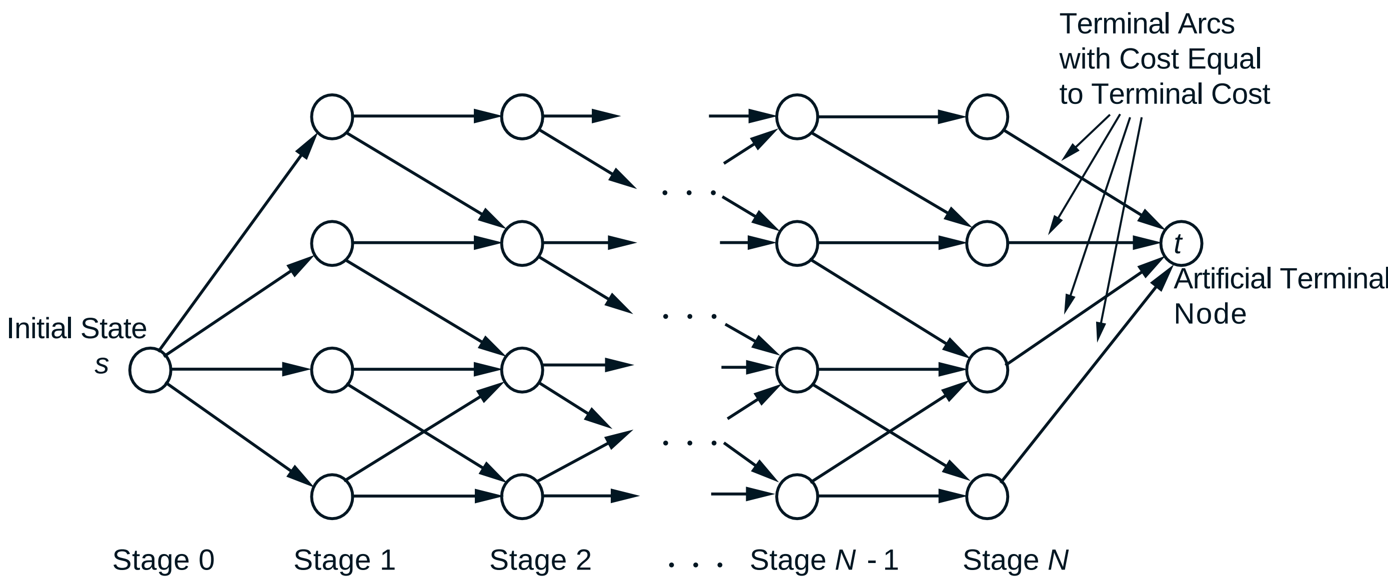
6.231 DYNAMIC PROGRAMMING

LECTURE 3

LECTURE OUTLINE

- Deterministic finite-state DP problems
- Backward shortest path algorithm
- Forward shortest path algorithm
- Shortest path examples
- Alternative shortest path algorithms

DETERMINISTIC FINITE-STATE PROBLEM



- States <==> Nodes
- Controls <==> Arcs
- Control sequences (open-loop) <==> paths from initial state to terminal states
- a_{ij}^k: Cost of transition from state $i \in S_k$ to state $j \in S_{k+1}$ at time k (view it as "length" of the arc)
- a_{it}^N: Terminal cost of state $i \in S_N$
- Cost of control sequence <==> Cost of the corresponding path (view it as "length" of the path)

BACKWARD AND FORWARD DP ALGORITHMS

- DP algorithm:

$$J_N(i) = a_{it}^N, \quad i \in S_N,$$

$$J_k(i) = \min_{j \in S_{k+1}} \big[a_{ij}^k + J_{k+1}(j)\big], \;\; i \in S_k, \; k = 0, \ldots, N-1$$

The optimal cost is $J_0(s)$ and is equal to the length of the shortest path from s to t

- Observation: An optimal path $s \to t$ is also an optimal path $t \to s$ in a "reverse" shortest path problem where the direction of each arc is reversed and its length is left unchanged

- Forward DP algorithm (= backward DP algorithm for the reverse problem):

$$\tilde{J}_N(j) = a_{sj}^0, \quad j \in S_1,$$

$$\tilde{J}_k(j) = \min_{i \in S_{N-k}} \big[a_{ij}^{N-k} + \tilde{J}_{k+1}(i)\big], \quad j \in S_{N-k+1}$$

The optimal cost is $\tilde{J}_0(t) = \min_{i \in S_N} \big[a_{it}^N + \tilde{J}_1(i)\big]$

- View $\tilde{J}_k(j)$ as optimal cost-to-arrive to state j from initial state s

A NOTE ON FORWARD DP ALGORITHMS

- There is no forward DP algorithm for stochastic problems
- Mathematically, for stochastic problems, we cannot restrict ourselves to open-loop sequences, so the shortest path viewpoint fails
- Conceptually, in the presence of uncertainty, the concept of "optimal-cost-to-arrive" at a state x_k does not make sense. For example, it may be impossible to guarantee (with prob. 1) that any given state can be reached
- By contrast, even in stochastic problems, the concept of "optimal cost-to-go" from any state x_k makes clear sense

GENERIC SHORTEST PATH PROBLEMS

- $\{1, 2, \ldots, N, t\}$: nodes of a graph (t: the *destination*)
- a_{ij}: cost of moving from node i to node j
- Find a shortest (minimum cost) path from each node i to node t
- Assumption: All cycles have nonnegative length. Then an optimal path need not take more than N moves
- We formulate the problem as one where we require exactly N moves but allow degenerate moves from a node i to itself with cost $a_{ii} = 0$

$J_k(i)$ = opt. cost of getting from i to t in $N-k$ moves

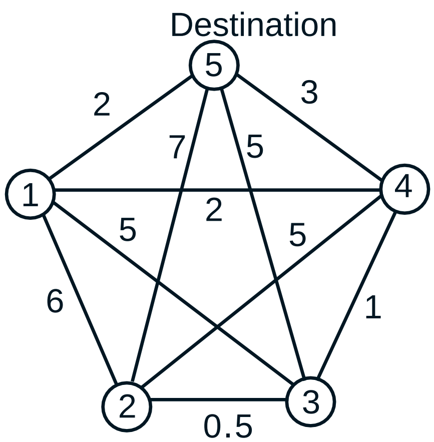
$J_0(i)$: Cost of the optimal path from i to t.

- DP algorithm:

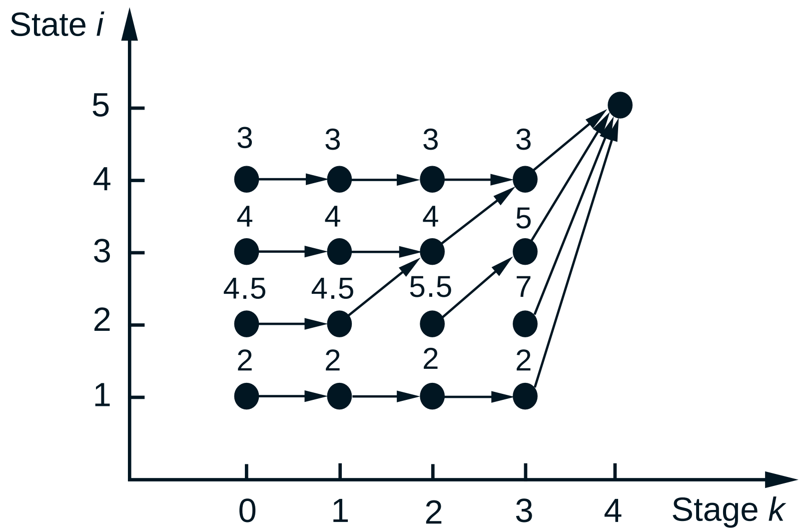
$$J_k(i) = \min_{j=1,\ldots,N} \big[a_{ij} + J_{k+1}(j)\big], \qquad k = 0, 1, \ldots, N-2,$$

with $J_{N-1}(i) = a_{it}$, $i = 1, 2, \ldots, N$

EXAMPLE



(a)



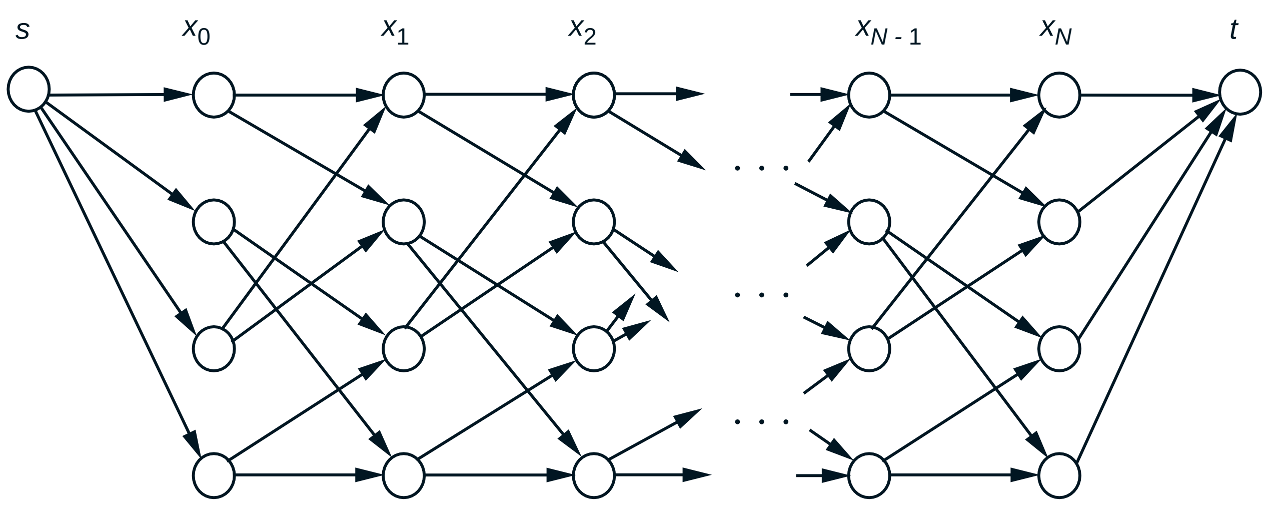
(b)

$$J_{N-1}(i) = a_{it}, \qquad i = 1, 2, \ldots, N,$$

$$J_k(i) = \min_{j=1,\ldots,N} \left[a_{ij} + J_{k+1}(j) \right], \qquad k = 0, 1, \ldots, N-2.$$

ESTIMATION / HIDDEN MARKOV MODELS

- Markov chain with transition probabilities p_{ij}
- State transitions are hidden from view
- For each transition, we get an (independent) observation
- $r(z; i, j)$: Prob. the observation takes value z when the state transition is from i to j
- Trajectory estimation problem: Given the observation sequence $Z_N = \{z_1, z_2, \ldots, z_N\}$, what is the "most likely" state transition sequence $\hat{X}_N = \{\hat{x}_0, \hat{x}_1, \ldots, \hat{x}_N\}$ [one that maximizes $p(X_N \mid Z_N)$ over all $X_N = \{x_0, x_1, \ldots, x_N\}$].



VITERBI ALGORITHM

- We have

$$p(X_N \mid Z_N) = \frac{p(X_N, Z_N)}{p(Z_N)}$$

where $p(X_N, Z_N)$ and $p(Z_N)$ are the unconditional probabilities of occurrence of (X_N, Z_N) and Z_N

- Maximizing $p(X_N \mid Z_N)$ is equivalent with maximizing $\ln(p(X_N, Z_N))$

- We have (using the "multiplication rule" for cond. probs)

$$p(X_N, Z_N) = \pi_{x_0} \prod_{k=1}^{N} p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k)$$

so the problem is equivalent to

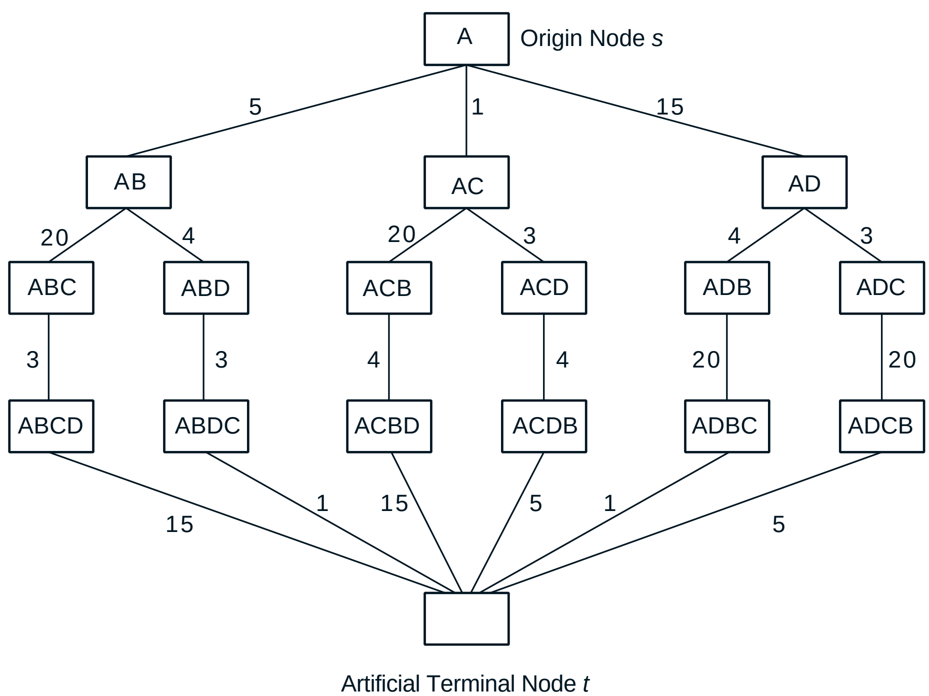
$$\text{minimize} - \ln(\pi_{x_0}) - \sum_{k=1}^{N} \ln\big(p_{x_{k-1}x_k} r(z_k; x_{k-1}, x_k)\big)$$

over all possible sequences $\{x_0, x_1, \ldots, x_N\}$.

- This is a shortest path problem.

GENERAL SHORTEST PATH ALGORITHMS

- There are many nonDP shortest path algorithms. They can all be used to solve deterministic finite-state problems
- They may be preferable than DP if they avoid calculating the optimal cost-to-go of EVERY state
- Essential for problems with HUGE state spaces.
- Combinatorial optimization is prime example (e.g., scheduling/traveling salesman)



	5	1	15
5		20	4
1	20		3
15	4	3	

LABEL CORRECTING METHODS

- Given: Origin s, destination t, lengths $a_{ij} \geq 0$.
- Idea is to progressively discover shorter paths from the origin s to every other node i
- Notation:
  - d_i (label of i): Length of the shortest path found (initially $d_s = 0$, $d_i = \infty$ for $i \neq s$)
  - UPPER: The label d_t of the destination
  - OPEN list: Contains nodes that are currently active in the sense that they are candidates for further examination (initially OPEN=$\{s\}$)

Label Correcting Algorithm

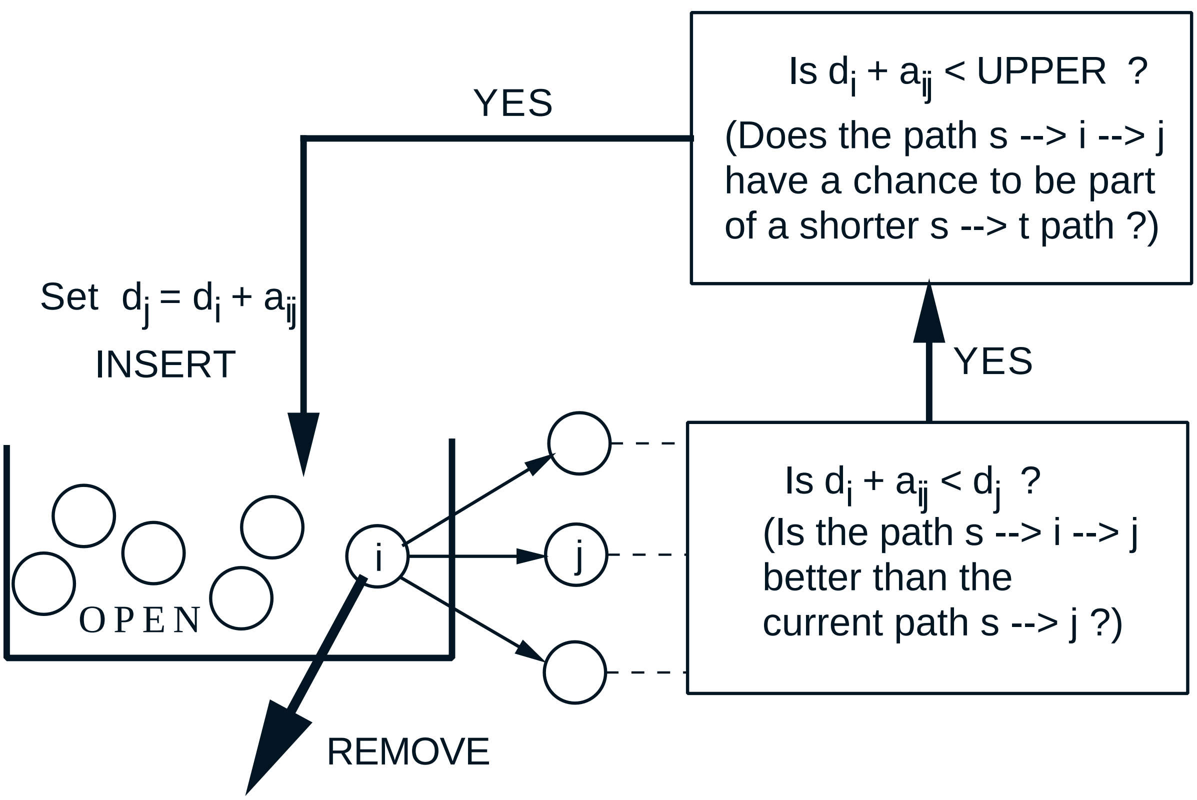
Step 1 (Node Removal): Remove a node i from OPEN and for each child j of i, do step 2

Step 2 (Node Insertion Test): If $d_i + a_{ij} < \min\{d_j, \text{UPPER}\}$, set $d_j = d_i + a_{ij}$ and set i to be the parent of j. In addition, if $j \neq t$, place j in OPEN if it is not already in OPEN, while if $j = t$, set UPPER to the new value $d_i + a_{it}$ of d_t

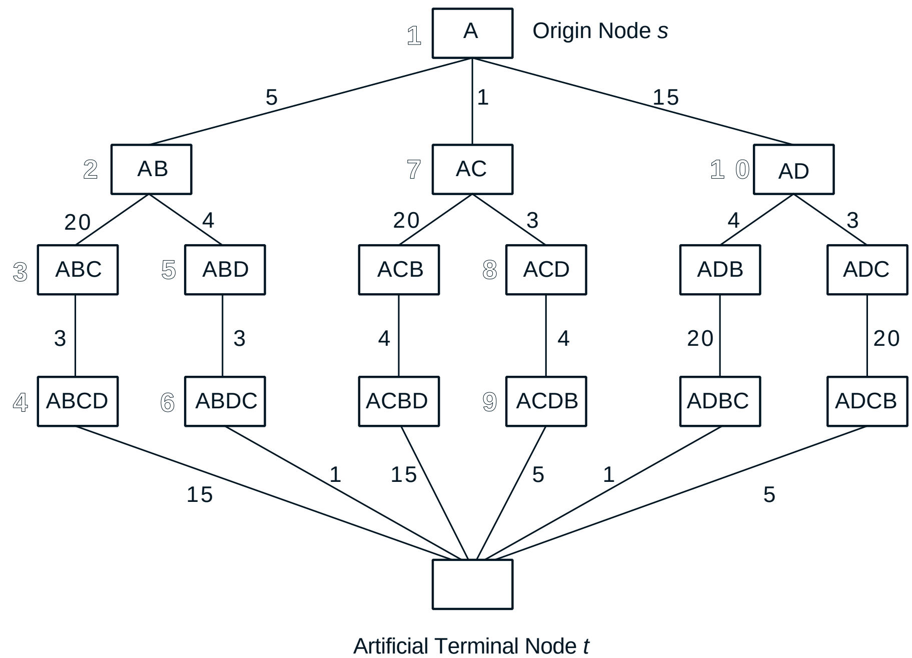
Step 3 (Termination Test): If OPEN is empty, terminate; else go to step 1

VISUALIZATION/EXPLANATION

- Given: Origin s, destination t, lengths $a_{ij} \geq 0$
- d_i (label of i): Length of the shortest path found thus far (initially $d_s = 0$, $d_i = \infty$ for $i \neq s$). The label d_i is implicitly associated with an $s \to i$ path
- UPPER: The label d_t of the destination
- OPEN list: Contains "active" nodes (initially OPEN=$\{s\}$)



EXAMPLE



Iter. No.	Node Exiting OPEN	OPEN after Iteration	UPPER
0	-	1	∞
1	1	2, 7,10	∞
2	2	3, 5, 7, 10	∞
3	3	4, 5, 7, 10	∞
4	4	5, 7, 10	43
5	5	6, 7, 10	43
6	6	7, 10	13
7	7	8, 10	13
8	8	9, 10	13
9	9	10	13
10	10	Empty	13

- Note that some nodes never entered OPEN

VALIDITY OF LABEL CORRECTING METHODS

Proposition: If there exists at least one path from the origin to the destination, the label correcting algorithm terminates with UPPER equal to the shortest distance from the origin to the destination

Proof: (1) Each time a node j enters OPEN, its label is decreased and becomes equal to the length of some path from s to j

(2) The number of possible distinct path lengths is finite, so the number of times a node can enter OPEN is finite, and the algorithm terminates

(3) Let $(s, j_1, j_2, \ldots, j_k, t)$ be a shortest path and let d^* be the shortest distance. If UPPER $> d^*$ at termination, UPPER will also be larger than the length of all the paths $(s, j_1, \ldots, j_m)$, $m = 1, \ldots, k$, throughout the algorithm. Hence, node j_k will never enter the OPEN list with d_{j_k} equal to the shortest distance from s to j_k. Similarly node j_{k-1} will never enter the OPEN list with $d_{j_{k-1}}$ equal to the shortest distance from s to j_{k-1}. Continue to j_1 to get a contradiction