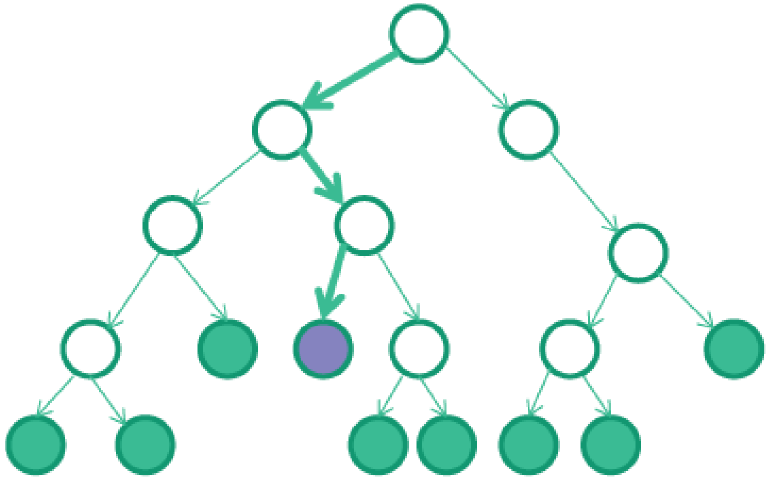
DSE 2256 DESIGN & ANALYSIS OF ALGORITHMS

Lecture 42

Coping with the Limitations of Algorithm Power using Branch-and-Bound

Assignment Problem, Knapsack Problem, Travelling Salesman Problem



Branch-and-bound

- An enhancement of backtracking.
- Applicable to optimization problems.
- For each node (partial solution) of a state-space tree, computes a bound on the value of the objective function for all descendants of the node (extensions of the partial solution)
- Uses the bound for:
  - Ruling out certain nodes as "nonpromising" to prune the state-space tree – if a node's bound is not better than the best solution seen so far
  - Guiding the search through state-space
- Compared to Backtracking, Branch-and-Bound traverses the tree in any manner, DFS or BFS.

Assignment Problem using Branch-and-bound

Select one element in each row of the cost matrix C so that:

- No two selected elements are in the same column
- The sum is minimized

Example

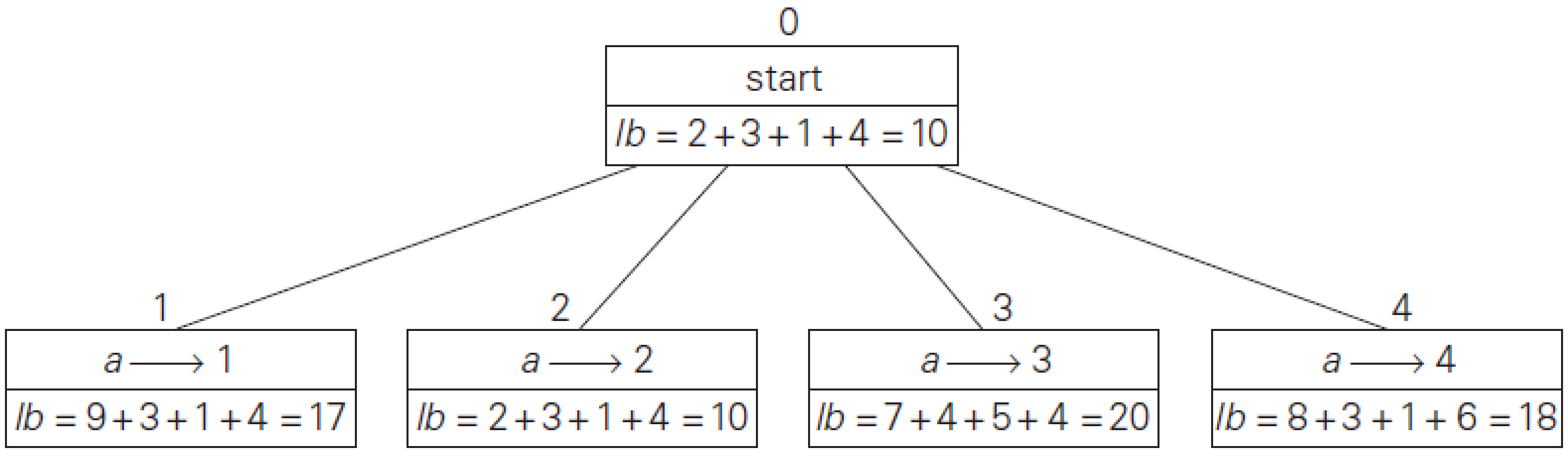
	Job 1	**Job 2**	**Job 3**	**Job 4**
Person *a*	9	2	7	8
Person *b*	6	4	3	7
Person *c*	5	8	1	8
Person *d*	7	6	9	4

Lower bound: Any solution to this problem will have total cost at least: 2 + 3 + 1 + 4 = 10 (scanning row-wise)

(or 5 + 2 + 1 + 4 = 12 (scanning column-wise))

Assignment Problem using Branch-and-bound

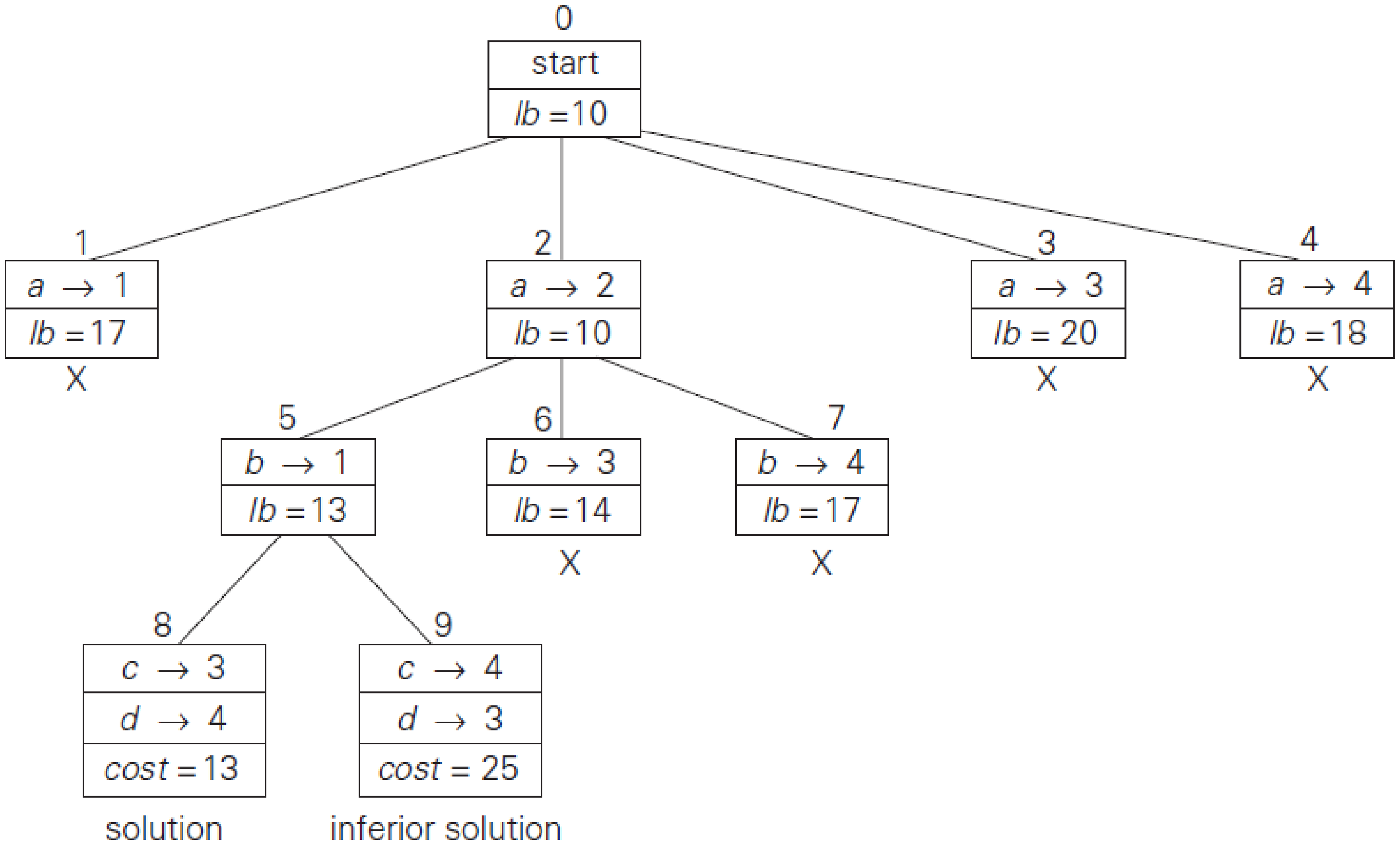
	Job 1	Job 2	Job 3	Job 4
Person *a*	9	2	7	8
Person *b*	6	4	3	7
Person *c*	5	8	1	8
Person *d*	7	6	9	4



Level 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

Assignment Problem using Branch-and-bound

	Job 1	Job 2	Job 3	Job 4
Person *a*	9	2	7	8
Person *b*	6	4	3	7
Person *c*	5	8	1	8
Person *d*	7	6	9	4



The complete state-space tree for the instance of the assignment problem solved with the best-first branch-and-bound algorithm

Knapsack Problem using Branch-and-bound

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

- Each node on the i^{th} level of this tree, $\mathbf{0 \le i \le n}$, represents all the subsets of *n* items that include a particular selection made from the first ***i*** ordered items

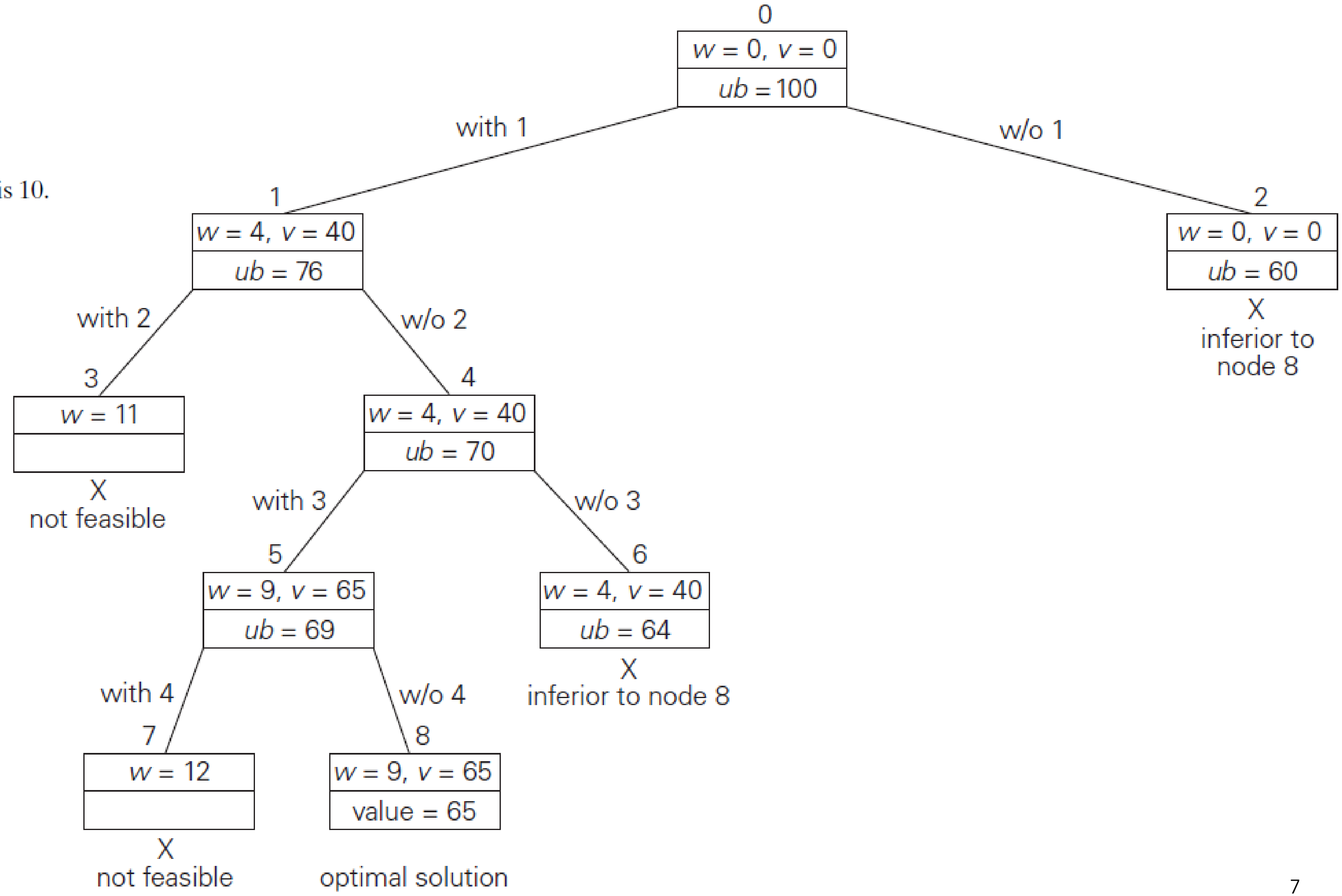
- Selection is uniquely determined by the path from the root to the node: a branch going to the left indicates the inclusion of the next item, and a branch going to the right indicates its exclusion.

- An upper bound value is required and computed as: $\boldsymbol{ub = v + (W - w)(v_i+1/w_i+1)}$

Knapsack Problem using Branch-and-bound

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

The knapsack's capacity W is 10.

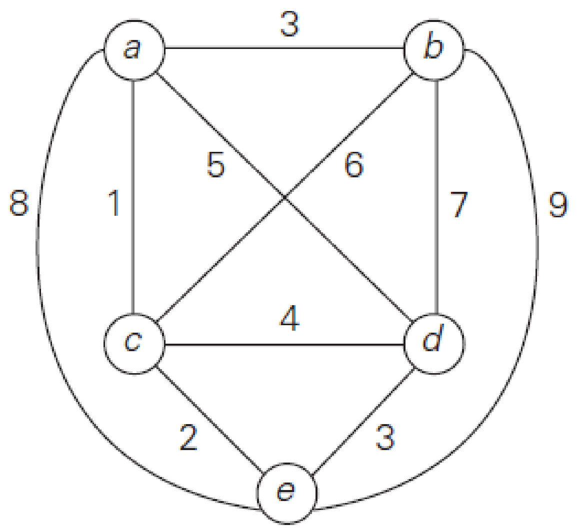


Travelling Salesman Problem using Branch-and-bound

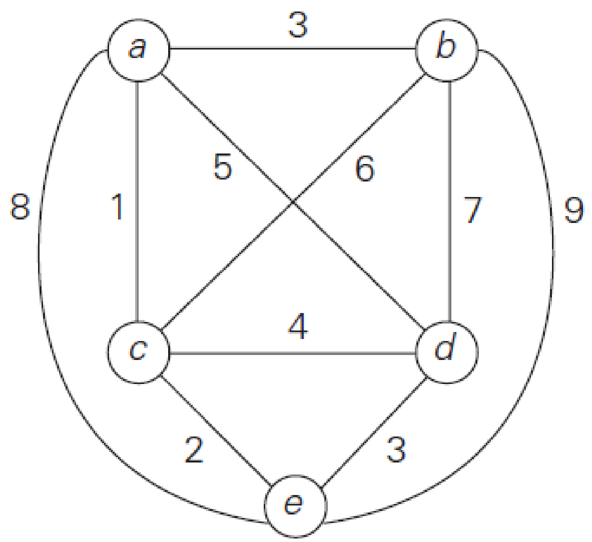
- For each city i (where 1≤ i ≤ n), find the sum s_i of the distances from city i to the two nearest city:
  - Compute the sums of these n numbers, divide the result by 2.
  - If all the distances are integers, round up the result to the nearest integer: $lb = \lceil s/2 \rceil$

<u>Example:</u> For the given graph,

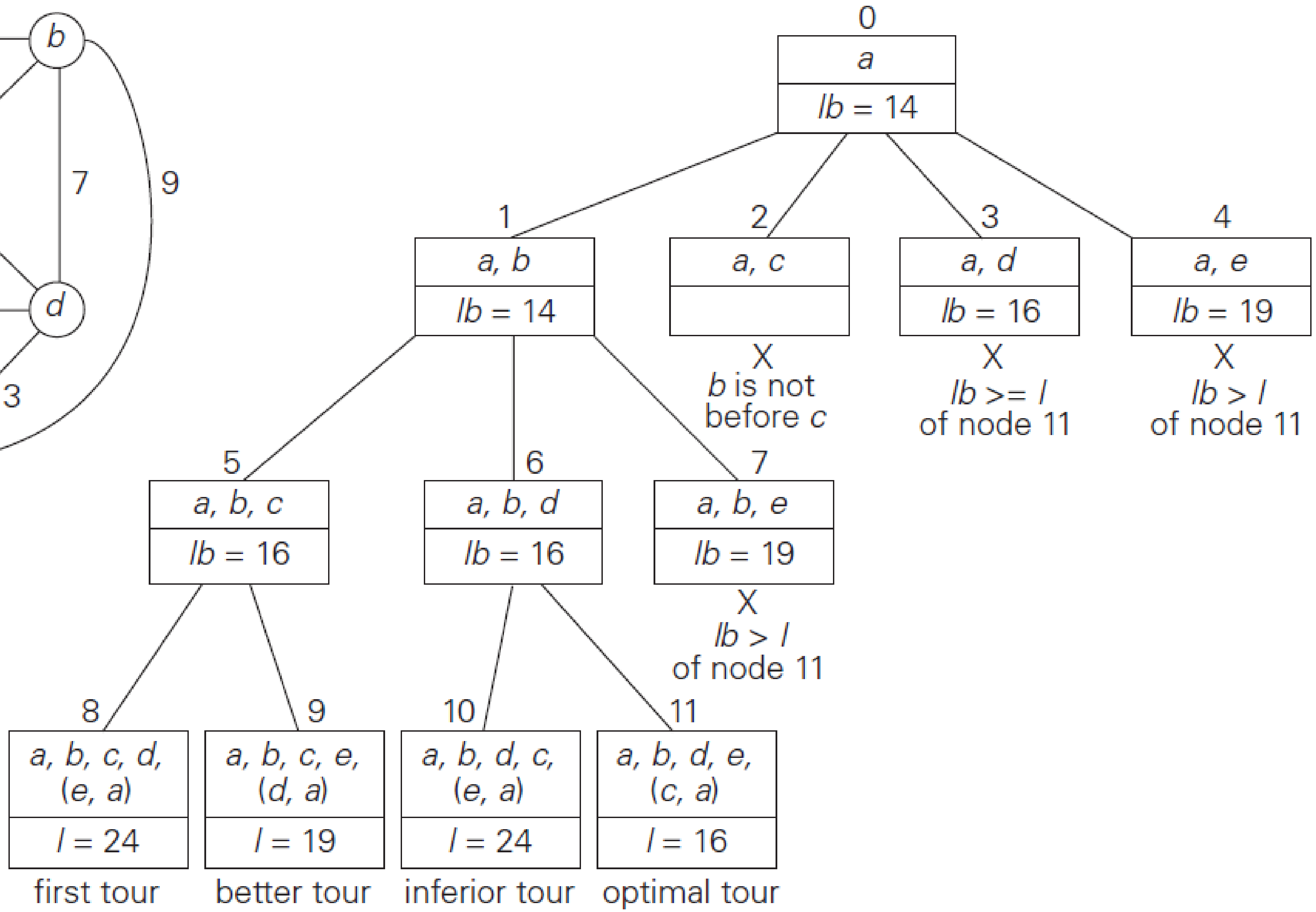
$$lb = \lceil [(1+3)+(3+6)+(1+2)+(3+4)+(2+3)]/2 \rceil = 14.$$



Travelling Salesman Problem using Branch-and-bound



(a)



Thank you!

Any queries?