Topics in Reinforcement Learning:
AlphaZero, ChatGPT, Neuro-Dynamic Programming, Model Predictive Control, Discrete Optimization
Arizona State University
Course CSE 691, Spring 2024


Dimitri P. Bertsekas
dpbertsekas@gmail.com

Lecture 5
Revisit Finite Horizon DP Problems - Deterministic Rollout
Outline

1. Finite Horizon Problems - Relation to Infinite Horizon
2. Rollout in General
3. Rollout for Deterministic Finite-State Problems
4. Cost Improvement Property of Rollout
5. Deterministic Rollout Variants and Extensions
Review: The Generic Finite Horizon DP Problem

Random Transition
\[ x_{k+1} = f_k(x_k, u_k, w_k) \]

Random Cost
\[ g_k(x_k, u_k, w_k) \]

- System \( x_{k+1} = f_k(x_k, u_k, w_k) \) with random “disturbance" \( w_k \) (e.g., physical noise, market uncertainties, demand for inventory, unpredictable breakdowns, etc)
- Cost function: \( E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k, w_k) \right\} \)
- Policies \( \pi = \{ \mu_0, \ldots, \mu_{N-1} \} \), where \( \mu_k \) is a “closed-loop control law" or “feedback policy"/a function of \( x_k \). A “lookup table" for the control \( u_k = \mu_k(x_k) \) to apply at \( x_k \).
- For given initial state \( x_0 \), minimize over all \( \pi = \{ \mu_0, \ldots, \mu_{N-1} \} \) the cost

\[ J_\pi(x_0) = E \left\{ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, \mu_k(x_k), w_k) \right\} \]

- Optimal cost function: \( J^*(x_0) = \min_\pi J_\pi(x_0) \). Optimal policy: \( J_{\pi^*}(x_0) = J^*(x_0) \)

We will be focusing on finite horizon: It’s most convenient for our algorithmic purposes (e.g., rollout) ... but nearly everything applies to infinite horizon
Review: The DP Algorithm

Produces the optimal costs $J_k^*(x_k)$ of the tail subproblems that start at $x_k$

Start with $J_N^*(x_N) = g_N(x_N)$, and for $k = 0, \ldots, N - 1$, let

$$J_k^*(x_k) = \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + J_{k+1}^*(f_k(x_k, u_k, w_k)) \right\}, \quad \text{for all } x_k.$$  

- The optimal cost $J^*(x_0)$ is obtained at the last step: $J_0^*(x_0) = J^*(x_0)$.
- The optimal policy is to use the minimizing $u_k^* = \mu_k^*(x_k)$ above.

Approximation in Value Space - Use of $\tilde{J}_{k+1}$ in Place of $J_{k+1}^*$

Sequentially, going forward, for $k = 0, 1, \ldots, N - 1$, observe $x_k$ and apply

$$\tilde{u}_k \in \arg \min_{u_k \in U_k(x_k)} E_{w_k} \left\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(f_k(x_k, u_k, w_k)) \right\}.$$  

There is also a multistep version.

There are many different ways to compute $\tilde{J}_{k+1}$ (e.g., on-line rollout, off-line training, problem approximation, heuristics, etc)
Infinite Horizon Problems: Review

Infinite number of stages, and stationary system and cost

- Cost of a policy \( \pi = \{\mu_0, \mu_1, \ldots\} \): The limit as \( N \to \infty \) of the \( N \)-stage costs

\[
J_\pi(x_0) = \lim_{N \to \infty} E_{w_k} \left\{ \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right\}
\]

- Optimal cost function \( J^*(x_0) = \min_{\pi} J_\pi(x_0) \).

- Bellman’s equation: \( J^*(x) = \min_{u \in \mathcal{U}(x)} E_w \left\{ g(x, u, w) + \alpha J^*(f(x, u, w)) \right\} \) for all \( x \)

- The nice case is discounted Markov Decision Problems (MDP): Finite state and action spaces, and \( \alpha < 1 \).

- Another nice case is Stochastic Shortest Path problems: Finite state and action spaces, \( \alpha = 1 \), and a cost-free and absorbing (goal/termination) state.
**Value iteration (VI):** Generates finite horizon opt. cost function sequence \( \{J_k\} \)

\[
J_k(x) = \min_{u \in U(x)} \mathbb{E}_w \left\{ g(x, u, w) + \alpha J_{k-1}(f(x, u, w)) \right\}, \quad J_0 \text{ is "arbitrary"}
\]

**Policy Iteration (PI):** Generates sequences of policies \( \{\mu^k\} \) and their cost functions \( \{J_{\mu^k}\} \); \( \mu^0 \) is “arbitrary”

The typical iteration starts with a policy \( \mu \) and generates a new policy \( \tilde{\mu} \) in two steps:

- **Policy evaluation step**, which computes \( J_\mu \) the cost function of the (base) policy \( \mu \)
- **Policy improvement step**, which computes the improved (rollout) policy \( \tilde{\mu} \) using the one-step lookahead minimization

\[
\tilde{\mu}(x) \in \arg \min_{u \in U(x)} \mathbb{E}_w \left\{ g(x, u, w) + \alpha J_\mu (f(x, u, w)) \right\}
\]

**Rollout** is a single policy iteration with policy evaluation performed by on-line simulation as needed
An Important Conceptual Idea: Finite Horizon can be Transformed to Infinite Horizon

As a result:

- The Bellman equation of the infinite horizon problem is the DP algorithm for the finite horizon problem
- Policy iteration/Newton step ideas apply to finite horizon problems
At State $x_k$

DP minimization

\[
\min_{u_k, \mu_{k+1}, \ldots, \mu_{k+\ell-1}} E \left\{ g_k(x_k, u_k, w_k) + \sum_{i=k+1}^{k+\ell-1} g_i(x_i, \mu_i(x_i), w_i) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right\}
\]

Rollout Control $\tilde{u}_k$

Rollout Policy $\tilde{\mu}_k$

First $\ell$ Steps

"Future"

$\tilde{J}_{k+\ell}(x_{k+\ell})$ is the Cost Function of Some Policy or Heuristic

- The policy used for rollout is called base policy
- The policy obtained by lookahead minimization is called rollout policy

Approximate variants

- $\tilde{J}_{k+\ell}(x_{k+\ell})$ may also approximate the cost function of the base policy
- Possibility of truncated rollout
Rollout is Important for this Course

Role of Rollout

- It provides important options for cost function approximation in the context of value space methods (a “good” option because $J_k^* \leq \tilde{J}_k$, based on visualizations)
- It is the basic building block of the fundamental PI algorithm (and approximate variants)

Reasons why it will be important:

- Rollout, in its pure form, is the RL method that is easiest to understand and apply
- Rollout is by far the most reliable
- It is very general: Applies to deterministic and stochastic problems, to finite horizon and infinite horizon
- Since it is a special case of approx. in value space, it relates to Newton’s method
- Deals well with on-line replanning, and provides a useful alternative to reoptimization in adaptive control
- It relates to model predictive control, and can be used to improve the stability of MPC schemes
- Truncated rollout can be combined with many of the RL methods used in practice [including self-learning (approximate PI), Q-learning, aggregation, and others]
System

\[ x_{k+1} = f_k(x_k, u_k), \quad k = 0, 1, \ldots, N - 1 \]

where \( x_k \): State, \( u_k \): Control chosen from some set \( U_k(x_k) \)

Cost function:

\[ g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \]

For given initial state \( x_0 \), minimize over control sequences \( \{u_0, \ldots, u_{N-1}\} \)

\[ J(x_0; u_0, \ldots, u_{N-1}) = g_N(x_N) + \sum_{k=0}^{N-1} g_k(x_k, u_k) \]

Optimal cost function \( J^*(x_0) = \min_{u_k \in U_k(x_k)} \sum_{k=0}^{N-1} J(x_0; u_0, \ldots, u_{N-1}) \)
Nodes correspond to states $x_k$

Each arc corresponds to a state-control pair $(x_k, u_k)$ [start node is $x_k$; end node is $x_{k+1} = f_k(x_k, u_k)$]

An arc corresponding to $(x_k, u_k)$ has a cost $g_k(x_k, u_k)$.

The cost to optimize is the sum of the arc costs from the initial node/state $x_0$ to a terminal node $t$.

The problem is equivalent to finding a minimum cost/shortest path from $x_0$ to $t$. 
A Combinatorial Example: The $N$ Queens Problem

Starting Position
Root Node $s$

Cost = 0

Dead-End Position

Cost = 1

Cost = 0

Artificial Terminal Node $t$
At state $x_k$, for every pair $(x_k, u_k)$, $u_k \in U_k(x_k)$, we generate a Q-factor

$$\tilde{Q}_k(x_k, u_k) = g_k(x_k, u_k) + H_{k+1}(f_k(x_k, u_k))$$

using the base heuristic [$H_{k+1}(x_{k+1})$ is the heuristic cost starting from $x_{k+1}$]

- We select the control $u_k$ with minimal Q-factor
- We move to next state $x_{k+1}$, and continue
- Multistep lookahead versions
- An important question: Is rollout cost improving? (Performs no worse than the base heuristic, from $x_0$)
A Multivehicle Routing Example

Base heuristic
Move each vehicle one step towards its closest task

Base heuristic moves both vehicles to node 4 and moves them together after that

Rollout operation at each stage, given the current pair of vehicle positions
- Consider all the possible pairs of moves from the current position
- Run the base heuristic from each pair
- Select the move of min total vehicle moves
- Rollout finds the optimal solution (in this example). A total of 6 moves compared with 10 for the base heuristic.
An Example: Search for an $N$-Arc Breakthrough Path in a Tree (e.g., Search Through a Maze)

Greedy base heuristic: If one arc is free use it; if both arcs are free use the right arc

- Complexity of the DP algorithm is $O(N2^N)$ (size of tree grows exponentially)
- Complexity of the greedy and rollout algorithms is $O(N)$ and $O(N^2)$, respectively
- Assuming arcs are blocked with given probability, the rollout algorithm has $O(N)$ times higher probability of breakthrough; see the textbook and the cited literature.
- This is qualitatively typical: Rollout improves performance of base heuristic substantially at the expense of polynomial amount of extra computation.
Cost improvement is not automatic: Special conditions must hold to guarantee that the rollout policy has no worse performance than the base heuristic.

Two such conditions are sequential consistency and sequential improvement.

The base heuristic is sequentially consistent if at a given state it chooses control that depends only on that state (and not on how we got to that state).

- If the heuristic generates the sequence
  \[ \{x_k, x_{k+1}, \ldots, x_N\} \]
  starting from state \(x_k\), it also generates the sequence
  \[ \{x_{k+1}, \ldots, x_N\} \]
  starting from state \(x_{k+1}\).

- The base heuristic is sequentially consistent if and only if it can be implemented with a legitimate DP policy \(\{\mu_0, \ldots, \mu_{N-1}\}\).

- “Greedy” heuristics are sequentially consistent (e.g., nearest neighbor for TSP).

- We will focus on a less restrictive condition: sequential improvement.
Implies cost improvement: \((\text{Cost of Rollout Policy}) \leq (\text{Cost of Base Heuristic})\)

- **Sequential improvement definition:** Best heuristic Q-factor \(\leq\) Heuristic cost, i.e.,

\[
\min_{u_k \in U_k(x_k)} \left[ g_k(x_k, u_k) + H_{k+1}(f_k(x_k, u_k)) \right] \leq H_k(x_k), \quad \text{for all } x_k
\]

where \(H_k(x_k): \text{cost of the trajectory generated by the heuristic starting from } x_k\)

- **Justification:** Rollout, upon reaching \(\tilde{x}_k\), has obtained a “current” trajectory \(R_k\).

Sequential improvement implies: \(\text{Cost of } R_k \geq \text{Cost of } R_{k+1}\)

- **Thus the current trajectory cannot get worse.** Since \(R_0\) corresponds to the base heuristic, \(R_N\) corresponds to the rollout, \(\text{Cost of } R_0 \geq \text{Cost of } R_N\)

- **Note that sequential consistency \(\rightarrow\) sequential improvement**
Base heuristic: Nearest neighbor (sequentially consistent and sequentially improving)

Cost of $R_0 \geq$ Cost of $R_1 \geq$ Cost of $R_2$

Matrix of Intercity Travel Costs

Initial State $x_0$

Terminal State $t$
A Fifteen-Minute Break

All our lectures will have a 15-minute break, somewhere in the middle
Catch our breath and think about issues relating to the first half of the lecture.
A short discussion/questions/answers period will follow each break.
Simplified Rollout Algorithm - Assuming Sequential Improvement

Simplified algorithm: Instead of control w/ minimal Q-factor, use any control with Q-factor \( \leq \) heuristic cost \( H_k(x_k) \)

- When at \( x_k \), choose as rollout control any \( \tilde{u}_k = \tilde{\mu}_k(x_k) \) such that

\[
g_k(x_k, \tilde{u}_k) + H_{k+1}(f_k(x_k, \tilde{u}_k) \leq H_k(x_k),
\]

where \( H_k(x_k) \) is the cost of the trajectory generated by the heuristic from \( x_k \).

- Can focus on a small subset of “promising” controls (save lots of computation)

Cost improvement for the simplified algorithm:

Let the rollout policy under the simplified algorithm be \( \tilde{\pi} = \{\tilde{\mu}_0, \ldots, \tilde{\mu}_{N-1}\} \), and let \( J_{k,\tilde{\pi}}(x_k) \) denote its cost starting from \( x_k \). Then for all \( x_k \) and \( k \), \( J_{k,\tilde{\pi}}(x_k) \leq H_k(x_k) \).

Proof: Again, the current trajectory cannot get worse,

\[
H_0(x_0) = \text{Cost of } R_0 \geq \cdots \geq \text{Cost of } R_k \geq \text{Cost of } R_{k+1} \geq \cdots \geq \text{Cost of } R_N
\]
Consider combining several heuristics in the context of rollout

- The idea is to construct a superheuristic, which runs all the heuristics at each state encountered, and selects the best out of the trajectories produced.
- The superheuristic can be viewed as the base heuristic for a rollout algorithm.
- It can be verified using the definitions, that if all the heuristics are sequentially improving, the same is true for the superheuristic.

**Proof:** Write the sequential improvement condition for each of the $M$ heuristics

$$\min_{u_k \in U_k(x_k)} \tilde{Q}_k^m(x_k, u_k) \leq H_k^m(x_k), \quad m = 1, \ldots, M,$$

and all $x_k$ and $k$, where $\tilde{Q}_k^m(x_k, u_k)$ and $H_k^m(x_k)$ are Q-factors and heuristic costs that correspond to the $m$th heuristic. By taking minimum over $m$, and interchanging the order of the minimization

$$\min_{u_k \in U_k(x_k)} \underbrace{\min_{m=1,\ldots,M} \tilde{Q}_k^m(x_k, u_k)}_{\text{Superheuristic Q-factor}} \leq \underbrace{\min_{m=1,\ldots,M} H_k^m(x_k)}_{\text{Superheuristic cost}},$$

which is the sequential improvement condition for the superheuristic.
The optimal trajectory \((x_0, u_0^*, x_1^*, u_1^*, x_2^*)\).

Assume the heuristic produces \((u_0^*, u_1^*)\) at \(x_0\), and \(\tilde{u}_1\) at \(x_1^*\).

Rollout uses the base heuristic to construct a trajectory starting from \(x_1^*\) and \(\tilde{x}_1\).

Then (Q-factor of \(u_0^*) > (Q\text{-factor of } \tilde{u}_0)\). So the rollout algorithm selects \(\tilde{u}_0\), and moves to a nonoptimal next state \(\tilde{x}_1 = f_0(x_0, \tilde{u}_0)\).

Thus in the absence of sequential improvement, the rollout can deviate from an already available good “current” trajectory.

This suggests a possible remedy: Follow the best “current” trajectory found even if rollout suggests following a different (but inferior) trajectory.
Fortified Rollout: Restores Cost Improvement for Base Heuristics that are not Sequentially Improving

Idea: At each step, follow the best trajectory computed thus far

- At state $x_k$: In addition to the permanent rollout trajectory $\overline{P}_k = \{x_0, u_0, \ldots, u_{k-1}, x_k\}$, also store a tentative best trajectory
  \[
  \overline{T}_k = \{x_0, \ldots, x_k, \overline{u}_k, \overline{x}_{k+1}, \overline{u}_{k+1}, \ldots, \overline{u}_{N-1}, \overline{x}_N\}
  \]

  $\overline{T}_k$ is the best end-to-end trajectory computed up to stage $k$

- We reject the minimum Q-factor choice $\tilde{u}_k$ if its complete trajectory is more costly than the current tentative best; otherwise we accept $\tilde{u}_k$, and update the tentative best trajectory.
Illustration of Fortified Algorithm

At $x_0$, the fortified rollout stores as initial tentative best trajectory the unique optimal trajectory $(x_0, u_0^*, x_1^*, u_1^*, x_2^*)$ generated by the base heuristic.

In the first rollout step, it computes the Q-factors of $u_0^*$ and $\tilde{u}_0$ by running the heuristic from $x_1^*$ and $\tilde{x}_1$.

Even though the rollout prefers $\tilde{u}_0$ to $u_0^*$, it discards $\tilde{u}_0$ in favor of $u_0^*$, which is dictated by the tentative best trajectory.

It then sets the permanent trajectory to $(x_0, u_0^*, x_1^*)$ and keeps the tentative best trajectory unchanged to $(x_0, u_0^*, x_1^*, u_1^*, x_2^*)$. 

Model-Free Rollout with an Expert for the General Discrete Optimization

\[ \min_{u_0 \in U_0, \ldots, u_{N-1} \in U_{N-1}} G(u_0, \ldots, u_{N-1}) \]

- Assume we do not know \( G \), and/or the constraint sets \( U_k \)
- Instead we have a base heuristic, which given a partial solution \( (u_0, \ldots, u_k) \), outputs all next controls \( \tilde{u}_{k+1} \), and generates from each a complete solution

\[ S_k(u_0, \ldots, u_k, \tilde{u}_{k+1}) = (u_0, \ldots, u_k, \tilde{u}_{k+1}, \ldots, \tilde{u}_{N-1}) \]

- Also, we have a human or software “expert" that can rank any two complete solutions without assigning numerical values to them.
- Deterministic rollout can be applied to this problem; we have all we need.
Given a sequence of nucleotides (molecules of “types" A,C,G,U), “fold" it in an “interesting" way (introduce pairings that result in an “interesting" structure).

Make a pairing decision at each nucleotide in sequence (open, close, do nothing).

Base heuristic: Given a partial folding, generates a complete folding (this is the partial folding software).

Two complete foldings can be compared by the expert software.

There is no explicit cost function here (it is internal to the expert software).
About the Next Lecture

We will cover:
- Rollout with multistep lookahead
- Rollout for constrained problems
- Applications in integer programming

Homework (due in two weeks): Exercise 1.3 (spiders and flies)

About your project:
- Read the guidelines for the term paper, posted at canvas
- Send us email for clarifications and questions
- Please send us by the end of the spring break a one-page-or-less proposal about your term paper, be it a read-and-report type or a mini-research project