6.231 DYNAMIC PROGRAMMING

LECTURE 15

LECTURE OUTLINE

- Rollout algorithms
- Cost improvement property
- Discrete deterministic problems
- Sequential consistency and greedy algorithms
- Sequential improvement

ROLLOUT ALGORITHMS

- *One-step lookahead policy*: At each k and state x_k, use the control $\overline{\mu}_k(x_k)$ that

$$\min_{u_k \in U_k(x_k)} E\Big\{g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big)\Big\},$$

where

  - $\tilde{J}_N = g_N$.
  - $\tilde{J}_{k+1}$: approximation to true cost-to-go J_{k+1}

- *Rollout algorithm*: When $\tilde{J}_k$ is the cost-to-go of some heuristic policy (called the *base policy*)

- Cost improvement property (to be shown): The rollout algorithm achieves no worse (and usually much better) cost than the base heuristic starting from the same state.

- Main difficulty: Calculating $\tilde{J}_k(x_k)$ may be computationally intensive if the cost-to-go of the base policy cannot be analytically calculated.
  - May involve Monte Carlo simulation if the problem is stochastic.
  - Things improve in the deterministic case.

EXAMPLE: THE QUIZ PROBLEM

- A person is given N questions; answering correctly question i has probability p_i, with reward v_i.
- Quiz terminates at the first incorrect answer.
- Problem: Choose the ordering of questions so as to maximize the total expected reward.
- Assuming no other constraints, it is optimal to use the *index policy*: Questions should be answered in decreasing order of the "index of preference" $p_i v_i/(1 - p_i)$.
- With minor changes in the problem, the index policy need not be optimal. Examples:
  - A limit ($< N$) on the maximum number of questions that can be answered.
  - Time windows, sequence-dependent rewards, precedence constraints.
- Rollout with the index policy as base policy: Convenient because at a given state (subset of questions already answered), the index policy and its expected reward can be easily calculated.

COST IMPROVEMENT PROPERTY

- Let

$$\overline{J}_k(x_k)\text{: Cost-to-go of the rollout policy}$$

$$H_k(x_k)\text{: Cost-to-go of the base policy}$$

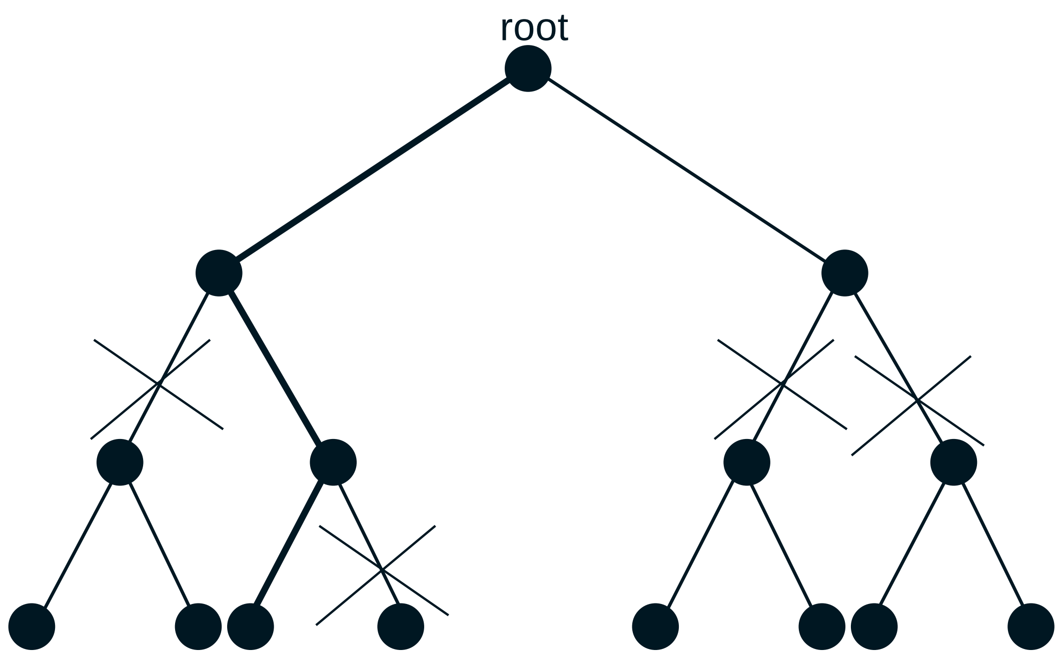
- We claim that $\overline{J}_k(x_k) \leq H_k(x_k)$ for all x_k and k
- Proof by induction: We have $\overline{J}_N(x_N) = H_N(x_N)$ for all x_N. Assume that

$$\overline{J}_{k+1}(x_{k+1}) \leq H_{k+1}(x_{k+1}), \quad \forall\, x_{k+1}.$$

Then, for all x_k

$$\begin{aligned}
\overline{J}_k(x_k) &= E\Big\{g_k\big(x_k, \overline{\mu}_k(x_k), w_k\big) + \overline{J}_{k+1}\Big(f_k\big(x_k, \overline{\mu}_k(x_k), w_k\big)\Big)\Big\} \\
&\leq E\Big\{g_k\big(x_k, \overline{\mu}_k(x_k), w_k\big) + H_{k+1}\Big(f_k\big(x_k, \overline{\mu}_k(x_k), w_k\big)\Big)\Big\} \\
&\leq E\Big\{g_k\big(x_k, \mu_k(x_k), w_k\big) + H_{k+1}\Big(f_k\big(x_k, \mu_k(x_k), w_k\big)\Big)\Big\} \\
&= H_k(x_k)
\end{aligned}$$

EXAMPLE: THE BREAKTHROUGH PROBLEM



- Given a binary tree with N stages.
- Each arc is either free or is blocked (crossed out in the figure).
- Problem: Find a free path from the root to the leaves (such as the one shown with thick lines).
- Base heuristic (greedy): Follow the right branch if free; else follow the left branch if free.
- For large N and given prob. of free branch: the rollout algorithm requires $O(N)$ times more computation, but has $O(N)$ times larger prob. of finding a free path than the greedy algorithm.

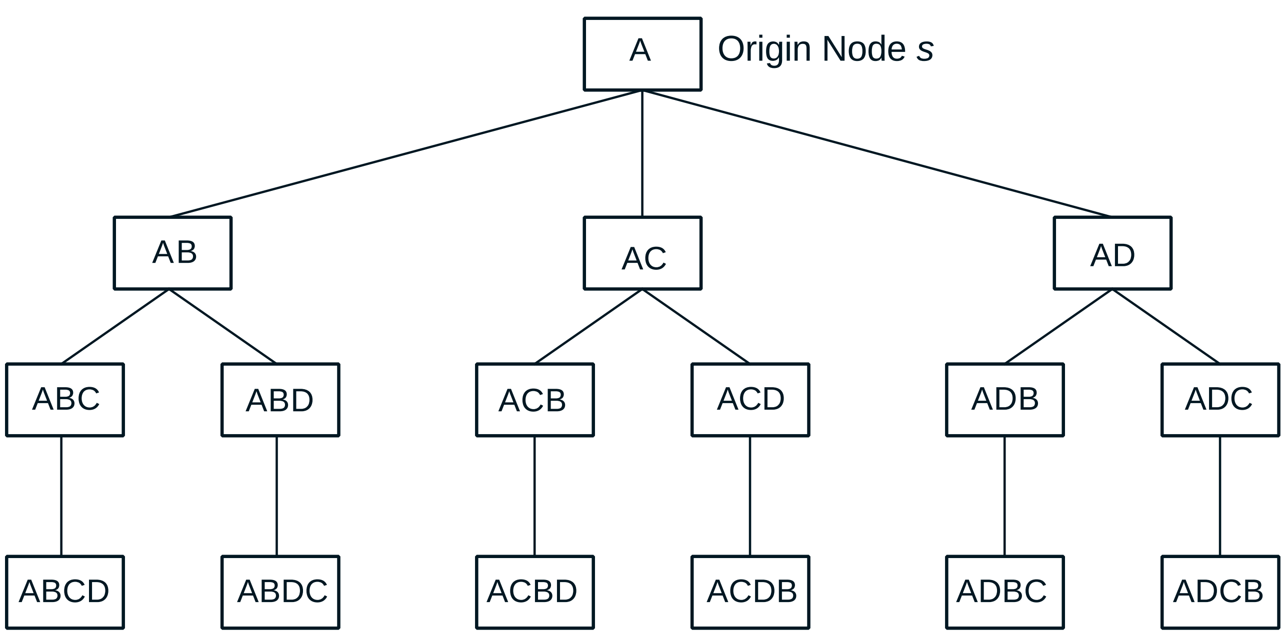
DISCRETE DETERMINISTIC PROBLEMS

- Any discrete optimization problem (with finite number of choices/feasible solutions) can be represented as a sequential decision process by using a tree.

- The leaves of the tree correspond to the feasible solutions.

- The problem can be solved by DP, starting from the leaves and going back towards the root.

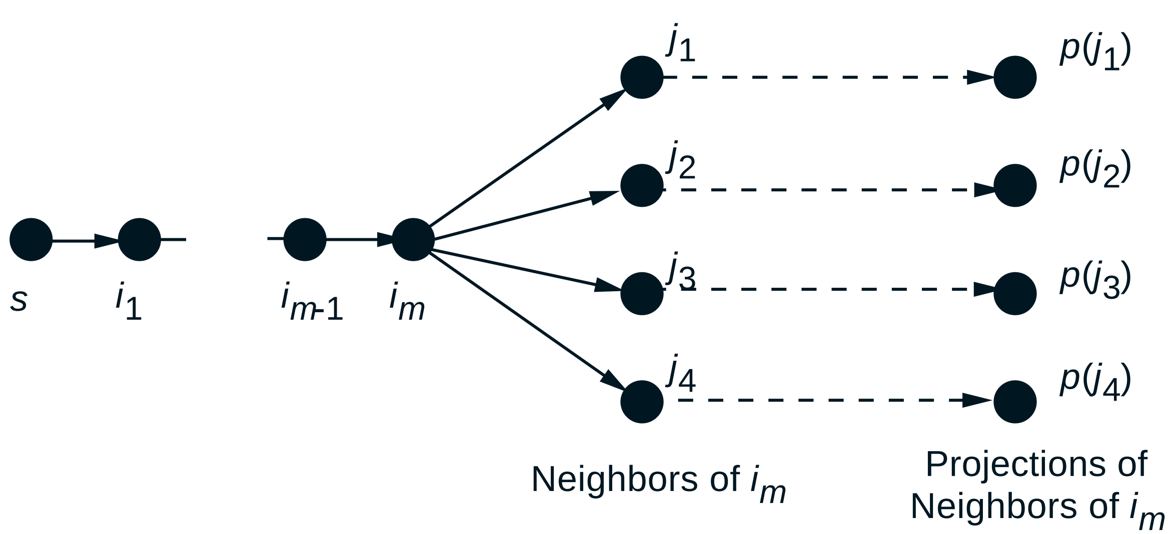
- Example: Traveling salesman problem. Find a minimum cost tour that goes exactly once through each of N cities.



Traveling salesman problem with four cities A, B, C, D

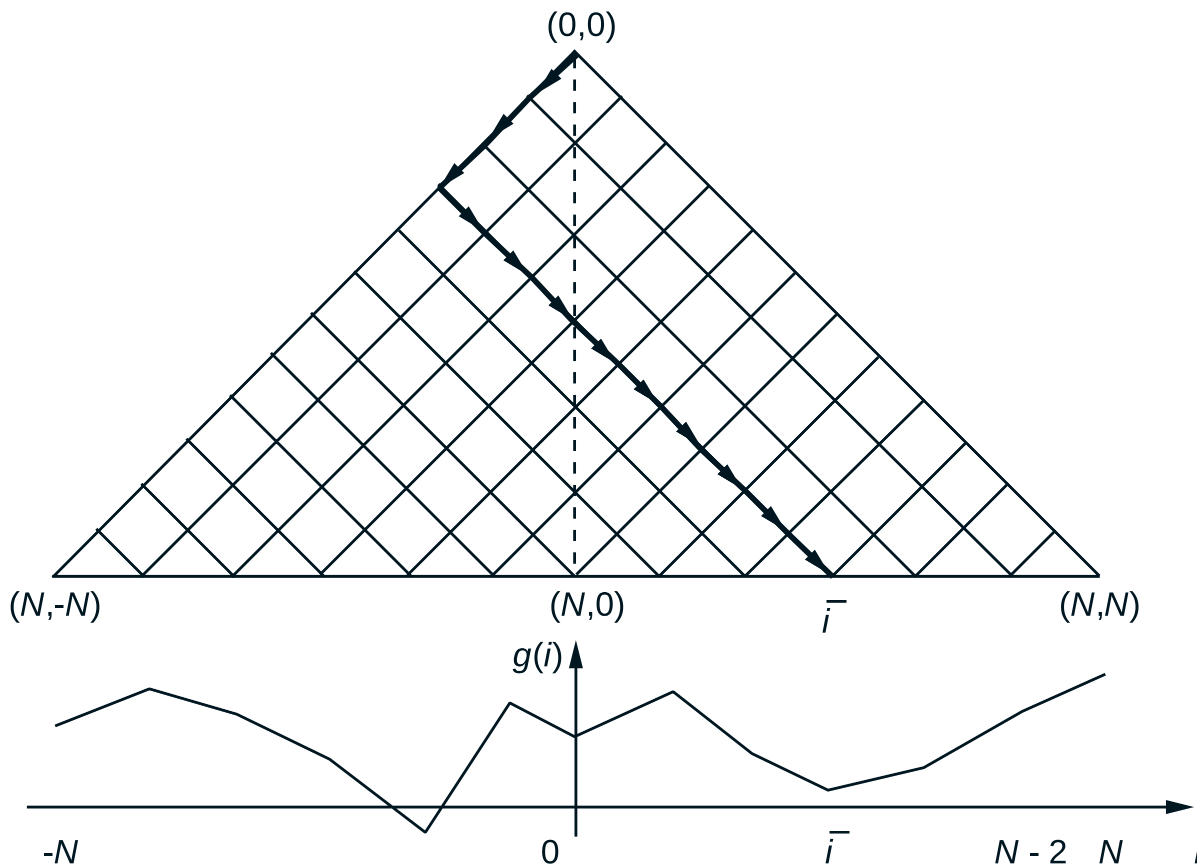
A CLASS OF GENERAL DISCRETE PROBLEMS

- Generic problem:
  - Given a graph with directed arcs
  - A special node s called the $origin$
  - A set of terminal nodes, called $destinations$, and a cost $g(i)$ for each destination i.
  - Find min cost path starting at the origin, ending at one of the destination nodes.
- Base heuristic: For any nondestination node i, constructs a path $(i, i_1, \ldots, i_m, \bar{i})$ starting at i and ending at one of the destination nodes $\bar{i}$. We call $\bar{i}$ the $projection$ of i, and we denote $H(i) = g(\bar{i})$.
- Rollout algorithm: Start at the origin; choose the successor node with least cost projection



EXAMPLE: ONE-DIMENSIONAL WALK

- A person takes either a unit step to the left or a unit step to the right. Minimize the cost $g(i)$ of the point i where he will end up after N steps.



- Base heuristic: Always go to the right. Rollout finds the rightmost $local\ minimum$.

- Base heuristic: Compare always go to the right and always go the left. Choose the best of the two. Rollout finds a $global\ minimum$.

SEQUENTIAL CONSISTENCY

- The base heuristic is *sequentially consistent* if for every node i, whenever it generates the path $(i, i_1, \ldots, i_m, \bar{i})$ starting at i, it also generates the path $(i_1, \ldots, i_m, \bar{i})$ starting at the node i_1 (i.e., all nodes of its path have the same projection).

- Prime example of a sequentially consistent heuristic is a *greedy algorithm*. It uses an *estimate* $F(i)$ of the optimal cost starting from i.

- At the typical step, given a path $(i, i_1, \ldots, i_m)$, where i_m is not a destination, the algorithm adds to the path a node i_{m+1} such that

$$i_{m+1} = \arg \min_{j \in N(i_m)} F(j)$$

- If the base heuristic is sequentially consistent, the cost of the rollout algorithm is no more than the cost of the base heuristic. In particular, if $(s, i_1, \ldots, i_{\bar{m}})$ is the rollout path, we have

$$H(s) \geq H(i_1) \geq \cdots \geq H(i_{\bar{m}-1}) \geq H(i_{\bar{m}})$$

where $H(i) =$ cost of the heuristic starting from i.

SEQUENTIAL IMPROVEMENT

- We say that the base heuristic is *sequentially improving* if for every non-destination node i, we have

$$H(i) \geq \min_{j \text{ is neighbor of } i} H(j)$$

- If the base heuristic is sequentially improving, the cost of the rollout algorithm is no more than the cost of the base heuristic, starting from any node.
- Fortified rollout algorithm:
  - Simple variant of the rollout algorithm, where we keep the best path found so far through the application of the base heuristic.
  - If the rollout path deviates from the best path found, then follow the best path.
  - Can be shown to be a rollout algorithm with sequentially improving base heuristic for a slightly modified variant of the original problem.
  - Has the cost improvement property.