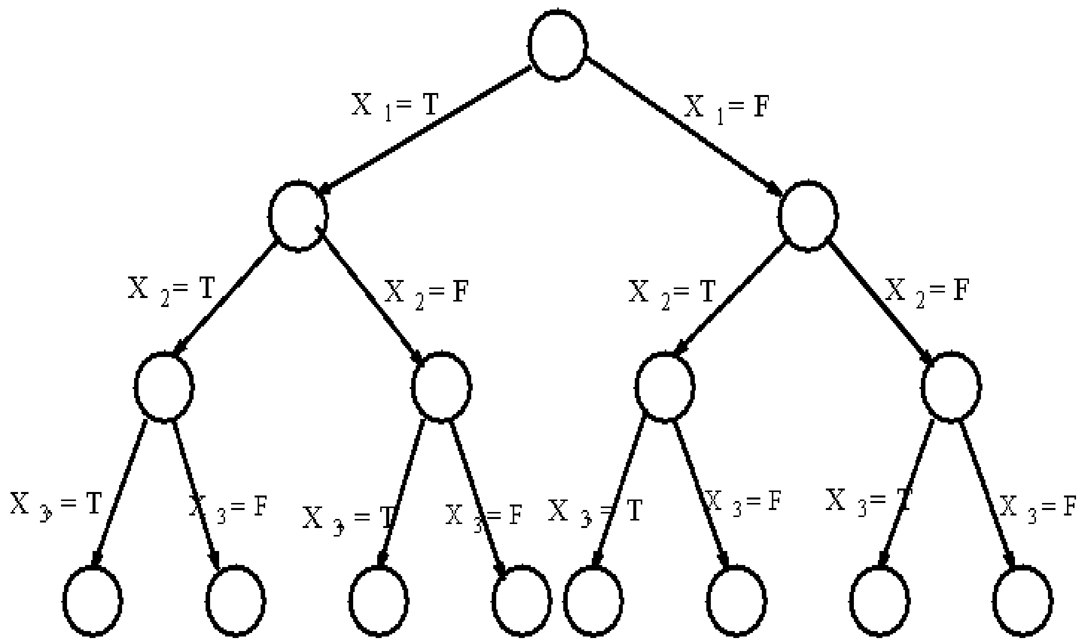
Chapter 5

Tree Searching Strategies

The Satisfiability Problem

x_1	x_2	x_3
F	F	F
F	F	T
F	T	F
F	T	T
T	F	F
T	F	T
T	T	F
T	T	T



Tree representation of 8 assignments.

If there are n variables $x_1, x_2, \ldots, x_n$, then there are 2^n possible assignments.

- An instance:

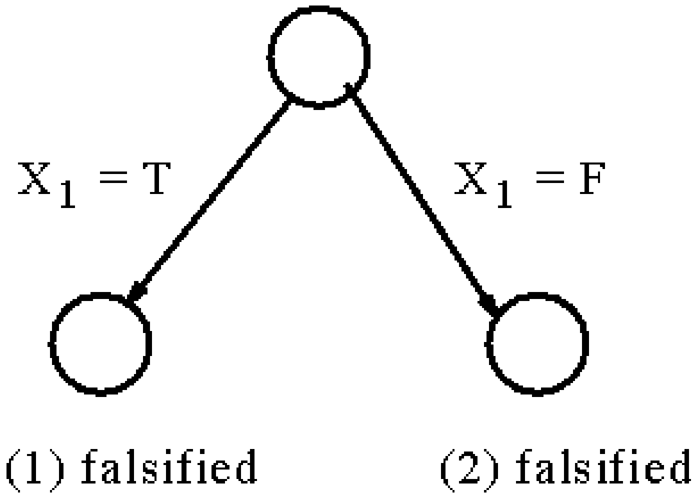
  $-x_1$..............(1)

  x_1..............(2)

  $x_2 \vee x_5$........(3)

  x_3..............(4)

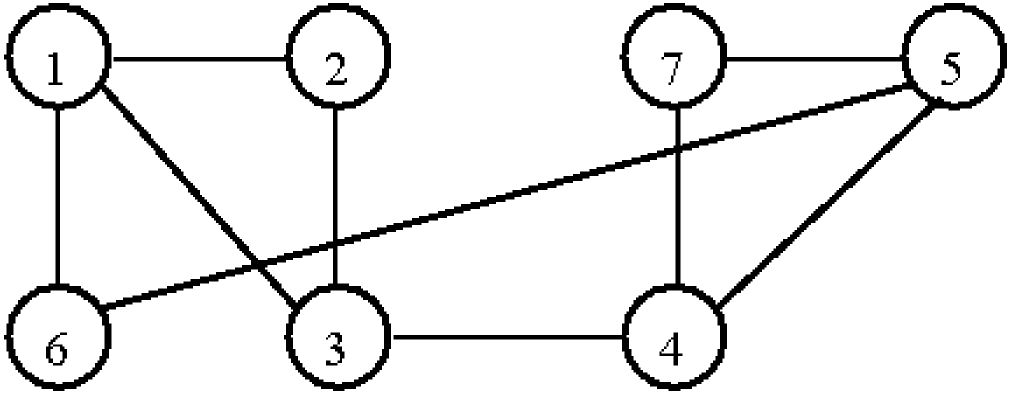
  $-x_2$..............(5)



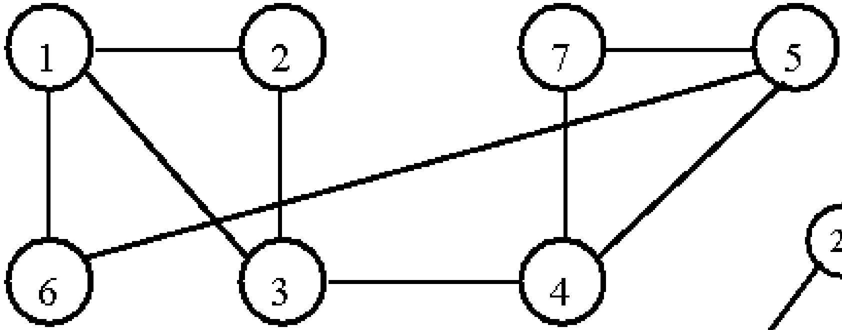
A partial tree to determine the satisfiability problem.

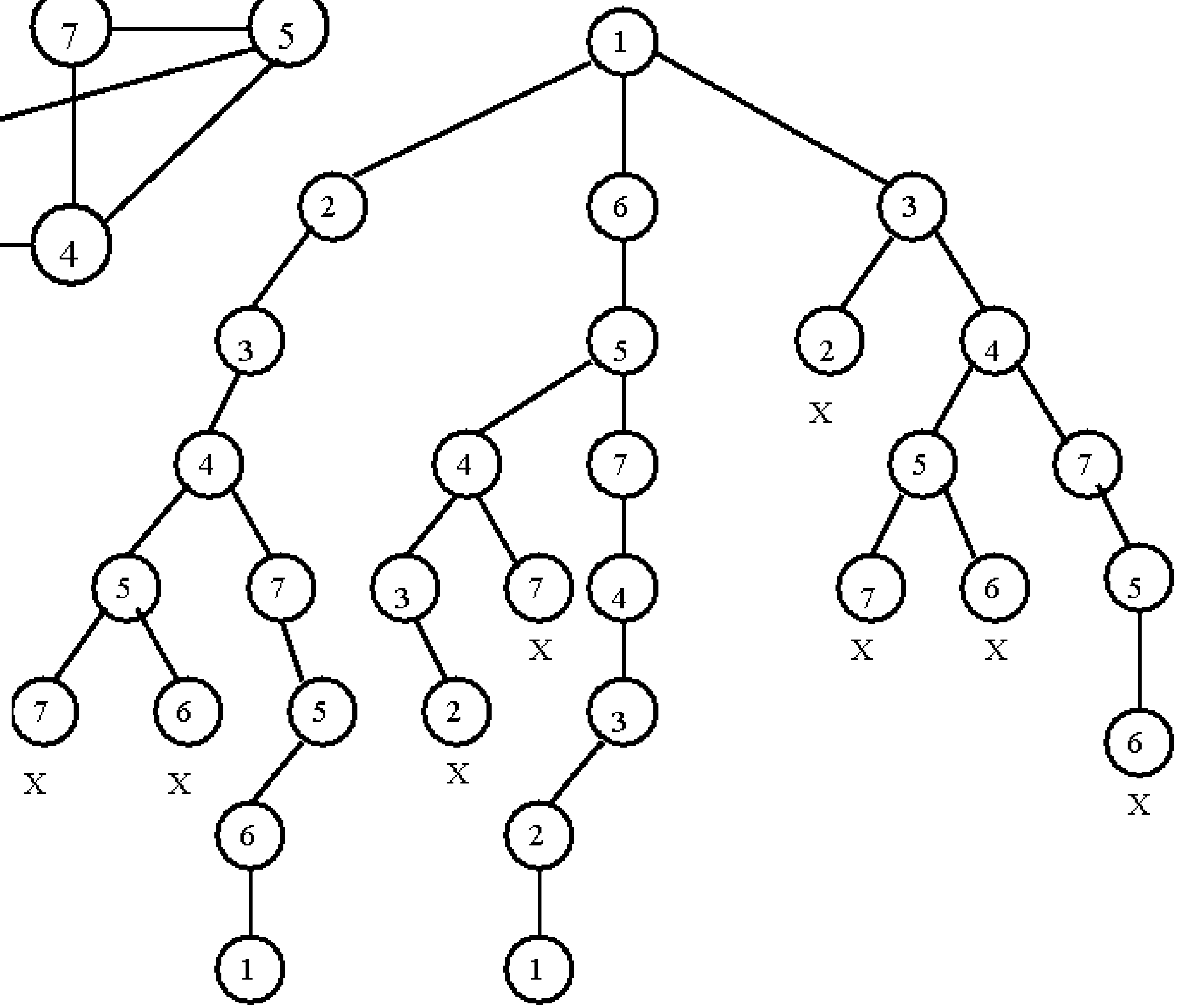
- We may not need to examine all possible assignments.

Hamiltonian Cycle Problem



A graph containing a Hamiltonian cycle.

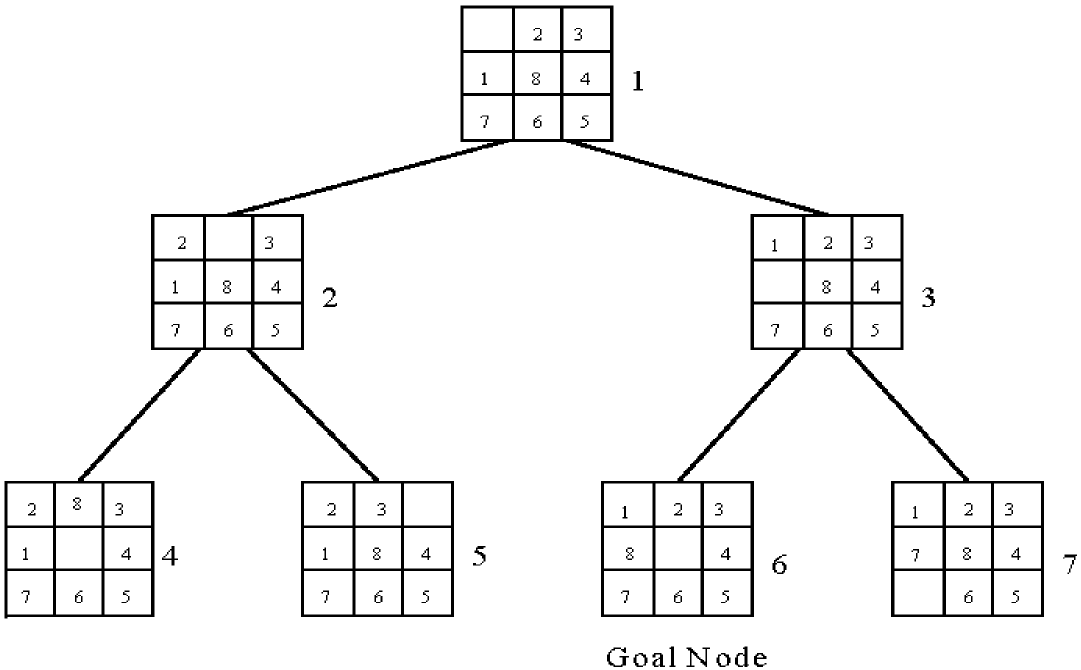




The tree representation of whether there exists a Hamiltonian cycle.

Breadth-First Search (BFS)

- 8-puzzle problem



- The breadth-first search uses a queue to hold all expanded nodes.

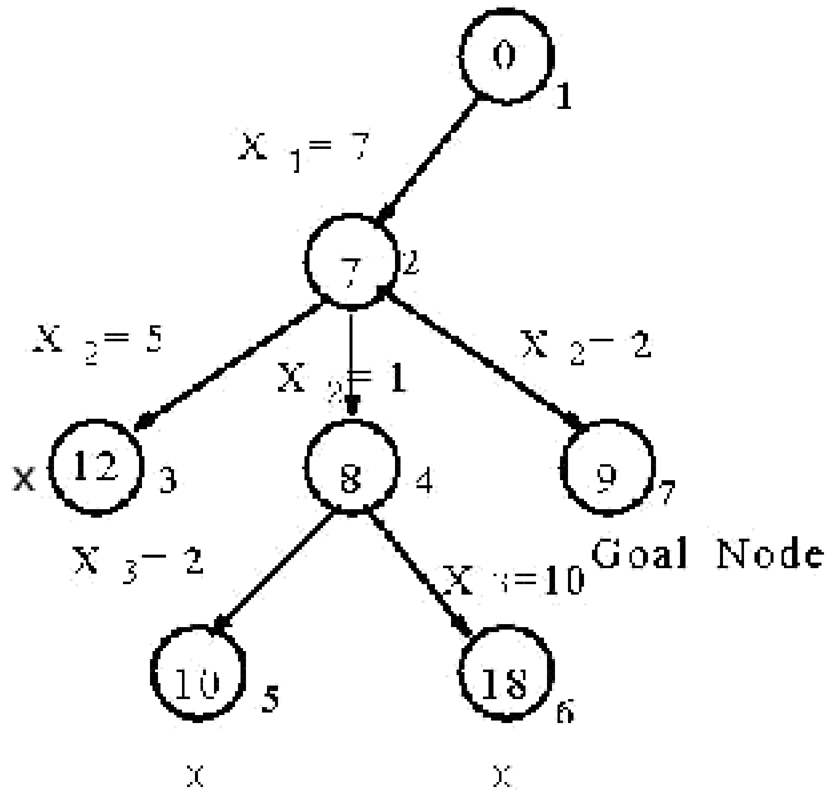
Depth-First Search (DFS)

- e.g. sum of subset problem

  S={7, 5, 1, 2, 10}

  $\exists\, S' \subseteq S \ni$ sum of S' = 9?

- A stack can be used to guide the depth-first search.



A sum of subset problem solved by depth-first search.

Hill Climbing

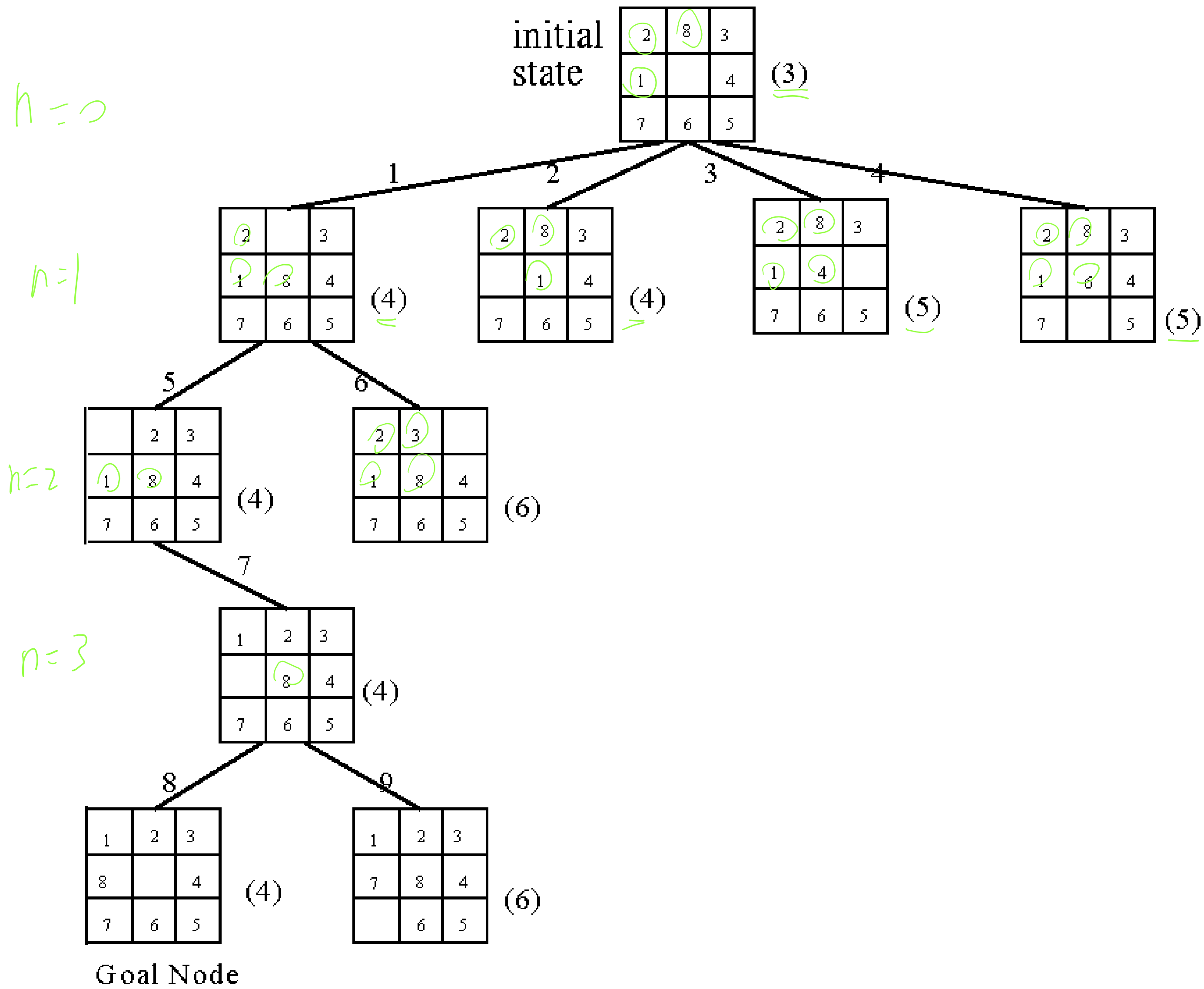
- A variant of depth-first search

  The method selects the locally optimal node to expand.
- e.g. 8-puzzle problem

  evaluation function $f(n) = d(n) + w(n)$,

  where $d(n)$ is the depth of node n

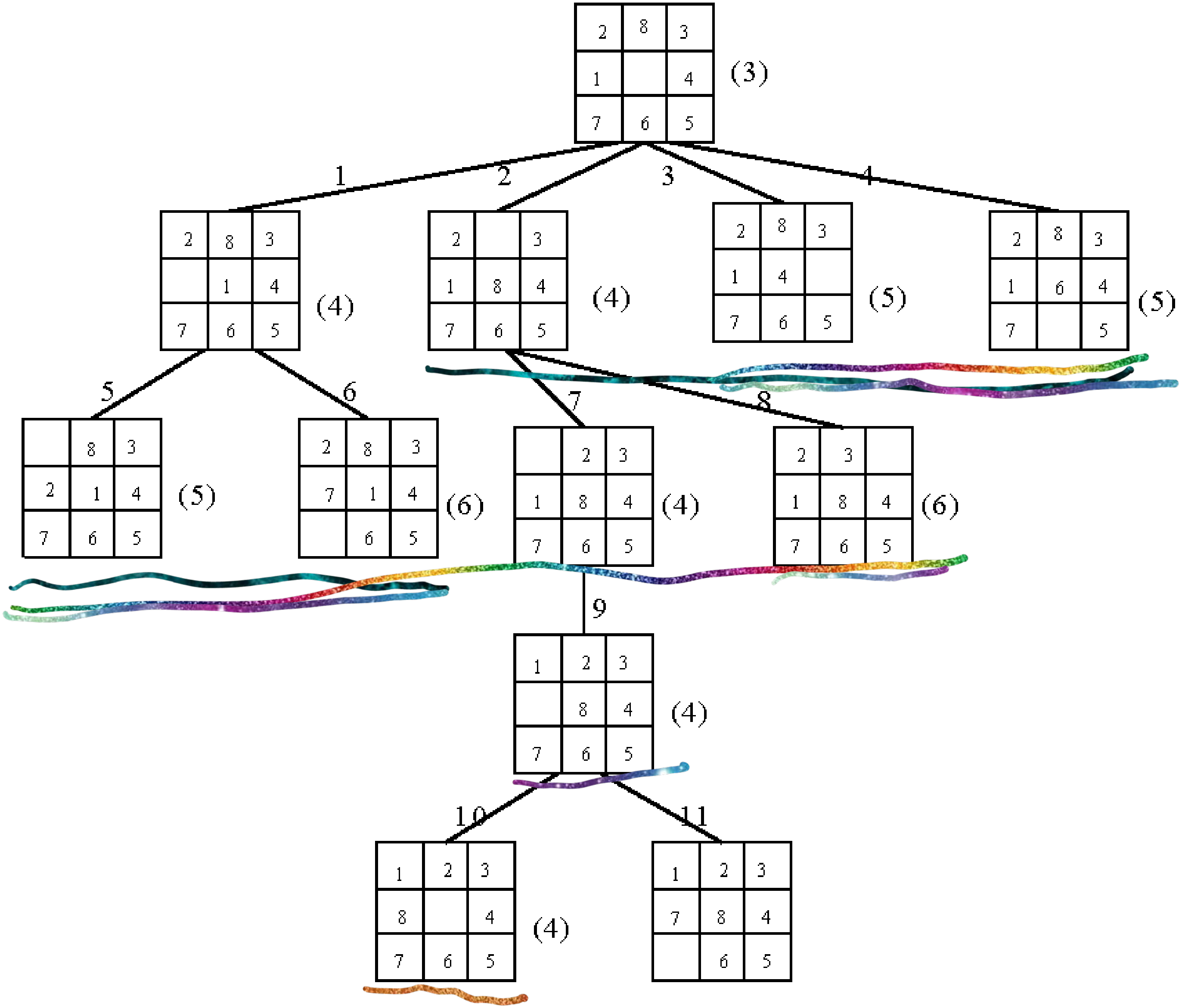
  $w(n)$ is # of misplaced tiles in node n.



An 8-puzzle problem solved by a hill climbing method.

Best-First Search Strategy

- Combine depth-first search and breadth-first search.
- Select the node with the best estimated cost among all nodes.
- This method has a global view.



An 8-puzzle problem solved by a best-first search scheme.

Best-First Search Scheme

Step1: Form a one-element list consisting of the root node.

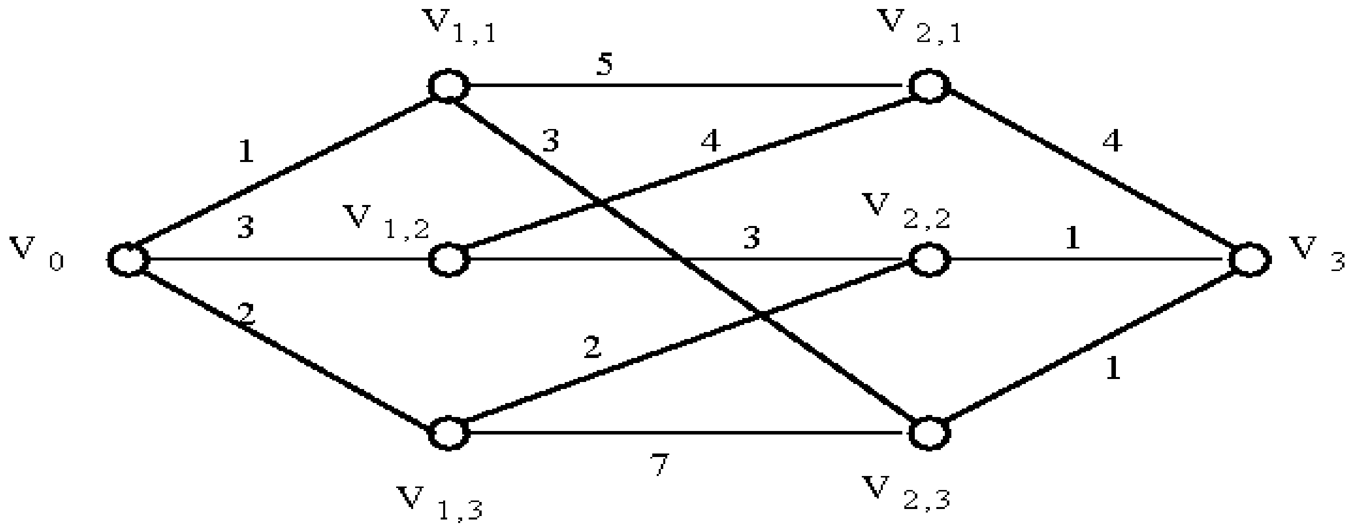
Step2: Remove the first element from the list. Expand the first element. If one of the descendants of the first element is a goal node, then stop; otherwise, add the descendants into the list.

Step3: Sort the entire list by the values of some estimation function.

Step4: If the list is empty, then failure. Otherwise, go to Step 2.

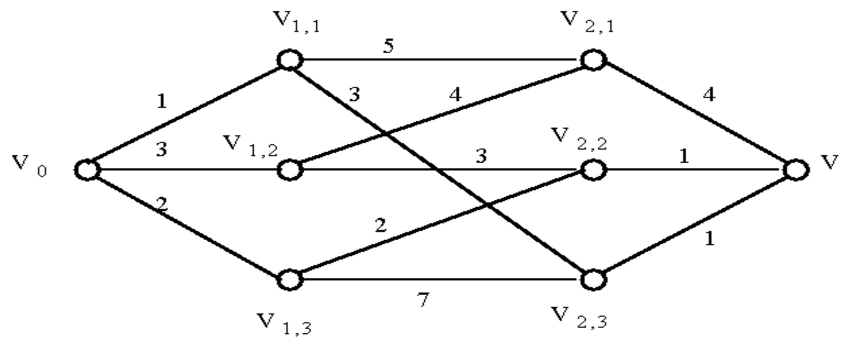
Branch-and-Bound Strategy

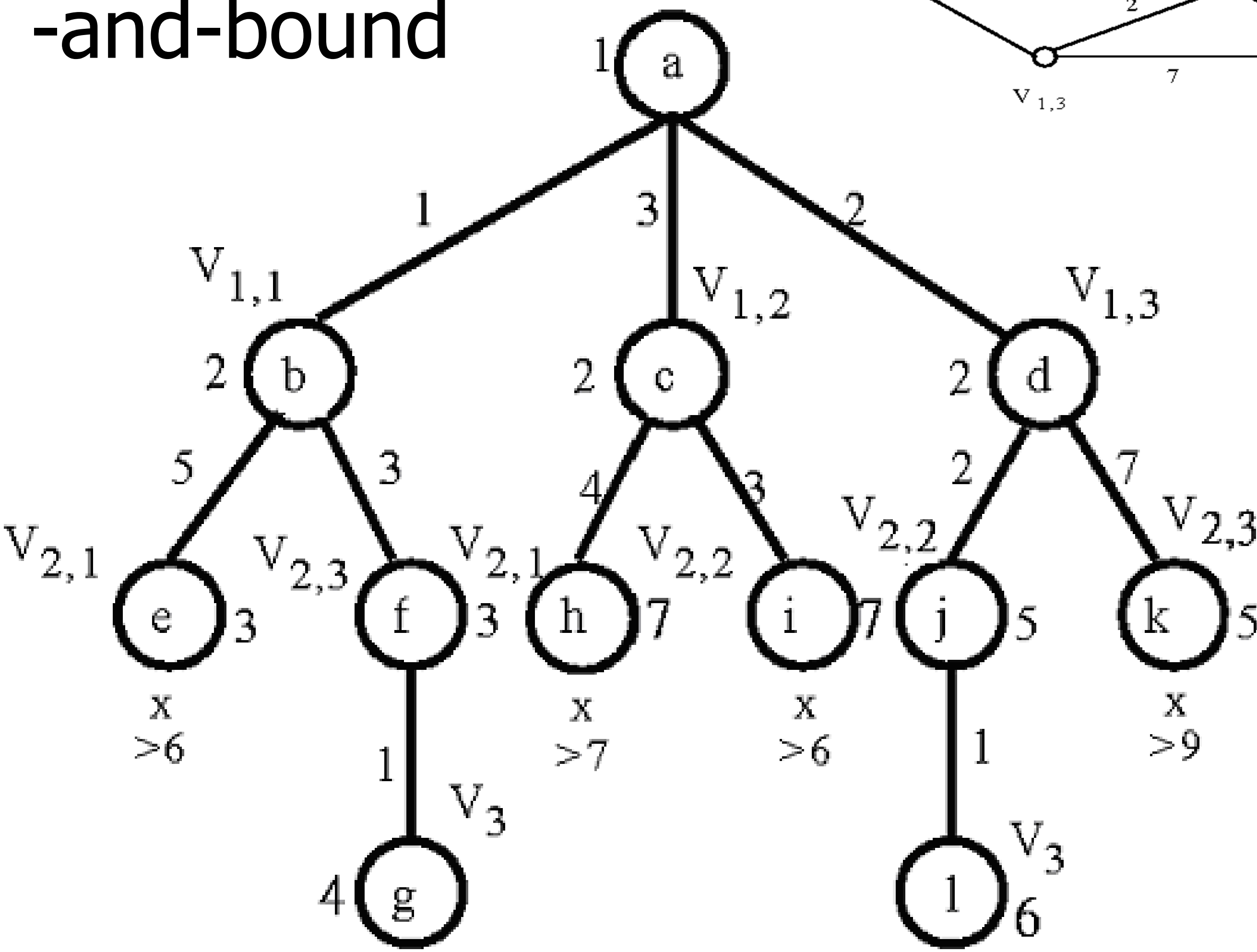
- This strategy can be used to efficiently solve optimization problems.
- e.g.



A multi-stage graph searching problem.

- Solved by branch -and-bound





Personnel Assignment Problem

- A linearly ordered set of persons $P=\{P_1, P_2, \ldots, P_n\}$ where $P_1<P_2<\ldots<P_n$
- A partially ordered set of jobs $J=\{J_1, J_2, \ldots, J_n\}$
- Suppose that P_i and P_j are assigned to jobs $f(P_i)$ and $f(P_j)$ respectively. If $f(P_i) \leq f(P_j)$, then $P_i \leq P_j$. Cost C_{ij} is the cost of assigning P_i to J_j. We want to find a feasible assignment with the minimum cost. i.e.

  $X_{ij} = 1$ if P_i is assigned to J_j

  $X_{ij} = 0$ otherwise.
- Minimize $\sum_{i,j} C_{ij}X_{ij}$

- e.g. A partial ordering of jobs

$$\begin{array}{ccc} J_1 & & J_2 \\ \downarrow & \boxed{\searrow} & \downarrow \\ J_3 & & J_4 \end{array}$$

- After topological sorting, one of the following topologically sorted sequences will be generated:

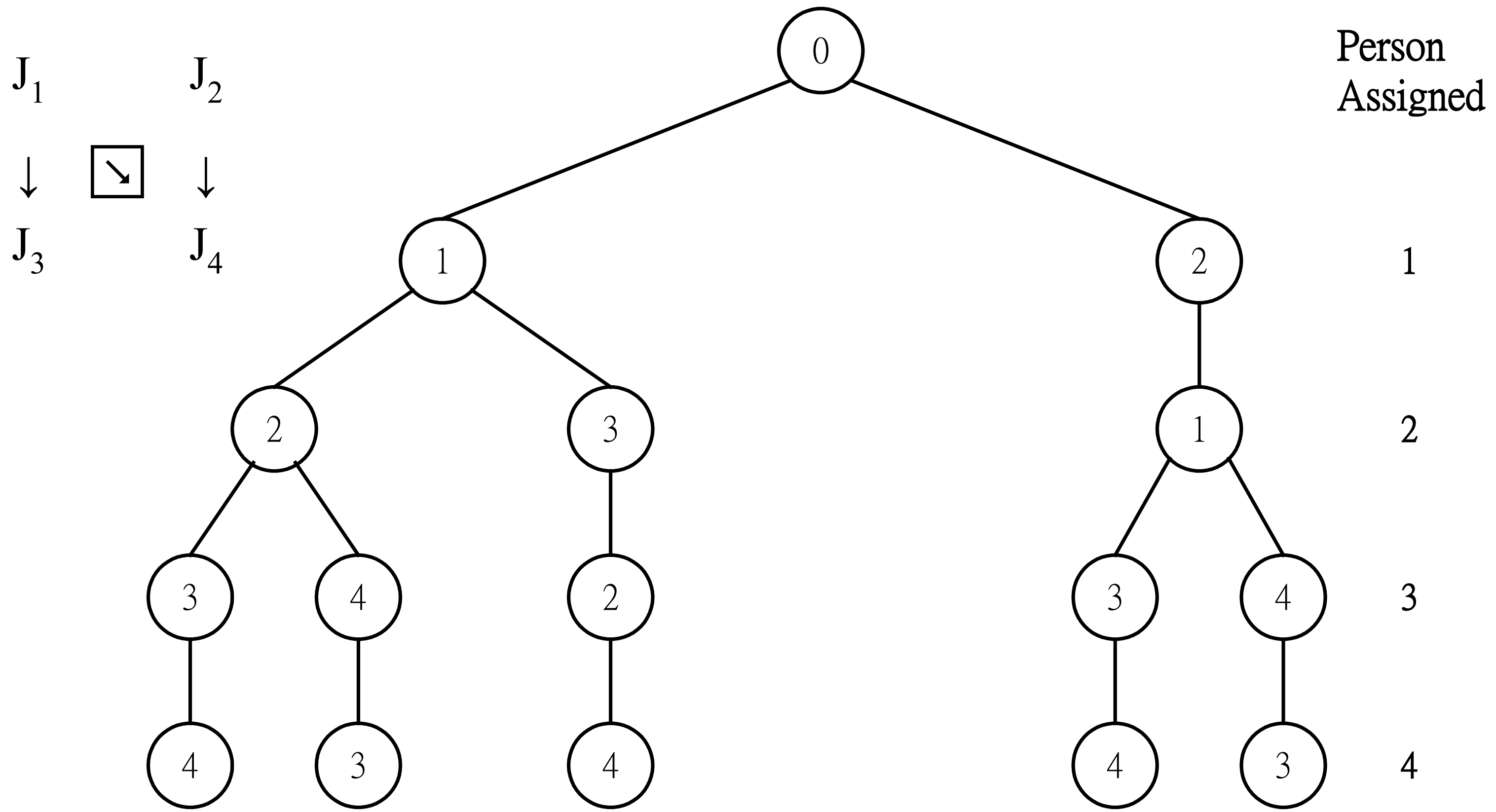
J_1,	J_2,	J_3,	J_4
J_1,	J_2,	J_4,	J_3
J_1,	J_3,	J_2,	J_4
J_2,	J_1,	J_3,	J_4
J_2,	J_1,	J_4	J_3

- One of feasible assignments:

  $P_1 \rightarrow J_1$, $P_2 \rightarrow J_2$, $P_3 \rightarrow J_3$, $P_4 \rightarrow J_4$

A Solution Tree

- All possible solutions can be represented by a solution tree.

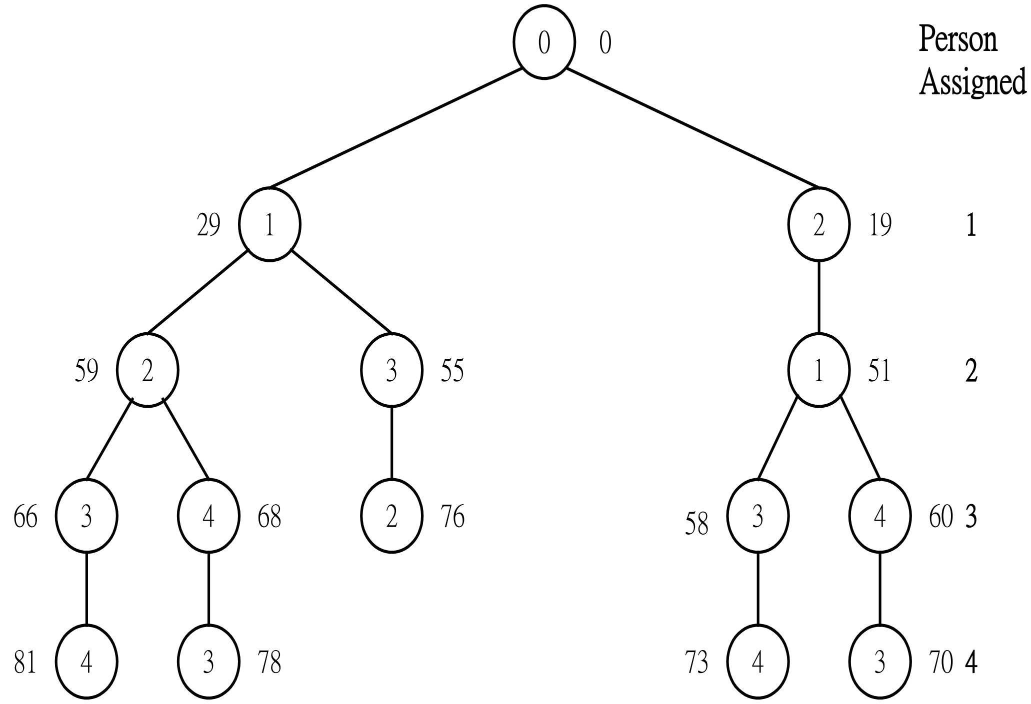


Cost Matrix

- Cost matrix

Jobs / Persons	1	2	3	4
1	29	19	17	12
2	32	30	26	28
3	3	21	7	9
4	18	13	10	15

- Apply the best-first search scheme:



Only one node is pruned away.

Reduced Cost Matrix

- Cost matrix

Jobs / Persons	1	2	3	4
1	29	19	17	12
2	32	30	26	28
3	3	21	7	9
4	18	13	10	15

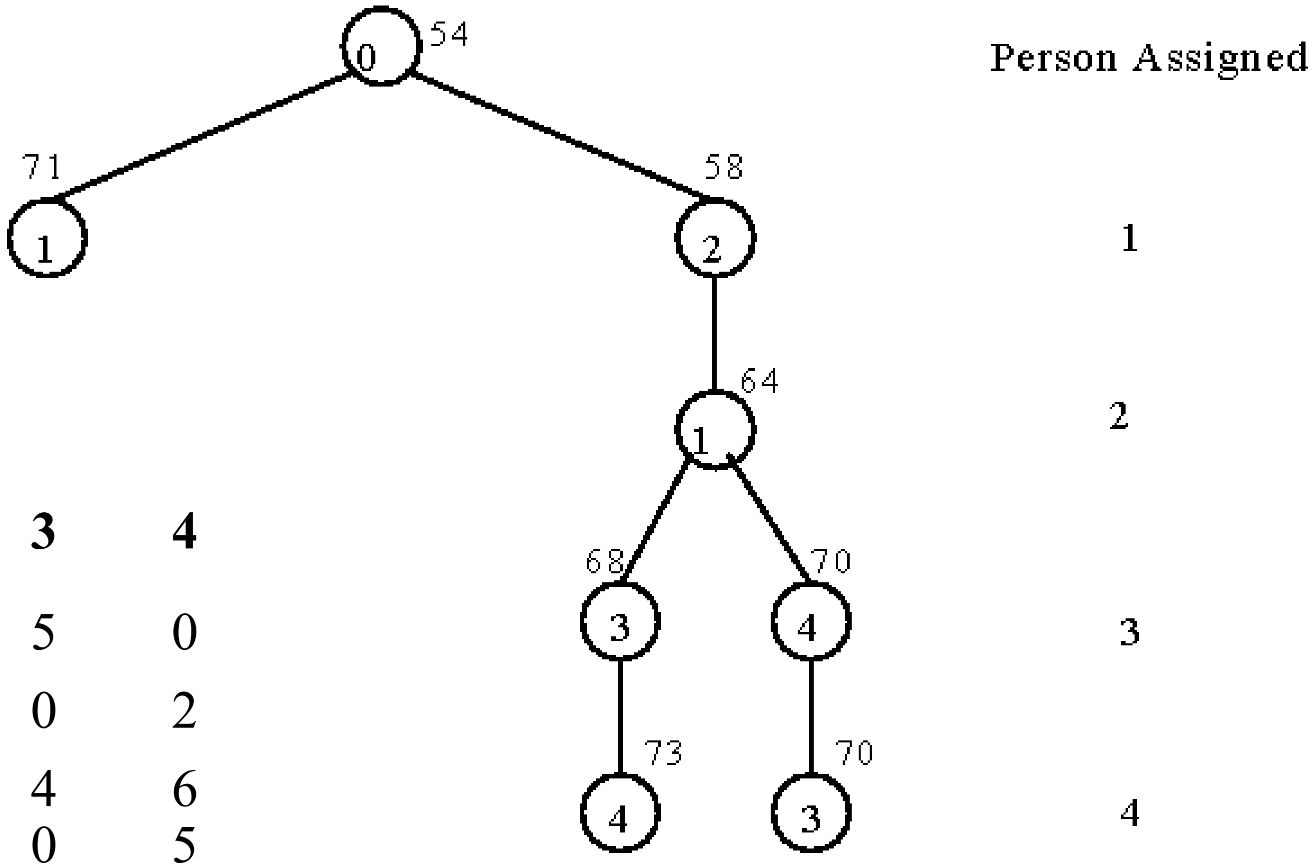
- Reduced cost matrix

Jobs / Persons	1	2	3	4	
1	17	4	5	0	(-12)
2	6	1	0	2	(-26)
3	0	15	4	6	(-3)
4	8	0	0	5	(-10)
		(-3)			

- A reduced cost matrix can be obtained: subtract a constant from each row and each column respectively such that each row and each column contains at least one zero.
- Total cost subtracted: 12+26+3+10+3 = 54
- This is a lower bound of our solution.

Branch-and-Bound for the Personnel Assignment Problem

- Bounding of subsolutions:



Jobs / Persons	1	2	3	4
1	17	4	5	0
2	6	1	0	2
3	0	15	4	6
4	8	0	0	5

The Traveling Salesperson Optimization Problem

- It is NP-complete.
- A cost matrix

i \ j	1	2	3	4	5	6	7
1	∞	3	93	13	33	9	57
2	4	∞	77	42	21	16	34
3	45	17	∞	36	16	28	25
4	39	90	80	∞	56	7	91
5	28	46	88	33	∞	25	57
6	3	88	18	46	92	∞	7
7	44	26	33	27	84	39	∞

- A reduced cost matrix

i \ j	1	2	3	4	5	6	7	
1	∞	0	90	10	30	6	54	(-3)
2	0	∞	73	38	17	12	30	(-4)
3	29	1	∞	20	0	12	9	(-16)
4	32	83	73	∞	49	0	84	(-7)
5	3	21	63	8	∞	0	32	(-25)
6	0	85	15	43	89	∞	4	(-3)
7	18	0	7	1	58	13	∞	(-26)

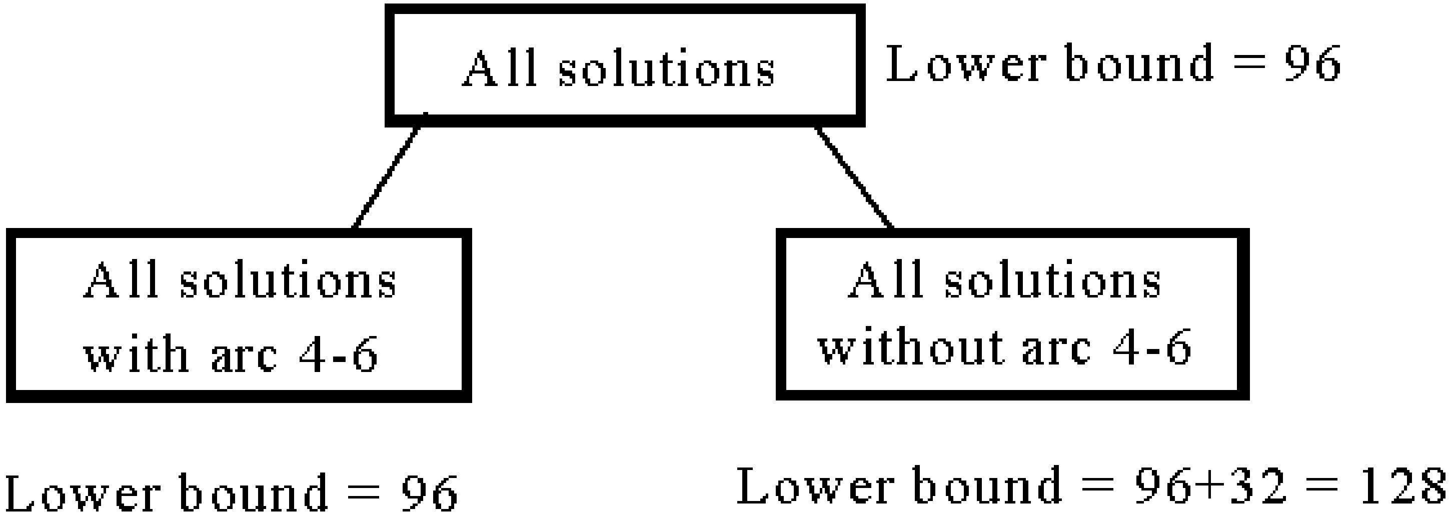
Reduced: 84

- Another reduced matrix

i \ j	1	2	3	4	5	6	7
1	∞	0	83	9	30	6	50
2	0	∞	66	37	17	12	26
3	29	1	∞	19	0	12	5
4	32	83	66	∞	49	0	80
5	3	21	56	7	∞	0	28
6	0	85	8	42	89	∞	0
7	18	0	0	0	58	13	∞
			(-7)	(-1)			(-4)

Total cost reduced: 84 + 7 + 1 + 4 = 96 (lower bound)

- The highest level of a decision tree:



- If we use arc 3-5 to split, the difference on the lower bounds is 17 + 1 = 18.

- A reduced cost matrix if arc (4,6) is included in the solution.

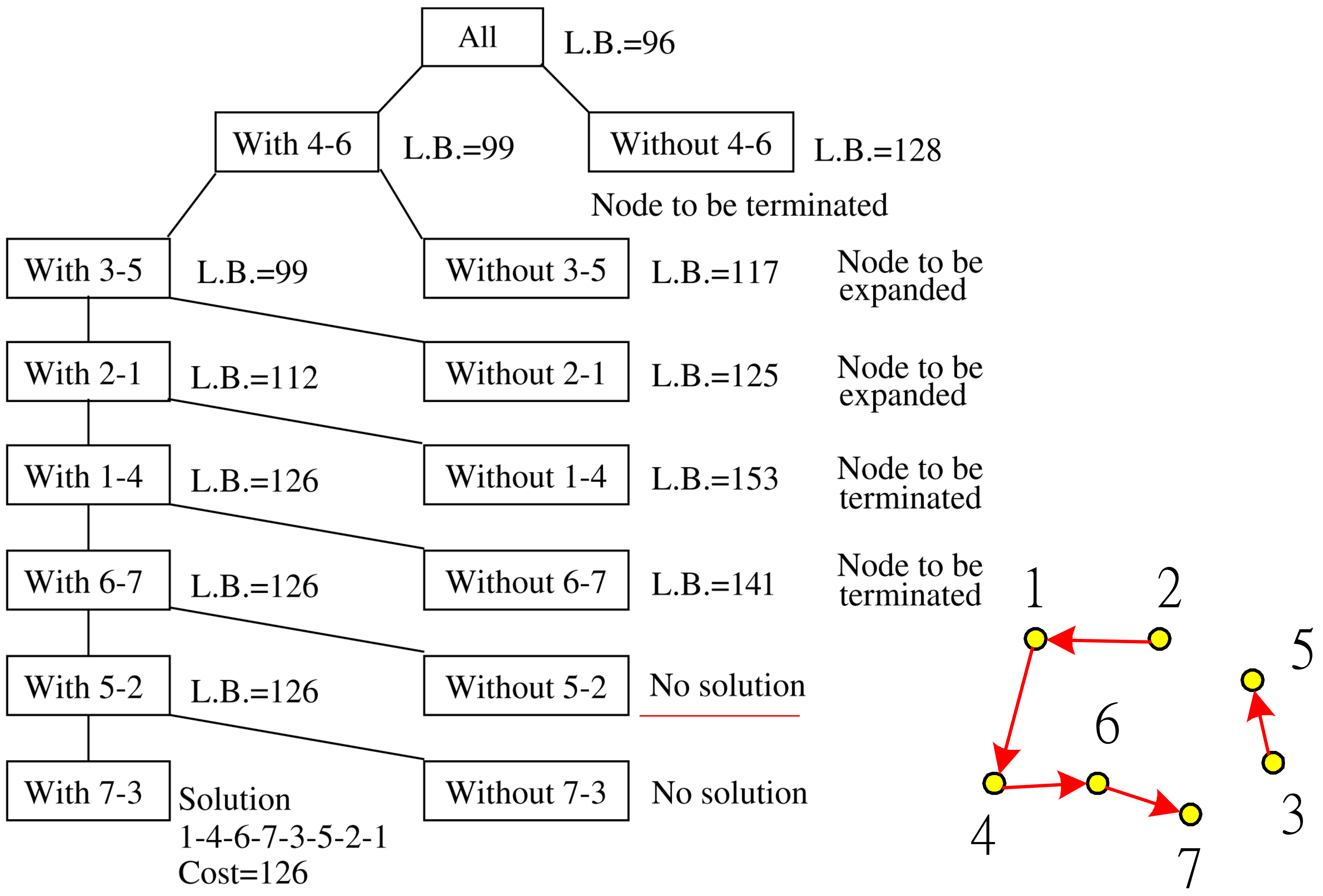
j / i	1	2	3	4	5	7
1	∞	0	83	9	30	50
2	0	∞	66	37	17	26
3	29	1	∞	19	0	5
5	3	21	56	7	∞	28
6	0	85	8	∞	89	0
7	18	0	0	0	58	∞

Arc (6,4) is changed to be infinity since it cannot be included in the solution.

- The reduced cost matrix for all solutions with arc 4-6.

i \ j	1	2	3	4	5	7	
1	∞	0	83	9	30	50	
2	0	∞	66	37	17	26	
3	29	1	∞	19	0	5	
5	0	18	53	4	∞	25	(-3)
6	0	85	8	∞	89	0	
7	18	0	0	0	58	∞	

- Total cost reduced: 96 + 3 = 99 (new lower bound).



A branch-and-bound solution of a traveling salesperson problem.

The 0/1 Knapsack Problem

- Positive integers $P_1, P_2, \ldots, P_n$ (profit)
  $W_1, W_2, \ldots, W_n$ (weight)
  M (capacity)

$$\text{Maximize } \sum_{i=1}^{n} P_i X_i$$

$$\text{subject to } \sum_{i=1}^{n} W_i X_i \le M \quad X_i = 0 \text{ or } 1,\ i = 1, \ldots, n.$$

The problem is modified:

$$\text{Minimize } -\sum_{i=1}^{n} P_i X_i$$

- e.g. $n = 6$, $M = 34$

i	1	2	3	4	5	6
P_i	6	10	4	5	6	4
W_i	10	19	8	10	12	8

$(P_i/W_i \geq P_{i+1}/W_{i+1})$

- A feasible solution: $X_1 = 1$, $X_2 = 1$, $X_3 = 0$, $X_4 = 0$, $X_5 = 0$, $X_6 = 0$
  $-(P_1+P_2) = -16$ (upper bound)
  Any solution higher than -16 cannot be an optimal solution.

Relax the Restriction

- Relax our restriction from $X_i = 0$ or 1 to $0 \le X_i \le 1$ (knapsack problem)

Let $-\sum_{i=1}^{n} P_i X_i$ be an optimal solution for 0/1 knapsack problem and $-\sum_{i=1}^{n} P_i X_i'$ be an optimal solution for knapsack problem.

$$\text{Let } Y = -\sum_{i=1}^{n} P_i X_i, \quad Y' = -\sum_{i=1}^{n} P_i X_i'.$$

$$\Rightarrow Y' \le Y$$

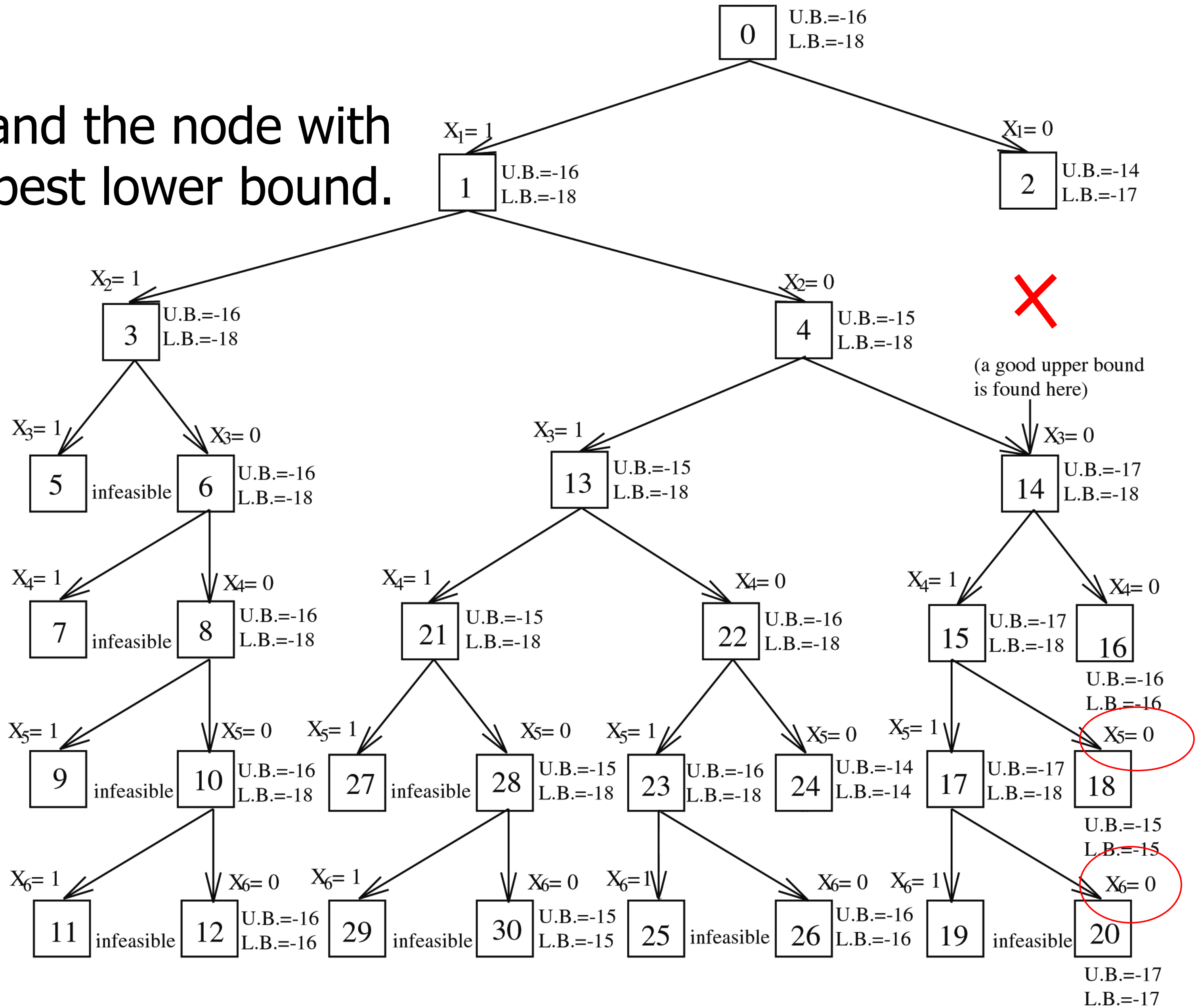
Upper Bound and Lower Bound

- We can use the greedy method to find an optimal solution for knapsack problem:

  $X_1 = 1, X_2 = 1, X_3 = 5/8, X_4 = 0, X_5 = 0, X_6 = 0$
  $-(P_1+P_2+5/8P_3) = -18.5$ (lower bound)
  -18 is our lower bound (only consider integers)

$\Rightarrow -18 \leq$ optimal solution ≤ -16
optimal solution: $X_1 = 1, X_2 = 0, X_3 = 0, X_4 = 1, X_5 = 1, X_6 = 0$
$-(P_1+P_4+P_5) = -17$

Expand the node with the best lower bound.



0/1 knapsack problem solved by branch-and-bound strategy.

The A^* Algorithm

- Used to solve optimization problems.
- Using the best-first strategy.
- If a feasible solution (goal node) is obtained, then it is optimal and we can stop.
- Cost function of node n : f(n)

  $f(n) = g(n) + h(n)$

  g(n): cost from root to node n.

  h(n): estimated cost from node n to a goal node.

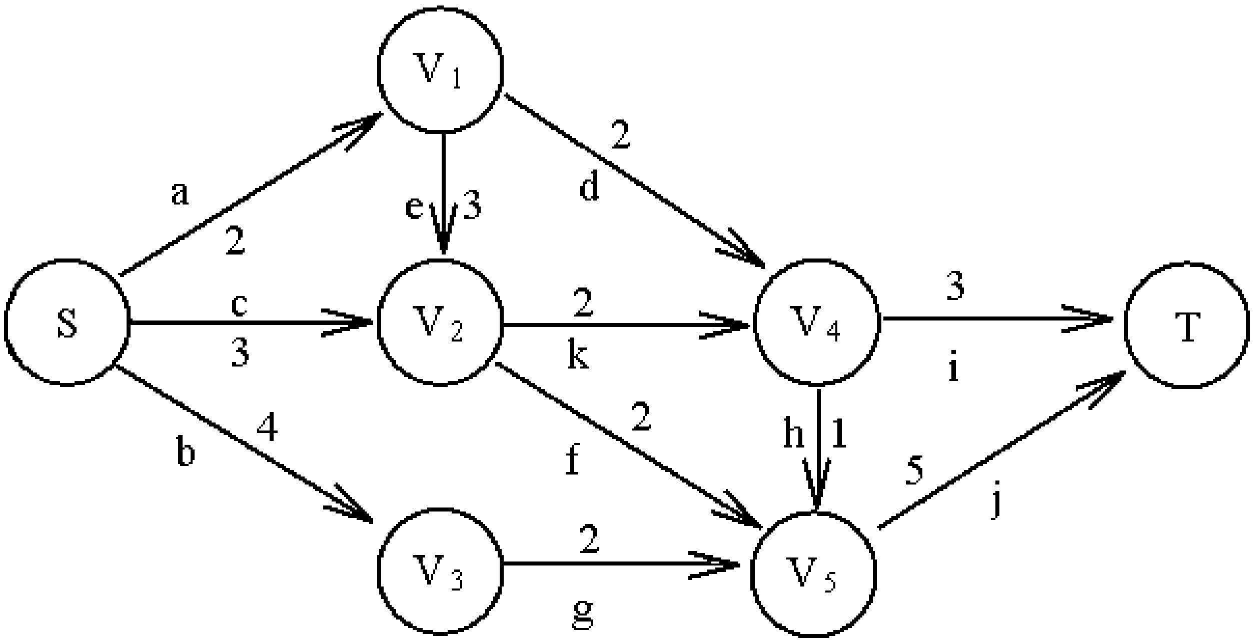
  $h^*(n)$: "real" cost from node n to a goal node.

- If we guarantee $h(n) \leq h^*(n)$, then

  $f(n) = g(n) + h(n) \leq g(n)+h^*(n) = f^*(n)$

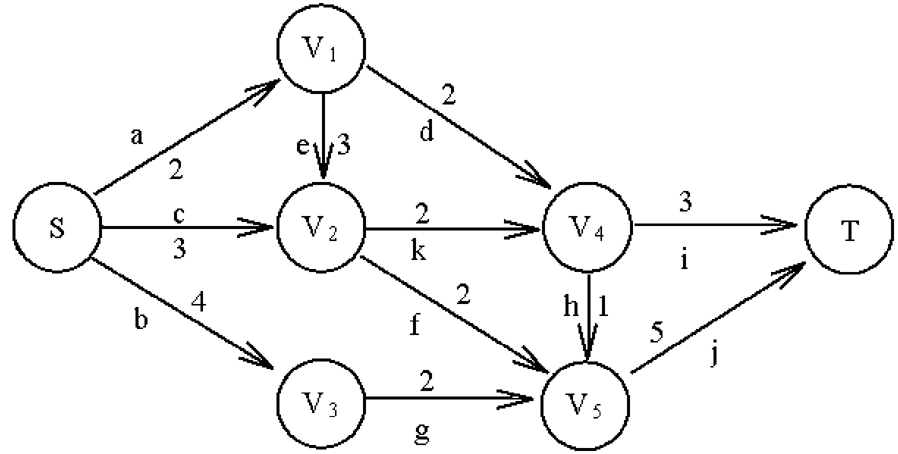
An Example for A* Algorithm

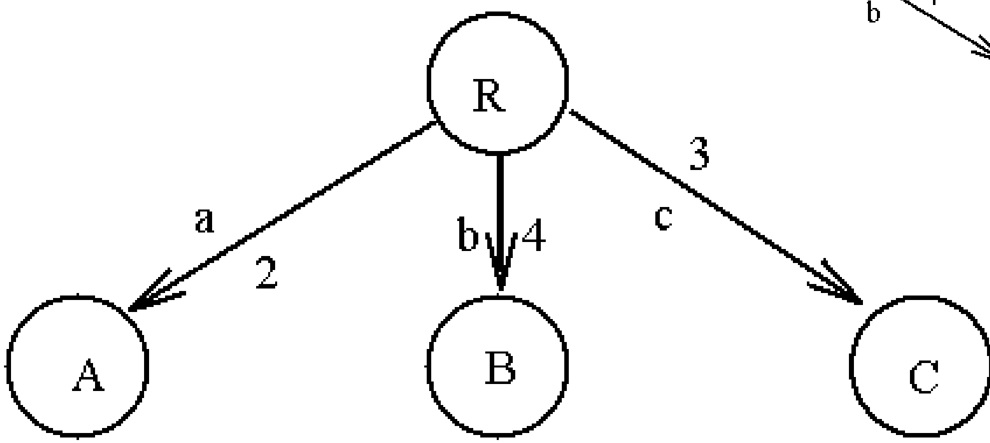
- A graph to illustrate A* algorithm.



- <u>Stop if the selected node is also a goal node.</u>

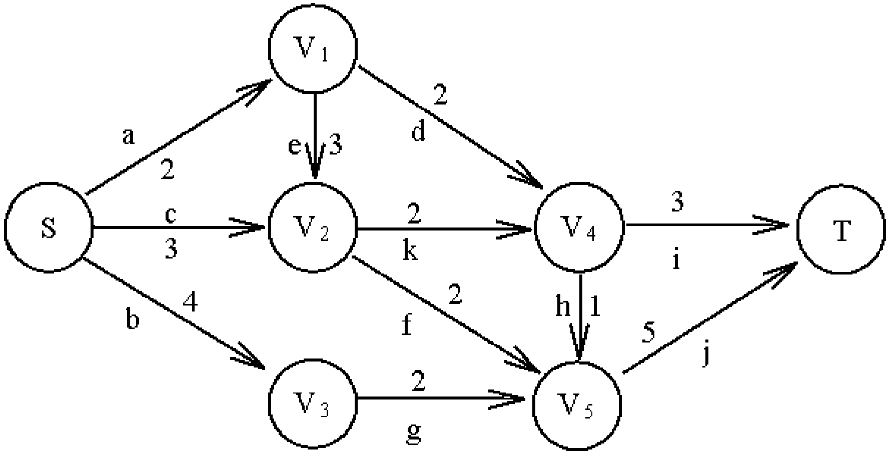
- Step 1:

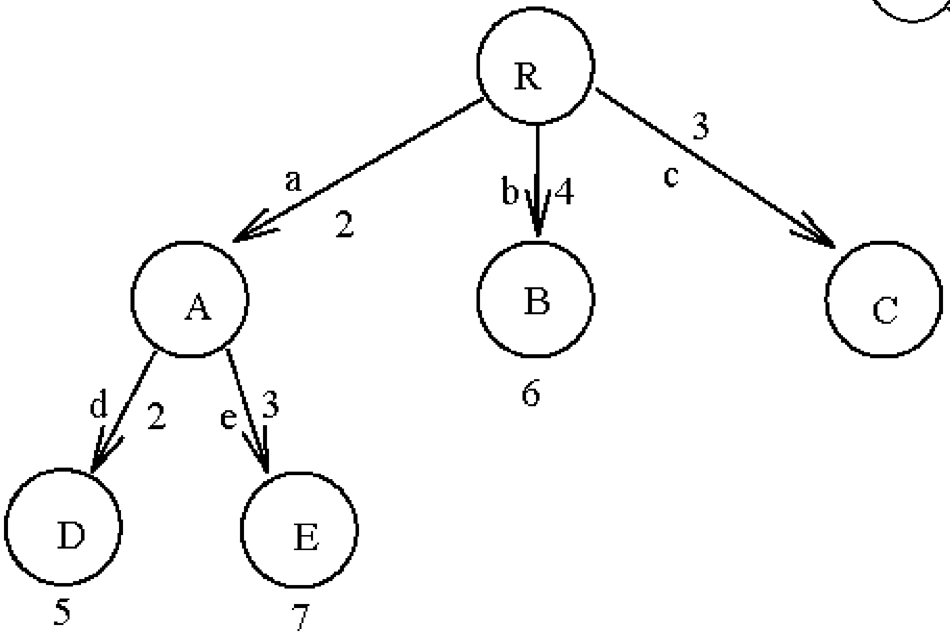




g(A)=2	h(A)=min{2,3}=2	f(A)=2+2=4
g(B)=4	h(B)=min{2}=2	f(B)=4+2=6
g(C)=3	h(C)=min{2,2}=2	f(C)= 3+2=5

- Step 2: Expand node A.

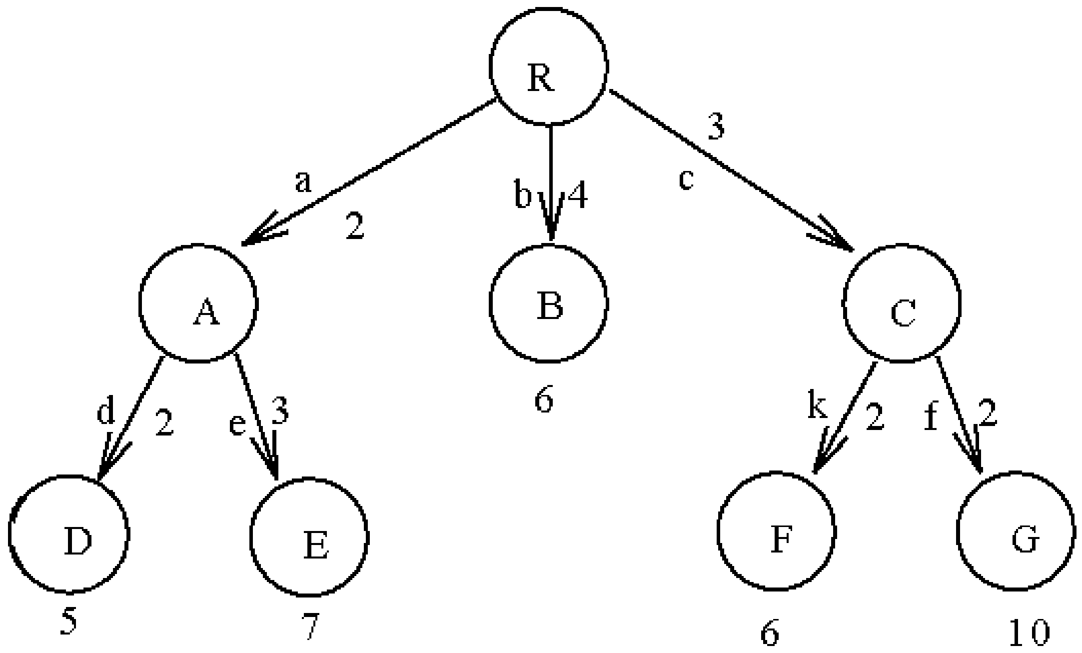




$g(D)=2+2=4 \quad h(D)=\min\{3,1\}=1 \quad f(D)=4+1=5$

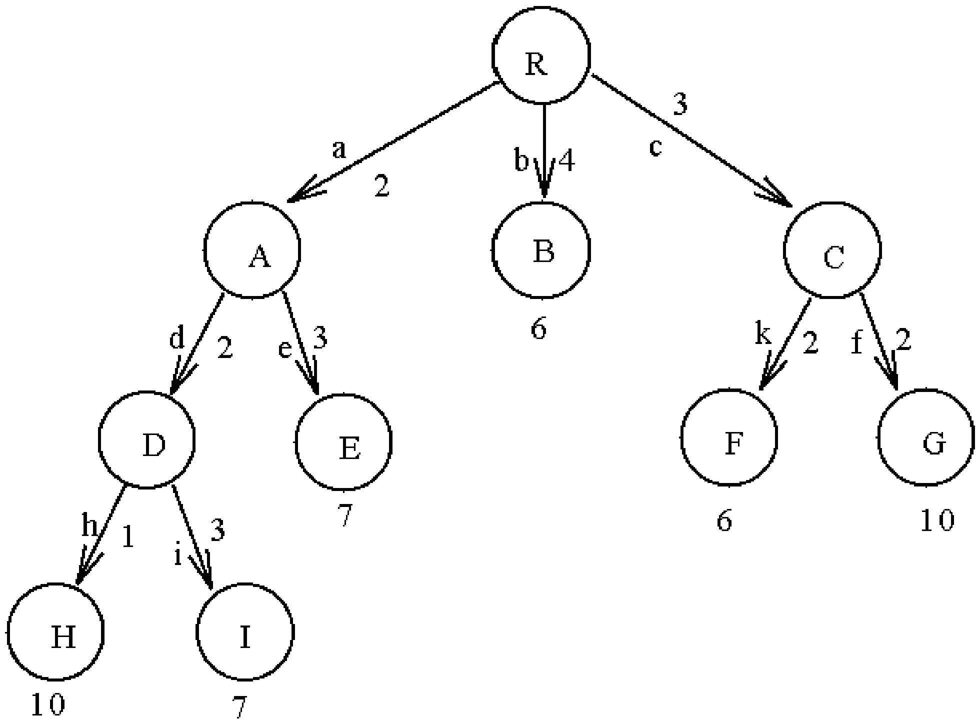
$g(E)=2+3=5 \quad h(E)=\min\{2,2\}=2 \quad f(E)=5+2=7$

- Step 3: Expand node C.



$$g(F)=3+2=5 \qquad h(F)=\min\{3,1\}=1 \qquad f(F)=5+1=6$$
$$g(G)=3+2=5 \qquad h(G)=\min\{5\}=5 \qquad f(G)=5+5=10$$

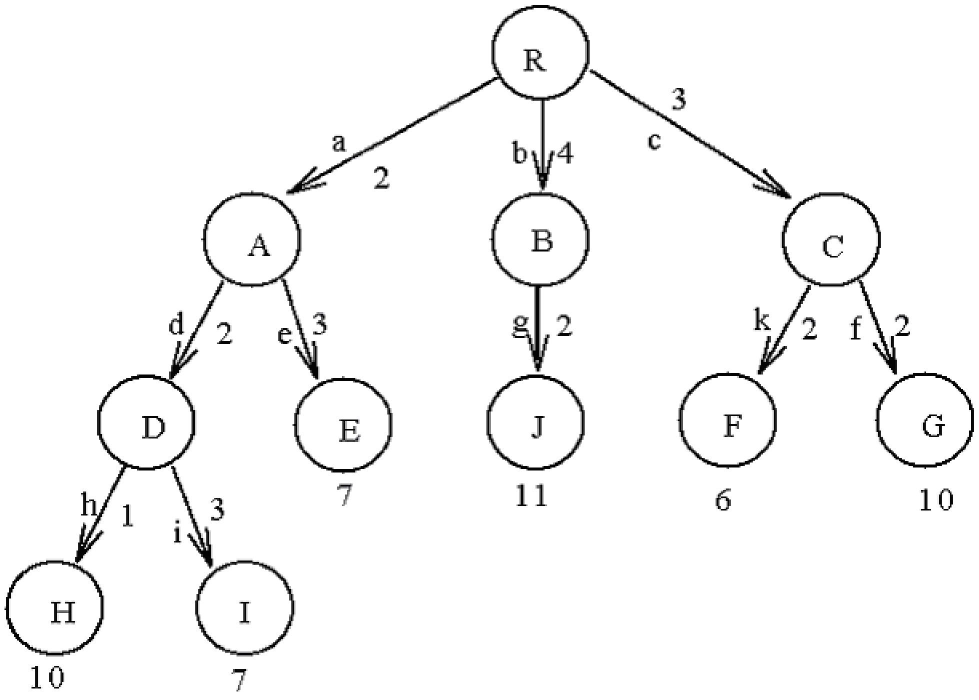
- Step 4: Expand node D.



g(H)=2+2+1=5 h(H)=min{5}=5 f(H)=5+5=10

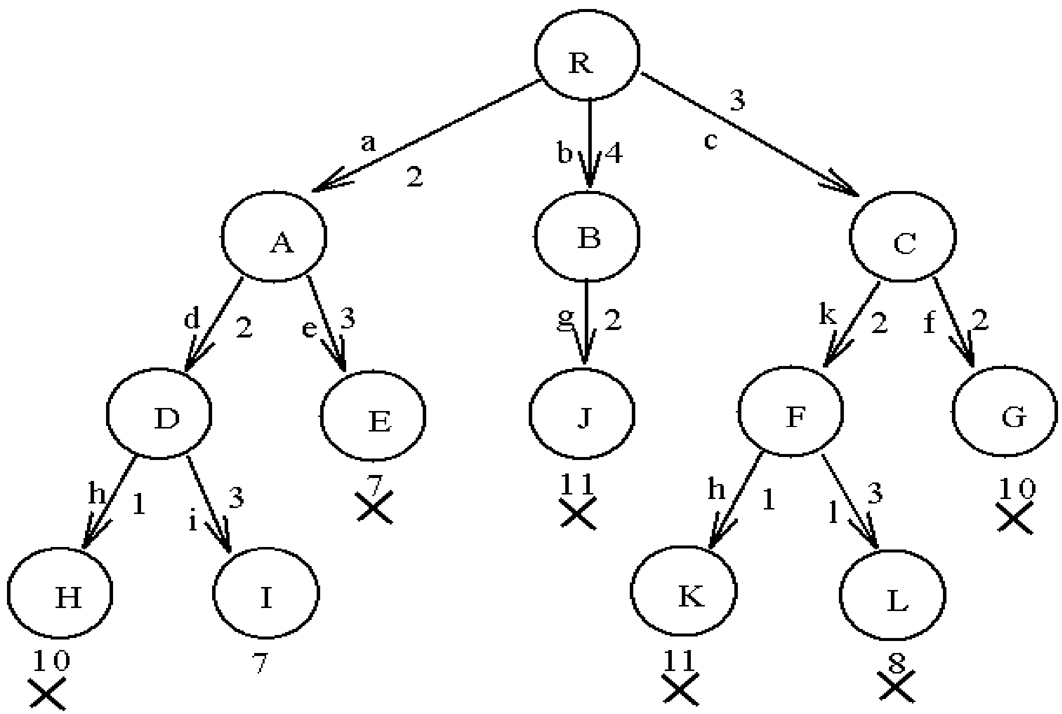
g(I)=2+2+3=7 h(I)=0 f(I)=7+0=7

- Step 5: Expand node B.



$$g(J)=4+2=6 \qquad h(J)=\min\{5\}=5 \qquad f(J)=6+5=11$$

- Step 6: Expand node F.



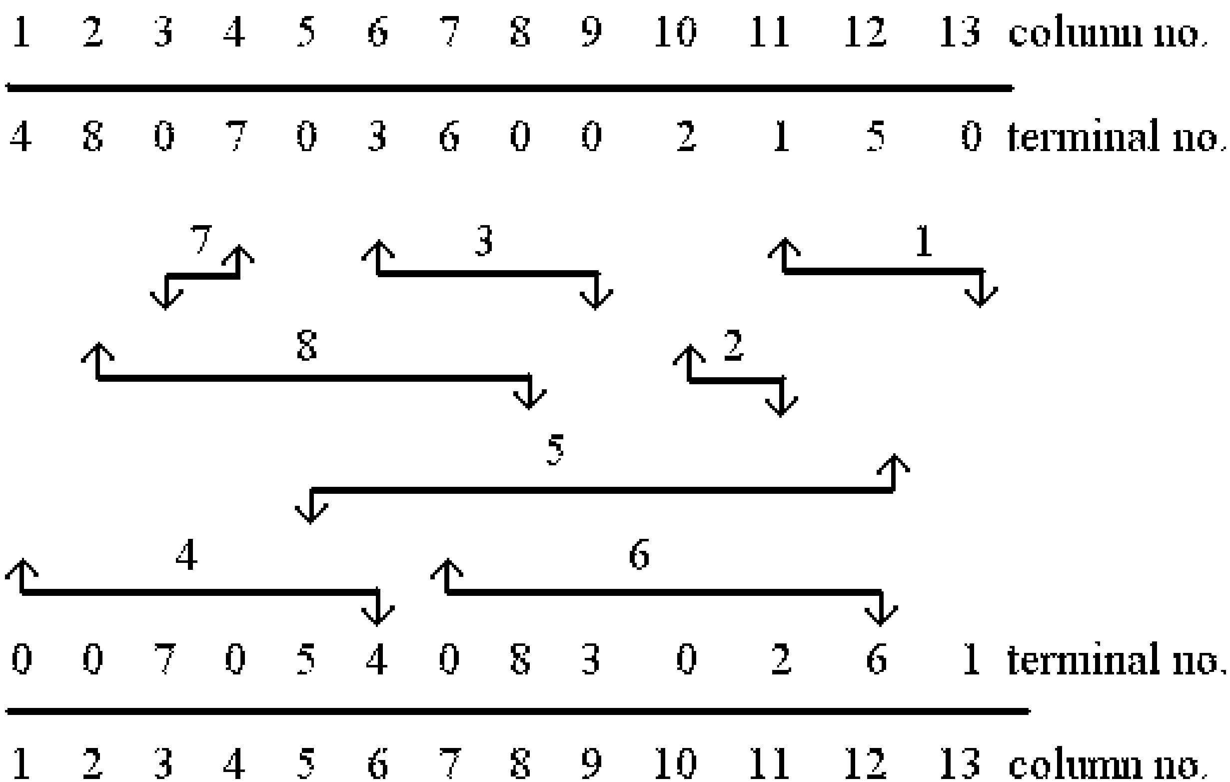
$f(n) \le f^*(n)$

g(K)=3+2+1=6 h(K)=min{5}=5 f(K)=6+5=11
g(L)=3+2+3=8 h(L)=0 f(L)=8+0=8

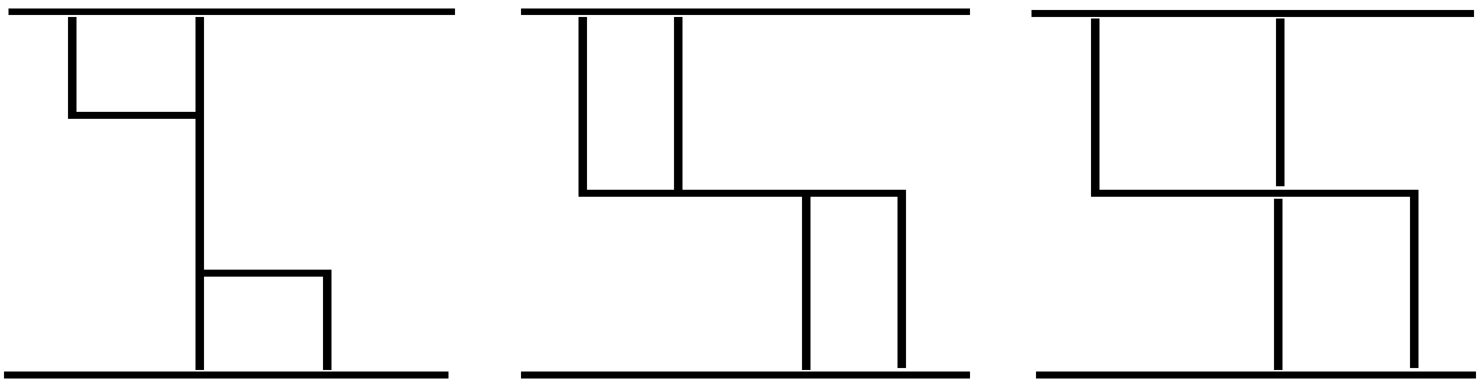
Node I is a goal node. Thus, the final solution is obtained.

The Channel Routing Problem

- A channel specification

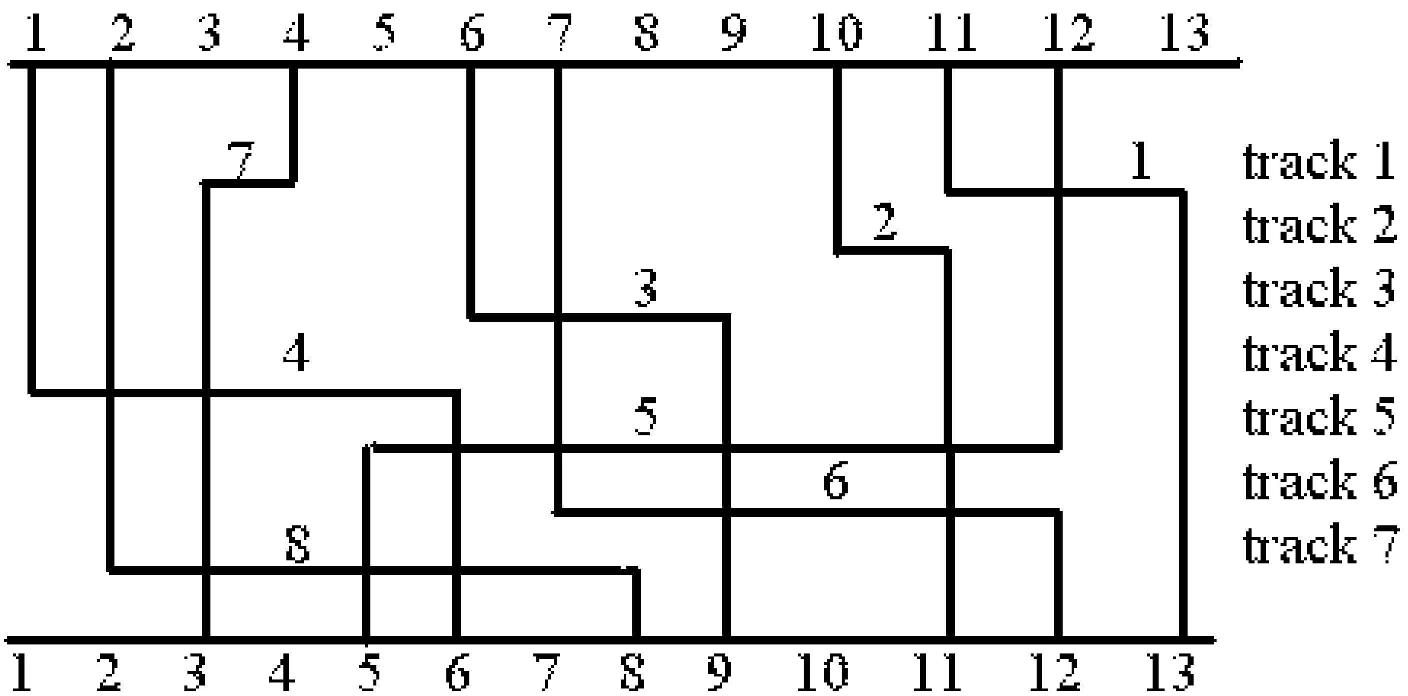


- Illegal wirings:



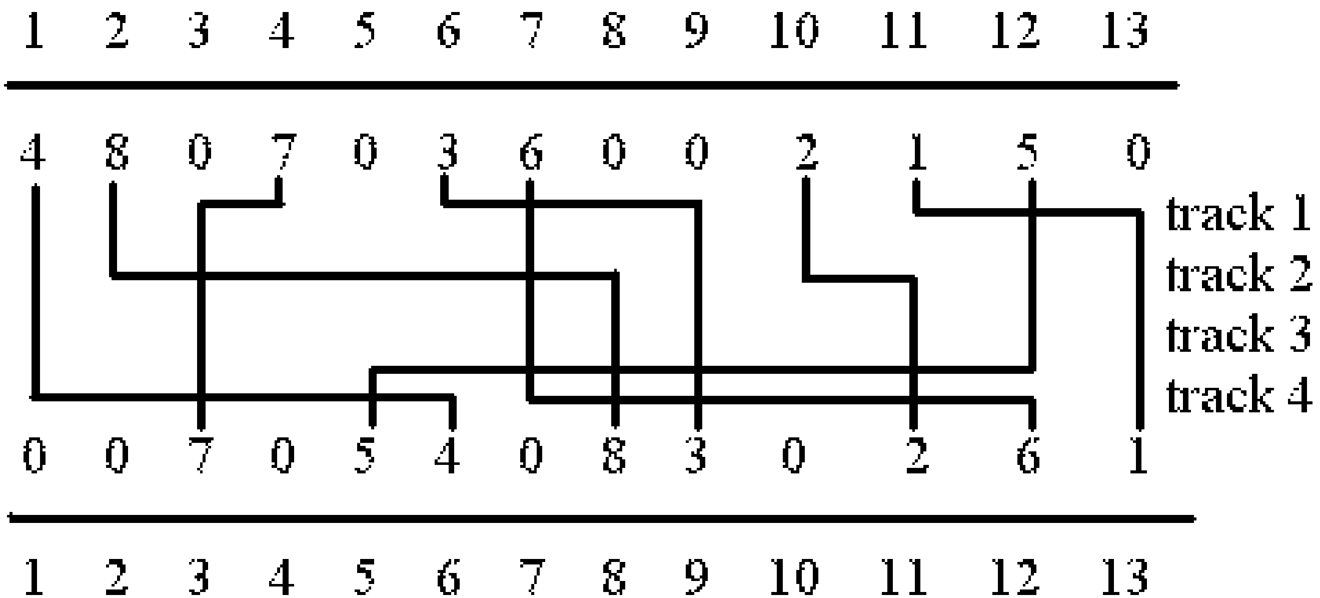
- We want to find a routing which minimizes the number of tracks.

A Feasible Routing



- 7 tracks are needed.

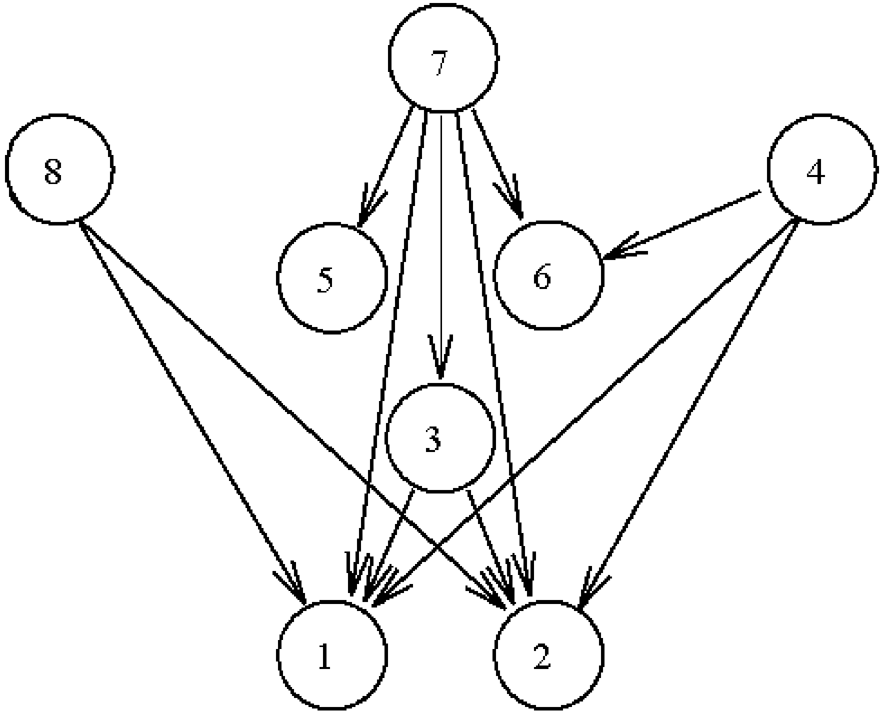
An Optimal Routing

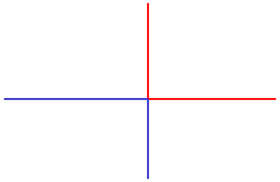


- 4 tracks are needed.
- This problem is NP-complete.

A* Algorithm for the Channel Routing Problem

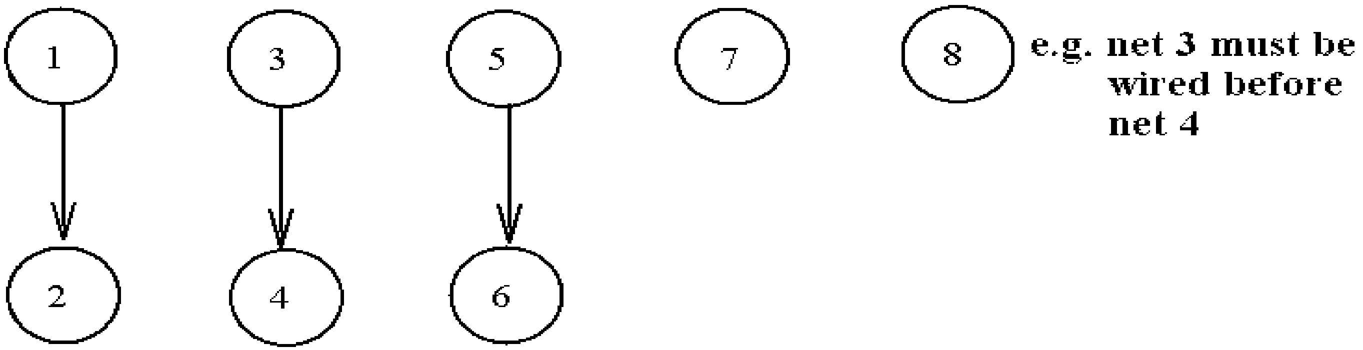
- Horizontal constraint graph (HCG).





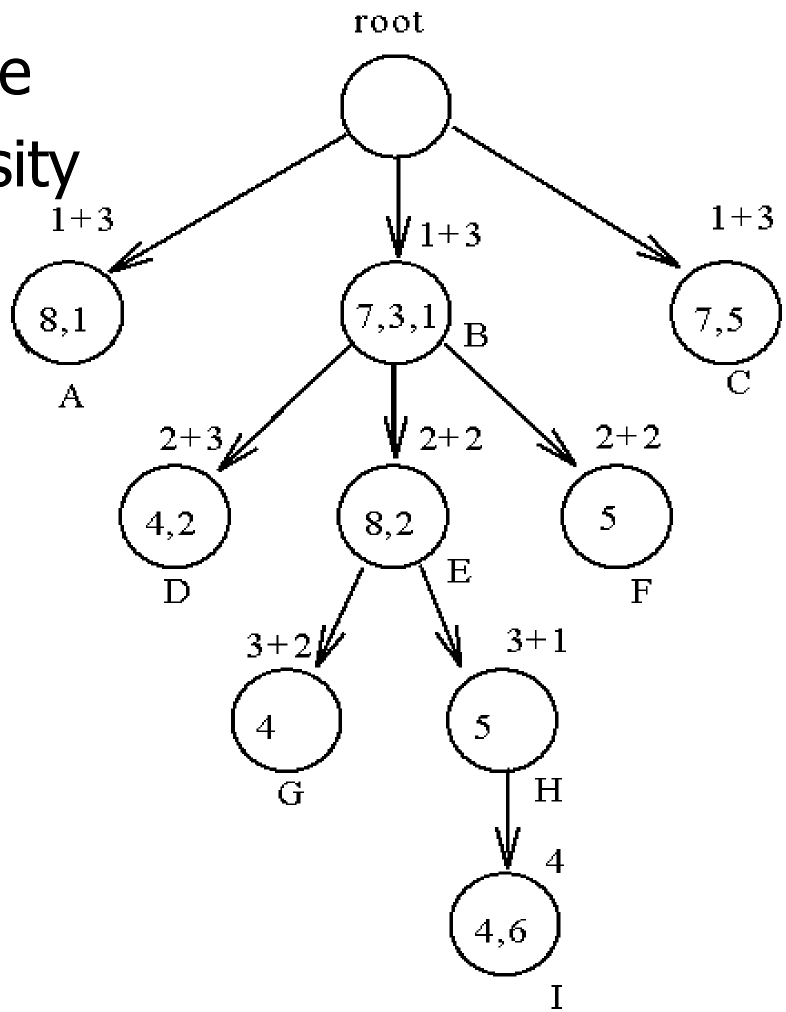
- e.g. net 8 must be to the left of net 1 and net 2 if they are in the same track.

- Vertical constraint graph:



- Maximum cliques in HCG: {1,8}, {1,3,7}, {5,7}. Each maximum clique can be assigned to a track.

- $f(n) = g(n) + h(n)$,
  - $g(n)$: the level of the tree
  - $h(n)$: maximal local density



A partial solution tree for the channel routing problem by using A^* algorithm.