IM2010: Operations Research
Integer Programming
(Chapter 9)

Ling-Chieh Kung

Department of Information Management
National Taiwan University

April 11, 2013

Road map

- **Integer programming formulation**.
- Linear relaxation.
- Branch and bound.
- Branch and bound for knapsack.

Integer programming formulation

- In some cases, when variables should only take integer values, we apply integer programming.
- Moreover, we may introduce integer variables (mostly **binary variables**) to enrich our formulation and model more complicated situations.
- Here we will study some widely adopted integer programming formulation techniques.

The knapsack problem

- We start our illustration with the classic **knapsack** problem.
- There are four items to select:

Item	1	2	3	4
Value ($)	16	22	12	8
Weight(kg)	5	7	4	3

- The knapsack capacity is 10 kg.
- We want to maximize the total value without exceeding the knapsack capacity.

The knapsack problem: basic formulation

- Let the decision variables be

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is selected} \\ 0 & \text{o/w} \end{cases}.$$

- The knapsack constraint:

$$5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 10.$$

- The objective function:

$$\max \quad 16x_1 + 22x_2 + 12x_3 + 8x_4.$$

- The complete formulation:

$$\begin{array}{rrcrcrcrcl} \max & 16x_1 & + & 22x_2 & + & 12x_3 & + & 8x_4 & & \\ \text{s.t.} & 5x_1 & + & 7x_2 & + & 4x_3 & + & 3x_4 & \leq & 10 \\ & \multicolumn{9}{c}{x_i \in \{0,1\} \quad \forall i = 1, ..., 4.} \end{array}$$

Some more requirements

- Integer programming allows us to implement some special selection rules.
- At least/most some items:
  - Suppose we must select **at least** one item among items 2, 3, and 4:

$$x_2 + x_3 + x_4 \geq 1.$$

  - Suppose we must select **at most** two items among items 1, 3, and 4:

$$x_1 + x_3 + x_4 \leq 2$$

Some more requirements

- Or:
  - Select item 2 or item 3:

$$x_2 + x_3 \geq 1.$$

  - Select item 2, otherwise, items 3 and 4 togehter:

$$2x_2 + x_3 + x_4 \geq 2.$$

- If-else:
  - If item 2 is not selected, do not select item 3:

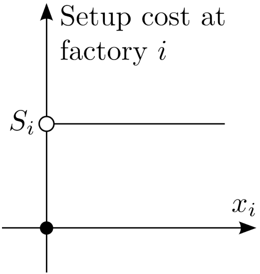
$$x_2 \geq x_3.$$

  - If item 1 is selected, do not select items 3 and 4:

$$2(1 - x_1) \geq x_3 + x_4.$$

Fixed-charge constraints

- Consider the following example:
- n factories, 1 market, 1 product.
  - Capacity of factory i: K_i.
  - Unit production cost at factory i: C_i.
  - **Setup cost** at factory i: S_i.
  - Demand: D.
  - We want to satisfy the demand with the minimum cost.
- One needs to pay the setup cost as long as any **positive** amount of products is produced.



Basic formulation

- Let the decision variables be

  x_i = production quantity at factory i, $i = 1, ..., n$,

  $$y_i = \begin{cases} 1 & \text{if some products are produced at factory } i,\ i = 1, ..., n. \\ 0 & \text{o/w.} \end{cases}$$

- Objective function:

$$\min \quad \sum_{i=1}^{n} C_i x_i + \sum_{i=1}^{n} S_i y_i.$$

- Capacity limitation:

$$x_i \leq K_i \quad \forall i = 1, ..., n.$$

- Demand fulfillment:

$$\sum_{i=1}^{n} x_i \geq D.$$

Setup costs

- How may we know whether we need to pay the setup cost at factory i?
  - If $x_i > 0$, y_i must be 1; if $x_i = 0$, y_i should be 0.
- So the relationship between x_i and y_i should be:

$$x_i \leq K_i y_i \quad \forall i = 1, ..., n.$$

  - If $x_i > 0$, y_i cannot be 0.
  - If $x_i = 0$, y_i can be 0 or 1. Why y_i will always be 0 when $x_i = 0$?
- Finally, binary and nonnegative constraints:

$$x_i \geq 0, y_i \in \{0, 1\} \quad \forall i = 1, ..., n.$$

Fixed-charge constraints

- The setup cost constraint $x_i \leq K_i y_i$ is known as a **fixed-charge constraint**.
- In general, a fixed-charge constraint is

$$x \leq My.$$

  - Both x and y are decision variables.
  - $y \in \{0, 1\}$ is determined by x.
  - M is a **large enough constant**.
  - When x is binary, $x \leq y$ is sufficient.
- We need to make M an **upper bound** of x.
  - For example, K_i is an upper bound of x_i in the factory example. Why?
  - What if there is no capacity limitation?

At least/most some constraints

- Using a similar technique, we may **flexibly** select constraints.
- Suppose satisfying one of the two constraints

$$g_1(x) \leq b_1 \quad \text{and} \quad g_2(x) \leq b_2$$

  is enough. How to formulate this situation?
- Let's define a binary variable

$$z = \begin{cases} 0 & \text{if } g_1(x) \leq b_1 \text{ is satisfied,} \\ 1 & \text{if } g_2(x) \leq b_2 \text{ is satisfied.} \end{cases}$$

- With M_i being an upper bound of each LHS, the following two constraints are what we need!

$$g_1(x) - b_1 \leq M_1 z$$
$$g_2(x) - b_2 \leq M_2(1 - z)$$

At least/most some constraints

- Suppose at least two of the three constraints

$$g_i(x) \leq b_i, \quad i = 1, 2, 3,$$

  must be satisfied. How to play the same trick?
- Let

$$z_i = \begin{cases} 1 & \text{if } g_i(x) \leq b_i \text{ must be satisfied,} \\ 0 & \text{if } g_i(x) \leq b_i \text{ may not be satisfied.} \end{cases}$$

- With M_i being an upper bound of each LHS, the following constraints are what we need:

$$\begin{aligned} & g_i(x) - b_i \leq M_i(1 - z_i) \quad \forall i = 1, ..., 3. \\ & z_1 + z_2 + z_3 \geq 2. \end{aligned}$$

If-else constraints

- In some cases, if $g_1(x) > b_1$ is satisfied, then $g_2(x) \leq b_2$ must also be satisfied.
- How to model this situation?
  - First, note that "if A then B" $\Leftrightarrow$ "(not A) or B".
  - So what we really want to do is $g_1(x) \leq b_1$ or $g_2(x) \leq b_2$.
  - So simply select at least one of $g_1(x) \leq b_1$ and $g_2(x) \leq b_2$!

Road map

- Integer programming formulation.
- Linear relaxation.
- Branch and bound.
- **Branch and bound for knapsack**.

Solving an integer program

- Suppose we are given an integer program, how may we solve it?
- The simplex method certainly does not work!
  - The feasible region is not "a" region.
  - It is not convex. In fact, it is discrete.
  - There is no way to "move along edges".
- But all we know is how to solve linear programs by the simplex method. How about solving a **linear relaxation** first?

Definition 1

For a given integer program, its linear relaxation is the resulting linear program after removing all the integer constraints.

Linear relaxation

- What is the linear relaxation of

$$\begin{array}{rrcrcr} \max & x_1 & + & x_2 & & \\ \text{s.t.} & x_1 & + & 3x_2 & \leq & 10 \\ & 2x_1 & - & x_2 & \geq & 5 \\ & \multicolumn{5}{c}{x_i \in \mathbb{Z}_+ \quad \forall i = 1, 2?} \end{array}$$

  - $\mathbb{Z}$ is the set of all integers. $\mathbb{Z}_+$ is the set of all nonnegative integers.
- The linear relaxation is

$$\begin{array}{rrcrcr} \max & x_1 & + & x_2 & & \\ \text{s.t.} & x_1 & + & 3x_2 & \leq & 10 \\ & 2x_1 & - & x_2 & \geq & 5 \\ & \multicolumn{5}{c}{x_i \geq 0 \quad \forall i = 1, 2.} \end{array}$$

Linear relaxation

- For the knapsack problem

$$\begin{array}{rccccccccc} \max & 16x_1 & + & 22x_2 & + & 12x_3 & + & 8x_4 & & \\ \text{s.t.} & 5x_1 & + & 7x_2 & + & 4x_3 & + & 3x_4 & \leq & 10 \\ & \multicolumn{9}{c}{x_i \in \{0,1\} \quad \forall i = 1, ..., 4,} \end{array}$$

  the linear relaxation is

$$\begin{array}{rccccccccc} \max & 16x_1 & + & 22x_2 & + & 12x_3 & + & 8x_4 & & \\ \text{s.t.} & 5x_1 & + & 7x_2 & + & 4x_3 & + & 3x_4 & \leq & 10 \\ & \multicolumn{9}{c}{x_i \in [0,1] \quad \forall i = 1, ..., 4,} \end{array}$$

  - $x_i \in [0,1]$ is equivalent to $x_i \geq 0$ and $x_i \leq 1$.

Linear relaxation provides a bound

- What kind of relationship do we have between an integer program and its linear relaxation?
- For a **minimization** integer program, the linear relaxation provides a **lower bound**.

Proposition 1

Let z^* and z' be the objective values associated to the optimal solutions of a minimization integer program and its linear relaxation, respectively, then $z' \leq z^*$.

Proof. The linear relaxation has the same objective function as the integer program does. However, its feasible region is at least weakly larger than that of the integer program. □

- For a **maximization** integer program, the linear relaxation provides an **upper bound**.

Linear relaxation may be optimal

- If we are lucky, the optimal solution to the linear relaxation may be **feasible** to the original integer program.
- When this happens, what does that imply?

Proposition 2

Let x' be the optimal solutions to the linear relaxation of an integer program. If x' is feasible to the integer program, it is optimal to the integer program.

Proof. Suppose x' is not optimal to the IP, there must be another feasible solution x'' that is better. However, as x'' is feasible to the IP, it is also feasible to the linear relaxation, which implies that x' cannot be optimal to the linear relaxation. $\square$

Linear relaxation

- In general, for any given mathematical program:
  - When we **relax** some constraints, a resulting optimal solution provides a **bound** to the original program.
  - If an optimal solution to the relaxed program is **feasible** to the original program, it is **optimal** to the original program.
- Therefore, one attempt of solving an integer program is to first solve its linear relaxation.
  - If we are **lucky** and get a solution feasible to the integer program, we can stop and report it!
  - What if the solution is not feasible to the integer program?
  - An optimal solution to the linear relaxation still provides some suggestions to our decision making.
  - If we really need an optimal solution to the integer program, how?

Rounding a fractional solution

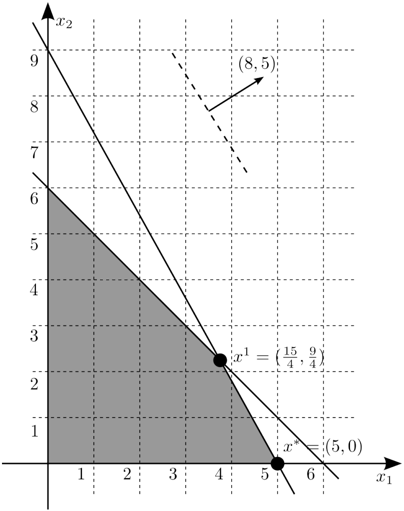
- Suppose we solve a linear relaxation with an optimal solution x'.
- x', however, has at least one variable violating the integer constraint in the original integer program.
- As we cannot find the true optimal solution x^* to the original integer program, we may choose to **round** the variable.
  - How do we know whether to round up or down?
  - Is the resulting solution always feasible?
  - Intuitively, the resulting solution should be close to x^*. Is it true?

Rounding a fractional solution

- Consider the following integer program

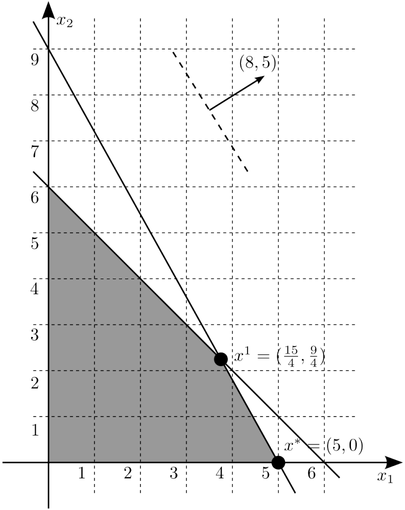
$$
(Q_0) \quad \begin{array}{rrcrcr}
\max & 8x_1 & + & 5x_2 & & \\
\text{s.t.} & x_1 & + & x_2 & \leq & 6 \\
 & 9x_1 & + & 5x_2 & \leq & 45 \\
 & \multicolumn{5}{c}{x_i \in \mathbb{Z}_+ \quad \forall i = 1, 2.}
\end{array}
$$

- The optimal solution is $x^* = (5, 0)$.
- The optimal solution to the linear relaxation is $x^1 = (\frac{15}{4}, \frac{9}{4})$.



Rounding a fractional solution

- For $x^1 = (\frac{15}{4}, \frac{9}{4})$:
  - Rounding up any variable results in infeasible solutions.
  - None of the four grip points around x' is optimal.
- We need a way that guarantees to find an optimal solution.
- The method we will introduce is the **branch-and-bound** algorithm.



Road map

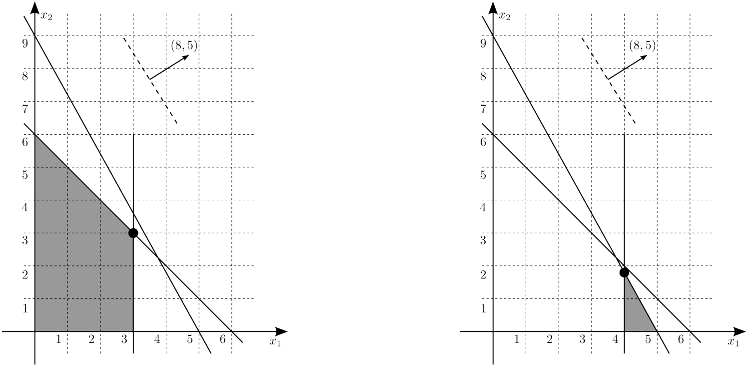
- Integer programming formulation.
- Linear relaxation.
- **Branch and bound**.
- Branch and bound for knapsack.

Rounding a fractional solution

- Recall that we obtain $x^1 = (\frac{15}{4}, \frac{9}{4})$. as the result of solving the linear relaxation.
  - We hate fractional values!
  - How may we remove fractional values?
- Consider x_1, for example.
  - Rounding up or down x_1 (i.e., adding $x_1 = 4$ or $x_1 = 3$ into the program) both **fail** to find the optimal solution.
  - Because we eliminate too large a search space!
  - Instead of adding equalities, we should add **inequalities**.
- What will happen if we add $x_1 \geq 4$ or $x_1 \leq 3$ into the program?
- We will **branch** this problem into two problems, one with an additional constraint.

Rounding a fractional solution

If we add $x_1 \leq 3$: If we add $x_1 \geq 4$:



- The optimal solution to the integer program must be contained in one of the above two feasible regions. Why?

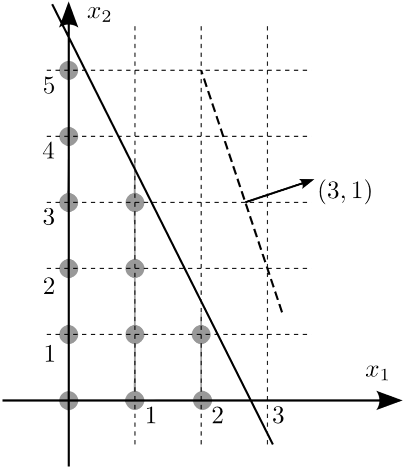
Rounding a fractional solution

- So when we solve the linear relaxation and find any variable fractional, we will **branch** this problem into two problems, one with an additional constraint.
- Note that the two new programs are still linear programs.
- Once we solved them:
  - If each of them results in a feasible solution to the original integer program, simply compare these two and choose the better one.
  - If any of them results in a variable violating the integer constraint, **branch** again on that variable.
  - Eventually compare all the feasible solutions we obtain.

Example

- Let's illustrate the branch-and-bound algorithm with the following example:

$$(P_0) \quad \begin{array}{rrcrcr} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & \multicolumn{5}{c}{x_i \in \mathbb{Z}_+ \quad \forall i = 1, 2.} \end{array}$$

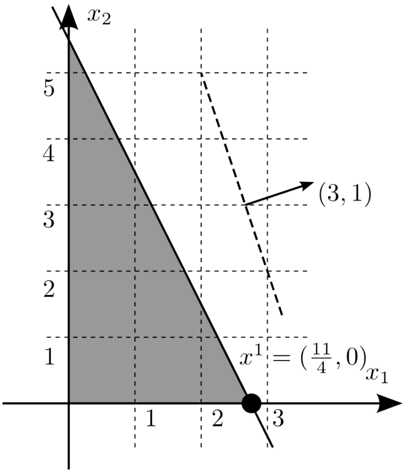


Subproblem 1

- First we solve the linear relaxation:

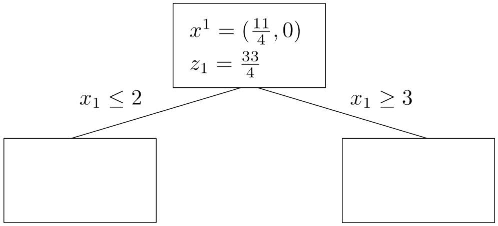
$$
(P_1) \quad \begin{array}{rrcrcr} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & \multicolumn{5}{l}{\color{blue}{x_i \geq 0} \quad \forall i = 1, 2.} \end{array}
$$

- The optimal solution is $x^1 = (\frac{11}{4}, 0)$.
- So we need to branch on x_1.



Branching tree

- The branch and bound algorithm produces a **branching tree**.
  - Each node represents a subproblem.
  - Each time we branch on a variable, we create two child nodes.

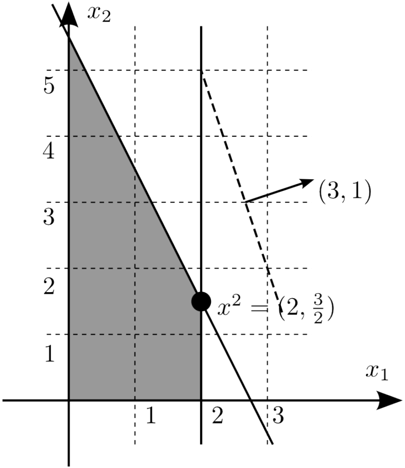


Subproblem 2

- When we add $x_1 \leq 2$:

$$
(P_2) \quad \begin{array}{rrcrcr}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
 & x_1 & & & \leq & 2 \\
 & \multicolumn{5}{l}{x_i \geq 0 \quad \forall i = 1, 2.}
\end{array}
$$

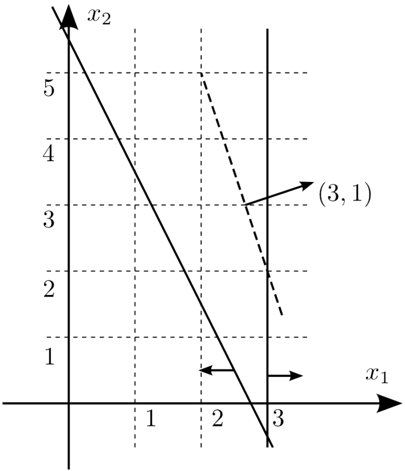
- The optimal solution $x^2 = (2, \frac{3}{2})$, so later we may need to branch on x_2.
- Before that, let's solve subproblem 3.



Subproblem 3

- When we add $x_1 \geq 3$:

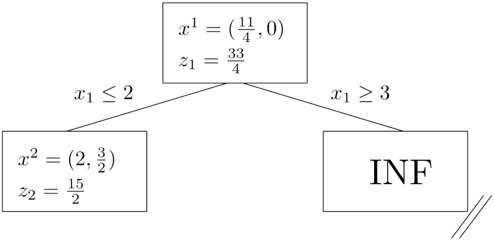
$$(P_3) \qquad \begin{array}{rrcrcr} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & \color{blue}{x_1} & & & \color{blue}{\geq} & \color{blue}{3} \\ & \multicolumn{5}{l}{x_i \geq 0 \quad \forall i = 1, 2.} \end{array}$$



- The problem is infeasible!
- This node is "dead" and does not produce any candidate solution.

Branching tree

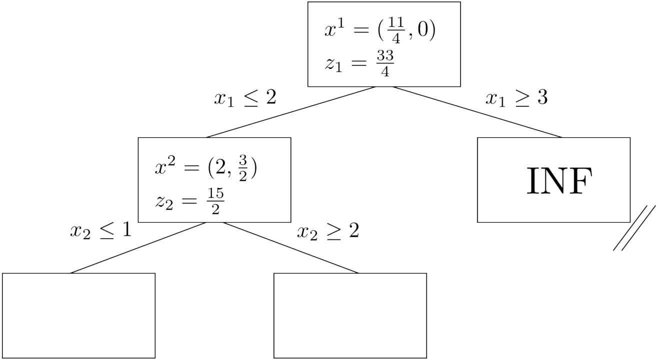
- The current progress can be summarized in the branching tree.



- Note that $z_2 = 7.5 < 8.25 = z_1$.
- In general, when we branch to the next level, the objective value associated with the optimal solution will **always** be weakly **lower** (for a maximization problem).
  - Why?

Branching tree

- As $x_2 = \frac{3}{2}$ in x^2, we will branch subproblem 2 on x_2.

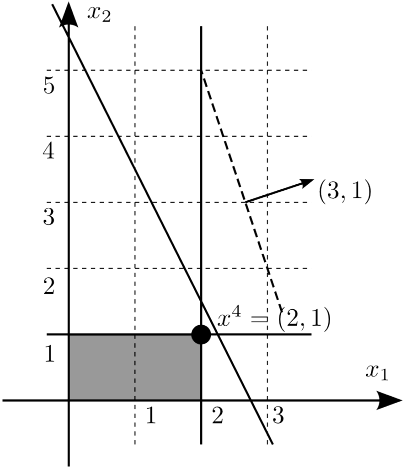


Subproblem 4

- When we add $x_2 \leq 1$:

$$
(P_4) \quad \begin{array}{rrcrcr}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
& x_1 & & & \leq & 2 \\
& & & \color{blue}{x_2} & \color{blue}{\leq} & \color{blue}{1} \\
& \multicolumn{5}{l}{x_i \geq 0 \quad \forall i = 1, 2.}
\end{array}
$$

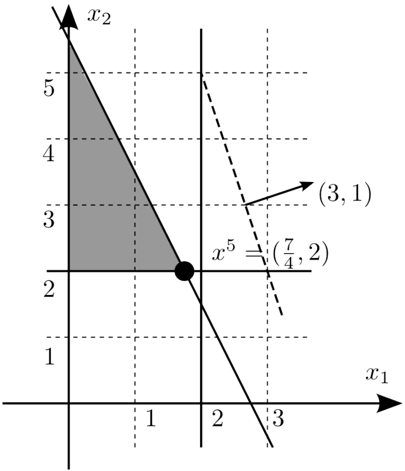
- Note that we add $x_2 \leq 1$ into subproblem 2, so $x_1 \leq 2$ is still there.



Subproblem 5

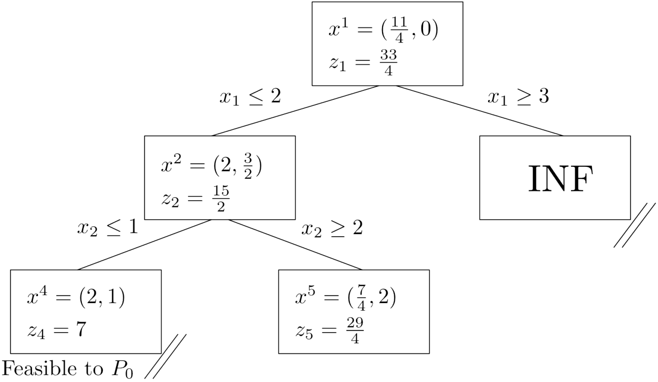
- When we add $x_2 \geq 2$:

$$(P_5) \quad \begin{array}{rrcrcr} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & x_1 & & & \leq & 2 \\ & & & \color{blue}{x_2} & \color{blue}{\geq} & \color{blue}{2} \\ & \multicolumn{5}{l}{x_i \geq 0 \quad \forall i = 1, 2.} \end{array}$$



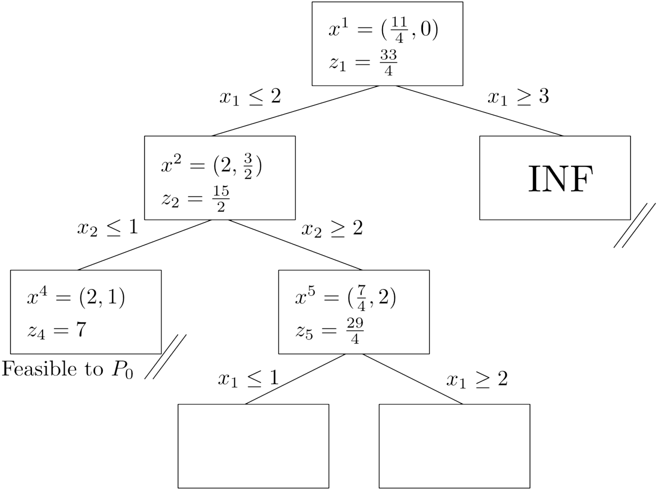
Branching tree

- x^4 satisfies all the integer constraints.
- It is a **candidate solution** to the original integer program.
- But branching subproblem 5 may result in a better solution.



Branching tree

- Let's branch subproblem 5 on x_1.

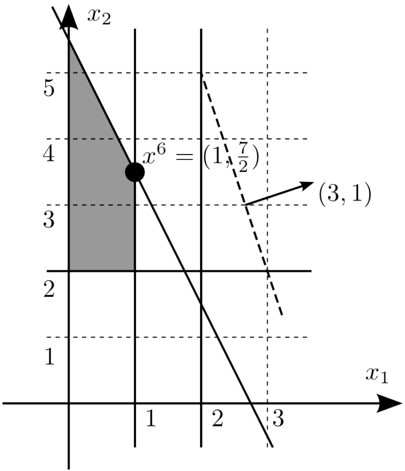


Subproblem 6

- When we add $x_1 \leq 1$:

$$
(P_6) \quad \begin{array}{rrcrcr}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
 & x_1 & & & \leq & 2 \\
 & & & x_2 & \geq & 2 \\
 & x_1 & & & \leq & 1 \\
 & x_i \geq 0 & & \forall i = 1, 2. & &
\end{array}
$$

- $x^6 = (1, \frac{7}{2})$. We may need to branch on x_2 again. However, let's solve subproblem 7 first.

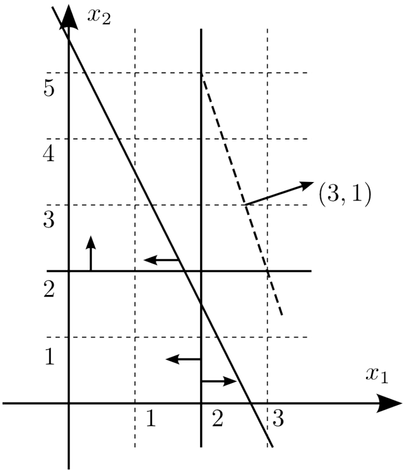


Subproblem 7

- When we add $x_1 \geq 2$:

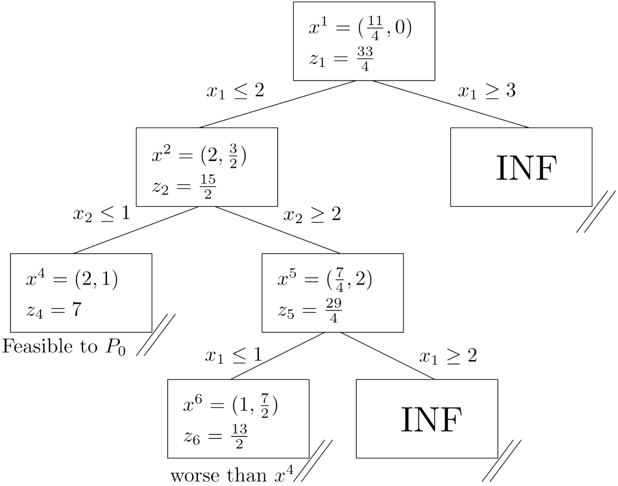
$$
(P_7) \quad
\begin{array}{rrcrcr}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
& x_1 & & & \leq & 2 \\
& & & x_2 & \geq & 2 \\
& x_1 & & & \geq & 2 \\
& x_i \geq 0 & & \forall i = 1, 2. & &
\end{array}
$$

- The problem is infeasible.
- The node is "dead".



Branching tree

- The only "alive" node is subproblem 6, with x_2 fractional.
- Before we branch subproblem 6, consider the following:



Bounding

- The current objective value of note 6 is $z_6 = \frac{13}{2}$.
- If we branch subproblem 6, all the candidate solution generated under it will have objective values weakly **lower** than $\frac{13}{2}$.
- However, $\frac{13}{2} < 7 = z_4$, and x_4 is already a candidate solution!
- So there is no need to branch subproblem 6. This is the "**bounding**" situation in the branch-and-bound algorithm.
  - This allows us to solve fewer subproblems.

Summary

- In running the branch-and-bound algorithm, we maintain a branching tree.
- If the solution of a subproblem is feasible to the original integer program, set it to the candidate solution if it is currently the best among all feasible solutions. Stop branching this node.
- If a subproblem is infeasible, stop branching this node.
- If the solution of a subproblem is not feasible to the original integer program:
  - If it is better than the current candidate solution, branch.
  - Otherwise, stop branching.

Another example

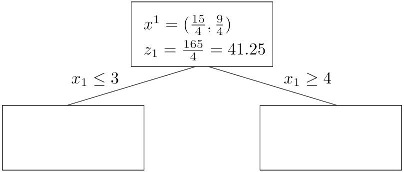
- Now let's go back to our motivating example:

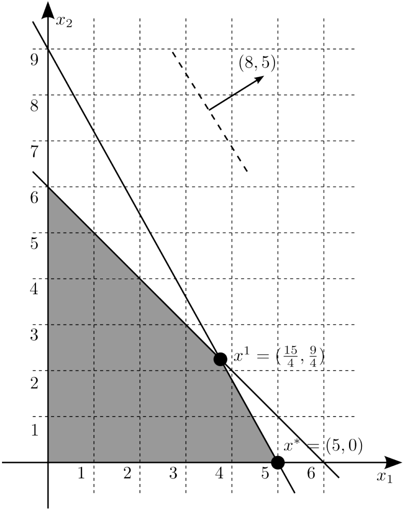
$$
(Q_0) \qquad \begin{array}{rrcrcr}
\max & 8x_1 & + & 5x_2 & & \\
\text{s.t.} & x_1 & + & x_2 & \leq & 6 \\
& 9x_1 & + & 5x_2 & \leq & 45 \\
& \multicolumn{5}{c}{x_i \in \mathbb{Z}_+ \quad \forall i = 1, 2.}
\end{array}
$$

- Let's solve it with the branch-and-bound algorithm.

Subproblem 1

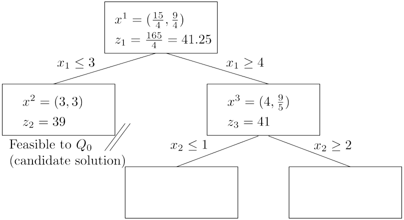
- $x^1 = (\frac{15}{4}, \frac{9}{4})$.
- We may branch on either variable. Let's branch on x_1.

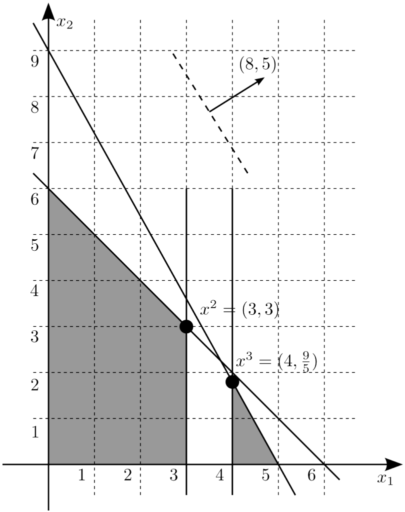




Subproblems 2 and 3

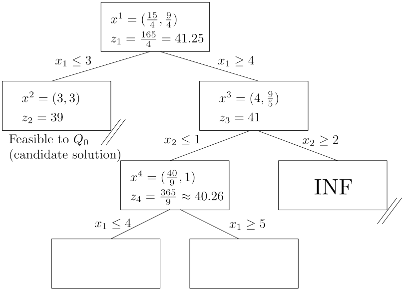
- Subproblem 2 generates a candidate solution.
- $x^3 = (4, \frac{9}{5})$. As $z_3 = 41 > z_2 = 39$, we should branch subproblem 3.

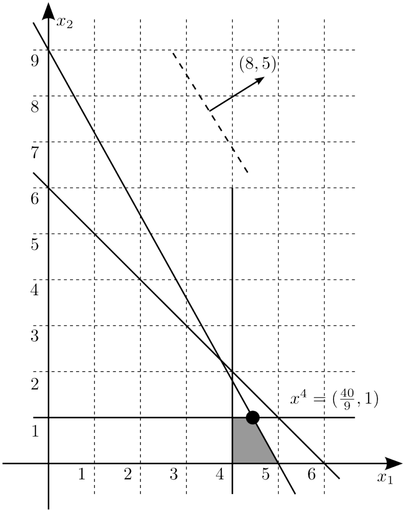




Subproblems 4 and 5

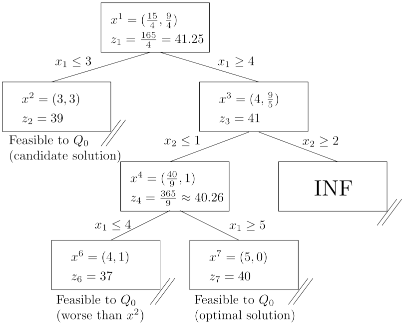
- $x^4 = (\frac{40}{9}, 1)$. As $z_4 = 40.25 > z_2 = 39$, we should branch subproblem 4.
- Subproblem 5 is infeasible.

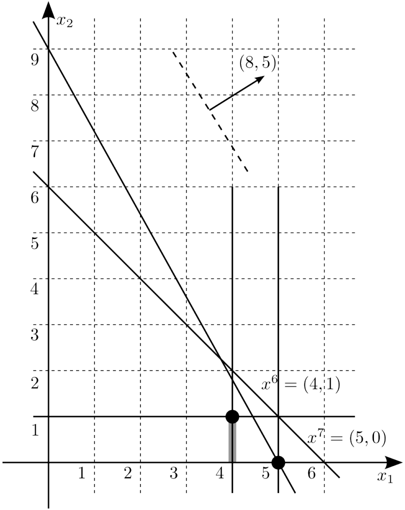




Subproblems 6 and 7

- $x^6 = (4, 1)$ but $z_6 = 37 < 39 = z_2$.
- $x^7 = (5, 0)$ and $z_7 = 40 > 39 = z_2$. As it is also the last node, x^7 is an optimal solution.





Remarks

- To select a node to branch:
  - Among all alive nodes, there are many different ways of selecting a node to branch.
  - One common approach is to branch the node with the highest objective value (for a maximization problem). Why?
  - Another popular approach is "once a node is branched, all its descendants are branched before any nondescendant. Why?
- The branch-and-bound algorithm guarantees to find an optimal solution, if one exists.
- However, it is an **exponential-time** algorithm.

Road map

- Integer programming formulation.
- Linear relaxation.
- Branch and bound.
- **Branch and bound for knapsack**.

Branch and bound for knapsack

- The branch-and-bound algorithm is particularly useful for solving the knapsack problem.
- Because the linear relaxation of a knapsack problem can be solved very easily.
- Consider the example

$$\begin{aligned}\max \quad & 5x_1 + 8x_2 + 3x_3 + 7x_4 \\ \text{s.t.} \quad & 3x_1 + 5x_2 + 2x_3 + 4x_4 \leq 6 \\ & x_i \in \{0, 1\} \quad \forall i = 1, ..., 4.\end{aligned}$$

How to solve its linear relaxation?

Branch and bound for knapsack

- The linear relaxation

$$
\begin{aligned}
\max \quad & 5x_1 + 8x_2 + 3x_3 + 7x_4 \\
\text{s.t.} \quad & 3x_1 + 5x_2 + 2x_3 + 4x_4 \leq 6 \\
& x_i \in [0, 1] \quad \forall i = 1, ..., 4.
\end{aligned}
$$

  can be solved **greedily** by sorting the variables according to the benefit-cost ratio.
  - The four ratios are $\frac{5}{3} \approx 1.67$, $\frac{8}{5} = 1.6$, $\frac{3}{2} = 1.5$, and $\frac{7}{4} = 1.75$.
  - x_4 has the highest priority then x_1, then x_2, then x_3.
  - First set $x_4 = 1$. Then set $x_1 = \frac{2}{3}$ (because setting $x_1 = 1$ violates the constraint). Then $x_2 = x_3 = 0$.
- Let's now use the branch-and-bound algorithm to solve this knapsack problem. For each node, we can use the above rule (instead of the simplex method) to find an optimal solution.

Solving the knapsack problem

$$
\begin{aligned}
\max \quad & 5x_1 + 8x_2 + 3x_3 + 7x_4 \\
\text{s.t.} \quad & 3x_1 + 5x_2 + 2x_3 + 4x_4 \leq 6, \quad x_i \in \{0, 1\} \quad \forall i = 1, ..., 4.
\end{aligned}
$$

- We branch subproblem 1 on x_1:
  - Note that $x_1 \leq 0$ is equivalent to $x_1 = 0$ for this binary variable.

$$\boxed{\begin{array}{l} x^1 = (\frac{2}{3}, 0, 0, 1) \\ z_1 = \frac{31}{2} \approx 10.33 \end{array}}$$

$x_1 = 0$:

$$\boxed{\begin{array}{l} x^2 = (0, \frac{2}{5}, 0, 1) \\ z_2 = \frac{51}{5} \approx 10.2 \end{array}}$$

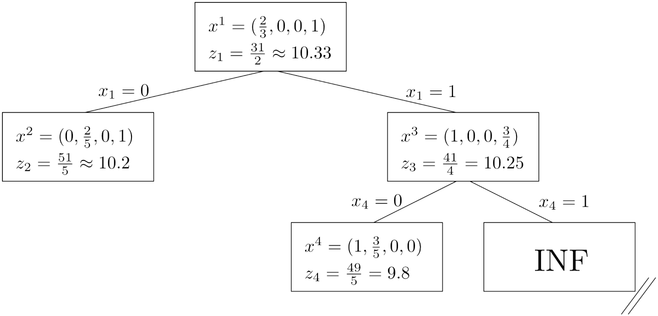
$x_1 = 1$:

$$\boxed{\begin{array}{l} x^3 = (1, 0, 0, \frac{3}{4}) \\ z_3 = \frac{41}{4} = 10.25 \end{array}}$$

Solving the knapsack problem

$$\begin{array}{rl} \max & 5x_1 + 8x_2 + 3x_3 + 7x_4 \\ \text{s.t.} & 3x_1 + 5x_2 + 2x_3 + 4x_4 \leq 6, \quad x_i \in \{0, 1\} \quad \forall i = 1, ..., 4. \end{array}$$

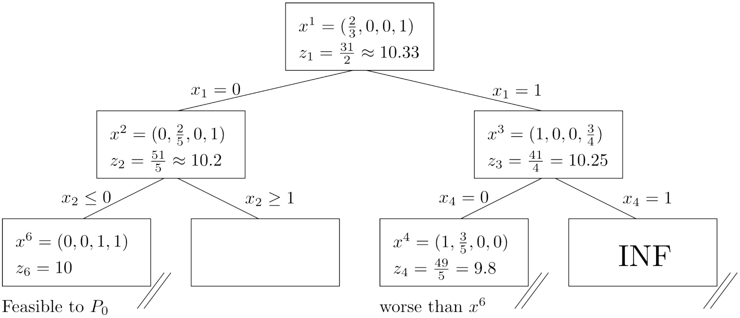
- We branch subproblem 3 first (why?)



Solving the knapsack problem

$$
\begin{array}{rl}
\max & 5x_1 + 8x_2 + 3x_3 + 7x_4 \\
\text{s.t.} & 3x_1 + 5x_2 + 2x_3 + 4x_4 \leq 6, \quad x_i \in \{0,1\} \quad \forall i = 1, ..., 4.
\end{array}
$$

- We branch subproblem 2 before we branch subproblem 4 (why?).
- Then, luckily, we will not need to branch subproblem 4.



Solving the knapsack problem

$$\begin{array}{rl} \max & 5x_1 + 8x_2 + 3x_3 + 7x_4 \\ \text{s.t.} & 3x_1 + 5x_2 + 2x_3 + 4x_4 \leq 6, \quad x_i \in \{0,1\} \quad \forall i = 1, ..., 4. \end{array}$$

- We do not need to branch subproblem 7.
- An optimal solution is found.