Branch and Bound: A Powerful Strategy for Integer Optimization

1. Introduction

During our **Combinatorial Optimization** class with **Professor Rosa Figueiredo**, we explored an essential algorithm for solving **integer programming problems**: the **Branch and Bound algorithm**.

Unlike the Simplex method, which deals with continuous variables, Branch and Bound is used when solutions must be **integers only**.

At its core, the idea is based on a classic principle in computer science:

> **Divide and conquer.**

Instead of searching the entire solution space blindly, the algorithm systematically **splits the problem** into smaller subproblems, evaluates them, and **eliminates** those that cannot lead to a better solution.

The “Divide and Conquer” Strategy

The heart of Branch and Bound is the idea of **breaking the problem** into smaller parts, solving each one, and keeping only the promising directions.

Let’s say you want to **maximize** a function over a set of integer solutions:

$$z^* = \max \{ c^T x \mid x \in S \}$$

You divide S into smaller subsets: $S = S_1 \cup S_2 \cup \ldots \cup S_{\square}$

Then, for each subset:

$z_1 = \max \{ c^T x \mid x \in S_1 \}$
$z_2 = \max \{ c^T x \mid x \in S_2 \}$
…
$z_{\square} = \max \{ c^T x \mid x \in S_{\square} \}$

The best solution is simply: $z^* = \max(z_1, z_2, \ldots, z_{\square})$

This approach **reduces the size of the search space** at each step.

You don’t explore everything — just what’s needed.

It’s especially useful when dealing with problems where **brute force** would take too long, like **binary variables**, **scheduling**, or the **Traveling Salesman Problem**.

3. The Enumeration Tree

To apply the **divide and conquer** strategy in practice, Branch and Bound builds a **tree structure**.

Each **node** in the tree represents a **subproblem**, and each **branching** step splits that subproblem into two or more **simpler ones**.

Let's say we're solving a problem with three binary variables: $x_1, x_2, x_3 \in \{0, 1\}$

There are $2^3 = 8$ possible combinations. Instead of listing them all, Branch and Bound builds a tree like this:

```
        S
       / \
    x_1=0  x_1=1
    / \     / \
x_2=0 x_2=1 x_2=0 x_2=1
  ...        ...
```

Each level of the tree **fixes one variable** (e.g., $x_1=0$ or $x_1=1$), and we move **downward** by continuing to fix variables until we reach a **leaf** — a complete solution.

4. Bounding: Eliminate What Doesn't Matter

At each node of the tree, the algorithm calculates two things:

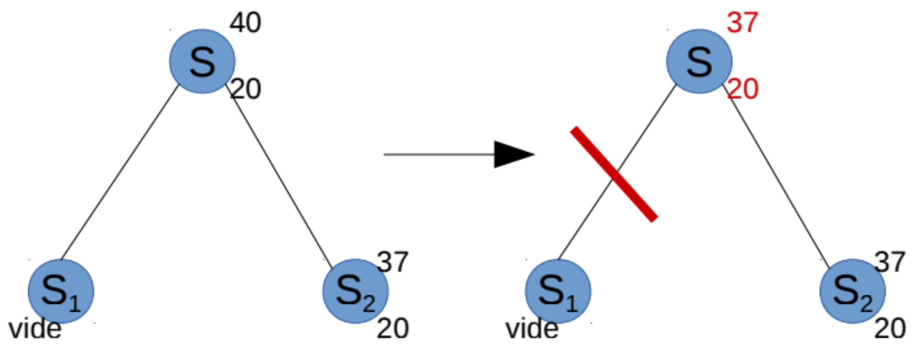
- **An upper bound**: the **best possible outcome** from that node (even if it's not integer)
- **A lower bound**: the **best feasible integer solution** found so far

These bounds allow the algorithm to make smart decisions:

Three Types of Pruning (Branch Elimination)

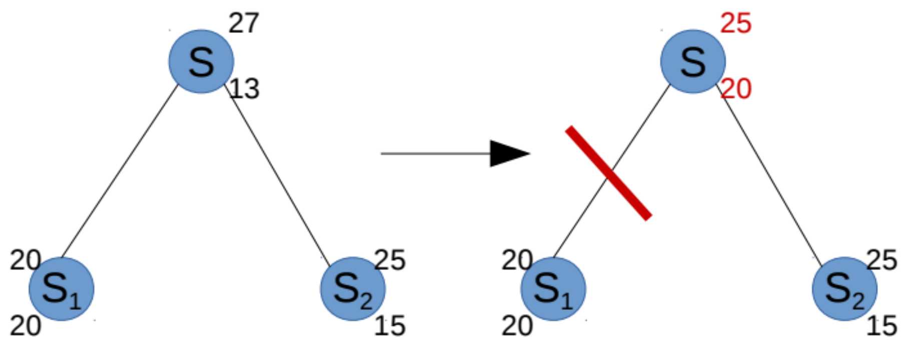
Figures adapted from the course slides of Prof. Rosa Figueiredo (2024) Avignon France.

By infeasibility:
If the node has **no feasible solution** (e.g., the constraints contradict each other), we eliminate it.



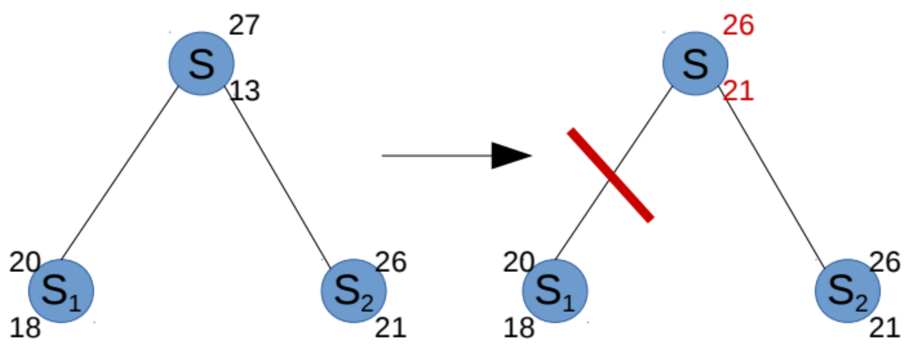
course of Prof. Rosa Figueiredo, Avignon France 2024

By optimality:
If the solution at a node is integer and **better than anything else so far**, it becomes the new **best**.



course of Prof. Rosa Figueiredo, Avignon France 2024

By bound:
If the upper bound at a node is **worse than the current best integer solution**, we **cut** that branch.
→ No need to explore it.



course of Prof. Rosa Figueiredo, Avignon France 2024

5. A Step-by-Step Example

We'll walk through the problem from class:

Maximize: $z = 4x_1 - x_2$
Subject to:
$7x_1 - 2x_2 \leq 14$
$x_2 \leq 3$
$2x_1 - 2x_2 \leq 3$
$x_1, x_2 \in \mathbb{Z}_+$

We'll solve it **step by step** using Branch and Bound:

- Start by solving the **relaxation** (ignore integrality),
- Then **branch** on the variable that isn't integer,
- Apply **bounding and pruning**,
- And finally, find the **optimal integer solution**.

Solve the Linear Relaxation

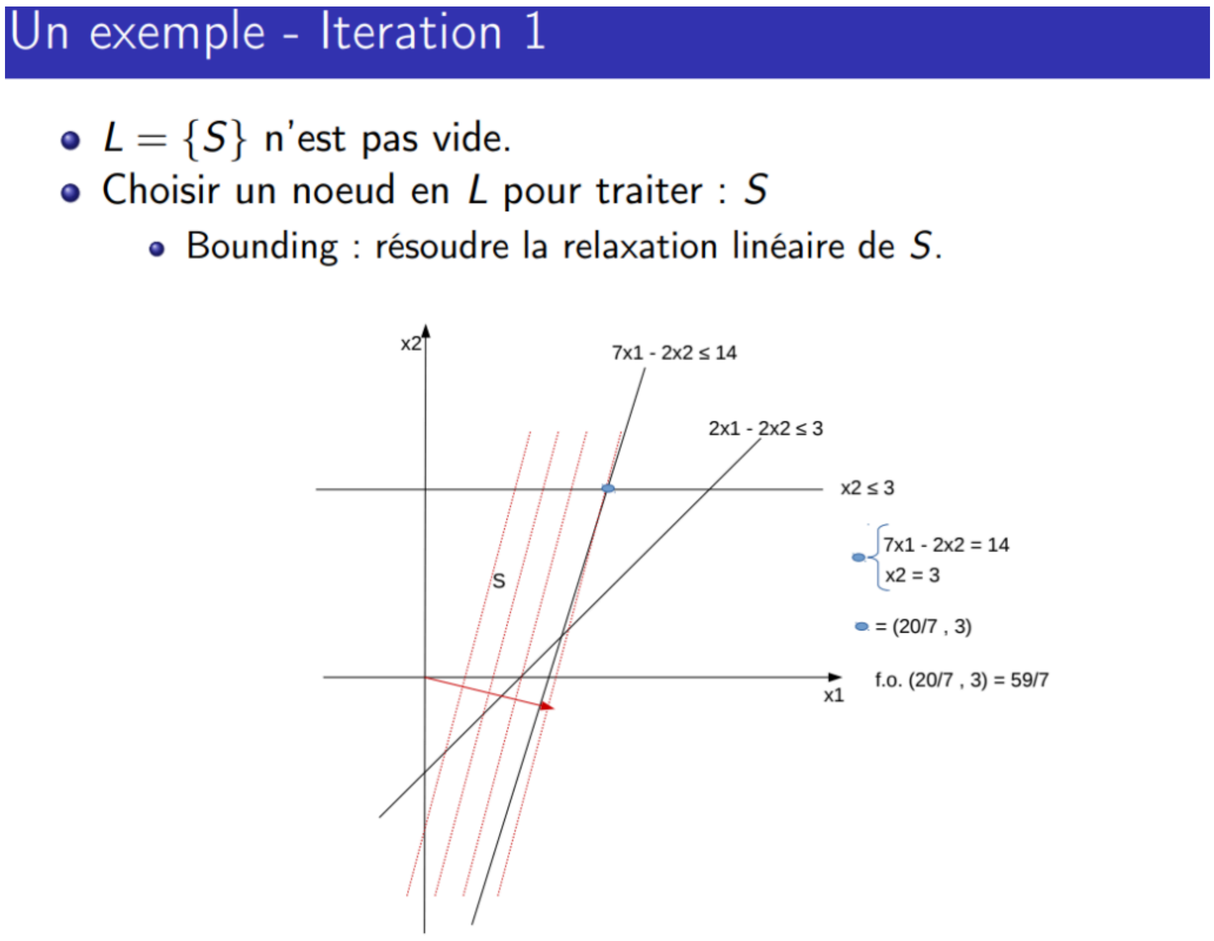
We ignore the integer constraints for now (we allow real values for variables).
This gives us a **relaxed linear problem**:

Objective:
Maximize: $z = 4x_1 - x_2$

Subject to:

$7x_1 - 2x_2 \leq 14$
$2x_1 - 2x_2 \leq 3$
$x_2 \leq 3$
$x_1 \geq 0, x_2 \geq 0$

We solve this LP using graphical methods or a solver (like in Julia or Python).



Un exemple - Iteration 1

- $L = \{S\}$ n'est pas vide.
- Choisir un noeud en L pour traiter : S
  - Bounding : résoudre la relaxation linéaire de S.

Figure from the course of Prof. Rosa Figueiredo, Avignon France 2024

$x_1 = 20/7 \approx 2.86$
$x_2 = 3$

$z = 4x_1 - x_2 = 59/7 \approx 8.43$

Since $x_1 = 20/7$ is not an integer, we **branch** on x_1.
We'll split the solution space into two new subproblems:

- One where $x_1 \leq 2$
- Another where $x_1 \geq 3$

We'll handle those in the next iteration.

Solve the Relaxation for Subproblem S_1:

We solve the **relaxed version** of S_1 (still allowing non-integer values), with the additional constraint: $x_1 \leq 2$

The feasible region is now smaller (as shown in the figure). The optimal solution for this relaxation is:

$x_1 = 2$
$x_2 = 0.5$

Objective value: $z = 4x_1 - x_2 = 4\times2 - 0.5 = 7.5$

However, this solution is **not integer** — so we **branch again**, this time on x_2.

Branching on x_2

Since $x_2 = 0.5$ is fractional, we split into two more subproblems:

- **S_3:** $x_2 \leq 0$
- **S_4:** $x_2 \geq 1$

These will be evaluated in the next iterations. These two branches will now be added to the list of subproblems to explore. In this way, we continue narrowing the search space — **always checking whether the solution is integer and optimal**.

Key Idea: At each step, we either find a better integer solution or eliminate subproblems that cannot lead to a better result (through bounding or infeasibility).

Exploring Subproblem S_2

At this point, the list of active subproblems is: L={S2,S3,S4}L = \{S_2, S_3, S_4\}L={S2,S3,S4}

Now, we pick the next node in the list to explore — **S_2**.

S_2 is the subset of solutions **from the original feasible set** where:

x1≥3

So we're now focusing only on the part of the search space where x1x_1x1 is greater than or equal to 3.

Bounding S_2

We try to solve the **relaxed linear version** of this subproblem.
But… it turns out that **S_2 is infeasible**. There's no solution that satisfies all the constraints along with

x1≥3

This means:

> **S_2 is eliminated** — it cannot contain any feasible (let alone optimal) integer solution.

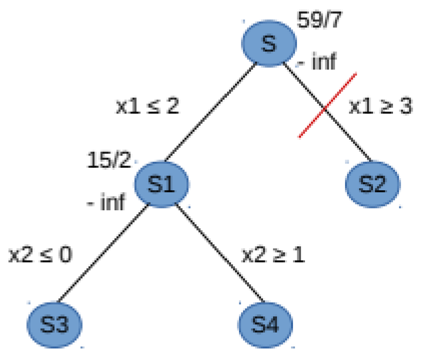


Figure from the course of Prof. Rosa Figueiredo, Avignon France 2024

Current list of subproblems after this step: L={S3,S4}

We now choose to explore **S_4**.

S_4 is a subproblem defined by adding two new constraints to narrow the search space:

- From before: $x1 \le 2$
- From branching: $x2 \ge 1$

This subset is a **refinement of S_1**, excluding the fractional solution we found earlier (2,1/2).

We solve the LP relaxation (allowing real numbers), and we find:

- Optimal solution: $x=2$, $y=1$
- Objective value: $z=4x-y=4(2)-1=7$

This time, the solution is **entirely integer**, so:

We've found a new feasible integer solution!

We now **update our current best solution**:

- New best objective value: **$z=7$**
- Current best solution: $(x1=2, x2=1)$

And because this is an integer solution that matches the upper bound for this node:

S_4 is eliminated by optimality!

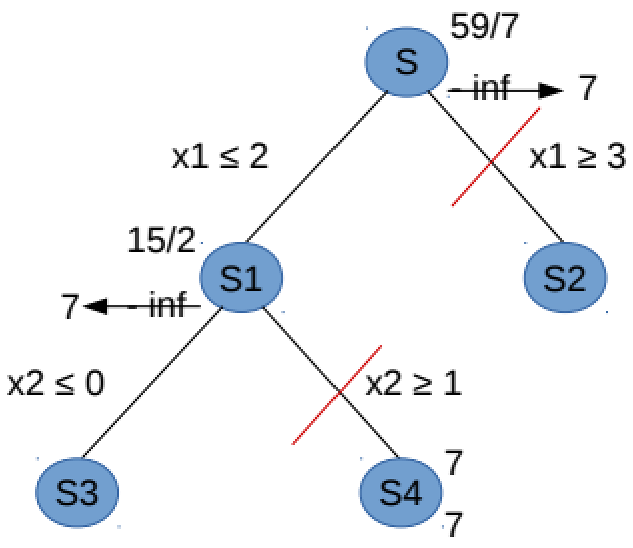


Figure from the course of Prof. Rosa Figueiredo, Avignon France 2024

Now we have L={S3}

Do we find a better solution in **S**$_3$, or is this one already optimal?

Subproblem **S**$_3$ was created from **S**$_1$ with this additional constraint:

- $x2\leq0$
- (we still have $x1\leq2$ from earlier)

Bounding: Solve the LP relaxation of S$_3$

We solve the relaxation again, and this time we find:

- Solution: x=32, y=0
- Objective value: $z=4\cdot3/2-0=6$

Not an integer solution
Worse than current best value z=7

runing (Elagage) by bound

We **compare this node's best value** (upper bound z=6)
with our current best solution, z=7.

Since 6 < 7:

We eliminate S$_3$ by bound — it can't contain a better solution.

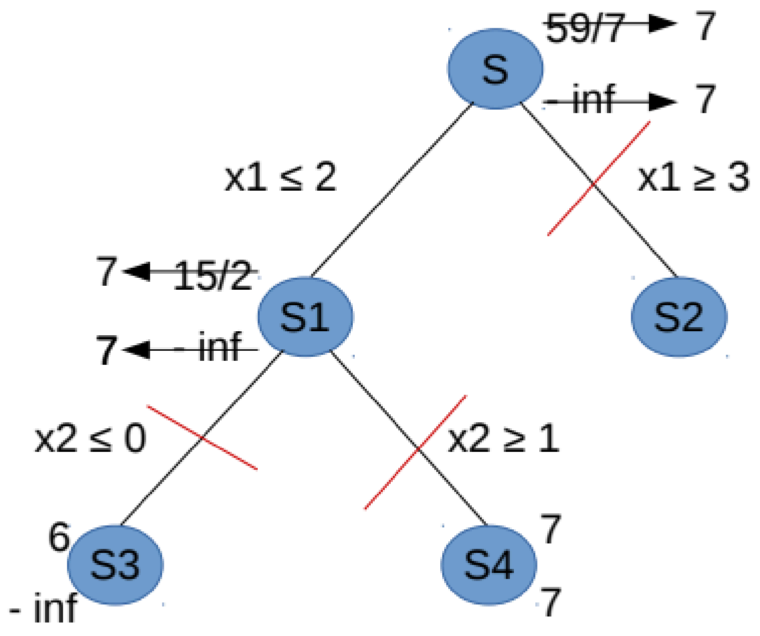


Figure from the course of Prof. Rosa Figueiredo, Avignon France 2024

Final result

Now the list L is empty.
We explored **all meaningful branches**, and our best integer solution is:

x1=2, x2=1, z=7

We're done!

Conclusion

The **Branch and Bound algorithm** is a powerful method for solving integer optimization problems. Unlike methods that explore all possible solutions, Branch and Bound smartly navigates the solution space by:

- **Dividing** the problem into subproblems (branching),
- **Bounding** their potential (using relaxations),
- And **eliminating** regions that cannot contain better solutions (pruning).

In this example, we saw how the algorithm uses logic and geometry to zero in on the optimal integer solution step by step. Even though the problem space could have hundreds or thousands of possibilities, only a few key branches were actually explored.

This strategy is not only efficient but also widely used in **real-world applications** such as scheduling, resource allocation, logistics, and combinatorial optimization.

I am running a few minutes late; my previous meeting is running over.