Branch and Bound

By: Aaryan Vinod Kumar

Branch and Bound (BnB) is a problem-solving paradigm used mainly for **optimization problems**, where the goal is to find the **best (maximum or minimum)** solution among many possibilities.

It systematically explores the solution space but **prunes** (cuts off) branches that cannot possibly lead to a better result - hence the name *Branch (explore)* and *Bound (limit)*.

In simple terms:

> *"Explore every path that could work - but abandon any that can't beat your best so far."*

It's an intelligent upgrade to brute force - exploring smartly and pruning early.

 Uploaded November 2025

Core Idea

Branch and Bound relies on **two key principles**:

1. **Branch:**
   Break the main problem into smaller subproblems (nodes in a decision tree).

2. **Bound:**
   Estimate a limit (upper or lower bound) on the best possible solution within each branch.

3. **Prune:**
   If the bound proves that branch can't outperform the current best solution, it's cut off from further exploration.

4. **Explore:**
   Continue branching only from the most promising nodes (those with best bounds).

Branch and Bound in Simple Terms

Imagine you're exploring different routes in a delivery network to find the **shortest path** that covers all cities.
You can:

- *Branch* - choose the next city to visit.

- *Bound* - calculate the minimum possible distance left for that route.

- *Prune* - drop routes that already exceed your best known route.

This saves time while still ensuring you find the optimal answer.

 Uploaded November 2025

General Steps

Branch and Bound is most effective for problems that:

- Have **a well-defined optimization goal** (min or max).
- Allow **bounding functions** to estimate best-case outcomes.
- Can be expressed as **state-space trees** (decisions forming a hierarchy).

Common characteristics:

- Uses **recursion or priority queues** to manage nodes.
- Relies on **bounding functions** to limit exploration.
- Guarantees the **optimal solution**.

 Uploaded November 2025

Popular Branch and Bound Problems

Problem	Description
Travelling Salesman Problem (TSP)	Find the shortest route visiting all cities exactly once.
Job Assignment Problem	Assign workers to jobs minimizing total cost.
N-Queens (optimized)	Solve with bounding to reduce search.
Integer Linear Programming	Optimize linear constraints with integer variables.
Graph Coloring (minimum colors)	Assign minimal colors while preventing adjacent conflicts.

Uploaded November 2025

Example 1: Travelling Salesman Problem

Problem:
Given n cities and the distances between each pair, find the shortest possible tour that visits every city exactly once and returns to the starting point.

Approach:

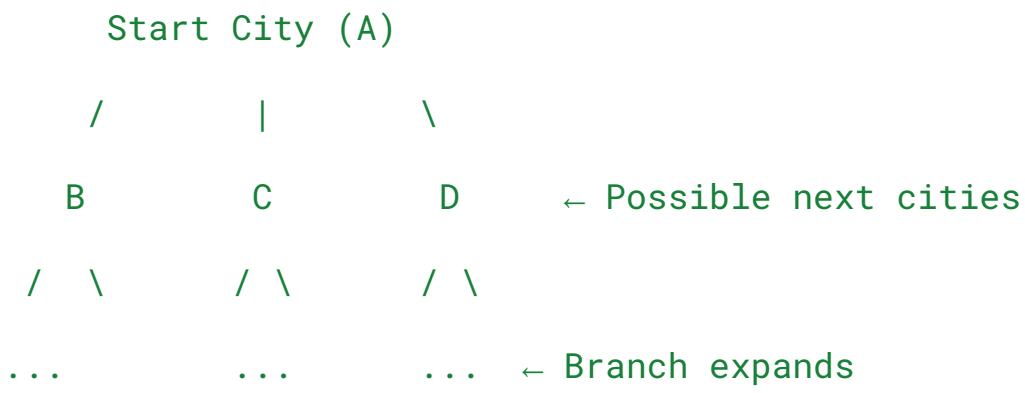
1. Each node represents a **partial tour**.
2. The **cost so far** and a **lower bound** of total cost are computed.
3. Nodes with total estimated cost higher than the best current route are **pruned**.
4. The **priority queue** ensures we always explore the most promising route first.

Typical Steps:

1. Start from the initial city.
2. For each unvisited city, **branch** by adding it to the path.
3. Compute the **lower bound** = cost so far + minimum possible future cost.
4. **Prune** routes whose bound ≥ current best route.
5. Continue exploring until all possible paths are checked or pruned.

 Uploaded November 2025

Conceptual Representation



Each node stores:

- Current path taken
- Cost so far
- Lower bound on remaining cost

Uploaded November 2025

Simplified Pseudocode

```
function branchAndBoundTSP():
    bestCost = ∞
    start from root (initial city)
    put root in priority queue (sorted by lower bound)

    while queue not empty:
        node = queue.pop()
        if node.bound >= bestCost:
            continue
        if node is complete tour:
            bestCost = min(bestCost, node.cost)
        else:
            for each unvisited city:
                child = extend(node, city)
                if child.bound < bestCost:
                    queue.push(child)
    return bestCost
```

 Uploaded November 2025

Complexity Analysis

Operation	Time Complexity
Building full tree (brute-force)	$O(n!)$
With bounding and pruning	$O(n^2 * 2^n)$ worst case $O(k \times n^2)$ where $k << 2^n$ average case
Space Complexity	$O(n^2)$

In practice, pruning drastically reduces the number of explored nodes, making it much faster than brute force.

Why Branch and Bound Works Here

Because it avoids exploring expensive or redundant paths, focusing only on those that can potentially yield a shorter route.
This ensures an **optimal solution with less work**.

Uploaded November 2025

Advantages of Branch and Bound

- **Guaranteed Optimality:** Finds the best possible solution.
- **Efficient Pruning:** Avoids exploring impossible or non-promising branches.
- **Flexible:** Works on many NP-hard optimization problems.
- **Generalizable:** Same approach can solve knapsack, TSP, and scheduling with slight modifications.

Limitations

- **High Memory Usage:** Needs to store multiple partial states.
- **Performance Depends on Bound Quality:** Weak bounds = minimal pruning.
- **Still Exponential in Worst Case:** Doesn't escape NP-hard nature, just reduces work.
- **Complex Implementation:** Requires designing good bounding functions.

Uploaded November 2025