branching that minimizes the sum of the solution times of all the created subproblems.

- How do we know how long it will take to solve each subproblem?
  Answer: We don't.
  Idea: Try to predict the difficulty of a subproblem.

- A good branching rule: The value of the linear programming relaxation changes a lot!

Which Node to Select?

- An important choice in branch and bound is the strategy for selecting the next subproblem to be processed.
- Goals:
  - Minimizing overall solution time.
  - Finding a good feasible solution quickly.
- Some commonly used search strategies:
  - Best First
  - Depth First
  - Hybrid Strategies
  - Best Estimate

The Best First Approach

- One way to minimize overall solution time is to try to minimize the size of the search tree. We can achieve this by choosing the subproblem with the best bound (lowest lower bound if we are minimizing).
- Drawbacks of Best First
  - Doesn’t necessarily find feasible solutions quickly since feasible solution are "more likely" to be found deep in the tree.
  - Node setup costs are high. The linear program being solved may change quite a bit from one node evaluation to the next.
  - Memory usage is high. It can require a lot of memory to store the candidate list, since the tree can grow "broad".

The Depth First Approach

- The depth first approach is to always choose the deepest node to process next. Just dive until you prune, then back up and go the other way.
- This avoids most of the problems with best first: The number of candidate nodes is minimized (saving memory). The node set-up costs are minimized.
- LPs change very little from one iteration to the next. Feasible solutions are usually found quickly.
- Drawback: If the initial lower bound is not very good, then we may end up processing lots of non-critical nodes.
- Hybrid Strategies: Go depth-first until you find a feasible solution, then do best first search.

Example

Consider the IP(1) in the previous example, an optimal LP tableau is obtained as in the following table:

	x_1	x_2	x_3	x_4	x_5	solution
x_1	1	0	$-\frac{1}{5}$	$\frac{1}{5}$	$-\frac{2}{5}$	$\frac{2}{5}$
x_2	0	1	$\frac{8}{5}$	$-\frac{3}{5}$	$\frac{1}{5}$	$\frac{19}{5}$
z	0	0	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{11}{5}$	$-\frac{71}{5}$

If the constraint $x_2 \leq 3$ is added, how can we obtain the new optimal solution from this optimal tableau?

Example

Consider the IP(1) in the previous example, an optimal LP tableau is obtained as in the following table:

	x_1	x_2	x_3	x_4	x_5	solution
x_1	1	0	$-\frac{1}{5}$	$\frac{1}{5}$	$-\frac{2}{5}$	$\frac{2}{5}$
x_2	0	1	$\frac{8}{5}$	$-\frac{3}{5}$	$\frac{1}{5}$	$\frac{19}{5}$
z	0	0	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{11}{5}$	$-\frac{71}{5}$

If the constraint $x_2 \leq 3$ is added, how can we obtain the new optimal solution from this optimal tableau?

Rewrite $x_2 \leq 3$ as $x_2 + s = 3$, where s is a slack variable. Hence $s = 3 - x_2 = 3 - (\frac{19}{5} - \frac{8}{5}x_3 - \left(-\frac{3}{5}\right) x_4 - \left(\frac{1}{5}\right) x_5)$, or $s - \frac{8}{5}x_3 + \frac{3}{5}x_4 - \frac{1}{5}x_5 = -\frac{4}{5}$. We add it to the tableau:

Complete Enumeration
Branch-and-Bound Algorithm

Example

$$s - \frac{8}{5}x_3 + \frac{3}{5}x_4 - \frac{1}{5}x_5 = -\frac{4}{5}$$

	x_1	x_2	s	x_3	x_4	x_5	solution
x_1	1	0	0	$-\frac{1}{4}$	$\frac{1}{5}$	$-\frac{2}{5}$	$\frac{2}{5}$
x_2	0	1	0	$\frac{8}{5}$	$-\frac{3}{5}$	$\frac{1}{5}$	$\frac{19}{5}$
s	0	0	1	$-\frac{8}{5}$	$\frac{3}{5}$	$-\frac{1}{5}$	$-\frac{4}{5}$
z	0	0	0	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{11}{5}$	$-\frac{71}{5}$

The tableau is optimal but not feasible as $s = -\frac{4}{5} < 0$.

We can use the dual simplex method to get the optimal solution.

Example

	x_1	x_2	s	x_3	x_4	x_5	solution
x_1	1	0	0	$-\frac{1}{4}$	$\frac{1}{5}$	$-\frac{2}{5}$	$\frac{2}{5}$
x_2	0	1	0	$\frac{8}{5}$	$-\frac{3}{5}$	$\frac{1}{5}$	$\frac{19}{5}$
s	0	0	1	$-\frac{8}{5}$	$\frac{3}{5}$	$-\frac{1}{5}$	$-\frac{4}{5}$
z	0	0	0	$\frac{3}{5}$	$\frac{2}{5}$	$\frac{11}{5}$	$-\frac{71}{5}$

Leaving variable: s
Entering variable: x_j is selected such that $|z_j/y_{js}|$ is the minimum amongst all $y_{js} < 0$. Since

$$\min\left\{\left|\frac{z_3}{y_{3s}}\right|, \left|\frac{z_5}{y_{5s}}\right|\right\} = \min\left\{\left|\frac{3/5}{-8/5}\right|, \left|\frac{11/5}{-1/5}\right|\right\} = \frac{3}{8}$$

$\Rightarrow$ The entering variable is x_3. Thus, we do pivot operation on $-\frac{8}{5}$.

Example

After the pivot operation on $-\frac{8}{5}$, we get

	x_1	x_2	s	x_3	x_4	x_5	solution
x_1	1	0	$-\frac{1}{8}$	0	$\frac{1}{8}$	$-\frac{3}{8}$	$\frac{1}{2}$
x_2	0	1	1	0	0	0	3
x_3	0	0	$-\frac{5}{8}$	1	$-\frac{3}{8}$	$\frac{1}{8}$	$\frac{1}{2}$
z	0	0	$\frac{3}{8}$	0	$\frac{5}{8}$	$\frac{17}{8}$	$-\frac{29}{2}$

which is both optimal and primal feasible. Hence we obtain the optimal solution $x_1 = \frac{1}{2}$, $x_2 = 3$, $x_3 = \frac{1}{2}$, $x_4 = x_5 = 0$ at node 1.

Rounding down to improve bounds

If all cost coefficients of a maximization problem are integer valued, then the optimal objective value (for the IP) is integer. And $z_{IP}(j) \leq \lfloor z_{LP}(j) \rfloor$.

A bad example

$$\begin{aligned} \text{Max} \quad & 2x_1 + 2x_2 + 2x_3 + \ldots + 2x_{100} \\ \text{s.t.} \quad & 2x_1 + 2x_2 + 2x_3 + \ldots + 2x_{100} \leq 101 \\ & x_i \in \{0, 1\} \text{ for } i = 1, 2, \ldots, 100. \end{aligned}$$

What would happen if we use branch-and-bound as described before?

Adding constraints to improve bounds

- A constraint is called a valid constraint if it is satisfied by all integer solutions of an IP (but possibly not the linear solution of its LP relaxation).
- Adding a valid inequality might improve the bound.

$$\begin{array}{rl} \text{Max} & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{s.t.} & 2x_1 + 2x_2 + 2x_3 + 2x_4 \leq 3 \\ & x_i \in \{0,1\} \text{ for } i = 1, \ldots, 4 \end{array} \qquad A$$

Opt LP(A)

$$x^* = (1, 0.5, 0, 0)$$
$$z_{LP} = 5.5$$

$$\begin{array}{rl} \text{Max} & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{s.t.} & x_1 + x_2 + x_3 + x_4 \leq 1.5 \\ & x_i \in \{0,1\} \text{ for } i = 1, \ldots, 4 \end{array} \qquad B$$

$$\begin{array}{rl} \text{Max} & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{s.t.} & x_1 + x_2 + x_3 + x_4 \leq 1 \\ & x_i \in \{0,1\} \text{ for } i = 1, \ldots, 4 \end{array} \qquad C$$

Opt LP(C)

$$x^* = (1, 0, 0, 0)$$
$$z_{LP} = 4$$

The solution for LP(C) is optimal for IP(A)!

Summary

- Making Branch-and-bound work well in practice requires lots of good ideas.
- There was not time in class to cover all of these ideas in any detail.
- The best idea for speeding up Branch-and-bound is to add valid inequalities, or improve the inequalities.