CS498: Algorithmic Engineering

Lecture 5: From LP Relaxations to Integer Solutions

Elfarouk Harb

University of Illinois Urbana-Champaign

Week 03 – 02/03/2026

Outline

1. Setup and Motivation
2. From LPs to Integer Programs
3. Example: 0–1 Knapsack
4. Before Branch and Bound: The Naïve Approach
5. Branch and Bound: The Core Idea

1. Setup and Motivation
2. From LPs to Integer Programs
3. Example: 0–1 Knapsack
4. Before Branch and Bound: The Naïve Approach
5. Branch and Bound: The Core Idea

Where We Are in the Course

Week 02:

- Linear programming in depth – duality, sensitivity, LP as approximation.
- LP relaxations for Assignment (exact), Vertex Cover (2-approx) and Independent Set (huge gap).
- Assignment problem was special: LP = exact integer solution.

Where We Are in the Course

Week 02:

- Linear programming in depth – duality, sensitivity, LP as approximation.
- LP relaxations for Assignment (exact), Vertex Cover (2-approx) and Independent Set (huge gap).
- Assignment problem was special: LP = exact integer solution.

This week:

- We want exact integer solutions, not approximations.
- We'll see how solvers *enforce integrality*.
- And why the strength of formulation decides speed.

Motivation: Relaxing vs Enforcing Integrality

Recall Vertex Cover LP:

$$\min \sum_{v} x_v \quad s.t.\ x_u + x_v \geq 1\ \forall (u, v) \in E,\ 0 \leq x_v \leq 1.$$

Motivation: Relaxing vs Enforcing Integrality

Recall Vertex Cover LP:

$$\min \sum_v x_v \quad s.t.\ x_u + x_v \geq 1\ \forall (u, v) \in E,\ 0 \leq x_v \leq 1.$$

LP gave fractional values like $x_v = 0.5$.
But our true decision was binary (cover vertex or not).

Motivation: Relaxing vs Enforcing Integrality

Recall Vertex Cover LP:

$$\min \sum_v x_v \quad s.t.\ x_u + x_v \geq 1\ \forall (u,v) \in E,\ 0 \leq x_v \leq 1.$$

LP gave fractional values like $x_v = 0.5$.
But our true decision was binary (cover vertex or not).
Idea: Force $x_v \in \{0,1\} \implies$ an **Integer Program**.

Motivation: Relaxing vs Enforcing Integrality

Recall Vertex Cover LP:

$$\min \sum_v x_v \quad s.t.\ x_u + x_v \geq 1\ \forall (u, v) \in E,\ 0 \leq x_v \leq 1.$$

LP gave fractional values like $x_v = 0.5$.
But our true decision was binary (cover vertex or not).
Idea: Force $x_v \in \{0, 1\} \implies$ an **Integer Program**.
LP = “relax your morals.”
IP = “follow the rules exactly.”

1 Setup and Motivation

2 From LPs to Integer Programs

3 Example: 0–1 Knapsack

4 Before Branch and Bound: The Naïve Approach

5 Branch and Bound: The Core Idea

Integer Linear Programs (Definition)

$$
\begin{aligned}
&\min\ c^\top x\\
&\text{s.t. } Ax \le b,\\
&\qquad x_i \in \mathbb{Z} \text{ for } i \in I.
\end{aligned}
$$

Integer Linear Programs (Definition)

$$
\begin{aligned}
&\min\ c^\top x \\
&\text{s.t. } Ax \le b, \\
&\qquad x_i \in \mathbb{Z} \text{ for } i \in I.
\end{aligned}
$$

Special cases:

- All x_i continuous or $I = \emptyset \rightarrow$ LP.
- $x_i \in \{0, 1\}$ for $i \in I \rightarrow$ **Binary Integer Linear Program**.

Integer Linear Programs (Definition)

$$\begin{aligned} &\min\ c^\top x \\ &\text{s.t. } Ax \le b, \\ &\qquad x_i \in \mathbb{Z} \text{ for } i \in I. \end{aligned}$$

Special cases:

- All x_i continuous or $I = \emptyset \to$ LP.
- $x_i \in \{0, 1\}$ for $i \in I \to$ **Binary Integer Linear Program**.

Usually we only need 0/1 decisions; other integers can be counted in bits.

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Example 1: Enumerated values

$$x \in \{-2, -1, 1, 2\}$$

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Example 1: Enumerated values

$$x \in \{-2, -1, 1, 2\}$$

represented by binaries $y_{-2}, y_{-1}, y_1, y_2 \in \{0, 1\}$:

$$y_{-2} + y_{-1} + y_1 + y_2 = 1, \qquad x = -2y_{-2} - y_{-1} + y_1 + 2y_2.$$

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Example 1: Enumerated values

$$x \in \{-2, -1, 1, 2\}$$

represented by binaries $y_{-2}, y_{-1}, y_1, y_2 \in \{0, 1\}$:

$$y_{-2} + y_{-1} + y_1 + y_2 = 1, \qquad x = -2y_{-2} - y_{-1} + y_1 + 2y_2.$$

Ensures exactly one discrete value of x is chosen.

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Example 1: Enumerated values

$$x \in \{-2, -1, 1, 2\}$$

represented by binaries $y_{-2}, y_{-1}, y_1, y_2 \in \{0, 1\}$:

$$y_{-2} + y_{-1} + y_1 + y_2 = 1, \qquad x = -2y_{-2} - y_{-1} + y_1 + 2y_2.$$

Ensures exactly one discrete value of x is chosen.

Example 2: Large integer range

$$x \in \{0, \dots, 20\} \quad \Rightarrow \quad x = y_1 + 2y_2 + 4y_3 + 8y_4 + 16y_5, \quad y_i \in \{0, 1\} \text{ and } x \leq 20.$$

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Example 1: Enumerated values

$$x \in \{-2, -1, 1, 2\}$$

represented by binaries $y_{-2}, y_{-1}, y_1, y_2 \in \{0, 1\}$:

$$y_{-2} + y_{-1} + y_1 + y_2 = 1, \qquad x = -2y_{-2} - y_{-1} + y_1 + 2y_2.$$

Ensures exactly one discrete value of x is chosen.

Example 2: Large integer range

$$x \in \{0, \dots, 20\} \quad \Rightarrow \quad x = y_1 + 2y_2 + 4y_3 + 8y_4 + 16y_5, \quad y_i \in \{0, 1\} \text{ and } x \leq 20.$$

Uses binary digits (“bits”) to represent the integer efficiently.

Representing General Integers with Binaries

Even if a variable takes several integer values, solvers internally reduce it to binary decisions.

Example 1: Enumerated values

$$x \in \{-2, -1, 1, 2\}$$

represented by binaries $y_{-2}, y_{-1}, y_1, y_2 \in \{0, 1\}$:

$$y_{-2} + y_{-1} + y_1 + y_2 = 1, \qquad x = -2y_{-2} - y_{-1} + y_1 + 2y_2.$$

Ensures exactly one discrete value of x is chosen.

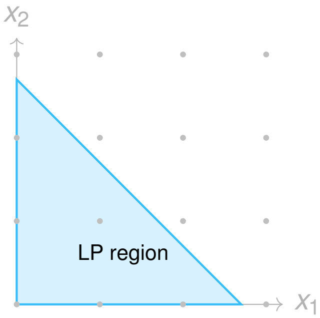
Example 2: Large integer range

$$x \in \{0, \dots, 20\} \quad \Rightarrow \quad x = y_1 + 2y_2 + 4y_3 + 8y_4 + 16y_5, \quad y_i \in \{0, 1\} \text{ and } x \leq 20.$$

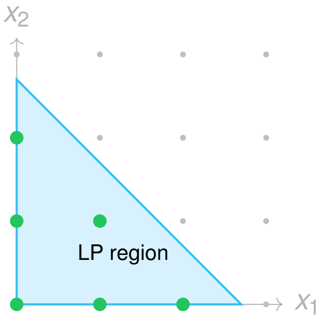
Uses binary digits (“bits”) to represent the integer efficiently.

Key point: Solvers enforce integrality through binary representations, so focusing on **binary programs** is sufficient in theory and practice.

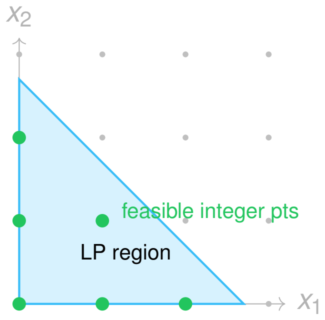
Geometry of IPs



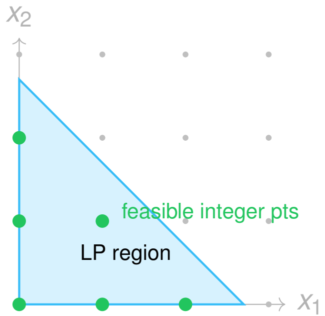
Geometry of IPs



Geometry of IPs



Geometry of IPs



Key idea:

- LP feasible set = convex polytope.
- IP feasible set = subset of lattice points.

Relaxation Relationship

Let F_{IP} be feasible region for IP. Let F_{LP} be feasible region for LP.

For a minimization problem:

$$\text{LP}^* = \min_{x \in F_{LP}} c^\top x \leq \min_{x \in F_{IP}} c^\top x = \text{IP}^*.$$

Relaxation Relationship

Let F_{IP} be feasible region for IP. Let F_{LP} be feasible region for LP.

For a minimization problem:

$$\text{LP}^* = \min_{x \in F_{LP}} c^\top x \leq \min_{x \in F_{IP}} c^\top x = \text{IP}^*.$$

For maximization problem:

$$\text{LP}^* = \max_{x \in F_{LP}} c^\top x \geq \max_{x \in F_{IP}} c^\top x = \text{IP}^*.$$

Relaxation Relationship

Let F_{IP} be feasible region for IP. Let F_{LP} be feasible region for LP.

For a minimization problem:

$$\text{LP}^* = \min_{x \in F_{LP}} c^\top x \leq \min_{x \in F_{IP}} c^\top x = \text{IP}^*.$$

For maximization problem:

$$\text{LP}^* = \max_{x \in F_{LP}} c^\top x \geq \max_{x \in F_{IP}} c^\top x = \text{IP}^*.$$

Meaning: LP solution is an **optimistic bound** on what's achievable with integers.

Relaxation Relationship

Let F_{IP} be feasible region for IP. Let F_{LP} be feasible region for LP.

For a minimization problem:

$$\mathrm{LP}^* = \min_{x \in F_{LP}} c^\top x \leq \min_{x \in F_{IP}} c^\top x = \mathrm{IP}^*.$$

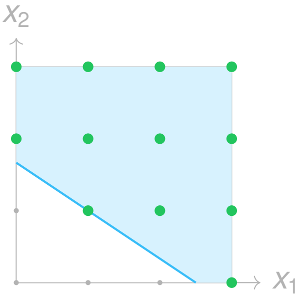
For maximization problem:

$$\mathrm{LP}^* = \max_{x \in F_{LP}} c^\top x \geq \max_{x \in F_{IP}} c^\top x = \mathrm{IP}^*.$$

Meaning: LP solution is an **optimistic bound** on what's achievable with integers. Sometimes bound = exact (answer is already integral, like assignment problem). Sometimes it's not $\rightarrow$ we need to *search.*

A Tiny Illustrative IP

$$\begin{aligned}
\min \quad & x_1 + x_2 \\
\text{s.t.} \quad & 2x_1 + 3x_2 \geq 5, \\
& 0 \leq x_i \leq 3, \quad i = 1, 2, \\
& x_1, x_2 \in \mathbb{Z}.
\end{aligned}$$

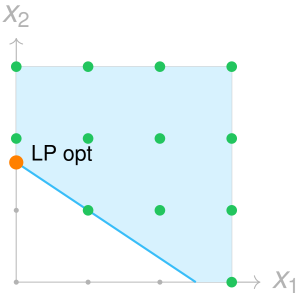


A Tiny Illustrative IP

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & 2x_1 + 3x_2 \geq 5, \\ & 0 \leq x_i \leq 3, \quad i = 1, 2, \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

LP solution:

$x_1 = 0,\ x_2 = \frac{5}{3},\ \text{obj} = \frac{5}{3} \approx 1.67.$



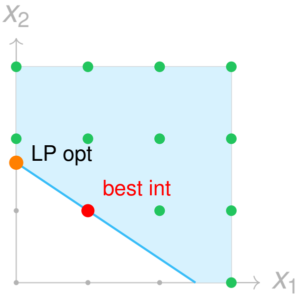
A Tiny Illustrative IP

$$\begin{aligned}
\min \quad & x_1 + x_2 \\
\text{s.t.} \quad & 2x_1 + 3x_2 \geq 5, \\
& 0 \leq x_i \leq 3, \quad i = 1, 2, \\
& x_1, x_2 \in \mathbb{Z}.
\end{aligned}$$

LP solution:

$x_1 = 0, \; x_2 = \frac{5}{3}, \; \text{obj} = \frac{5}{3} \approx 1.67.$

Best integer solution: $(x_1, x_2) = (1, 1)$ (or $(0, 2)$) $\Rightarrow \text{obj} = 2.$



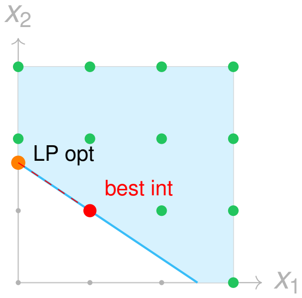
A Tiny Illustrative IP

$$\begin{aligned} \min \quad & x_1 + x_2 \\ \text{s.t.} \quad & 2x_1 + 3x_2 \geq 5, \\ & 0 \leq x_i \leq 3, \quad i = 1, 2, \\ & x_1, x_2 \in \mathbb{Z}. \end{aligned}$$

LP solution:
$x_1 = 0,\ x_2 = \frac{5}{3},\ \text{obj} = \frac{5}{3} \approx 1.67.$

Best integer solution: $(x_1, x_2) = (1, 1)$ (or $(0, 2)$) $\Rightarrow$ obj $= 2.$

Additive gap $= 2 - \frac{5}{3} = \frac{1}{3} \approx 0.33 \Rightarrow$ we'll use this gap as a *bound*.



1 Setup and Motivation

2 From LPs to Integer Programs

3 Example: 0–1 Knapsack

4 Before Branch and Bound: The Naïve Approach

5 Branch and Bound: The Core Idea

Knapsack Problem

You own a truck with capacity $W = 4$.

Knapsack Problem

You own a truck with capacity $W = 4$.

Items:

Item	Value v_i	Weight w_i
1	2	1
2	2	2
3	5	2
4	6	3

Knapsack Problem

You own a truck with capacity $W = 4$.

Items:

Item	Value v_i	Weight w_i
1	2	1
2	2	2
3	5	2
4	6	3

Which items should we pack to maximize value without exceeding capacity?

Variables and Model (1/2)

Define binary decision variables:

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$$

Variables and Model (1/2)

Define binary decision variables:

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$$

Constraint: total weight ≤ 4

$$x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4.$$

Variables and Model (1/2)

Define binary decision variables:

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is chosen,} \\ 0 & \text{otherwise.} \end{cases}$$

Constraint: total weight ≤ 4

$$x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4.$$

Objective:

$$\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4.$$

Bruteforce Attempt (2/2)

$$\begin{aligned}
&\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4 \\
&\text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \le 4, \\
&\qquad x_i \in \{0, 1\}.
\end{aligned}$$

Bruteforce Attempt (2/2)

$$\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4$$
$$\text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4,$$
$$x_i \in \{0, 1\}.$$

Feasible integer solutions (respect weight ≤ 4):

(x_1, x_2, x_3, x_4)	Weight	Value
$(0,0,0,0)$	0	0
$(0,0,0,1)$	3	6
$(0,0,1,0)$	2	5
$(0,1,0,0)$	2	2
$(0,1,1,0)$	4	7
$(1,0,0,0)$	1	2
$(1,0,0,1)$	4	8
$(1,0,1,0)$	3	7
$(1,1,0,0)$	3	4

Bruteforce Attempt (2/2)

$$\begin{aligned} \max\ & 2x_1 + 2x_2 + 5x_3 + 6x_4 \\ \text{s.t. } & x_1 + 2x_2 + 2x_3 + 3x_4 \le 4, \\ & x_i \in \{0, 1\}. \end{aligned}$$

Examples of infeasible (too heavy) combinations:

(x_1, x_2, x_3, x_4)	Weight	Value
$(0, 0, 1, 1)$	5	11 (too heavy)
$(0, 1, 0, 1)$	5	8 (too heavy)
$(1, 1, 1, 1)$	8	15 (too heavy)

Bruteforce Attempt (2/2)

$$\begin{aligned} \max\ & 2x_1 + 2x_2 + 5x_3 + 6x_4 \\ \text{s.t.}\ & x_1 + 2x_2 + 2x_3 + 3x_4 \le 4, \\ & x_i \in \{0, 1\}. \end{aligned}$$

Examples of infeasible (too heavy) combinations:

(x_1, x_2, x_3, x_4)	Weight	Value
$(0, 0, 1, 1)$	5	11 (too heavy)
$(0, 1, 0, 1)$	5	8 (too heavy)
$(1, 1, 1, 1)$	8	15 (too heavy)

Best feasible integer solution:

$$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1) \quad \Rightarrow \quad \text{value } = 8.$$

1 Setup and Motivation

2 From LPs to Integer Programs

3 Example: 0–1 Knapsack

4 Before Branch and Bound: The Naïve Approach

5 Branch and Bound: The Core Idea

Brute Force Search Procedure

1. Generate all possible assignments $x \in \{0, 1\}^k$.
2. Check feasibility of each against constraints.
3. Keep the feasible one with best objective.

Brute Force Search Procedure

1. Generate all possible assignments $x \in \{0, 1\}^k$.
2. Check feasibility of each against constraints.
3. Keep the feasible one with best objective.

Pseudo-code:

```
best_val = float("-inf")
best_x = None
for x in itertools.product([0,1], repeat=k):
  if is_feasible(x):
      val = objective(x)
      if val > best_val:
          best_val = val
          best_x = x
```

Brute Force Search Procedure

1. Generate all possible assignments $x \in \{0, 1\}^k$.
2. Check feasibility of each against constraints.
3. Keep the feasible one with best objective.

Pseudo-code:

```
best_val = float("-inf")
best_x = None
for x in itertools.product([0,1], repeat=k):
  if is_feasible(x):
      val = objective(x)
      if val > best_val:
          best_val = val
          best_x = x
```

For k binary variables, that's 2^k possibilities.
Works fine for $k \leq 20 \ldots$ catastrophic after that.
Exponential time: even $k = 50$ gives 1.1×10^{15} cases.

Why It's Too Slow

- Number of subproblems grows exponentially in number of integer vars.
- Most subproblems are infeasible or clearly bad.
- We're wasting time exploring hopeless regions.

Why It's Too Slow

- Number of subproblems grows exponentially in number of integer vars.
- Most subproblems are infeasible or clearly bad.
- We're wasting time exploring hopeless regions.

Idea for improvement:

- Don't check every integer vector.
- Use LP relaxation to *bound* the best achievable value in a region.
- Explore only promising regions $\rightarrow$ **Branch and Bound!**

Why It's Too Slow

- Number of subproblems grows exponentially in number of integer vars.
- Most subproblems are infeasible or clearly bad.
- We're wasting time exploring hopeless regions.

Idea for improvement:

- Don't check every integer vector.
- Use LP relaxation to *bound* the best achievable value in a region.
- Explore only promising regions $\rightarrow$ **Branch and Bound!**

So B&B = "smart brute force guided by LPs."

1 Setup and Motivation

2 From LPs to Integer Programs

3 Example: 0–1 Knapsack

4 Before Branch and Bound: The Naïve Approach

5 Branch and Bound: The Core Idea

LP Relaxation

Relax integrality: replace $x_i \in \{0, 1\}$ with $0 \leq x_i \leq 1$.

$$\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4 \quad \text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4.$$

LP Relaxation

Relax integrality: replace $x_i \in \{0, 1\}$ with $0 \leq x_i \leq 1$.

$$\max \; 2x_1 + 2x_2 + 5x_3 + 6x_4 \quad \text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4.$$

Fractional solution:

$$x_1 = 1, \; x_2 = 0, \; x_3 = 1, \; x_4 = \tfrac{1}{3}, \qquad z_{\text{LP}} = 9.$$

LP Relaxation

Relax integrality: replace $x_i \in \{0, 1\}$ with $0 \leq x_i \leq 1$.

$$\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4 \quad \text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \leq 4.$$

Fractional solution:

$$x_1 = 1,\ x_2 = 0,\ x_3 = 1,\ x_4 = \tfrac{1}{3}, \qquad z_{\text{LP}} = 9.$$

LP provides an optimistic upper bound of 9: “You can’t do better than 9!”

When LP Is Not Enough

Relaxed LP gives:

- an **upper bound** on the integer optimum (for maximization),
- and a fractional solution x^*.

When LP Is Not Enough

Relaxed LP gives:

- an **upper bound** on the integer optimum (for maximization),
- and a fractional solution x^*.

If x^* is integer — done!

When LP Is Not Enough

Relaxed LP gives:

- an **upper bound** on the integer optimum (for maximization),
- and a fractional solution x^*.

If x^* is integer — done!

If not, we must somehow **enforce integrality** by exploring discrete choices.

When LP Is Not Enough

Relaxed LP gives:

- an **upper bound** on the integer optimum (for maximization),
- and a fractional solution x^*.

If x^* is integer — done!

If not, we must somehow **enforce integrality** by exploring discrete choices.

This is where **Branch and Bound (B&B)** enters.

Branch and Bound: Conceptual Loop

Algorithmic skeleton:

1. Solve LP relaxation $\rightarrow$ gives bound z_{LP} and solution x^*.

Branch and Bound: Conceptual Loop

Algorithmic skeleton:

1. Solve LP relaxation $\rightarrow$ gives bound z_{LP} and solution x^*.
2. If x^* integral $\rightarrow$ update best integer solution (incumbent).

Branch and Bound: Conceptual Loop

Algorithmic skeleton:

1. Solve LP relaxation $\rightarrow$ gives bound z_{LP} and solution x^*.
2. If x^* integral $\rightarrow$ update best integer solution (incumbent).
3. Else, pick a fractional variable x_j.

Branch and Bound: Conceptual Loop

Algorithmic skeleton:

1. Solve LP relaxation $\rightarrow$ gives bound z_{LP} and solution x^*.
2. If x^* integral $\rightarrow$ update best integer solution (incumbent).
3. Else, pick a fractional variable x_j.
4. Create two branches:

$$x_j = 0, \quad x_j = 1.$$

Branch and Bound: Conceptual Loop

Algorithmic skeleton:

1. Solve LP relaxation $\rightarrow$ gives bound z_{LP} and solution x^*.
2. If x^* integral $\rightarrow$ update best integer solution (incumbent).
3. Else, pick a fractional variable x_j.
4. Create two branches:
$$x_j = 0, \quad x_j = 1.$$
5. Solve LPs for each subproblem, get new bounds.

Branch and Bound: Conceptual Loop

Algorithmic skeleton:

1. Solve LP relaxation $\rightarrow$ gives bound z_{LP} and solution x^*.
2. If x^* integral $\rightarrow$ update best integer solution (incumbent).
3. Else, pick a fractional variable x_j.
4. Create two branches:
$$x_j = 0, \quad x_j = 1.$$
5. Solve LPs for each subproblem, get new bounds.
6. Prune branches that are infeasible or worse than best integer so far.

Tracking the Search

Each node in branch-and-bound has its own upper bound.

Tracking the Search

Each node in branch-and-bound has its own upper bound.
For a maximization problem, at any node:

$$\text{Upper Bound } UB_{\text{node}} = \max \text{possible value of the integer solution at that node.}$$

Tracking the Search

Each node in branch-and-bound has its own upper bound.
For a maximization problem, at any node:

$$\text{Upper Bound } UB_{\text{node}} = \max \text{possible value of the integer solution at that node.}$$

We also maintain the best global integer lower bound found so far, called the *incumbent*:

$$\text{Incumbent LB} = \max\{\textbf{integer} \text{ feasible solution found in the search}\}.$$

Tracking the Search

Each node in branch-and-bound has its own upper bound.
For a maximization problem, at any node:

$$\text{Upper Bound } UB_{\text{node}} = \max \text{possible value of the integer solution at that node.}$$

We also maintain the best global integer lower bound found so far, called the *incumbent*:

$$\text{Incumbent LB} = \max\{\textbf{integer} \text{ feasible solution found in the search}\}.$$

At every node we always have:

$$LB \;\leq\; z^*_{\text{node}} \;\leq\; UB_{\text{node}},$$

where z^*_{node} is the unknown optimal integer value in that subtree.

Tracking the Search

Each node in branch-and-bound has its own upper bound.
For a maximization problem, at any node:

$$\text{Upper Bound } UB_{\text{node}} = \max \text{possible value of the integer solution at that node.}$$

We also maintain the best global integer lower bound found so far, called the *incumbent*:

$$\text{Incumbent LB} = \max\{\textbf{integer} \text{ feasible solution found in the search}\}.$$

At every node we always have:

$$LB \;\leq\; z^*_{\text{node}} \;\leq\; UB_{\text{node}},$$

where z^*_{node} is the unknown optimal integer value in that subtree.
At the root, our goal is to shrink the gap $UB_{\text{root}} - LB$ until $UB_{\text{root}} = LB$, proving optimality.

Knapsack Problem at Root

Consider the 0–1 knapsack:

$$\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \le 4, \quad x_i \in \{0, 1\}.$$

Knapsack Problem at Root

Consider the 0–1 knapsack:

$$\max\ 2x_1 + 2x_2 + 5x_3 + 6x_4$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \le 4, \quad x_i \in \{0,1\}.$$

LP relaxation (allow $0 \le x_i \le 1$).

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = \tfrac{1}{3},$$

$$z_{\text{LP}} = 2 \cdot 1 + 2 \cdot 0 + 5 \cdot 1 + 6 \cdot \tfrac{1}{3} = 2 + 0 + 5 + 2 = 9.$$

So initially:

$$UB_{\text{root}} = 9, \quad LB = -\infty \text{ (no incumbent yet).}$$

Knapsack Problem at Root

Consider the 0–1 knapsack:

$$\max \; 2x_1 + 2x_2 + 5x_3 + 6x_4$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 + 3x_4 \le 4, \quad x_i \in \{0, 1\}.$$

LP relaxation (allow $0 \le x_i \le 1$).

$$x_1 = 1, \quad x_2 = 0, \quad x_3 = 1, \quad x_4 = \tfrac{1}{3},$$

$$z_{\text{LP}} = 2 \cdot 1 + 2 \cdot 0 + 5 \cdot 1 + 6 \cdot \tfrac{1}{3} = 2 + 0 + 5 + 2 = 9.$$

So initially:

$$UB_{\text{root}} = 9, \quad LB = -\infty \text{ (no incumbent yet).}$$

Fractional variable: $x_4 = \frac{1}{3} \Rightarrow$ branch on x_4.

First Branch: Fixing x_4

Branch on the fractional variable x_4:

$$\text{Node A: } x_4 = 1, \qquad \text{Node B: } x_4 = 0.$$

First Branch: Fixing x_4

Branch on the fractional variable x_4:

$$\text{Node A: } x_4 = 1, \qquad \text{Node B: } x_4 = 0.$$

We will explore:

- The left subtree (Node A, $x_4 = 1$) first.
- Then the right subtree (Node B, $x_4 = 0$).

Left Subtree: Node A ($x_4 = 1$)

At Node A, we fix $x_4 = 1$. The constraint becomes:

$$x_1 + 2x_2 + 2x_3 + 3 \cdot 1 \leq 4 \;\Rightarrow\; x_1 + 2x_2 + 2x_3 \leq 1.$$

The LP relaxation at Node A is:

$$\max \; 2x_1 + 2x_2 + 5x_3 + 6 \cdot 1 = 6 + 2x_1 + 2x_2 + 5x_3$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 \leq 1, \quad 0 \leq x_1, x_2, x_3 \leq 1.$$

Left Subtree: Node A ($x_4 = 1$)

At Node A, we fix $x_4 = 1$. The constraint becomes:

$$x_1 + 2x_2 + 2x_3 + 3 \cdot 1 \leq 4 \;\Rightarrow\; x_1 + 2x_2 + 2x_3 \leq 1.$$

The LP relaxation at Node A is:

$$\max\; 2x_1 + 2x_2 + 5x_3 + 6 \cdot 1 = 6 + 2x_1 + 2x_2 + 5x_3$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 \leq 1, \quad 0 \leq x_1, x_2, x_3 \leq 1.$$

$$x_1 = 0,\; x_2 = 0,\; x_3 = \tfrac{1}{2},\; x_4 = 1,$$
$$z_{\text{LP}} = 6 + 5 \cdot \tfrac{1}{2} = 6 + 2.5 = 8.5.$$

So

$$UB_A = 8.5.$$

The solution is fractional in $x_3 \Rightarrow$ branch on x_3.

Left Subtree: Children of Node A ($x_4 = 1$)

Branch on x_3 at Node A:

$$\text{Node A1: } x_4 = 1,\ x_3 = 1, \qquad \text{Node A2: } x_4 = 1,\ x_3 = 0.$$

Left Subtree: Children of Node A ($x_4 = 1$)

Branch on x_3 at Node A:

$$\text{Node A1: } x_4 = 1,\ x_3 = 1, \qquad \text{Node A2: } x_4 = 1,\ x_3 = 0.$$

Node A1: $x_4 = 1,\ x_3 = 1.$

$$x_1 + 2x_2 + 2 \cdot 1 + 3 \cdot 1 \leq 4 \ \Rightarrow\ x_1 + 2x_2 + 5 \leq 4 \ \Rightarrow\ x_1 + 2x_2 \leq -1,$$

which is impossible. So Node A1 is **infeasible** and thus **fathomed**.

Left Subtree: Children of Node A ($x_4 = 1$)

Branch on x_3 at Node A:

$$\text{Node A1: } x_4 = 1,\ x_3 = 1, \qquad \text{Node A2: } x_4 = 1,\ x_3 = 0.$$

Node A1: $x_4 = 1,\ x_3 = 1.$

$$x_1 + 2x_2 + 2 \cdot 1 + 3 \cdot 1 \leq 4 \ \Rightarrow\ x_1 + 2x_2 + 5 \leq 4 \ \Rightarrow\ x_1 + 2x_2 \leq -1,$$

which is impossible. So Node A1 is **infeasible** and thus **fathomed**.

Node A2: $x_4 = 1,\ x_3 = 0.$ LP relaxation at Node A2:

$$\max\ 6 + 2x_1 + 2x_2 \quad \text{s.t. } x_1 + 2x_2 \leq 1,\ 0 \leq x_1, x_2 \leq 1.$$

Left Subtree: Children of Node A ($x_4 = 1$)

Branch on x_3 at Node A:

$$\text{Node A1: } x_4 = 1,\ x_3 = 1, \qquad \text{Node A2: } x_4 = 1,\ x_3 = 0.$$

Node A1: $x_4 = 1,\ x_3 = 1.$

$$x_1 + 2x_2 + 2 \cdot 1 + 3 \cdot 1 \leq 4 \ \Rightarrow\ x_1 + 2x_2 + 5 \leq 4 \ \Rightarrow\ x_1 + 2x_2 \leq -1,$$

which is impossible. So Node A1 is **infeasible** and thus **fathomed**.

Node A2: $x_4 = 1,\ x_3 = 0$. LP relaxation at Node A2:

$$\max\ 6 + 2x_1 + 2x_2 \quad \text{s.t. } x_1 + 2x_2 \leq 1,\ 0 \leq x_1, x_2 \leq 1.$$

The LP solution is $x_2 = 0,\ x_1 = 1$:

$$x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 1, \quad z = 6 + 2 = 8.$$

Left Subtree: Children of Node A ($x_4 = 1$)

Branch on x_3 at Node A:

$$\text{Node A1: } x_4 = 1,\ x_3 = 1, \qquad \text{Node A2: } x_4 = 1,\ x_3 = 0.$$

Node A1: $x_4 = 1,\ x_3 = 1.$

$$x_1 + 2x_2 + 2 \cdot 1 + 3 \cdot 1 \leq 4 \ \Rightarrow\ x_1 + 2x_2 + 5 \leq 4 \ \Rightarrow\ x_1 + 2x_2 \leq -1,$$

which is impossible. So Node A1 is **infeasible** and thus **fathomed**.
Node A2: $x_4 = 1,\ x_3 = 0$. LP relaxation at Node A2:

$$\max\ 6 + 2x_1 + 2x_2 \quad \text{s.t. } x_1 + 2x_2 \leq 1,\ 0 \leq x_1, x_2 \leq 1.$$

The LP solution is $x_2 = 0,\ x_1 = 1$:

$$x_1 = 1,\ x_2 = 0,\ x_3 = 0,\ x_4 = 1, \quad z = 6 + 2 = 8.$$

This is an **integer** solution, so we update the incumbent

$$LB = 8.$$

Summary of Left Subtree (Node A, $x_4 = 1$)

In the left subtree ($x_4 = 1$):

- Node A1: infeasible $\Rightarrow UB_{\text{A1}} = -\infty$.
- Node A2: LP solution is integer solution

$$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1), \quad z = 8.$$

So $UB_{\text{A2}} = 8$ and update global incumbent to $LB = 8$.

Summary of Left Subtree (Node A, $x_4 = 1$)

In the left subtree ($x_4 = 1$):

- Node A1: infeasible $\Rightarrow UB_{\text{A1}} = -\infty$.
- Node A2: LP solution is integer solution

$$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1), \quad z = 8.$$

So $UB_{\text{A2}} = 8$ and update global incumbent to $LB = 8$.

Thus, for Node A overall:

$$UB_A = \max\{UB_{\text{A1}}, UB_{\text{A2}}\} = \max\{-\infty, 8\} = 8,$$

We now turn to the right subtree, Node B ($x_4 = 0$), starting from:

$$UB_{\text{root}} = 9, \quad LB = 8.$$

Right Subtree: Node B ($x_4 = 0$)

At Node B, we fix $x_4 = 0$. The constraint simplifies to:

$$x_1 + 2x_2 + 2x_3 + 3 \cdot 0 \leq 4 \;\Rightarrow\; x_1 + 2x_2 + 2x_3 \leq 4.$$

Right Subtree: Node B ($x_4 = 0$)

At Node B, we fix $x_4 = 0$. The constraint simplifies to:

$$x_1 + 2x_2 + 2x_3 + 3 \cdot 0 \leq 4 \;\Rightarrow\; x_1 + 2x_2 + 2x_3 \leq 4.$$

The LP relaxation at Node B is:

$$\max \; 2x_1 + 2x_2 + 5x_3$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 \leq 4, \quad 0 \leq x_1, x_2, x_3 \leq 1.$$

Right Subtree: Node B ($x_4 = 0$)

At Node B, we fix $x_4 = 0$. The constraint simplifies to:

$$x_1 + 2x_2 + 2x_3 + 3 \cdot 0 \leq 4 \;\Rightarrow\; x_1 + 2x_2 + 2x_3 \leq 4.$$

The LP relaxation at Node B is:

$$\max\; 2x_1 + 2x_2 + 5x_3$$

$$\text{s.t. } x_1 + 2x_2 + 2x_3 \leq 4, \quad 0 \leq x_1, x_2, x_3 \leq 1.$$

$$x_1 = 1,\; x_2 = \tfrac{1}{2},\; x_3 = 1,\; x_4 = 0,$$
$$z_{\text{LP}} = 2 \cdot 1 + 2 \cdot \tfrac{1}{2} + 5 \cdot 1 = 2 + 1 + 5 = 8.$$

So $UB_B = 8$. However, our incumbent is already $LB = 8$. Therefore $UB_B = 8 \leq LB = 8$.

Conclusion: Even though the LP at Node B is fractional, *no integer solution in this subtree can improve the incumbent*. We **fathom Node B by bound**, without branching further.

Final Summary of the B&B Tree

We have explored all necessary branches:

- Left subtree (Node A, $x_4 = 1$):
  - Node A1, $x_3 = 1, x_4 = 1$: infeasible (fathomed by infeasibility).
  - Node A2, $x_3 = 0, x_4 = 1$: LP solution is integer solution

    $$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1), \quad z = 8.$$

    This solution becomes the incumbent: $LB = 8$. No branching.

Final Summary of the B&B Tree

We have explored all necessary branches:

- Left subtree (Node A, $x_4 = 1$):
  - Node A1, $x_3 = 1, x_4 = 1$: infeasible (fathomed by infeasibility).
  - Node A2, $x_3 = 0, x_4 = 1$: LP solution is integer solution

    $$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1), \quad z = 8.$$

    This solution becomes the incumbent: $LB = 8$. No branching.
- Right subtree (Node B, $x_4 = 0$):
  - LP solution at Node B: fractional but with $UB_B = 8 \leq LB = 8$.
  - Node B is **fathomed by bound** without any further branching, even though its LP solution is not integral.

Final Summary of the B&B Tree

We have explored all necessary branches:

- Left subtree (Node A, $x_4 = 1$):
  - Node A1, $x_3 = 1, x_4 = 1$: infeasible (fathomed by infeasibility).
  - Node A2, $x_3 = 0, x_4 = 1$: LP solution is integer solution

    $$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1), \quad z = 8.$$

    This solution becomes the incumbent: $LB = 8$. No branching.
- Right subtree (Node B, $x_4 = 0$):
  - LP solution at Node B: fractional but with $UB_B = 8 \leq LB = 8$.
  - Node B is **fathomed by bound** without any further branching, even though its LP solution is not integral.
- Update $UB_{\text{root}} = \max(UB_A, UB_B) = \max(8, 8) = 8 = LB$. Terminate!

Final Summary of the B&B Tree

We have explored all necessary branches:

- Left subtree (Node A, $x_4 = 1$):
  - Node A1, $x_3 = 1, x_4 = 1$: infeasible (fathomed by infeasibility).
  - Node A2, $x_3 = 0, x_4 = 1$: LP solution is integer solution

    $$(x_1, x_2, x_3, x_4) = (1, 0, 0, 1), \quad z = 8.$$

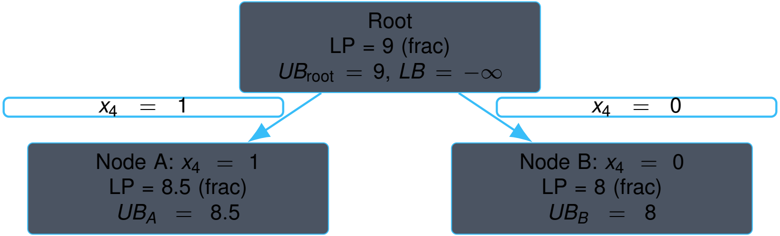
    This solution becomes the incumbent: $LB = 8$. No branching.
- Right subtree (Node B, $x_4 = 0$):
  - LP solution at Node B: fractional but with $UB_B = 8 \leq LB = 8$.
  - Node B is **fathomed by bound** without any further branching, even though its LP solution is not integral.
- Update $UB_{\text{root}} = \max(UB_A, UB_B) = \max(8, 8) = 8 = LB$. Terminate!

Best integer value found:

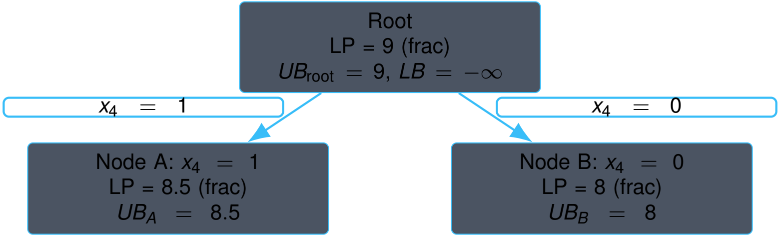
$$z^* = LB = 8, \quad x^* = (1, 0, 0, 1).$$

Root bounds at termination: $UB_{\text{root}} = 8, \quad LB = 8$. Since $UB_{\text{root}} = LB$, the

Evolving Tree (Stage 1)



Evolving Tree (Stage 1)

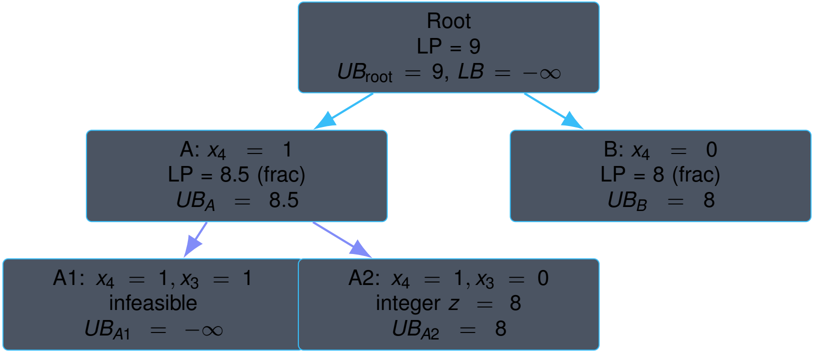


Both children are fractional.

We next expand Node A.

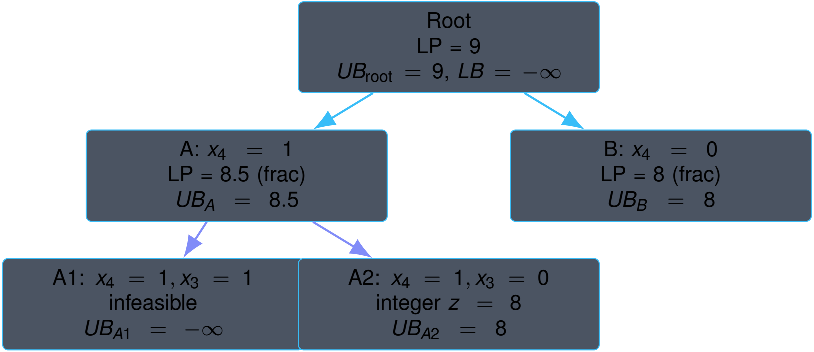
Evolving Tree (Stage 2a)

Expand Node A (fractional in x_3):



Evolving Tree (Stage 2a)

Expand Node A (fractional in x_3):



Node A1 is infeasible $\Rightarrow$ it contributes no feasible integer solution.

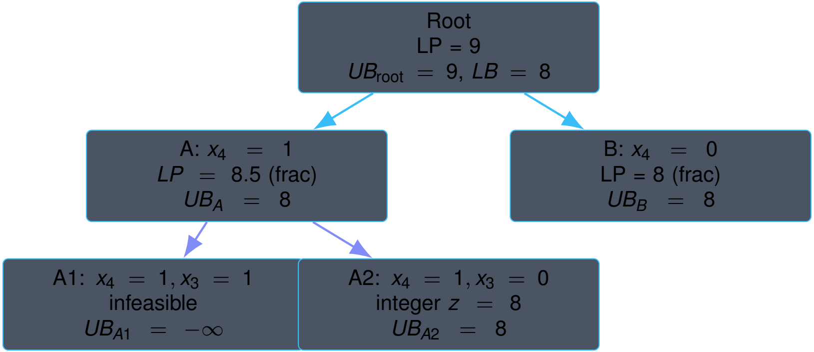
Node A2 has an **integer** LP solution with value $z = 8$.

Evolving Tree (Stage 2b)

Using the information from A1 and A2, we update bounds at Node A and the root.

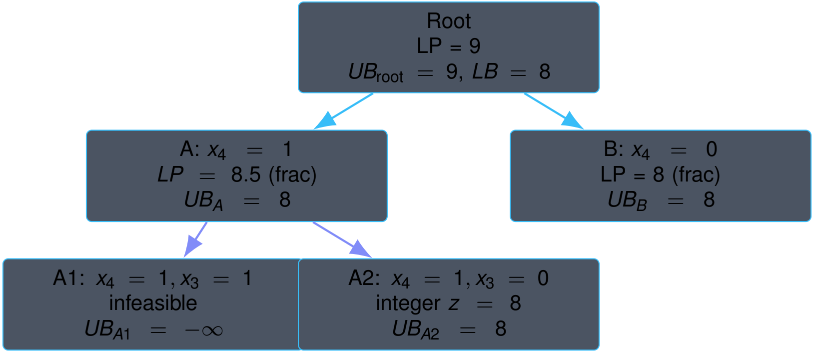
Evolving Tree (Stage 2b)

Using the information from A1 and A2, we update bounds at Node A and the root.



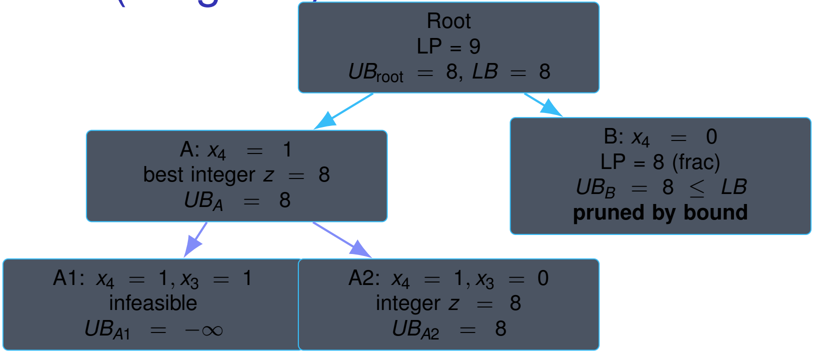
Evolving Tree (Stage 2b)

Using the information from A1 and A2, we update bounds at Node A and the root.

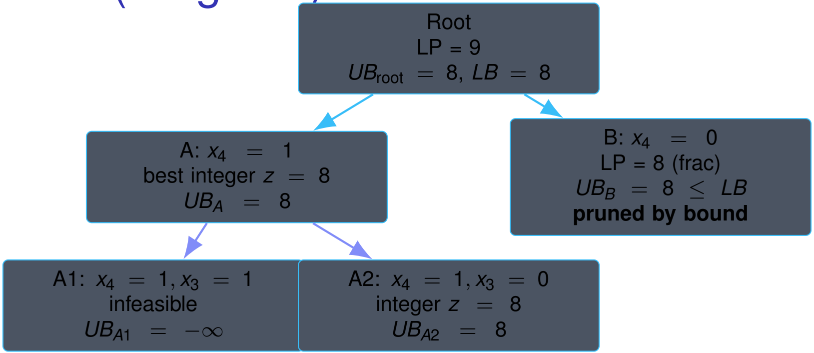


From A2, we obtain an incumbent with value $z = 8 \Rightarrow$ update global $\mathbf{LB} = 8$.
Node A's subtree cannot contain any solution better than 8, so $UB_A = 8$.
Node B remains active and will be considered next.

Evolving Tree (Stage 2c)



Evolving Tree (Stage 2c)



Since $UB_B = 8 \leq LB = 8$, Node B cannot contain any better integer solution.
Node B is **fathomed by bound**.
Update $UB_{\text{root}} = \max(UB_A, UB_B) = \max(8, 8) = 8$.
All nodes are now resolved, and $UB_{\text{root}} = LB = 8$, proving optimality of the incumbent $x^* = (1, 0, 0, 1)$ with value 8.

Bounding and Fathoming Summary

Fathoming rules (for maximization)

A node can be skipped (“fathomed”) if:

- LP is infeasible (e.g. Node A1), or
- LP bound $\leq$ incumbent (not worth exploring) [e.g. Node B] (Fathomed by bound), or
- LP solution is integral (update incumbent) [e.g. Node A2].

Bounding and Fathoming Summary

Fathoming rules (for maximization)

A node can be skipped ("fathomed") if:

- LP is infeasible (e.g. Node A1), or
- LP bound $\leq$ incumbent (not worth exploring) [e.g. Node B] (Fathomed by bound), or
- LP solution is integral (update incumbent) [e.g. Node A2].

Each LP bound tightens the global UB.

Algorithmic Behavior

- Each LP call provides a bound.
- Each integer solution provides an incumbent.
- We prune aggressively when LP can't beat the incumbent.

Algorithmic Behavior

- Each LP call provides a bound.
- Each integer solution provides an incumbent.
- We prune aggressively when LP can't beat the incumbent.

Visualize B&B as a dialogue:

LP: "At best, I can get 11in this node"   LB: "I already have 12!"

The gap tells us how much hope remains.

Node Selection Strategies

The order that we process nodes in B&B matters. We did the order root $\rightarrow A \rightarrow A1 \rightarrow A2 \rightarrow B$. **Two classical options:**

- **Depth-First Search (DFS):** quickly finds feasible integers (good for LB).
- **Best-Bound (Best-First):** explores the node with highest current UB bound.

Node Selection Strategies

The order that we process nodes in B&B matters. We did the order root $\to A \to A1 \to A2 \to B$. **Two classical options:**

- **Depth-First Search (DFS):** quickly finds feasible integers (good for LB).
- **Best-Bound (Best-First):** explores the node with highest current UB bound.

Modern solvers use hybrids: DFS early $\to$ Best-Bound once good incumbent found.

Branching Variable Choice

If there is only one variable that is fractional, just branch on it. But what if there is more than one?
Which variable to branch on?

- Most fractional ($x_i \approx 0.5$) $\to$ balances search.
- Greatest effect on objective (pseudo-costs).
- Domain-specific heuristics (e.g., branching on vertex degree).

Also, which branch to explore first? (i.e. $x_3 = 0$ or $x_3 = 1$ first?).

Branching Variable Choice

If there is only one variable that is fractional, just branch on it. But what if there is more than one?
Which variable to branch on?

- Most fractional ($x_i \approx 0.5$) $\rightarrow$ balances search.
- Greatest effect on objective (pseudo-costs).
- Domain-specific heuristics (e.g., branching on vertex degree).

Also, which branch to explore first? (i.e. $x_3 = 0$ or $x_3 = 1$ first?).
Smart branching = smaller tree = faster solve. Tons of heuristics.

Summary of B&B Workflow

1. Solve LP relaxation.
2. If integral, update incumbent. If fractional $\rightarrow$ branch.
3. Update bounds (both UB and LB of all nodes).
4. Prune by infeasibility or domination (fathoming).
5. Repeat until UB = LB.

Gurobi Example: Integer Knapsack

```python
import gurobipy as gp
from gurobipy import GRB

values  = [10, 7, 4]
weights = [5, 4, 3]
W = 7

m = gp.Model("knapsack_ip")
x = m.addVars(3, vtype=GRB.BINARY, name="x") #Only new thing

m.addConstr(sum(weights[i]*x[i] for i in range(3)) <= W)
m.setObjective(sum(values[i]*x[i] for i in range(3)), GRB.MAXIMIZE)
m.optimize()
```

Extracting Solver Statistics

After `m.optimize()`:

```
print("Optimal value:", m.ObjVal)
print("Nodes explored:", m.NodeCount)
print("Best bound:", m.ObjBound)
print("Gap:", m.MIPGap)
for v in x.values():
    print(v.VarName, v.X)
```