Recall that the branching technique can be represented as an enumeration tree. If the number of nodes in the enumeration tree is large then it takes more time to solve the IP. So, we will use bounding techniques to cleverly prune the tree, i.e., we will cut off some parts of the tree without losing the optimal solution. As a convention, we will denote the sub-problem corresponding to feasible region $S$ by $z_S := \{c^T x : x \in S\}$.

### 21.1 Pruning Enumeration Trees

We will see various pruning techniques based on bounds. The subproblem corresponding to node $S$ may not be solvable efficiently. However, typically we will be able to get some feasible solution and solve a relaxation quickly.

- Solving a relaxation of a maximization problem gives an upper bound $u$ on the objective value of the original problem.
- The objective value of a feasible solution to a maximization problem gives a lower bound $l$ on the objective value of the original problem.

Therefore, for $S$, we will be able to get $l, u$ such that $l \leq z_S \leq u$ quickly. We will use these bounds to cleverly prune the tree. We will illustrate the pruning rules through examples.

**Example 1:** See Figure 21.1
Figure 21.1: The bounds of the sub-problems allow us to (1) update the bounds for \( z_S \) and (2) then prune subproblem \( S_1 \) as it has already been solved to optimality.

In Figure 21.1 we found better lower and upper bounds for \( z \) using the upper and lower bounds for subproblems. Furthermore, branch \( S_1 \) has achieved optimality because \( c^1 = l^1 \). So, we do not have to explore branch \( S_1 \) further.

**Observation** (Updating bounds). Suppose \( l^i \leq z^i \leq u^i \ \forall i \in [k] \). Then

\[
\max\{l^i : i \in [k]\} \leq z_S \leq \max\{u^i : i \in [k]\}.
\]

**Observation** (Pruning by optimality). If \( l^i = u^i \) for some \( i \), then the optimum solution for the \( i^{th} \) subproblem \( z^i := \max\{c^T x : x \in S_i\} \) is known, so we do not have to explore branch \( S_i \) further, i.e., prune \( S_i \).

**Example 2:** See Figure 21.2

Figure 21.2: The bounds of the sub-problem allow us to prune \( S_1 \) as we can provably say that the optimal solution is not coming from that sub-problem.

In Figure 21.2, the branch rooted at \( S_1 \) does not contain an optimal solution to the original problem because \( u^1 \leq l \) (any solution under \( S_1 \) has value \( \leq 20 = u^1 \) and we already have a solution of value \( \geq 21 = l \)). So, prune the branch \( S_1 \) and update the bounds.

**Observation** (Pruning by bound). If \( u^i < l \) for some \( i \), then the optimal solution for \( z^i = \max\{c^T x : x \in S_i\} \) can never be an optimal solution to the original problem, i.e., prune \( S_i \).

**Observation** (Pruning by infeasibility). If \( S_i = \emptyset \), then prune \( S_i \).

In spite of pruning rules, we may not be able to prune sometimes.

**Example 3:** See Figure 21.3
21.2 LP-based B&B

A popular way to obtain upper bounds is by solving the LP-relaxation of the subproblems. We will illustrate LP-based B&B through an example.

Example.

Consider \( z = \max \{4x_1 - x_2 : x \in P \cap \mathbb{Z}^5 \} \)

where \( P := \{ x \in \mathbb{R}^5 : \begin{align*}
3x_1 - 2x_2 + x_3 &= 14 \\
x_2 + x_4 &= 3 \\
2x_1 - 2x_2 + x_5 &= 3 \\
x &\geq 0
\} \).

Let \( S := P \cap \mathbb{Z}^5 \).

Bounding:

1. Upper bound: Solve the LP-relaxation. An optimal solution is \( \tilde{x} = (\frac{9}{7}, 3, \frac{13}{2}, 0, 0) \). Therefore, \( u = 15 \).

2. Lower Bound: Lower bound for a maximization problem is given by a feasible solution. If we do not have a feasible solution yet, then \( l = -\infty \) is a conventional lower bound. Therefore, the tree looks as below.

Since \( l < u \), we do not have optimality/infeasibility so we need to branch, i.e., split the feasible region.

Branching:

Common way to split is to use an integer variable that is taking a fractional value in the current LP solution. Let

\[
S_1 := \{ x \in S : x_j \leq \lceil \tilde{x}_j \rceil \}
\]

\[
S_2 := \{ x \in S : x_j \leq \lfloor \tilde{x}_j \rfloor \}.
\]

Note that \( S = S_1 \cup S_2 \) and \( S_1 \cap S_2 = \emptyset \) and \( \tilde{x} \) is infeasible for both \( S_1 \) and \( S_2 \).
Which node to explore next? i.e., which subproblem to solve next?

Choosing an active node: Say we explore $S_2$.

**Bound:** Solve $u^2 := \max \{4x_1 - x_2 : x \in P_2 \}$ where $P_2 = \{x \in P : x_1 \geq 5\}$. It is infeasible. So, we can prune $S_2$.

Choose an active node: $S_1$

**Bound:** Solve $u^1 := \max \{4x_1 - x_2 : x \in P_1 \}$ where $P_1 := \{x \in P : x \leq 4\}$. An optimal solution is $\hat{x} = (4, \frac{5}{2}, 7, 4, 0)$. Therefore, $u^1 = \frac{27}{2}$. We note that $u^1 \leq u$, so we update the bounds.

Branch: $S_{11} := \{x \in S_1 : x_2 \leq 2\}$ and $S_{12} := \{x \in S_1 : x_2 \geq 3\}$.

Note that again $S_1 = S_{11} \cup S_{12}$ and $S_{11} \cap S_{12} = \emptyset$ and $\hat{x}$ from $S_1$ is infeasible for $S_{11}$ and $S_{12}$. 
Choose an active node: $S_{12}$

**Bound:** Solve $u_{12} := \max \{4x_1 - x_2 : x \in P_{12} \}$ where $P_{12} := \{ x \in P_1 : x_2 \geq 3 \}$. An optimal solution is $\bar{x} = (4, 3, 8, 1, 0)$ which implies that $u_{12} = 13$ and $l_{12} = 13$. We can prune $S_{12}$ by optimality and update the bounds.

Choose an active node: $S_{11}$

**Bound:** Solve $u_{11} := \max \{4x_1 - x_2 : x \in P_{11} \}$ where $P_{11} := \{ x \in P_1 : x_2 \leq 2 \}$. An optimal solution is $\bar{x} = (\frac{7}{2}, 2, \frac{15}{2}, 1, 0)$ which implies that $u_{11} = 12$. We can prune $S_{11}$ by bound and update the bounds.

Therefore, the optimal objective value is 13 and an optimal solution is $(4, 3, 8, 1, 0)$.

### 21.2.1 Implementation Considerations in LP-based B&B

1) **Re-optimization:** LP($S$) and LP($S_1$) differs only in one constraint. To optimize for LP($S_1$) start from the optimal solution for LP($S$) and use dual simplex.

2) **Storing the tree:** Instead of storing all nodes of the tree, store only the active nodes.
3) **Bounding:** For upper bound use LP-relaxation and for lower bound use heuristics and approximation algorithms.

4) **Branching:** Branch on a variable that is fractional in the optimal solution to the LP-relaxation. If there are several such variables, use the most fractional one. There are numerous other branching rules based on estimating the cost of a variable to become an integer.

5) **Choosing an active node to explore:** We chose arbitrarily in the example.

- Depth first strategy:
  - *Intuition:* Tree can be pruned significantly only if there is a good feasible solution which gives a good lower bound.
  - *Advantage:* Can reoptimize fast using dual simplex.
- Breadth first strategy:
  - *Intuition:* We would like to minimize the number of explored nodes so choose an active node with the best upper bound.
- In practice, we employ a combination of both. Initially, follow the depth first strategy to obtain a good feasible solution. Then do a mix of the two strategies.

### 21.2.2 LP-based B&B: Finiteness

Next, we will address the question of whether LP-based B&B will terminate in finite time. We will answer this affirmatively by showing that the enumeration tree in LP-based B&B is finite.

**Lemma 0.1.** Suppose \( P = \{ x \in \mathbb{R}^n : Ax \leq b, x \geq 0 \} \) is bounded. Then the enumeration tree in an LP-based B&B will be finite. In particular, if \( w_j := \lceil \max \{ x_j : x \in P \} \rceil \) then every path in the tree can contain at most \( \sum_{j=1}^{n} w_j \) nodes. Therefore,

\[
\text{Depth of tree} \leq \sum_{j=1}^{n} w_j.
\]

**Proof.** After adding the constraint \( x_j \leq d \) the only other constraint for \( x_j \) that can subsequently appear are \( x_j \leq d' \) and \( x_j \geq d' + 1 \) for some \( d' \in \{0, \ldots, d - 1\} \) (see figure below). Therefore, the largest number of branches involving \( x_j \) is at most \( w_j \).

\[\square\]