Probabilistic Algorithms

Paul Cotofrei

information management institute

PA 2016

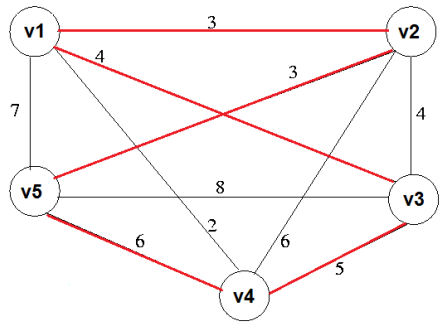
Outline

Traveling Salesperson Problem

- TSP is a famous discrete optimization problem
- Basic problem is to find best way for salesperson to hit every city in territory once and only once
  - Setting arises in many problems of optimization on networks (communications, transportation, etc.)
- If a tour involves n cities, there are $(n-1)!/2$ possible solutions
  - Extremely rapid growth in solution space as n increases
  - Problem is extremely hard
  - A central role in combinatorial optimisation
  - Standard testbed for new algorithmic ideas

Mathematical formulation

- Let be a graph $G = (V, E, d)$, where V is the set of n vertices $\{v_1, v_2, .., v_n\}$, $E \subset V \times V$ is the set of edges, and $d : E \to R^+$ is function assigning to each edge $e = (v_i, v_j)$ the length $d(e)$ (the distance between the two vertices).
- A path in G is a list of vertex $\{u_1, ..., u_k\}$ such that each $(u_i, u_{i+1}) \in E, i = 1..k-1$. If $u_1 = u_k$, the path is cyclic
- A Hamiltonian cycle is a cyclic path that visits every vertex of G (except for its starting point) exactly once, i.e. $H = \{u_1, ..., u_n, u_1\}$



- The length of a path $p = \{u_1, ..., u_k\}$ is the sum $\sum_{i=1}^{k-1} d(u_i, u_{i+1})$

TSP: find a Hamiltonian cycle with minimal path length in G.

Distance metrics

- Suppose that the vertices v_i (cities, nodes) are points in R^2 with coordinates (x_i, y_i)
- If the distance $d(i,j)$ between two vertices v_i and v_j is not given explicitly, it must be calculated according to a L_p metric
  - General case: $d(i,j) = (|x_i - x_j|^p + |y_i - y_j|^p)^{1/p}$
  - Euclidian distance ($p = 2$): $d(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
  - Manhattan distance ($p = 1$): $d(i,j) = |x_i - x_j| + |y_i - y_j|$
  - Tschebyscheff distance ($p = \infty$): $d(i,j) = \max\{|x_i - x_j|, |y_i - y_j|\}$

TSP Instance

- We may consider the graph G as a complete graph (there is an edge between any two vertices).
  - if G is not complete, we can always construct a complete graph G' such that the TSP for G' has exactly the same solutions as the one for G. Why ?
- If G complete graph, then the distance function may be represented by a matrix $A_{n\times n}$ such that $d(v_i, v_j) = d_A(i, j) = A(i, j)$
- The simplest representation for a TSP instance: a permutation σ of the set $V = \{1, 2, ..., n\}$
  - The starting node is $\sigma(1)$, the second node in the path is $\sigma(2)$, etc..
  - The length of the Hamiltonian cycle is

$$H(\sigma) = \sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1))$$

  - The loss function $L() = H(\sigma)$ is defined on $\mathcal{S}_n$ (the set of all permutations of V) with values in R^+

TSP Instance

- A second representation for a TSP instance: an adjacency matrix η of dimension $n \times n$

$$\eta(i,j) = \begin{cases} 1 & \text{if } j \text{ is the successor or the predecessor of } i \\ 0 & \text{otherwise} \end{cases}$$

- The length of the Hamiltonian cycle

$$H(\eta) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} d(i,j)\eta(i,j)$$

- **Note**: not every possible adjacency matrix represent a feasible solution! The necessary constraints are:
  - In the path, each node has a single successor and a single predecessor:

$$\forall j = 1..n, \sum_{i=1}^{n} \eta(i,j) = 2 \text{ and } \forall i = 1..n, \sum_{j=1}^{n} \eta(i,j) = 2 \qquad (1)$$

  - All nodes must be in one cycle:

$$\forall U \subset V = \{1,2,..,n\}, \text{ with } U \neq \emptyset \text{ and } U \neq V, \sum_{i,j \in U} \eta(i,j) < 2|U| \qquad (2)$$

- The domain of the loss function $L() = H(\eta)$ is the set of $n \times n$ matrices with values in $\{0,1\}$ satisfying the constraints (1) and (2)

TSP Instance

- Another representation: an incidence matrix $\bar{\eta}$ of dimension $n \times n$

$$\bar{\eta}(i,a) = \bar{\eta}_{i,a} = \begin{cases} 1 & \text{if the node number } i \text{ is the } a^{th} \text{ node in the cycle} \\ 0 & \text{otherwise} \end{cases}$$

  with $\bar{\eta}_{i,n+1} \equiv \bar{\eta}_{i,1}, \forall i$

- The length of the Hamiltonian cycle

$$H(\bar{\eta}) = \sum_{i,j,a} d(i,j)\bar{\eta}_{i,a}\bar{\eta}_{j,a+1}$$

- Not every possible incidence matrix represent a feasible solution. The necessary constraints are:
  - Each node is visited exactly once: $\sum_a \bar{\eta}_{i,a} = 1 \forall i$
  - Each position in the cycle contains exactly one node: $\sum_i \bar{\eta}_{i,a} = 1 \forall a$
- The domain of the loss function $L()$ is the set of all $n \times n$ matrices with values in $\{0,1\}$, if $L()$ includes also penalty functions:

$$L(\bar{\eta}) = H(\bar{\eta}) + \lambda_1 \sum_i \left(\sum_a \bar{\eta}_{i,a} - 1 \right)^2 + \lambda_2 \sum_a \left(\sum_i \bar{\eta}_{i,a} - 1 \right)^2$$

Outline

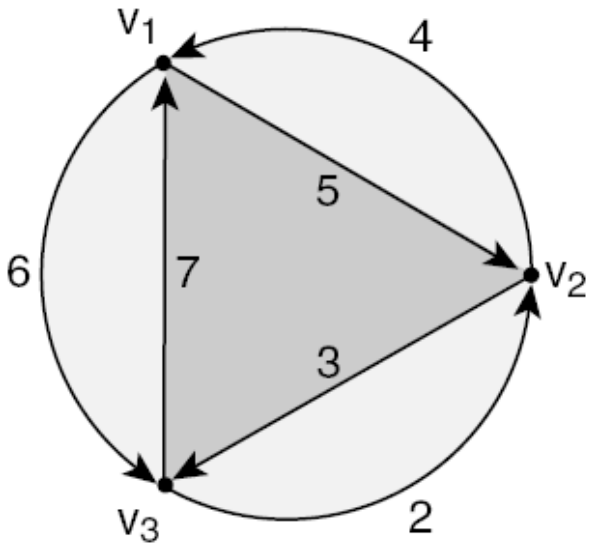
Asymmetric TSP

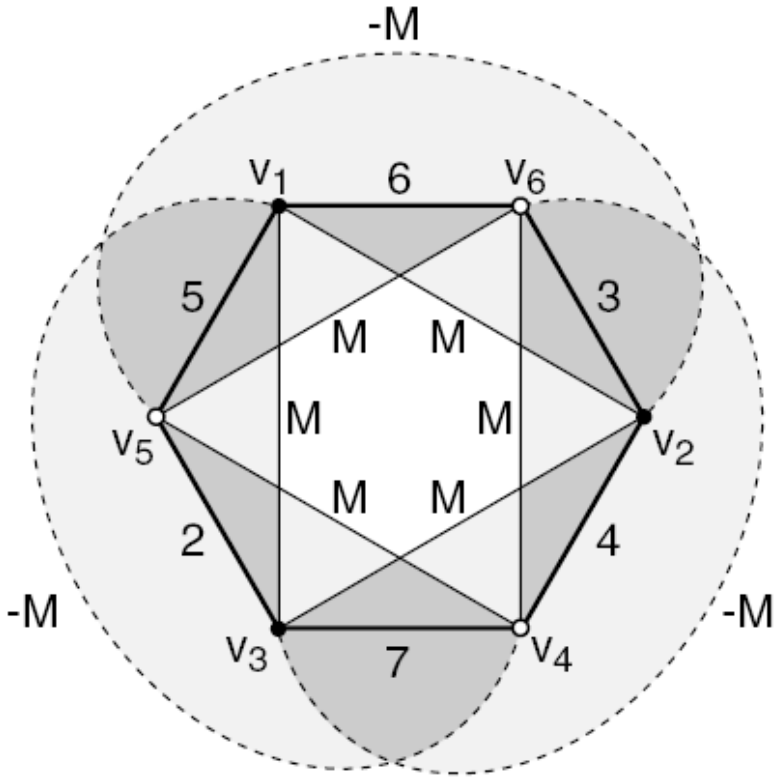
- A TSP instance is called *symmetric* if the distance matrix A is symmetric, i.e. $d_A(i,j) = d_A(j,i), \forall i,j = 1..n$.
- Asymmetric TSP (ATSP): $d_A(i,j) \neq d_A(j,i)$ for some nodes i,j
  - Mapping ATPS in TPS
  - Define a new distance matrix d_S of dimension $2n \times 2n$: each node of index $i \leq n$ is replicated by the node with index $n+i$

$$\begin{cases} d_S(i,n+j) = d_S(n+j,i) = d_A(i,j) & \forall i \neq j, i,j \leq n; \\ d_S(i,i) = 0 & \forall i \leq 2n \\ d_S(i,n+i) = d_S(n+i,i) = -M & \forall i \leq n \\ d_S(i,j) = M & \text{otherwise} \end{cases}$$

  - M must be a large number (e.g. $\sum_{i,j \leq n} d_A(i,j)$)
- Every configuration of the ATSP of length l can be transformed into a configuration of the symmetric TSP of length $l - n * M$
  - replace the edge (i,j) of the ATSP configuration by the edge $(i,n+j)$ followed by the edge $(n+j,j)$
- Any optimum configuration of the symmetric TSP contains all n edges with length $-M$ (why?)
  - After removing the edges with length $-M$, the result is an optimum configuration for ATSP

An example





- $\underbrace{(v_1v_2, v_2v_3, v_3v_1)}_{ATSP} \Rightarrow \underbrace{(v_1v_5, v_5v_2, v_2v_6, v_6v_3, v_3v_4, v_4v_1)}_{STPS}$

Time dependent TPS

- Time dependent TSP (TdTSP): $d(i, j)$ depends on time
- TSP with time window (TSPTW)
- Constraints:
  - the starting node is fixed
  - arrive at node i within a time interval $[t_S(i), t_E(i)]$;
  - wait if arrive too early;
  - consider time service $\tau(i)$
  - velocity is considered

Vehicle Routing Problem

- Multiple traveling salesman problem (MTSP): m salesmen starting from the same node (warehouse) and performing roundtrip such that every node is visited only once.
  - constraints on time window for nodes (customers): MTSPTW
- Vehicle Routing Problem: each salesman is a truck with a given capacity
  - each customer has demands to deliver or to pick up (fixed quantity of goods)
  - the goods are nonsplittable
  - $\sigma(i, j)$ - the node i in the roundtrip of the truck j; if truck j visits n_j nodes, then $\sigma(1, j) = \sigma(n_j, j) \equiv 1$
  - $H(\sigma) = \sum_{j=1}^{m} \sum_{i=1}^{n_j - 1} d(\sigma(i, j), \sigma(i + 1, j))$
- Constraints: on capacity (not mixing some goods), on time window, on nodes (excluded customers, multiple warehouses), on path capacity (bridges), etc..
- Probabilistic VRP: each node has a probability $p(i)$ to have a demand (taxi company problem)

Outline

Branch & Bound for TSP

- The domain $\Theta = \{\sigma | \sigma(1), \ldots, \sigma(n)$ - a permutation of $1..n\}$
- The loss function

$$L(\sigma) = \sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1))$$

- Another form for L:

$$L(\sigma) = \frac{1}{2} \sum_{i=1}^{n} \left(d(\sigma(i-1), \sigma(i)) + d(\sigma(i), \sigma(i+1)) \right),$$

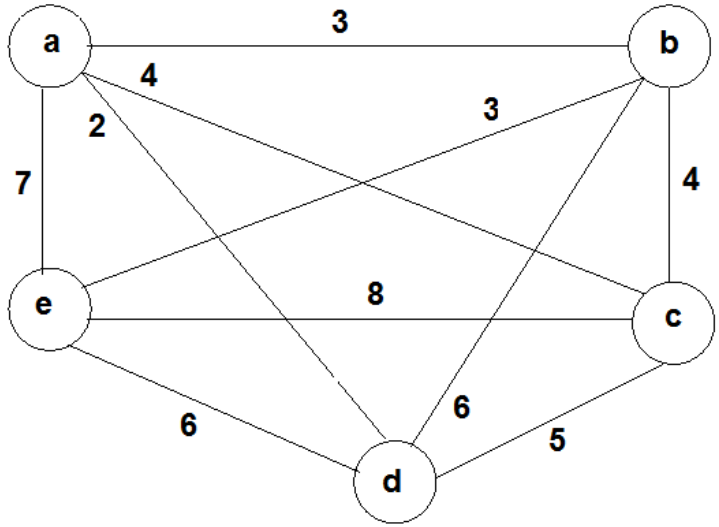
where $\sigma(n+1) = \sigma(1)$ and $\sigma(0) = \sigma(n)$

- A lower bound for $L(\sigma)$:

$$L(\sigma) \geq \frac{1}{2} \sum_{i=1}^{n} \left(\min_j d(\sigma(j), \sigma(i)) + \min_{k \neq j} d(\sigma(i), \sigma(k)) \right), \qquad (3)$$

where $\min_j d(\sigma(j), \sigma(i))$ and $\min_{k \neq j} d(\sigma(i), \sigma(k))$ are the two shortest edges adjacent to $\sigma(i)$

An exemple



Node	Shortest edges	Total length
a	(a, d), (a, b)	2+3=5
b	(b, a), (b, e)	3+3=6
c	(c, a), (c, b)	4+4=8
d	(d, a), (d, c)	2+5=7
e	(e, b), (e, d)	3+6=9

Lower bound for any solution σ: $L(\sigma) \geq \frac{1}{2}(5+6+8+7+9) = 17.5$

Lower bound for a subset of solutions

- Consider a subset S of solutions defined by
  - the set of edges (denoted S^{add}) that are included in each solution from S, and
  - the set of edges (denoted S^{exc}) that are not included in any solution from S
- Example: $S = \{\sigma | (a,b) \in \sigma, (a,e) \in \sigma, (b,c) \notin \sigma\}$
  - $(a,b)(b,d)(d,c)(c,e)(e,a) \in S$, but $(a,b)(b,c)(c,d)(d,e)(e,a) \notin \sigma$
  - S is completely defined by $S^{add} = \{(a,b),(a,e)\}$ and $S^{exc} = \{(b,c)\}$
  - For the set Θ, we have $\Theta^{add} = \Theta^{exc} = \varnothing$
- A lower bound for $L(S) = \{L(\sigma), \sigma \in S\}$ is obtained by altering the choice of the two shortest edges in the formula (3)
- Exemple: consider the previous set S

Node	Shortest edges	Total length
a	(a,b), (a,e)	3+7=10
b	(b,a), (b,e)	3+3=6
c	~~(c,b)~~ (c,a), (c,d)	4+5=9
d	(d,a), (d,c)	2+5=7
e	(e,a), (e,b)	7+3=10

Lower bound for any solution $\sigma \in S$: $L(S) \geq \frac{1}{2}(10+6+9+7+10) = 21$

Inferences on the subset S

- A set $S \subset \Theta$ defined by S^{add} and S^{exc} can be split in two disjoint subsets S_1 and S_2 by the following method:
  - consider an edge e not included in $S^{add} \cup S^{exc}$ (so e is a "free" edge for any $\sigma \in S$)
  - S_1 is defined by $S_1^{add} = S^{add} \cup \{e\}$ and $S_1^{ext} = S^{exc}$,
  - S_2 is defined by $S_2^{add} = S^{add}$ and $S_2^{ext} = S^{exc} \cup \{e\}$.
- Example : for $S = \{\sigma | (a,b) \in \sigma, (a,e) \in \sigma, (b,c) \notin \sigma\}$, the edge (c,d) is a "free" edge;
  - $S_1 = \{\sigma | (a,b) \in \sigma, (a,e) \in \sigma, (c,d) \in \sigma, (b,c) \notin \sigma\}$
  - $S_2 = \{\sigma | (a,b) \in \sigma, (a,e) \in \sigma, (b,c) \notin \sigma, (c,d) \notin \sigma\}$

Inferences on the subset S

- By adding or excluding an edge, the following two inference rules must be always satisfied:

  R1. An edge $e = (i, j)$ can not be added if

  a) S^{add} contains already two adjacent edges for the node i or the node j, or

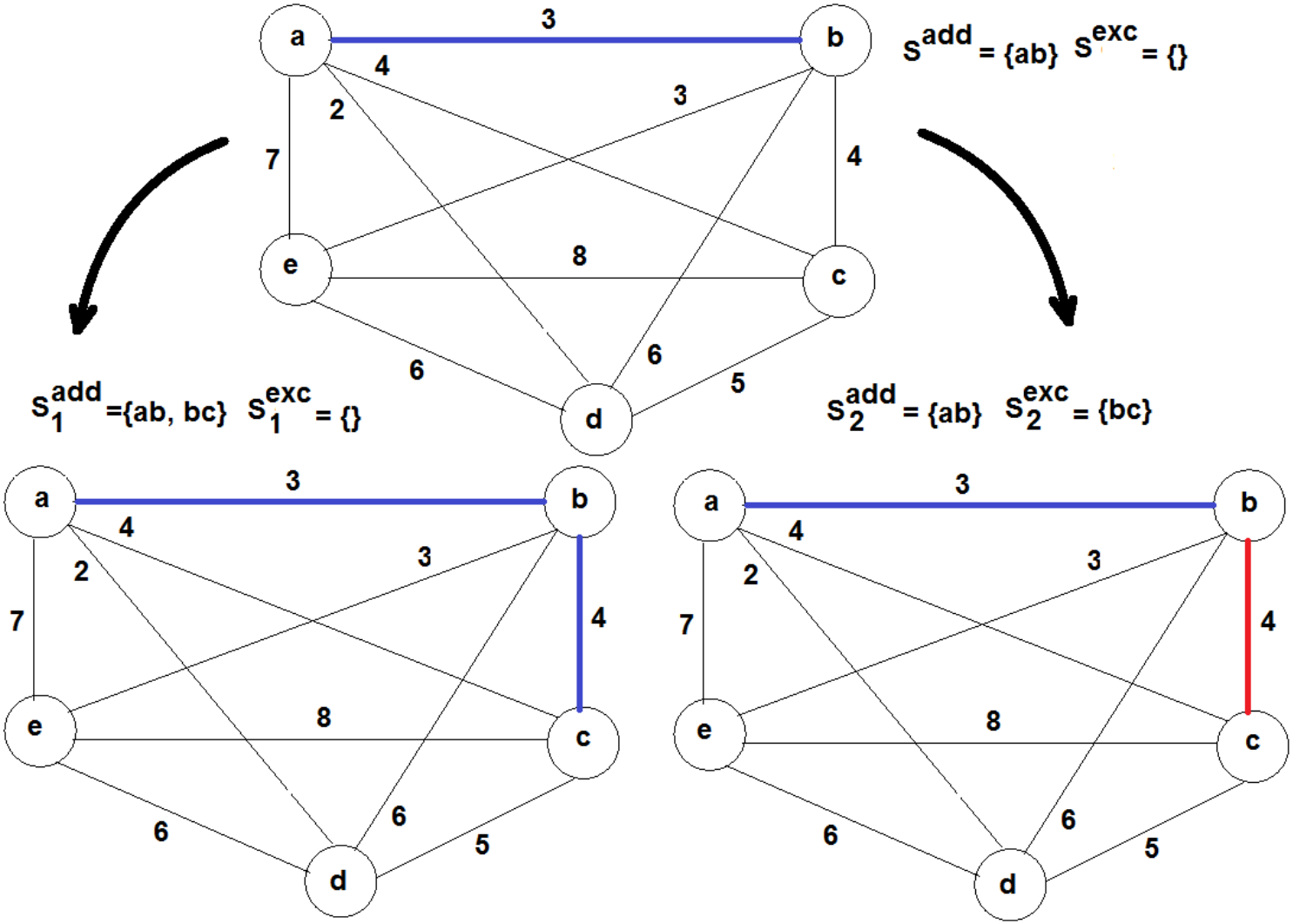
  b) we may construct a cycle with the edges from S^{add} and the new edge;

  R2. An edge $e = (i, j)$ can not be excluded if would be impossible for the node i or the node j to have two adjacent edges;

- An edge which can not be added must be excluded (included in S^{exc})

- An edge which can not be excluded must be added (included in S^{add}).

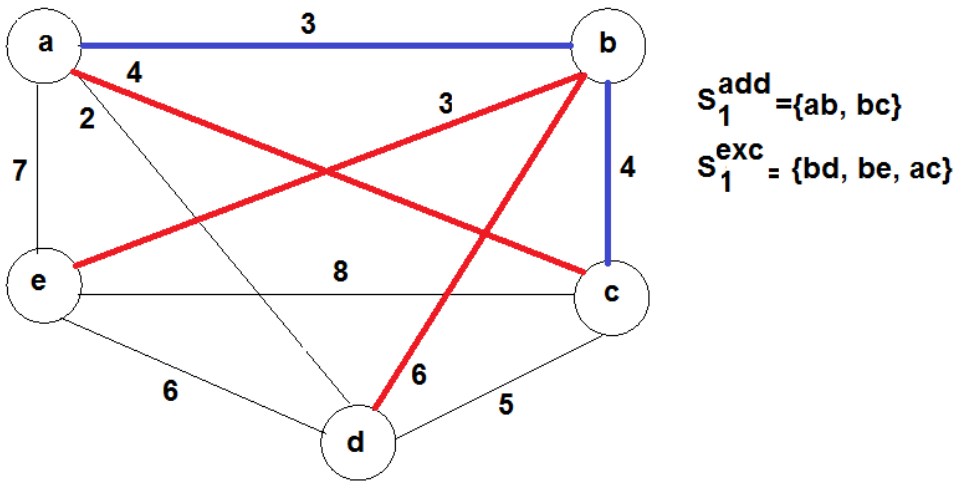
Inference example



The set S defined by $S^{add} = \{ab\}$ and $S^{exc} = \{\}$ is split by the edge bc in two subsets, S_1 and S_2.

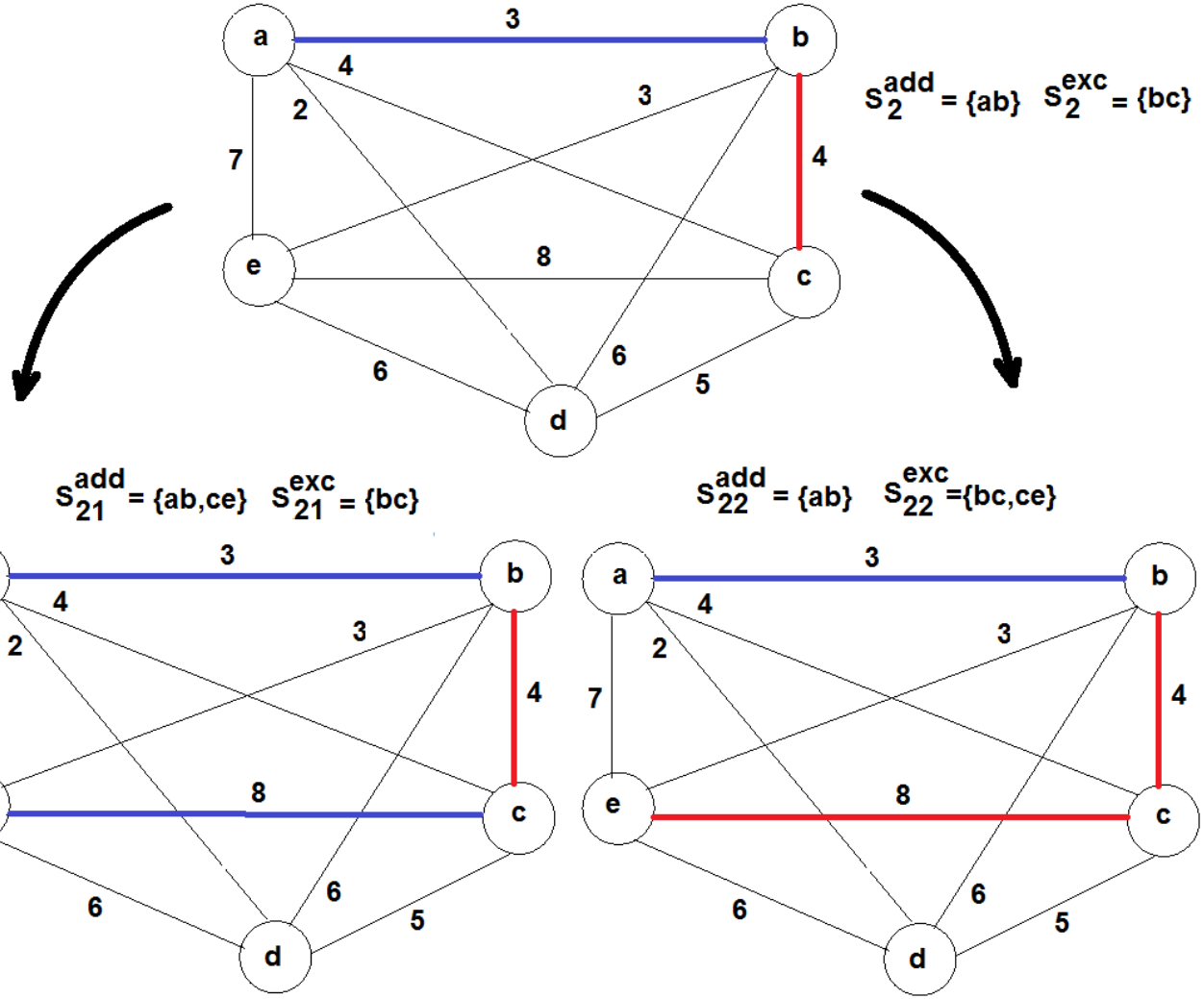
Inference example (cont.)

- The subset S_1 is defined by $S_1^{add} = \{ab, bc\}$ and $S_1^{exc} = \{\}$;
  - In the subset S_1, the node b has already two adjacents edges, so all other adjacent edges of b (i.e., bd, be) must be excluded; S_1^{exc} becomes $\{be, bd\}$
  - In the subset S_1, the edge ac may form a cycle with ab and bc, so it must be excluded; S_1^{exc} becomes $\{be, bd, ac\}$;



Inference example (cont.)

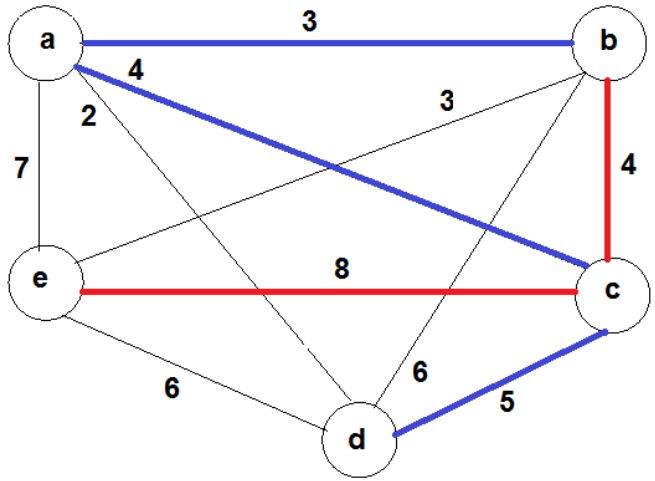
- The subset S_2 is defined by $S_2^{add} = \{ab\}$ and $S^{exc} = \{bc\}$; let's split the set S_2 by the edge ce.



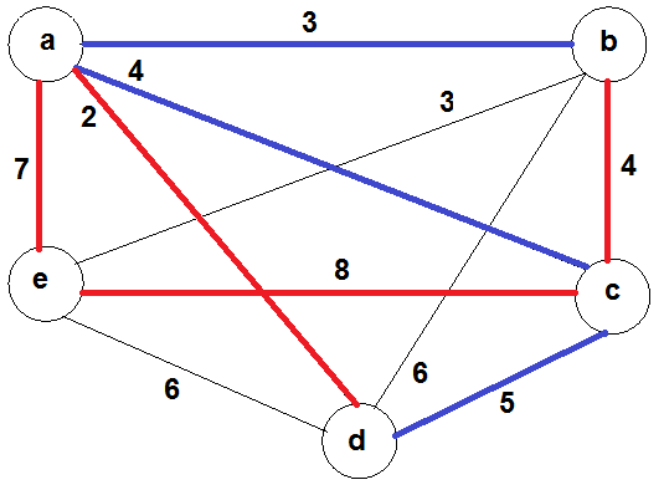
- The subset S_{21} is defined by $S_{21}^{add} = \{ab, ce\}$ and $S_{21}^{exc} = \{bc\}$;
- The subset S_{22} is defined by $S_{22}^{add} = \{ab\}$ and $S_{22}^{exc} = \{bc, ce\}$;

Inference example (cont.)

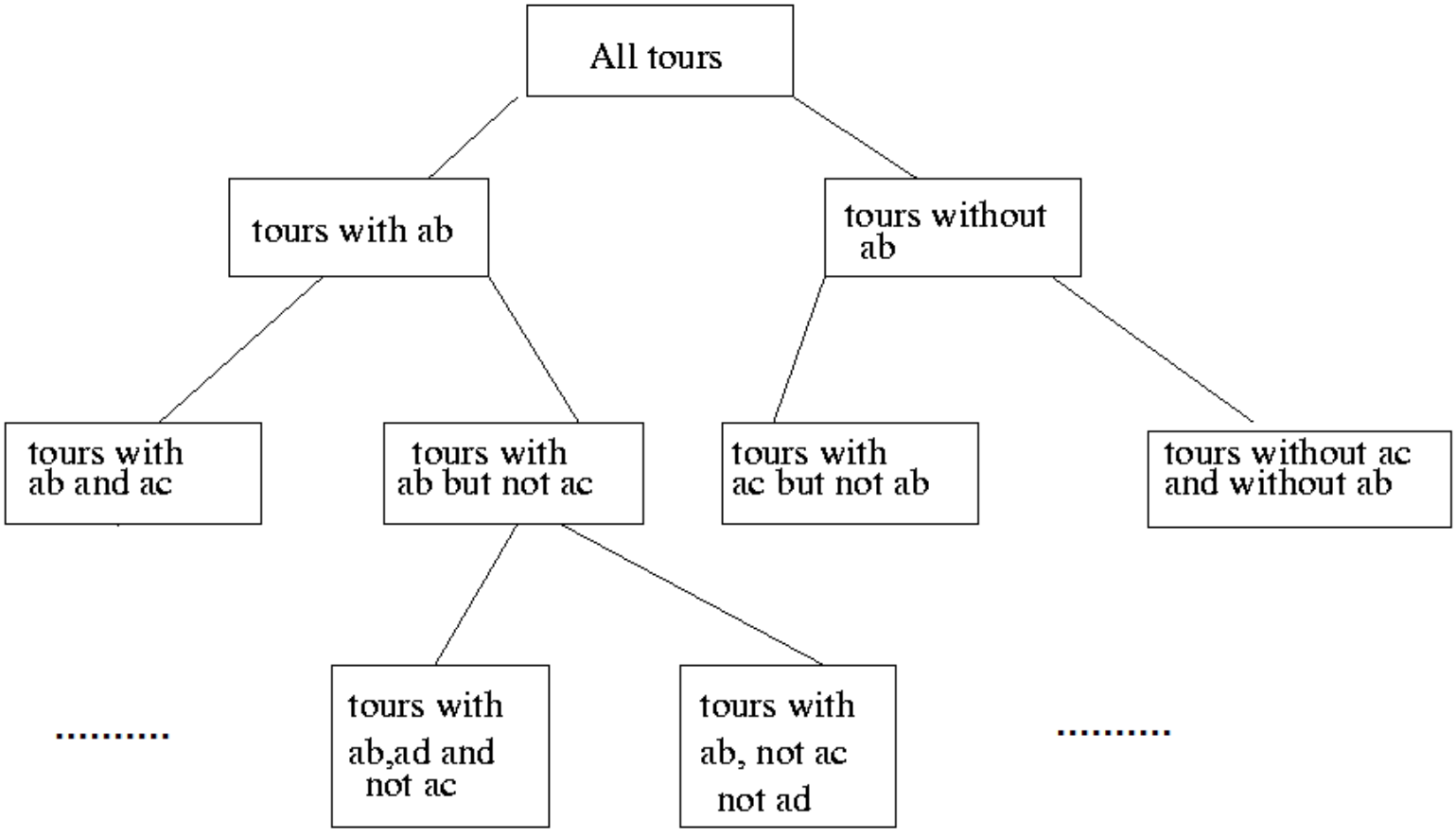
- In the subset S_{22}, the node c has two adjacent edges which are excluded, i.e. cb and ce; because the total number of adjacent edges of each node is 4, the remained two adjacent edges of c can't never be excluded and must be added to S_{22}^{add}, that becomes $\{ab, ca, cd\}$.



- Now the set S_{22}^{add} contains two adjacent edges for the node a (ab and ac) so all other adjacent edges of this node must be excluded; $S_{22}^{exc} = \{bc, ce, ae, ad\}$



A solution tree for TSP



The edges are considered in lexicographic order

Branch & Bound for TSP

Step 0:

- Define tree root by $S = \Theta$ ($S^{add} = S^{exc} = \{\}$); Add this node to the list of open nodes L_P
- Calculate the lower bound of S (denoted $LB(S)$) using formula (3);
- Calculate the upper bound of TSP instance (denoted UB) as
  (a) a very big number M, or
  (b) the length of a randomly chosen tour, or
  (c) the half sum of the longest two adjacent edges for each node;

Step 1: WHILE $L_P \neq \{\}$

Select Take the node S from L_P with the lowest value of $LB(S)$;

Branch Split the tree node in two child nodes (S_1 and S_2) using the first "free" edge in lexicographic order; append these nodes to L_P

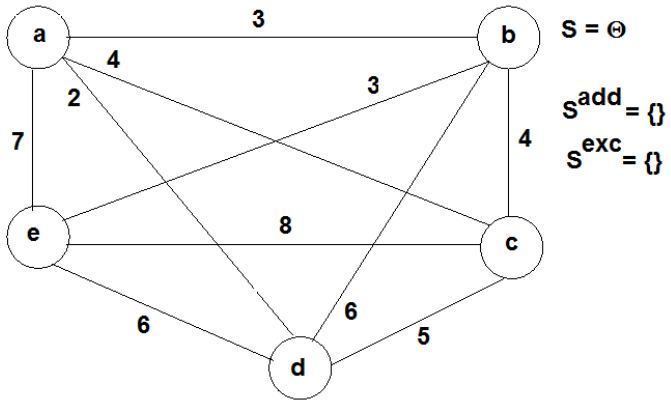
Bound Apply the inference rules to update the sets S^{add} and S^{exc} for each child and calculate $LB(S_1)$ and $LB(S_2)$;

Update/Pruning If S_i, $i = 1..2$ is a leaf node (contains a single solution) and if $LB(S_i) < UB$, then update UB and cut all nodes S from L_P having $LB(S) \geq UB$

Step 2 : Return the optimal solution (the last node which updated UB)

A running example

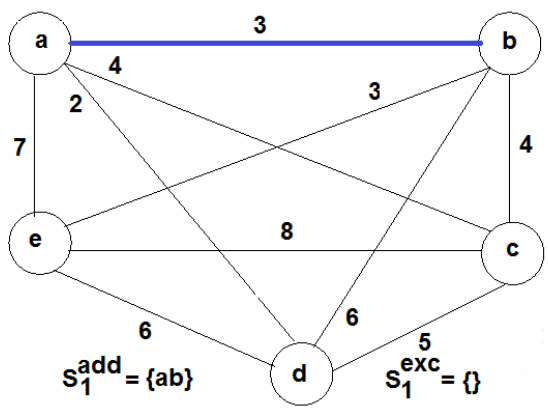
Step 0 Θ is the set of permutations for $\{a, b, c, d, e\}$;
$S = \Theta$, $S^{add} = S^{exc} = \{\}$, $L_P = \{S\}$
$LB(S) = 17.5$, $UB = 30.5$ (the half of the longest two adjacent edges for each node)

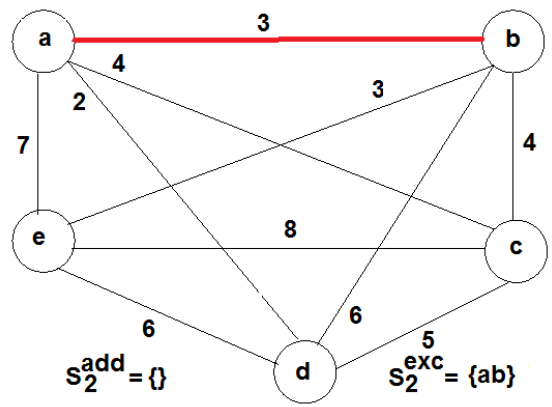


Select Select S;

Branch Split S in two nodes along the edge ab: $S_1^{add} = \{ab\}$, $S_1^{ext} = \{\}$, $S_2^{add} = \{\}$, $S_2^{exc} = \{ab\}$; $L_P = \{S_1, S_2\}$

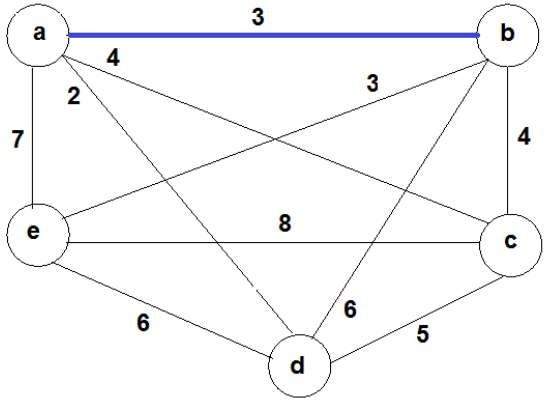
Bound $LB(S_1) = 17.5$, $LB(S_2) = 18.5$





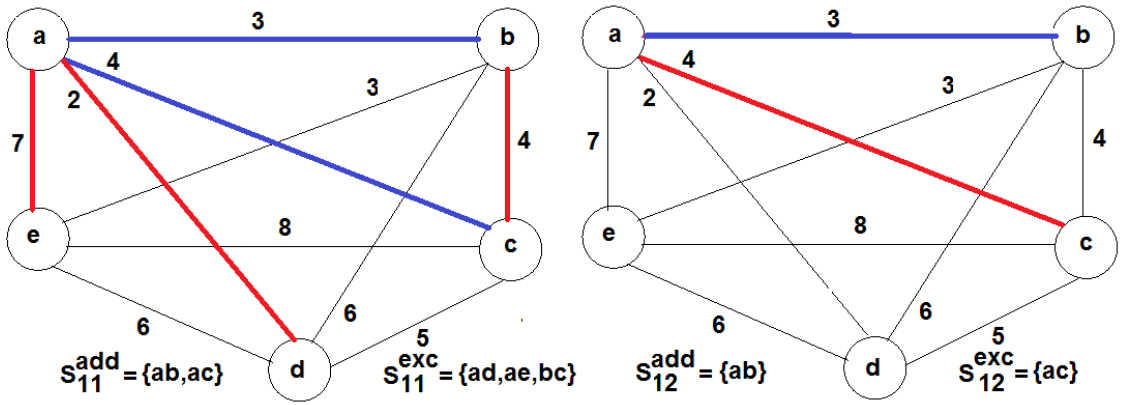
A running example (cont.)

Select S_1 has the minimum $LB(\cdot)$;



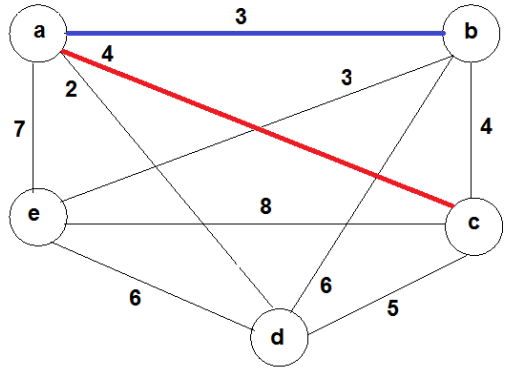
Branch Split S_1 in two nodes along the edge ac: $S_{11}^{add} = \{ab, ac\}$, $S_{11}^{ext} = \{\}$, $S_{12}^{add} = \{ab\}$, $S_{12}^{exc} = \{ac\}$; $L_P = \{S_2, S_{11}, S_{12}\}$

Bound Inference rules: for S_{11}, bc forms a cycle with ab and ac, and the node a has already two adjacent edges in S_{11}^{add}, so $S_{11}^{exc} = \{ad, ae, bc\}$;
$LB(S_{11}) = 21$, $LB(S_{12}) = 18$



A running example (cont.)

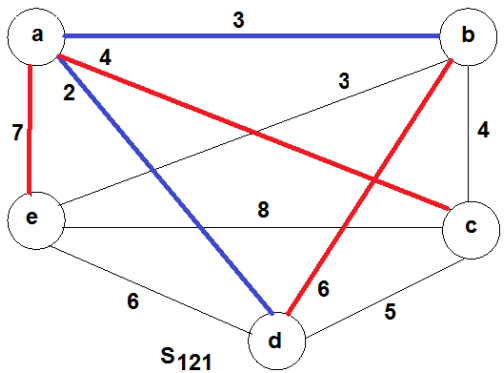
Select S_{12} has the minimum $LB(\cdot)$;

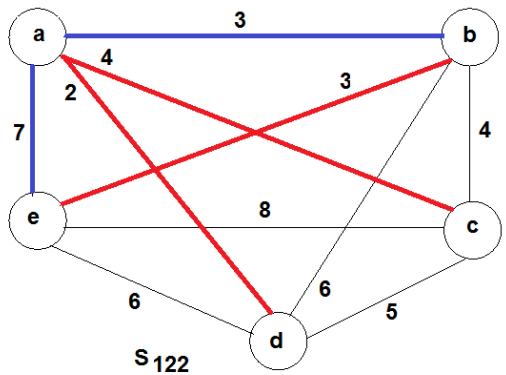


Branch Split S_{12} in two nodes along the edge ad: $S_{121}^{add} = \{ab, ad\}$, $S_{122}^{ext} = \{ac\}$, $S_{122}^{add} = \{ab\}$, $S_{12}^{exc} = \{ac, ad\}$; $L_P = \{S_2, S_{11}, S_{121}, S_{122}\}$

Bound Inference rules:

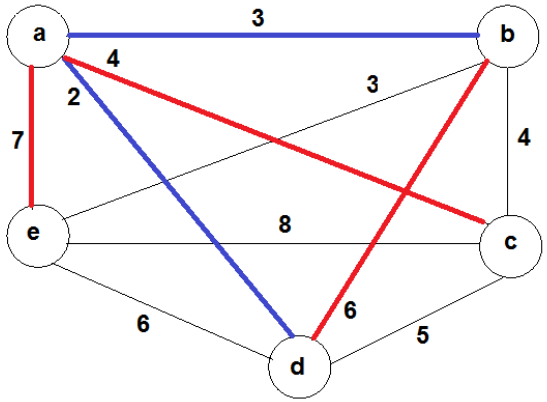
- for S_{121}, bd forms a cycle with ab and ad, and the node a has already two adjacent edges in S_{121}^{add}, so $S_{121}^{exc} = \{ac, ae, bd\}$;
- for S_{122}, the node a has already two excluded adjacent edges (ac and ad), so $S_{122}^{add} = \{ab, ae\}$; be forms a cycle with ab and ae, so $S_{122}^{exc} = \{ac, ad, be\}$
- $LB(S_{121}) = 18$, $LB(S_{122}) = 25$





A running example (cont.)

Select S_{121} has the minimum $LB(\cdot)$;

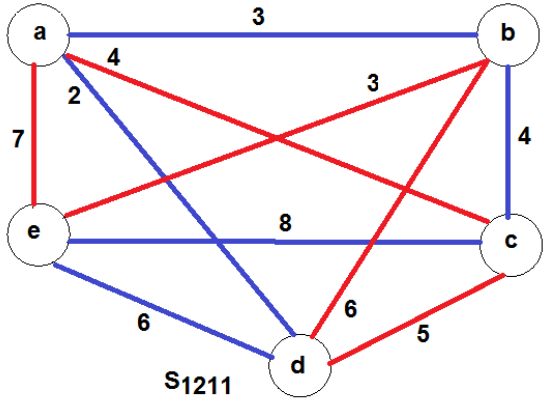


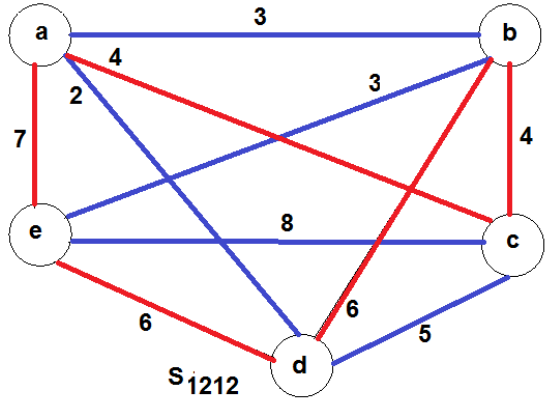
Branch Split S_{121} in two nodes along the edge bc: $S^{add}_{1211} = \{ab, ad, bc\}$, $S^{ext}_{1211} = \{ac, ae, bd\}$, $S^{add}_{1212} = \{ab, ad\}$, $S^{exc}_{12} = \{ac, ae, bc, bd\}$;
$L_P = \{S_2, S_{11}, S_{1211}, S_{1212}, S_{122}\}$

Bound Inference rules:

- for S_{1211}, cd forms a cycle with ab, bc and ad, and the node b has already two adjacent edges in S^{add}_{1211}, so $S^{exc}_{1211} = \{ac, ae, bd, cd, be\}$;
- for S_{1212}, the nodes b and c has already two excluded adjacent edges (bc and bd, respectively cb and ca), so $S^{add}_{1212} = \{ab, ad, be, ce, cd\}$;

A running example (cont.)



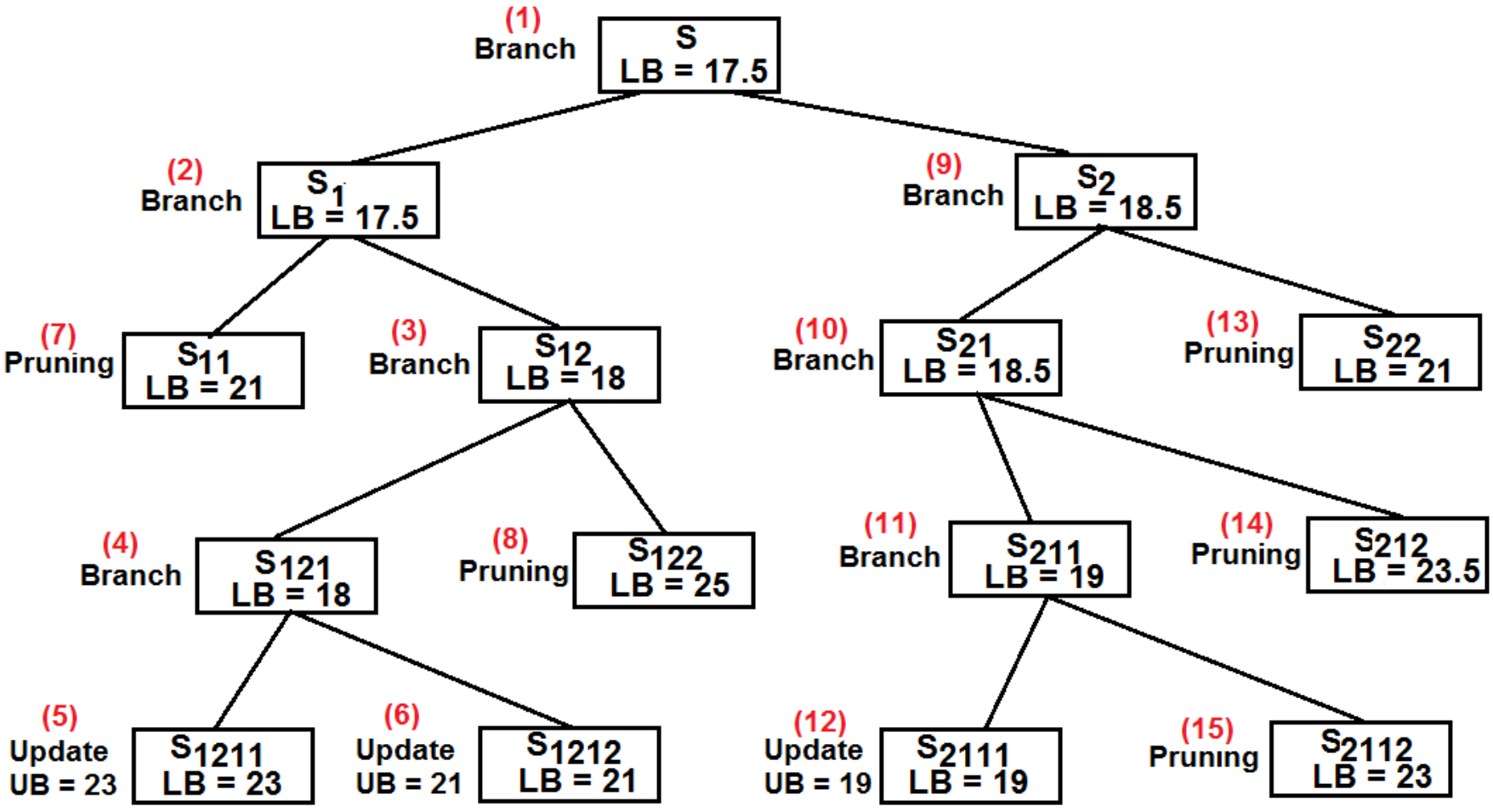


Update/Pruning

- The only solution satisfying S_{1211} is ab, bc, ce, ed, da, so S_{1211} is a leaf with $LB(S_{1211}) = 23$
- $LB(S_{1211}) = 23 < 30.5 = UB$, so $UB = 23$
- The only solution satisfying S_{1212} is $[ab, be, ec, cd, da]$, so S_{1212} is a leaf with $LB(S_{1212}) = 21$
- $LB(S_{1212}) = 21 < 23 = UB$, so $UB = 21$
- The list L_P contains actually S_2 with $LB(S_2) = 18.5$, S_{11} with $LB(S_{11}) = 21$, S_{122} with $LB(S_{122}) = 25$, S_{1211} with $LB(S_{1211}) = 23$ and S_{1212} with $LB(S_{1212}) = 21$. As $UB = 21$, the following nodes S_{11}, S_{122}, S_{1211} and S_{1212} will be cut, so $L_P = \{S_2\}$.

The algorithm continue by developing the sub-tree with root S_2 until no more nodes are included in L_P.

The Complete Branch & Bound Tree



The optimal solution is the tour [*ac*, *cb*, *be*, *ed*, *da*] (defined by the set S_{2111}) with a total length of 19.