# Branch and Bound

Dedan Kimathi University of Technology

Dr. Jane Kuria

# Introduction to Branch and Bound

Definition; Branch and Bound is a powerful algorithm design paradigm. paradigm. It efficiently solves combinatorial optimization problems by problems by systematically exploring possible solutions.

Purpose: To systematically explore the solution space to find the optimal solution while eliminating large portions of the search space that cannot contain the optimal solution..

# Key Concepts

### Pruning

Discarding subproblems that cannot cannot yield a better solution than the than the current best..

### Branching

Dividing the problem into smaller smaller subproblems for easier handling.

### Bounding

Using estimates to eliminate non-promising branches quickly.

# General Steps of Branch and Bound

1

### Initialization

Set parameters and prepare initial conditions for the algorithm.

2

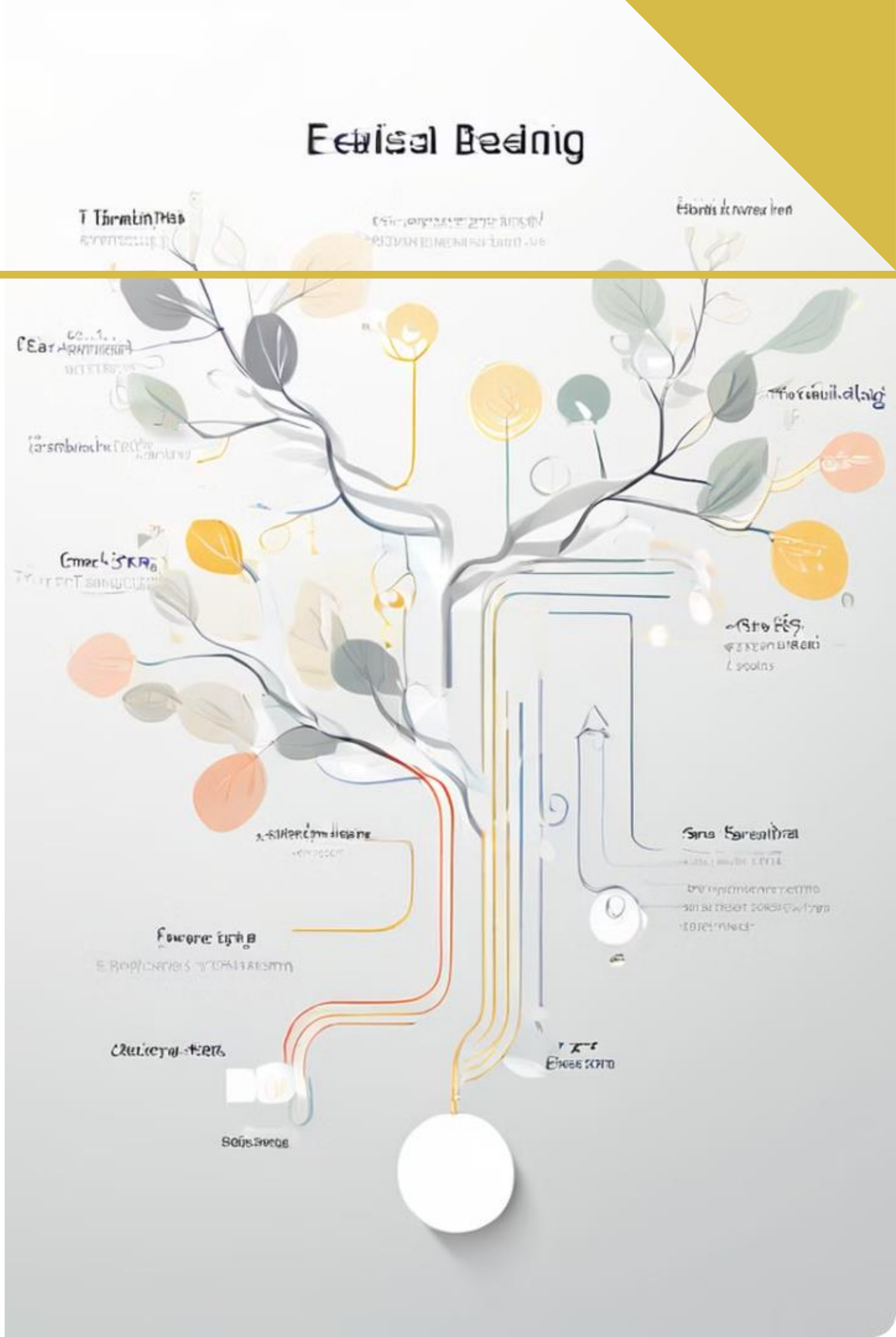
### Branching

Create subproblems by branching into feasible regions.

3

### Bounding

Evaluate bounds for the branches to find promising solutions.



# General Steps of Branch and Bound...

5

## Selection

Choose the most promising subproblem to explore next..

6

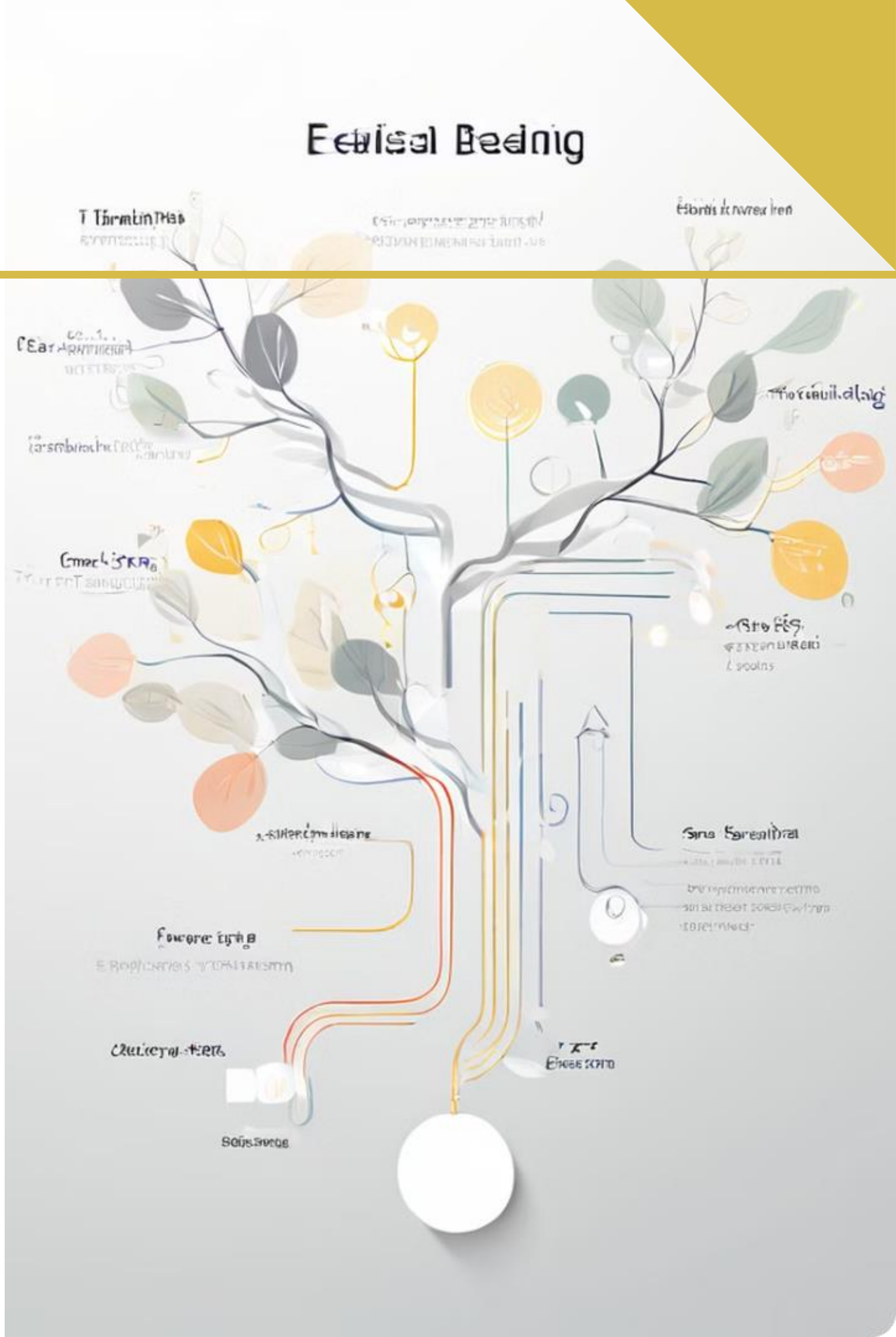
## Pruning

Discard subproblems that are not promising..

7

## Termination

Repeat until all subproblems are either solved or pruned..



# Branching Strategies

### Depth-First Search

Explore branches deeply before backtracking. Efficient for some problem types.

### Breadth-First Search

Explore all neighbors before going deeper. Guarantees the shortest path in some cases.

### Best-First Search

Select the most promising node based on heuristics. Effective for optimal searching.

# Bounding Techniques

1. **Upper Bound**

   Estimates the maximum possible value for a solution.

2. **Lower Bound**

   Estimates the minimum possible value for a solution.

3. **Relaxation Methods**

   Simplifying constraints to establish bounds more easily.

# Pruning and Fathoming

### Pruning

This technique eliminates branches that do not lead to feasible solutions.

### Fathoming

Determining a branch's inactivity by proving that it cannot yield a better solution.

### Efficiency

Pruning and fathoming significantly reduce computation time.

# Depth-First and Breadth-First Search

| Search Method | Strengths | Weaknesses |
|---|---|---|
| Depth-First | Saves memory, ideal for complex trees | Can get trapped in deep branches |
| Breadth-First | Finds shortest path, complete search | Uses more memory |

# Branching Strategies

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

### Branching

At each step, decide whether to include an item in the knapsack..

### Bounding

Calculate an upper bound on the maximum possible value that can be obtained from the current state..

### Pruning

Discard branches that cannot yield a better solution than the current best.

# The Knapsack Problem

Given n items of known weights $w_i$. And values $v_i$,i=1,2,...,n,and a knapsack of capacity W. find the most valuable subset of the items that fit in the knapsack

Order the items of a given instance in descending order by their value-to-weight ratios..

$$v_1/w_1 \geq v_2/w_2 \geq \cdots \geq v_n/w_n.$$

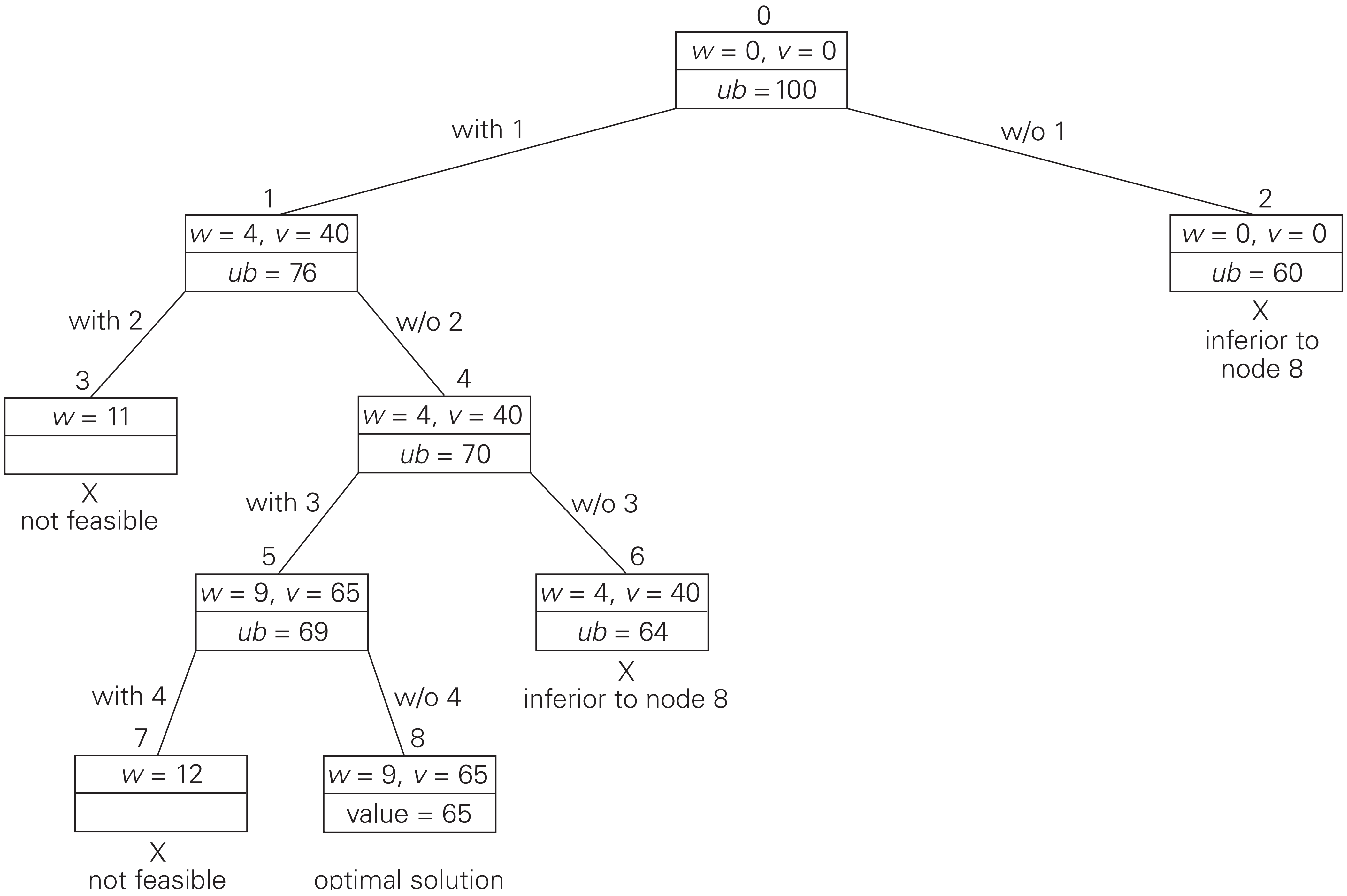
compute the upper bound ub is to add to v, the total value of the items already selected, the product of the remaining capacity of the knapsack W − w and the best per unit payoff among the remaining items, which is vi+1/wi+1:.

$$ub = v + (W - w)(v_{i+1}/w_{i+1}).$$

| **item** | **weight** | **value** | $\frac{\textbf{value}}{\textbf{weight}}$ |
|---|---|---|---|
| 1 | 4 | \$40 | 10 |
| 2 | 7 | \$42 | 6 |
| 3 | 5 | \$25 | 5 |
| 4 | 3 | \$12 | 4 |

The knapsack capacity $w = 10$

State-space tree of the best-first branch-and-bound algorithm for the instance of the knapsack problem.



# Assignment Problem

The Assignment Problem involves assigning n tasks to n agents such that the total cost is minimized.
The cost of assigning task i to agent j is given in a cost matrix. The goal is to find the optimal assignment with the minimum total cost.

### Initialization

Start with an empty assignment. Calculate the lower bound for the root node (an empty assignment).

### Pruning

Use the bound to prune branches that cannot yield a better solution than the current best.

### Branching

At each node, assign a task to an agent.
Create child nodes for each possible assignment...

### Pruning

Calculate the lower bound for each node by considering the minimum additional cost of completing the remaining tasks..

# The Knapsack Problem

Assigning n people to n jobs so that the total cost of the assignment is as small as possible

instance of the assignment problem is specified by an $n \times n$ cost matrix C

select one element in each row of the matrix so that no two selected elements are in the same column and their sum is the smallest possible

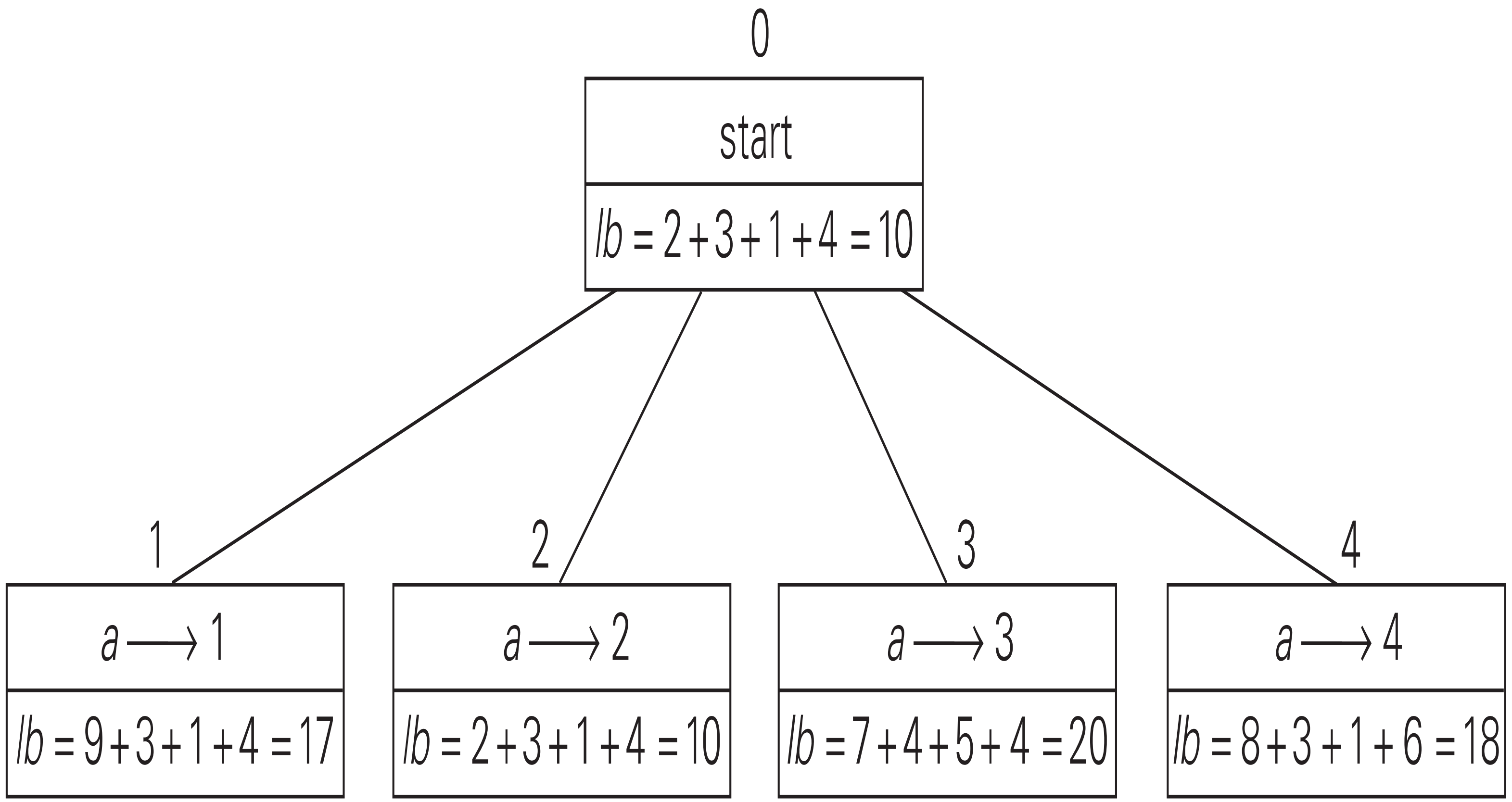
# Example

$$\begin{array}{c} \\ C = \end{array}\begin{array}{cccc} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \\ \end{array}$$

$$C = \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \begin{array}{l} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{array}$$

Best-First Branch-and-Bound is an optimization algorithm that explores the

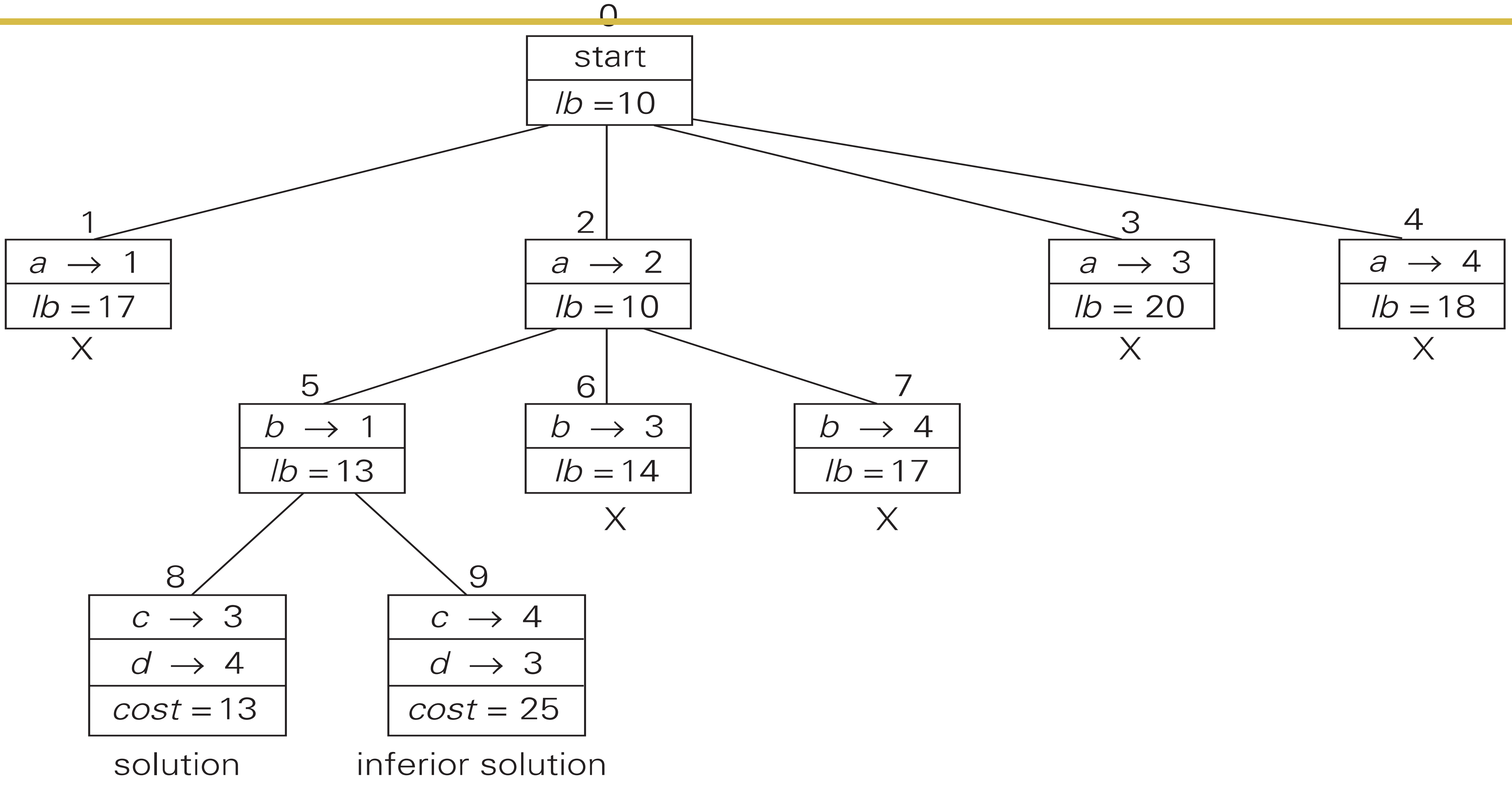
search space by always expanding the most promising node first, based on a heuristic or bound.




Levels 0 and 1 of the state-space tree for the instance of the assignment problem being solved with the best-first branch-and-bound algorithm.

The number above a node shows the order in which the node was generated. A node's fields indicate the job number assigned to person a and the lower bound value, lb, for this node.

## Final Solution



# Branching Strategies

Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

### Branching

At each step, decide whether to include an item in the knapsack..

### Bounding

Calculate an upper bound on the maximum possible value that can be obtained from the current state..

### Pruning

Discard branches that cannot yield a better solution than the current best.

## Advantages

Can solve a wide range of optimization problems.

Provides optimal solutions.

## Disadvantages

Can be computationally expensive.

Performance highly depends on the quality of bounds and branching strategy

# Applications of Branch and Bound

1. **Integer Programming**

   Solving optimization problems involving integer constraints.

2. **Combinatorial Optimization:**

   Problems like TSP, Knapsack, Job Scheduling, etc..

3. **Resource Allocation**

   Distributing resources effectively in various applications.

# Conclusion and Future Directions

Branch and Bound continues to evolve, adapting to new challenges. Advances in computational power enhance its capability.

Research aims to refine strategies for better performance.

# THANK YOU

Dedan Kimathi University of Technology

*Dr Jane Kuria*

*Private Bag- Dedan kimathi, Nyeri-Mweiga road*

*Telephone:* ***+254-721709511***

*Email:* ***jane.kuria@dkut.ac.ke***

*Website:* ***www.dkut.ac.ke***