Design and Analysis of Algorithms

L44: Branch and Bound

Job Assignment Problem
TSP Problem

Dr. Ram P Rustagi
Sem IV (2019-H1)
Dept of CSE, KSIT/KSSEM
rprustagi@ksit.edu.in

Resources

- Text book 1: Levitin
  - Sec 12.1, ***12.2***
- Text book 2: Horowitz
  - Sec **8.2**
  - R1: Introduction to Algorithms
    - Cormen et al.

Example use cases

- Job Assignment problem
- Traveling Salesperson Problem
- `0-1` Knapsack problem

Exact Solution Strategies

- Exhaustive search (brute force)
  - Search for all possible combinations (exponential time)
    - e.g. $O(k^n)$, $O(n!)$, $O(n^n)$
  - Useful only for small instances
- *Dynamic programming*
  - Applicable to some problems
    - Where problem can be recursively mapped to smaller other problems
- *Backtracking*
  - Build a state space tree
  - Eliminates unnecessary cases from consideration
  - Solutions may still take exponential time
- *Branch-and-bound*
  - Further refines backtracking for optimization problems

Branch and Bound

- Additional mechanisms in addition to backtracking
  - Provide a bound on the best value of objective function for every node of the state-space tree
  - The value of best solution so far
    - at the current node of state space tree
- Approach
  - Compare the node's bound value with the value of best solution seen so far.
    - If the bound is not better, terminate the search (prune the solution)
      - not smaller than the best sol^n in a minimization problem
      - not greater than the best sol^n in a maximization problem

Branch and Bound

- Termination criteria of the search path in state space tree using branch-n-bound algo:
  - The value of node's bound is worse than the value of best solution seen so far
  - The node represents no feasible solution because of the constraints of the problem are already violated
  - The subset of feasible solutions represented by the node consists of a single point
    - i.e. reached the end of solution and no more choices
    - Compare the value of objective function with that the best solution seen so far
      - Update the latter if former is better than latter.

Assignment Problem

- Consider a problem of assigning `n` jobs to `n` people so that cost is minimised.
  – The cost of each job done by each person is given
  – Represented in a matrix.
- Consider an example below for job assignment costs of 4 persons

	J_1	J_2	J_3	J_4
P_a	9	2	7	8
P_b	6	4	3	7
P_c	5	8	1	8
P_d	7	6	9	4

Assignment Problem

	J_1	J_2	J_3	J_4
P_a	9	2	7	8
P_b	6	4	3	7
P_c	5	8	1	8
P_d	7	6	9	4

- Problem can be stated as follows
  - Select one element in each row
    - such that no two selected elements are in same column
    - and, their sum is smallest possible.
- Solution with branch and bound
  - Consider the lowest possible sum
  - Take the lowest element in each row.
  - This may not be optimal but can act as lower bound
    - Two elements may belong to same column
  - Smallest possible values for above example
    - $P_a(J_2)=2+P_b(J_3)=3+P_c(J_3)=1+P_d(J_4)=4$
    - $=2+3+1+4=10$
    - It is not legitimate though
      - (P_b and P_c assgined same job J_3)

Assignment Problem

	J_1	J_2	J_3	J_4
P_a	9	2	7	8
P_b	6	4	3	7
P_c	5	8	1	8
P_d	7	6	9	4

- Backtracking:
  - Generate a child of last promising node
  - i.e. last active node (called E-node)
- Branch and Bound approach
  - Generate the child of most promising node
    - among non-terminated live leaves in current tree
    - Achieved by comparing lower bounds of all live nodes
  - Intuitively, it is better to consider a node with best bound as most promising
    - It may not lead to optimal solution
    - It may lie in other branch of the tree.

Assignment Problem

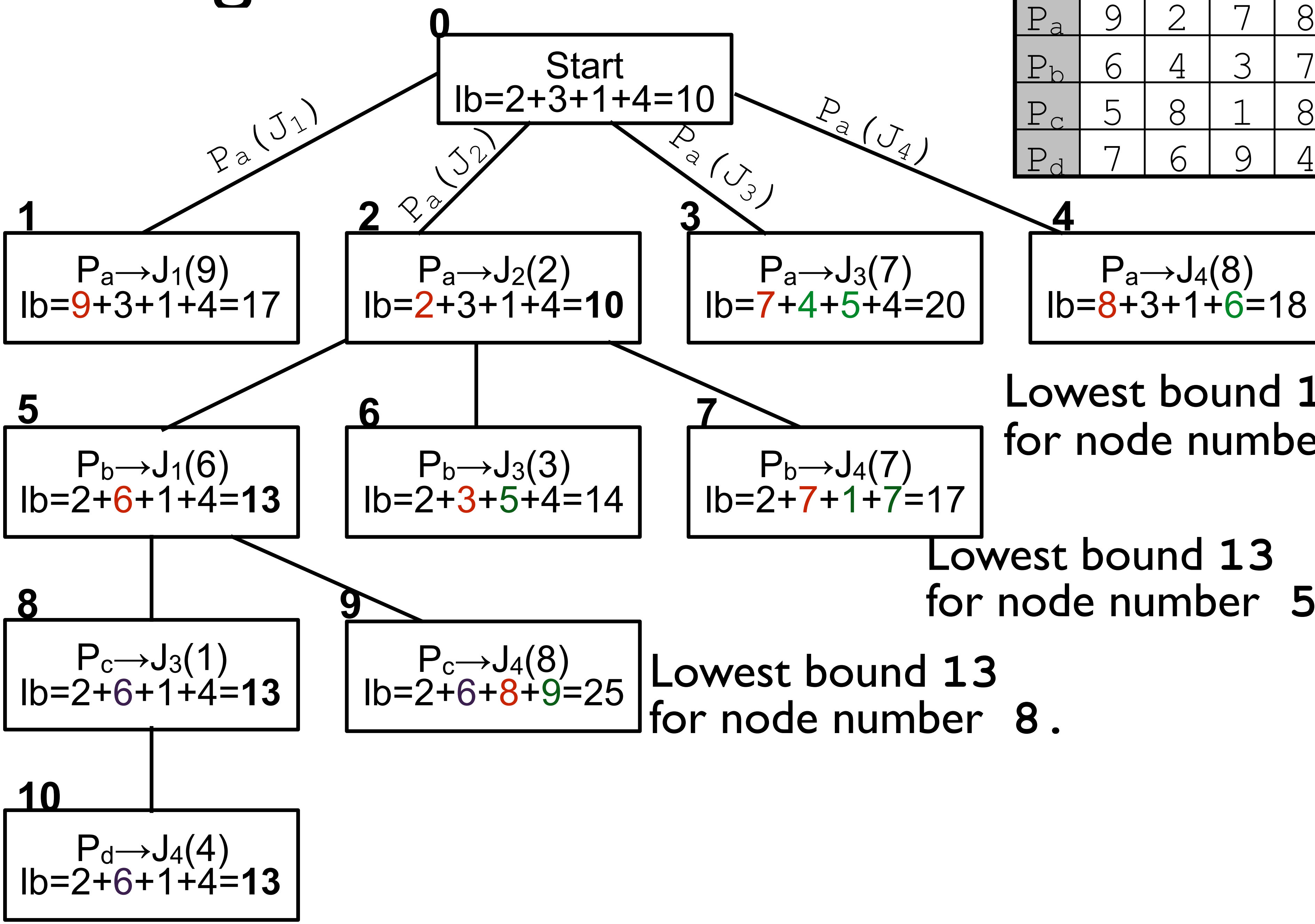
	J_1	J_2	J_3	J_4
P_a	9	2	7	8
P_b	6	4	3	7
P_c	5	8	1	8
P_d	7	6	9	4

Start
lb=2+3+1+4=10

- Start at root
  - no elements selected from matrix
  - Lower bound lb=10
- It (root) has 4 live nodes for P_a
  $J_1=9$, $J_2=2$, $J_3=7$, $J_4=8$,
  - Lower bounds for each of these are
    - LB for $P_a(J_1=9)=9+3+1+4=17$
      $P_a(J_1=9)+P_b(J_3=3)+P_c(J_3=1)+P_d(J_4=4)$
    - LB for $P_a(J_2=2)=2+3+1+4=10$
    - LB for $P_a(J_3=7)=7+4+5+4=20$
    - LB for $P_a(J_4=8)=8+3+1+6=18$
- Most promising node for P_a is 2.
  - Explore this node further.

Assignment Problem

	J_1	J_2	J_3	J_4
P_a	9	2	7	8
P_b	6	4	3	7
P_c	5	8	1	8
P_d	7	6	9	4

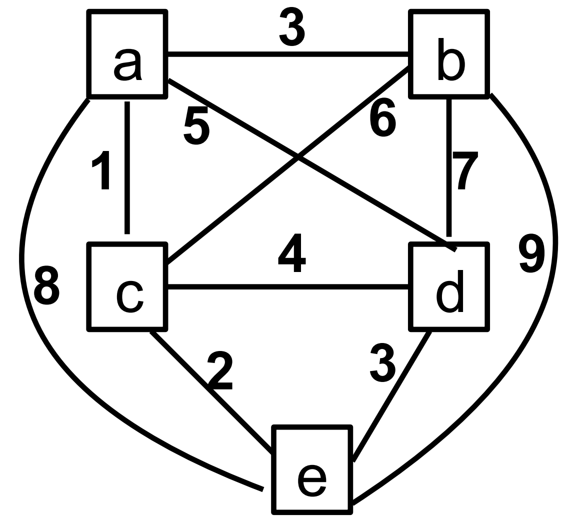


Lowest bound 10 for node number 2.

Lowest bound 13 for node number 5.

Lowest bound 13 for node number 8.

Traveling Salesperson Problem

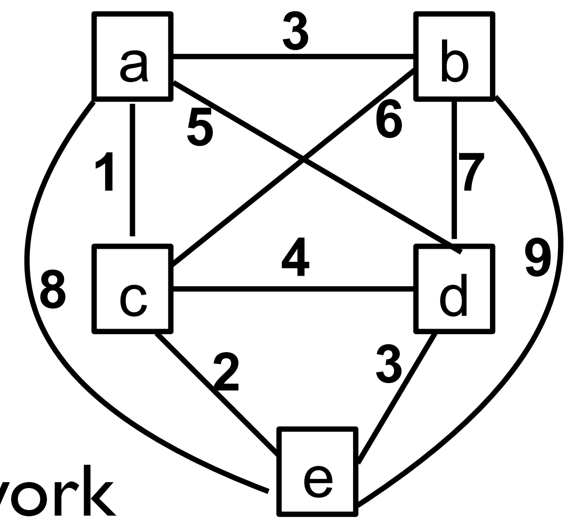


- BB approach: define a lower bound
- Simple approach:
  - Take the lowest edge cost
  - Multiply it by number of nodes
    - LB=1*5=5
- More informative but less obvious
  - Does not require much computation too
  - For each node, find two nearest nodes
    - Find the average of two
  - Sum this average (ceiling) for all nodes

LB=((1+3)+(3+6)+(1+2)+(3+4)+(2+3))/2=14

- When any tour includes a particular edge,
  - Update the lower bound accordingly
    - Using the included edge

Traveling Salesperson Problem

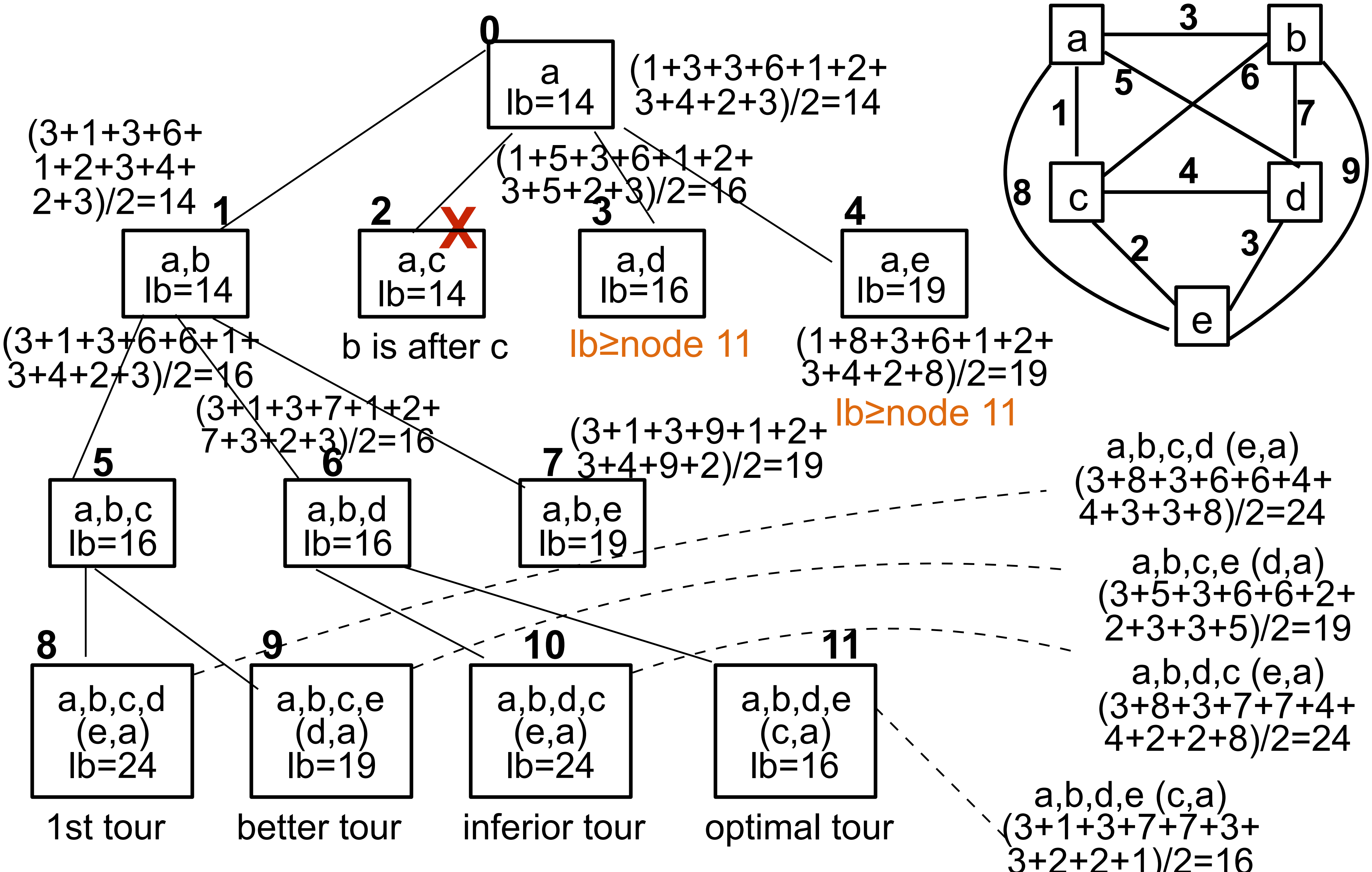


- When any tour includes a particular edge,
  - compute the lower bound using that edge.
- Example: Consider `(a,d)` is included.

$$LB=\lceil((\mathbf{5}+1)+(3+6)+(1+2)+(\mathbf{5}+3)+(2+3))/2\rceil=\lceil 31/2\rceil=16$$

- Further, to reduce the amount of potential work
  - We can consider that tour starts at node `a`, and
  - Since graph is undirected, impose the restriction
    - Generate tours in which `b` appears before `c`.
  - After visiting `n-1(=4)` nodes,
    - Tour has to visit the last unvisited node, and
    - Return to the starting node.

Traveling Salesperson Problem



Branch and Bound

- Finding a good bound function is a challenging task
  - May not be always easy to find one
- Bounding function should be easy to compute
- It should not be too simple
  - It may fail to prune the many branches of state space tree as soon as possible
- Finding the balance between two requirements (easy to compute, and not too simplistic)
  - may require intensive experimentation
  - with a wide variety of problem in question

Summary

- Assignment problem
- Traveling Salesperson Problem