Chapter 12: Coping with the Limitations of Algorithm Power

There are two principal approaches to tackling NP-hard problems or other "intractable" problems:

- Use a strategy that guarantees solving the problem exactly but doesn't guarantee to find a solution in polynomial time

- Use an approximation algorithm that can find an approximate (sub-optimal) solution in polynomial time

Exact solutions

The exact solution approach includes the strategies:

- ***Exhaustive search* (brute force)**
  - useful only for small instances
- **Dynamic programming**
  - Applicable for some problems, e.g., knapsack problem, TSP
- ***Backtracking***
  - eliminates some cases from consideration
  - yields solutions in reasonable time for many instances but worst case is still exponential
- ***Branch-and-bound***
  - Only applicable for optimization problems
  - further cuts down on the search
  - fast solutions for most instances
  - worst case is still exponential

Need a state-space tree

Nodes: partial solutions
Edges: choices in completing solutions

Backtracking

Construct the *state-space tree*:

- nodes: partial solutions
- edges: choices in completing solutions

Explore the state space tree using depth-first search (DFS)

“Prune” non-promising subtrees

- DFS stops exploring subtree rooted at nodes leading to no solutions and
- “backtracks” to its parent node

Branch and Bound

An enhancement of backtracking.

Applicable to optimization problems

Uses a lower bound for the value of the objective function for each node (partial solution) to:

- no solution can beat the lower bound
- guide the search through state-space
- rule out certain branches as "unpromising" – do not explore these subtrees
- using a "best-first" rule

Example: The assignment problem

For 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

Cost matrix

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

- ***no two selected elements are in the same column; and***
- ***the sum is minimized***

If using exhaustive search, the permutation of n persons ⇨ $\Theta(n!)$

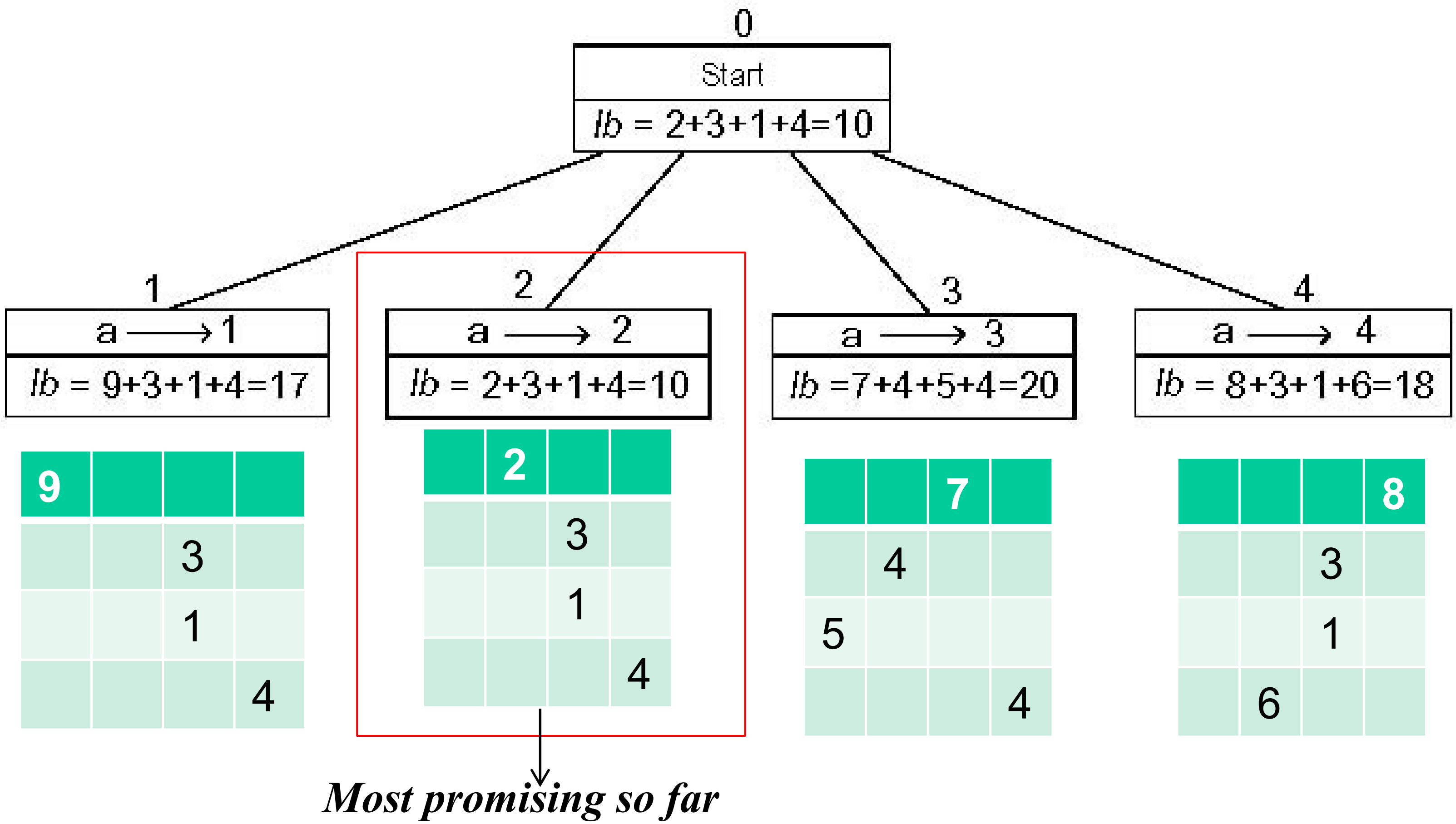
Example: The assignment problem

	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

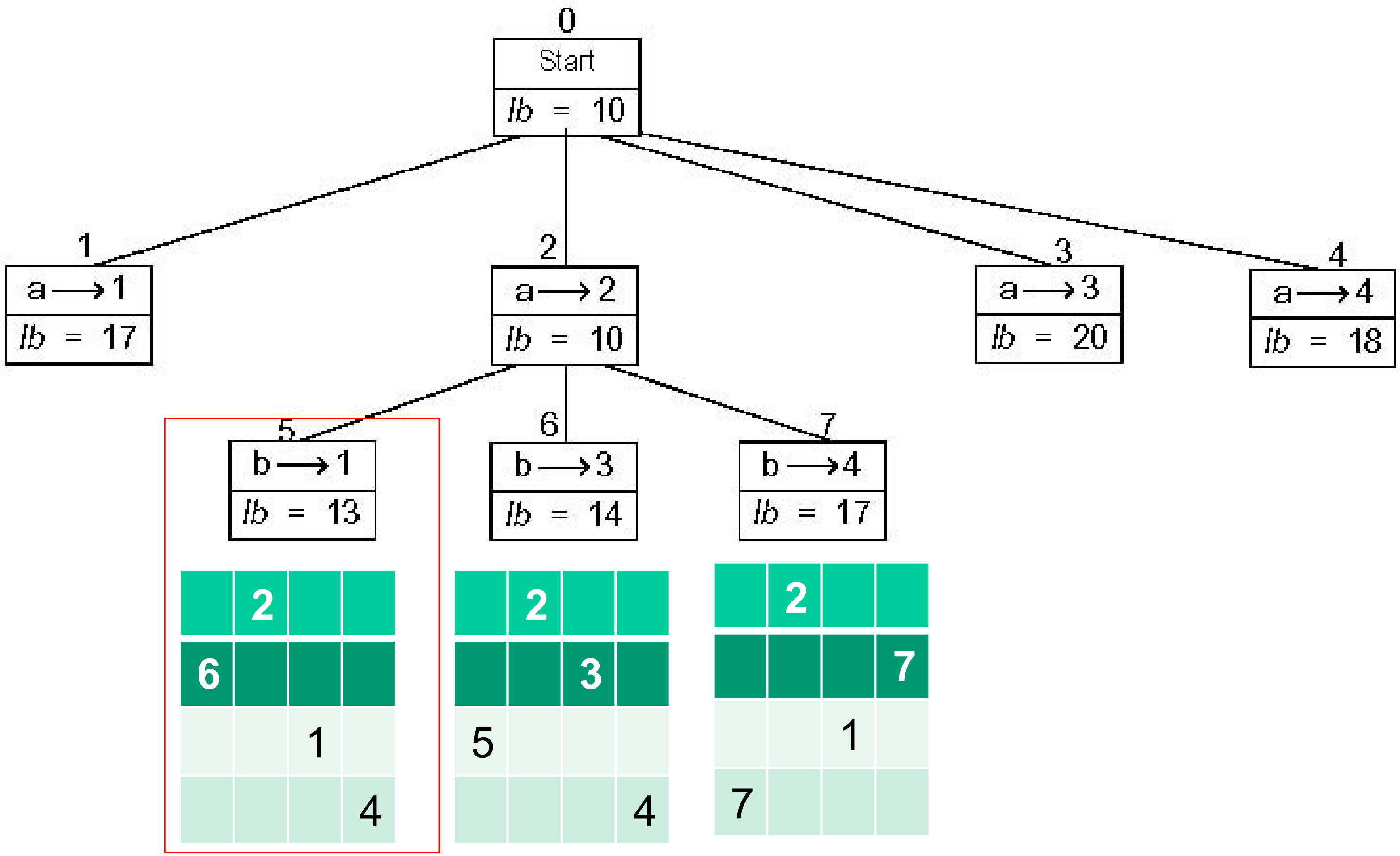
Lower bound*: Any solution to this problem will have total cost of *at least*:** ***The summation of the smallest elements in each row

No solution can beat the lower bound!

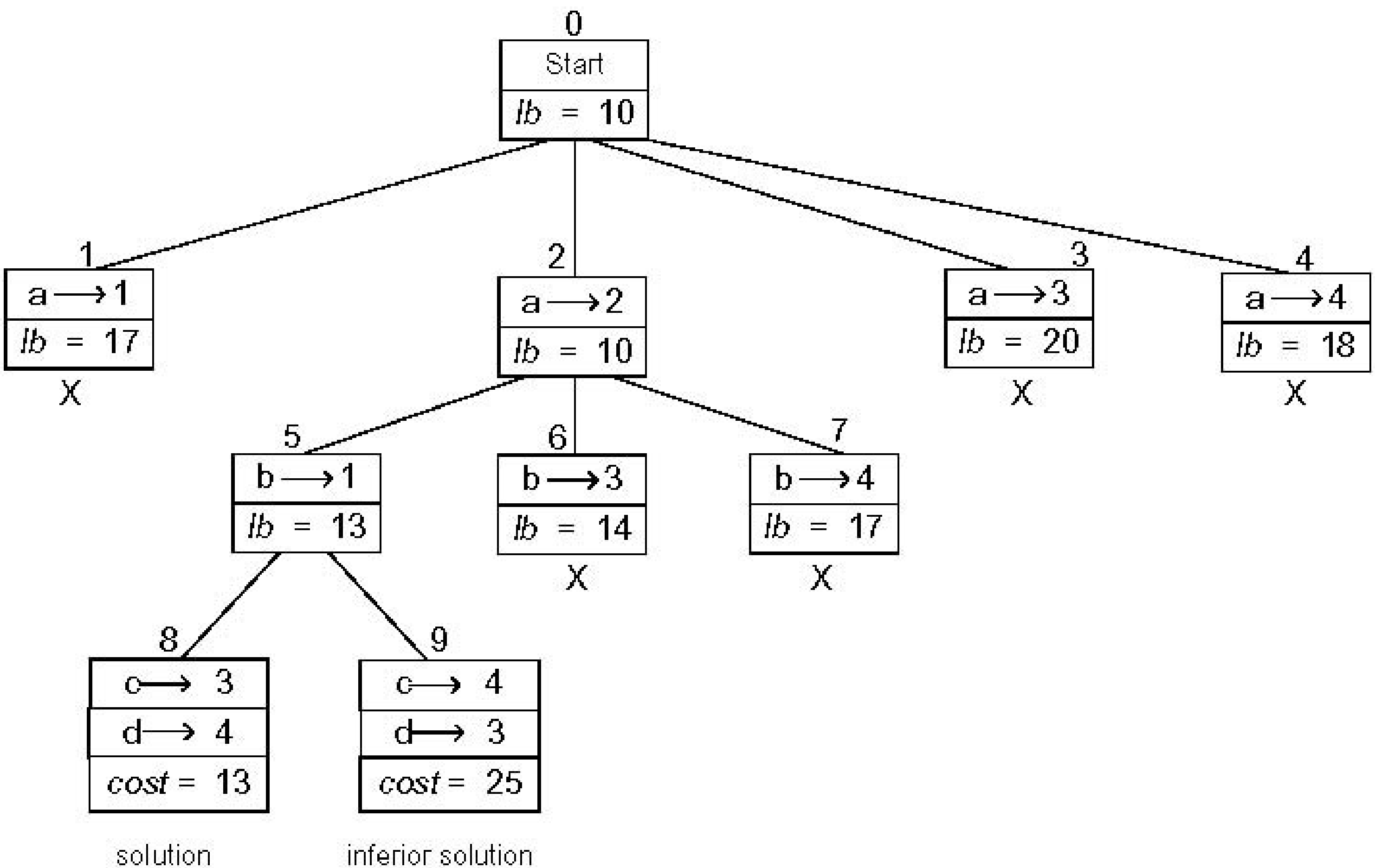
Assignment problem: lower bounds



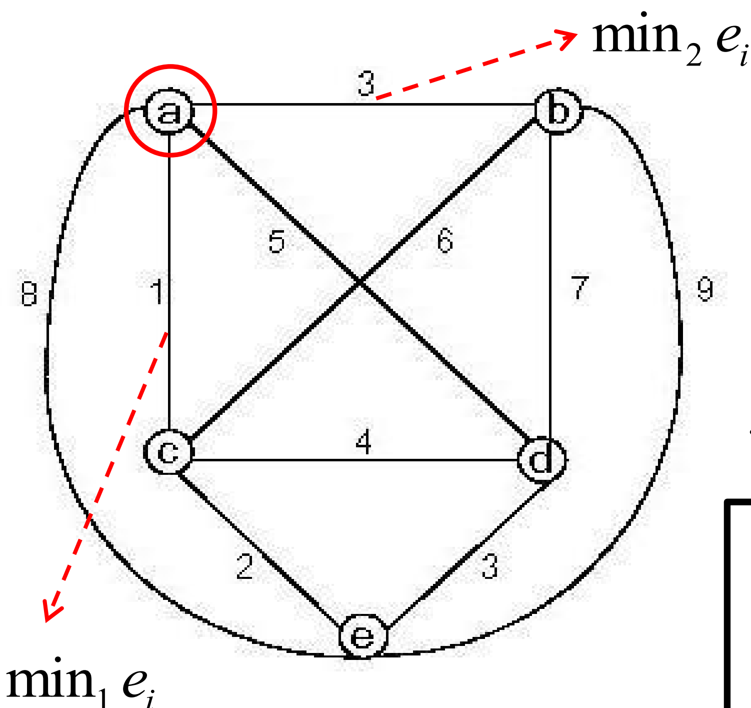
State-space levels 0, 1, 2



Complete state-space



Traveling salesman example:



How to find the lower bound for each step?

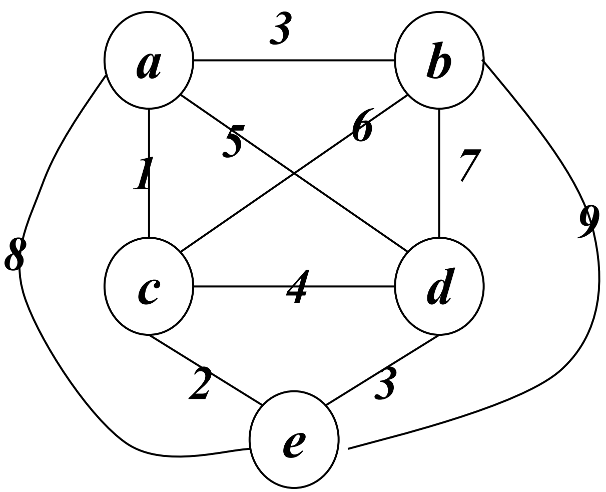
$$lb = \left\lceil \sum_{i=1}^{N} (\min_1 e_i + \min_2 e_i) / 2 \right\rceil$$

for N nodes

Constraints:

- ***start from a***
- ***b should be visited before c***
- ***After visiting n-1 vertices, the last vertex must be visited and go back to a***

Traveling salesman example:

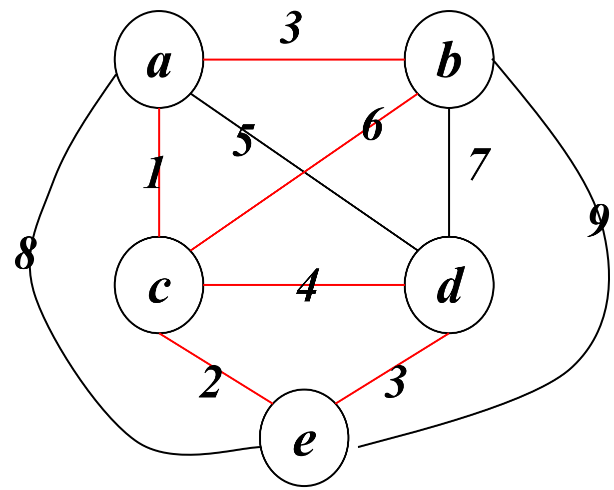


	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

a

$$\left\lceil \left[(1+3)+(3+6)+(1+2)+(3+4)+(2+3)\right]/2 \right\rceil$$

a (lb=14)

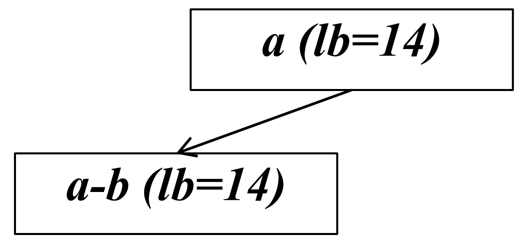


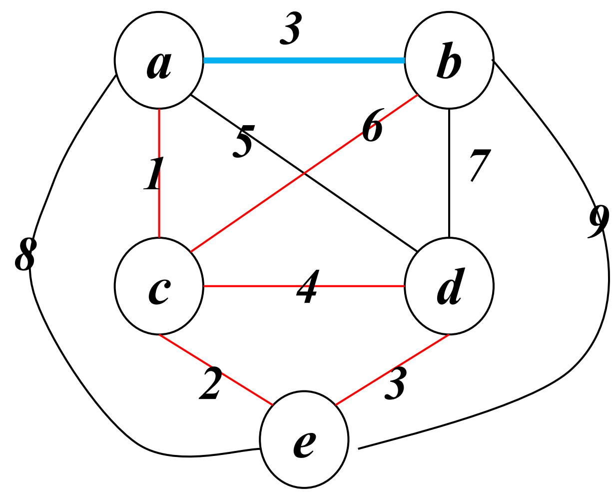
Traveling salesman example:

	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

a-b

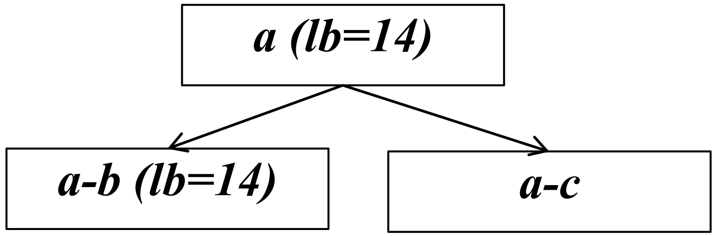
$$\left\lceil \left[(1+\overset{a\text{-}b}{\boxed{3}})+(\overset{b\text{-}a}{\boxed{3}}+6)+(1+2)+(3+4)+(2+3)\right]/2 \right\rceil$$





Traveling salesman example:

	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0



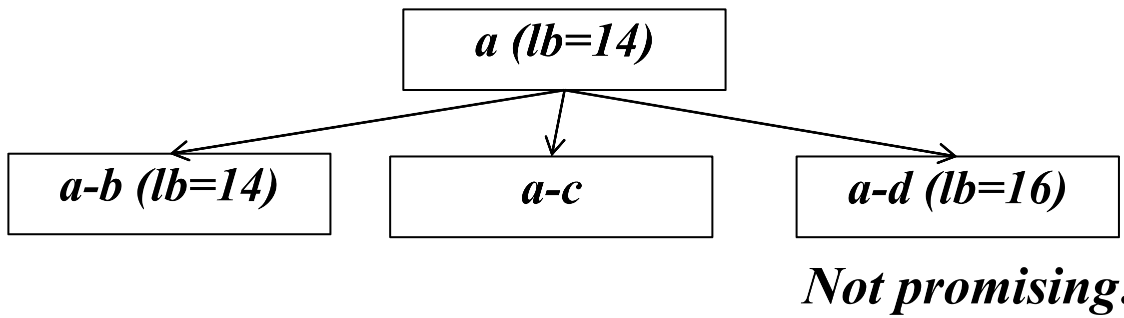
Does not satisfy the constraint b should be visited before c

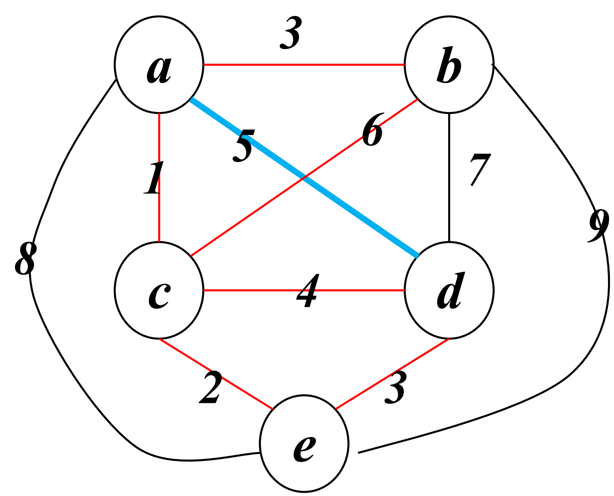
Traveling salesman example:

	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

a-d

$$\left\lceil \left[(1+\overset{a\text{-}d}{\boxed{5}})+(3+6)+(1+2)+(3+\overset{d\text{-}a}{\boxed{5}})+(2+3)\right]/2 \right\rceil$$



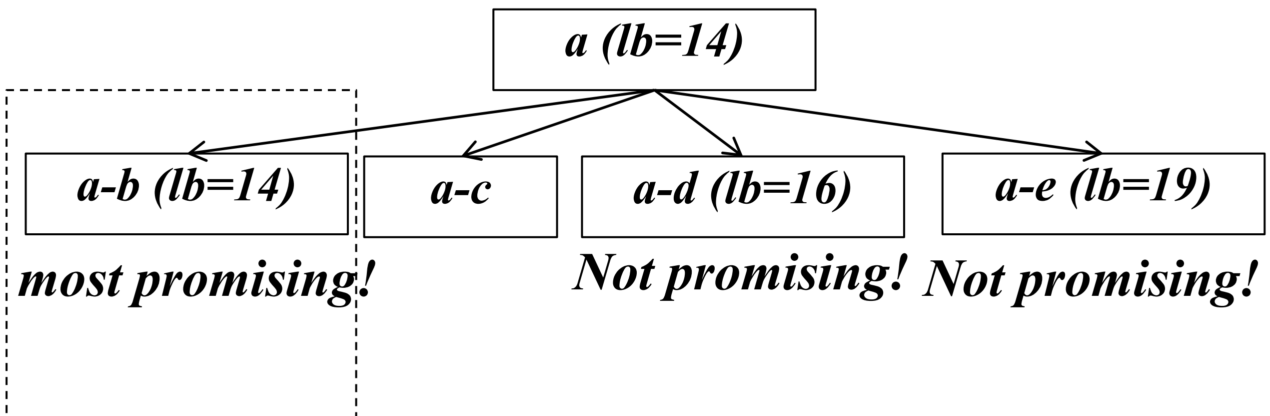


Traveling salesman example:

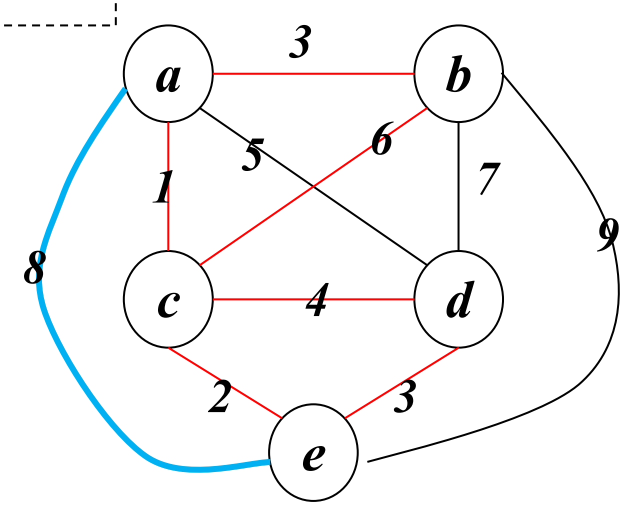
	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

a-e

$$\left\lceil \left[(1+\overset{a\text{-}e}{\boxed{8}}) + (3+6) + (1+2) + (3+4) + (2+\overset{e\text{-}a}{\boxed{8}}) \right] / 2 \right\rceil$$



a-b is the most promising, explore it

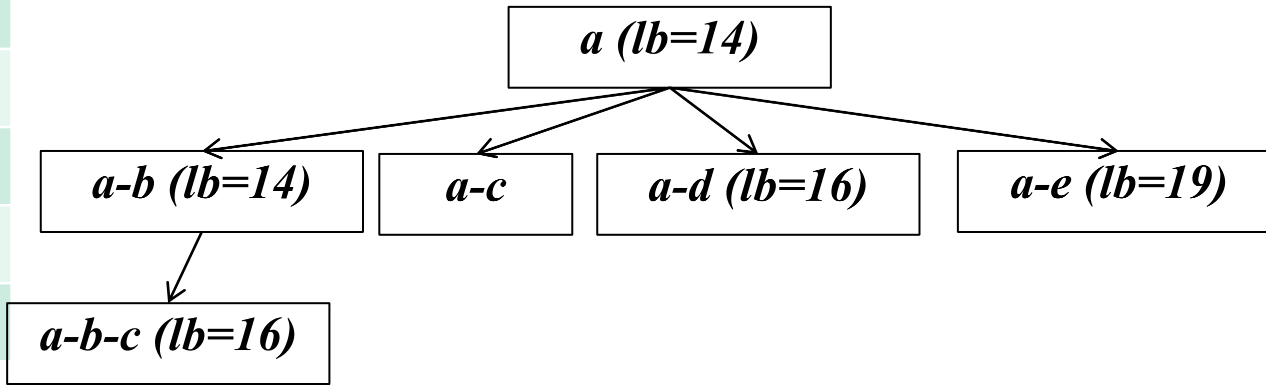


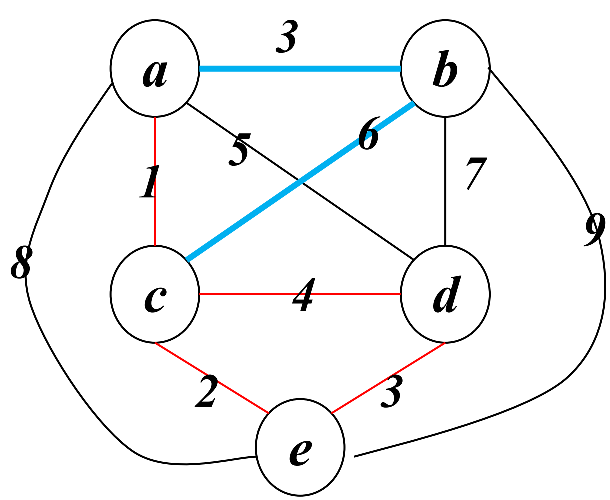
Traveling salesman example:

	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

a-b-c

$$\left\lceil \left[(1+\overset{a\text{-}b}{\boxed{3}})+(\overset{b\text{-}a}{\boxed{3}}+\overset{b\text{-}c}{\boxed{6}})+(1+\overset{c\text{-}b}{\boxed{6}})+(3+4)+(2+3)\right]/2 \right\rceil$$

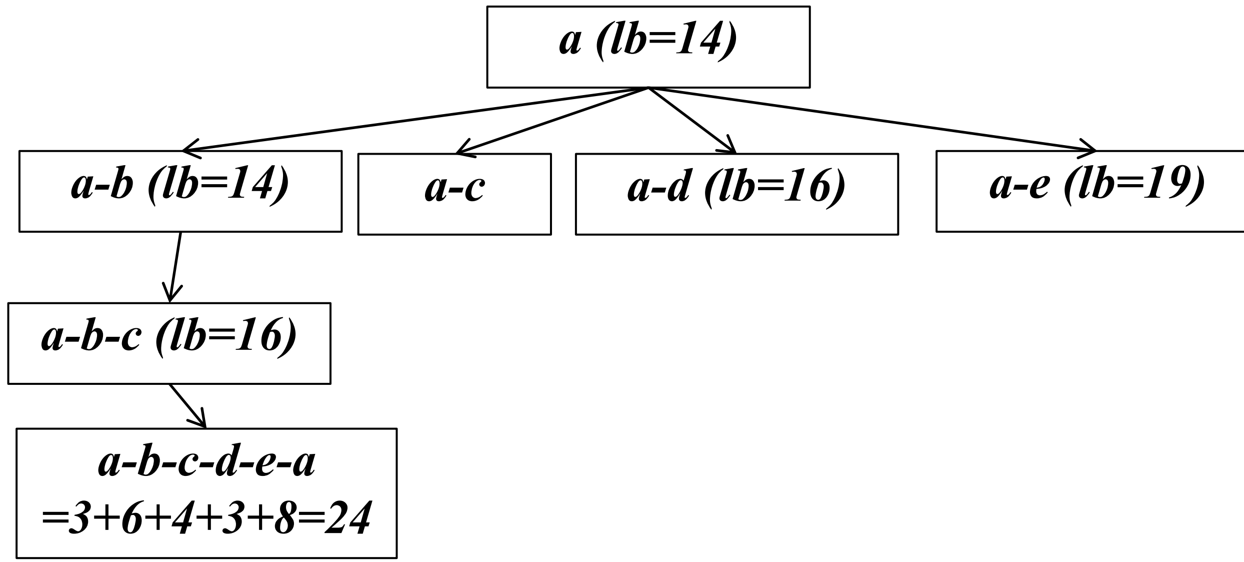


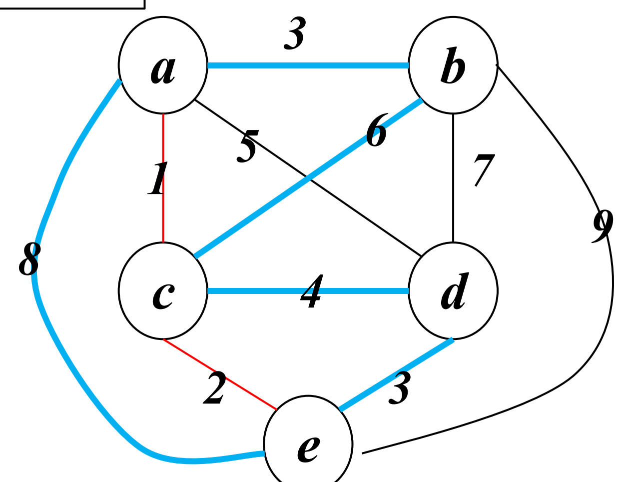


Traveling salesman example:

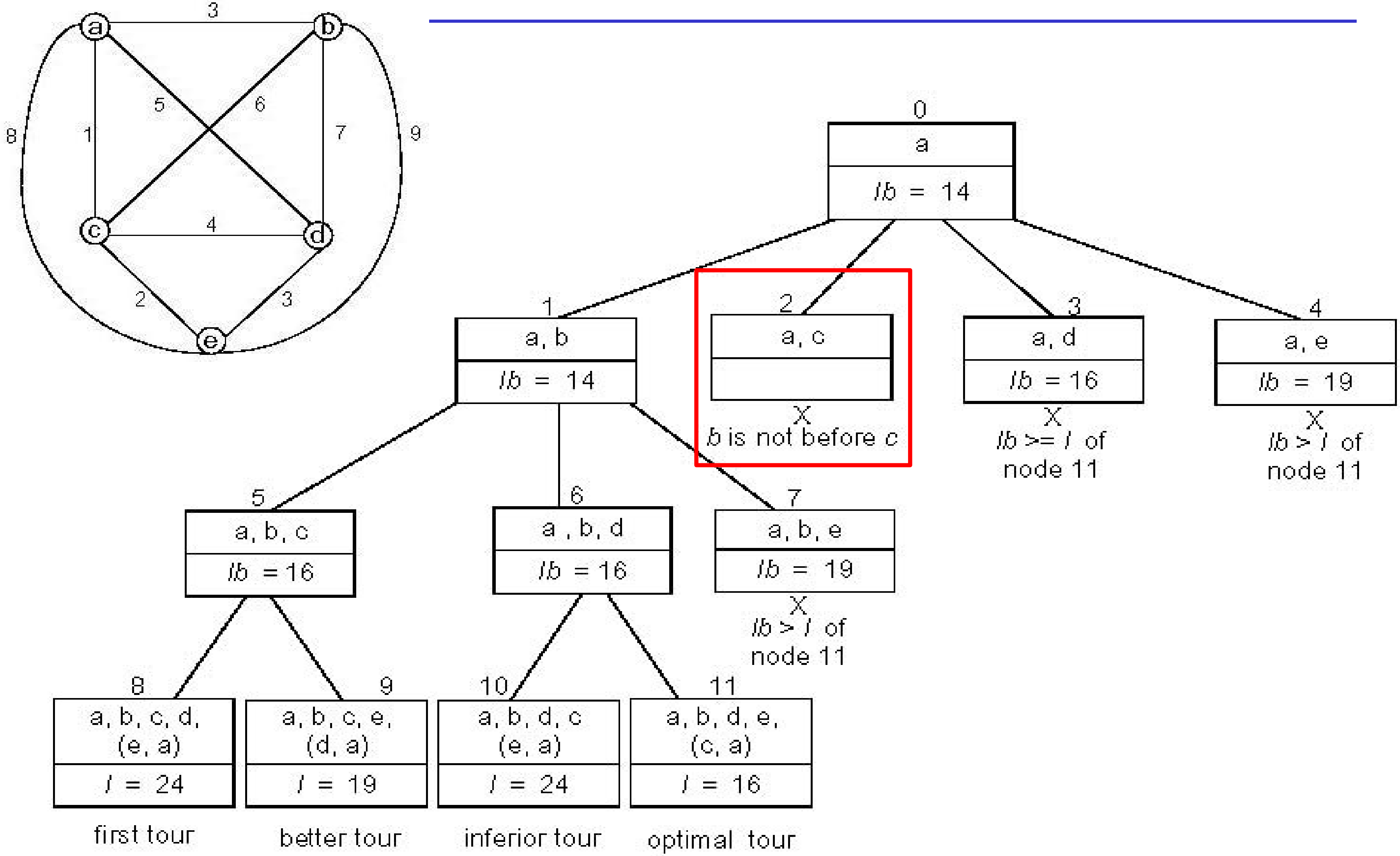
	a	b	c	d	e
a	0	3	1	5	8
b	3	0	6	7	9
c	1	6	0	4	2
d	5	7	4	0	3
e	8	9	2	3	0

a-b-c-d-e-a





Traveling salesman example:



Discussion on TSP using Branch-Bound

For every node except the n-1th vertex, we need to compute its corresponding lower bound.

For the n-1th vertex, we need to compute the total length

What operations we need?

- ***Find the minimum cost of each row***
- ***Calculate the summations***
- ***Compare with the best partial solution so far***

Can we improve the efficiency?

Yes. Just update the cost involving the change.