Topics in Reinforcement Learning: Rollout and Approximate Policy Iteration

ASU, CSE 691, Spring 2020

Dimitri P. Bertsekas
dbertsek@asu.edu

Lecture 4

Outline

Recall Approximation in Value Space and the Three Approximations

Approximate Q-Factor $\tilde{Q}_k(x_k, u_k)$

Min Approximation

First Step

“Future”

$$\boxed{\text{At } x_k} \rightarrow \min_{u_k} E\Big\{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(x_{k+1}) \Big\}$$

$E\{\cdot\}$ Approximation

Cost-to-Go Approximation

ONE-STEP LOOKAHEAD

At State x_k

DP minimization

First ℓ Steps

“Future”

$$\min_{u_k, \mu_{k+1}, \ldots, \mu_{k+\ell-1}} E\left\{ g_k(x_k, u_k, w_k) + \sum_{m=k+1}^{k+\ell-1} g_k\big(x_m, \mu_m(x_m), w_m\big) + \tilde{J}_{k+\ell}(x_{k+\ell}) \right\}$$

Lookahead Minimization

Cost-to-go Approximation

MULTISTEP LOOKAHEAD

The Pure Form of Rollout: For a Suboptimal Base Policy π, Use $\tilde{J}_{k+\ell}(x_{k+\ell}) = J_{k+\ell,\pi}(x_{k+\ell})$

At State x_k
DP minimization

$$\min_{u_k,\mu_{k+1},\dots,\mu_{k+\ell-1}} E\left\{\overbrace{g_k(x_k,u_k,w_k)+\sum_{i=k+1}^{k+\ell-1} g_i\big(x_i,\mu_i(x_i),w_i\big)}^{\text{First } \ell \text{ Steps}} + \overbrace{\tilde{J}_{k+\ell}(x_{k+\ell})}^{\text{“Future”}}\right\}$$

Rollout Control $\tilde{u}_k$
Rollout Policy $\tilde{\mu}_k$

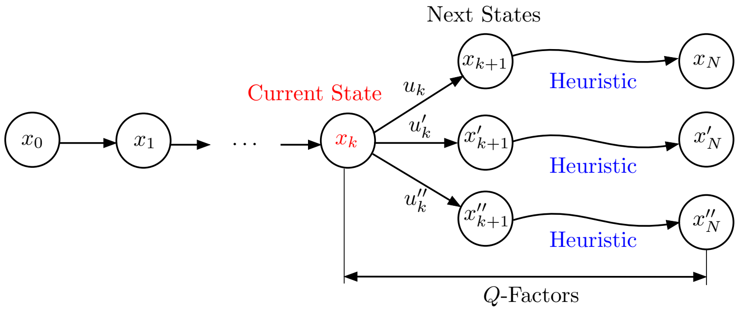
Lookahead Minimization

Base Policy Cost

Use a suboptimal/heuristic policy at the end of limited lookahead

- The suboptimal policy is called base policy.
- The lookahead policy is called rollout policy.
- The aim of rollout is policy improvement (i.e., rollout policy performs better than the base policy); true under some assumptions. In practice: good performance, very reliable, very simple to implement.
- Rollout in its “standard" forms involves simulation and on-line implementation.
- The simulation can be prohibitively expensive (so further approximations may be needed); particularly for stochastic problems and multistep lookahead.

Deterministic Rollout: At x_{k+1}, Use a Heuristic with Cost $H_{k+1}(x_{k+1})$



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

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

  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.
- The scheme is very general: The heuristic can be anything (stage- or state-dependent)! May not necessarily correspond to a policy.

Criteria for Cost Improvement of a Rollout Algorithm - Sequential Consistency, and Sequential Improvement

Remember:

Any heuristic (no matter how inconsistent or silly) is in principle admissible to use as base heuristic.

So it is important to properties guaranteeing 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 and only if it can be implemented with a legitimate DP policy $\{\mu_0, \ldots, \mu_{N-1}\}$.
- Example: Greedy heuristics tend to be sequentially consistent.
- A sequentially consistent heuristic is also sequentially improving. See next slide.
- We will see that any heuristic can be “fortified" to become sequentially improving.

Policy Improvement for Sequentially Improving Heuristics

Sequential improvement holds if for all x_k (Best heuristic Q-factor $\leq$ Heuristic cost):

$$\min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + H_{k+1}\big(f_k(x_k, u_k)\big)\Big] \leq H_k(x_k),$$

where $H_k(x_k)$ is the cost of the trajectory generated by the heuristic starting from x_k.
True for sequ. consistent heuristic [$H_k(x_k)$ =one of the Q-factors minimized on the left].

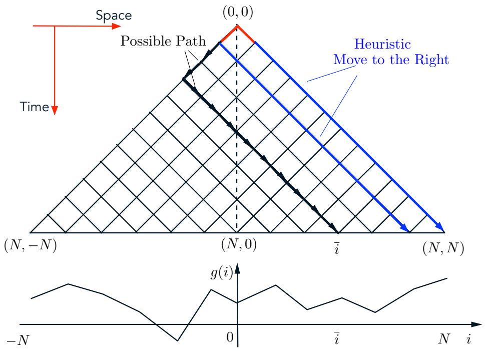
Cost improvement property for a sequentially improving base heuristic:

Let the rollout policy be $\tilde{\pi} = \{\tilde{\mu}_0, \dots, \tilde{\mu}_{N-1}\}$, 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 by induction: It holds for $k = N$, since $J_{N,\tilde{\pi}} = H_N = g_N$. Assume that it holds for index $k+1$. We show that it holds for index k:

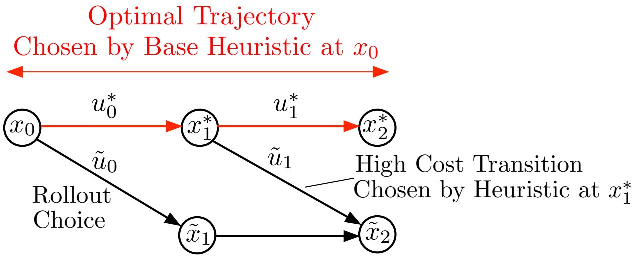
$$\begin{aligned}
J_{k,\tilde{\pi}}(x_k) &= g_k\big(x_k, \tilde{\mu}_k(x_k)\big) + J_{k+1,\tilde{\pi}}\Big(f_k\big(x_k, \tilde{\mu}_k(x_k)\big)\Big) \quad \text{(by DP equation)}\\
&\leq g_k\big(x_k, \tilde{\mu}_k(x_k)\big) + H_{k+1}\big(f_k(x_k, \tilde{\mu}_k(x_k))\big) \quad \text{(by induction hypothesis)}\\
&= \min_{u_k \in U_k(x_k)} \Big[g_k(x_k, u_k) + H_{k+1}\big(f_k(x_k, u_k)\big)\Big] \quad \text{(by definition of rollout)}\\
&\leq H_k(x_k) \quad \text{(by sequential improvement condition)}
\end{aligned}$$

A Working Break: Challenge Question



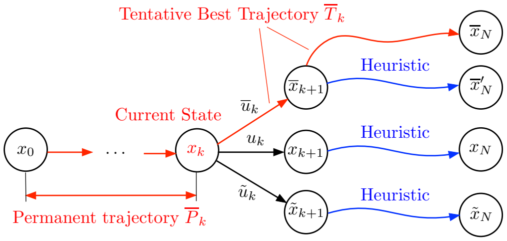
- Walk on a line of length $2N$ starting at position 0. At each of N steps, move one unit to the left or one unit to the right.
- Objective is to land at a position i of small cost $g(i)$ after N steps.
- Question: Consider a base heuristic that takes steps to the right only. Is it sequentially consistent or sequentially improving? How will the rollout perform compared to the base heuristic?
- Compare with a superheuristic/combination of two heuristics: 1) Move only to the right, and 2) Move only to the left. Base heuristic chooses the path of best cost.

A Counterexample to Cost Improvement



- Suppose at x_0 there is a unique optimal trajectory $(x_0, u_0^*, x_1^*, u_1^*, x_2^*)$.
- Suppose the base heuristic produces this optimal trajectory starting at x_0.
- Rollout uses the base heuristic to construct a trajectory starting from x_1^* and $\tilde{x}_1$.
- Suppose the heuristic's trajectory starting from x_1^* is “bad" (has high cost).
- Then the rollout algorithm rejects the optimal control u_0^* in favor of the other control $\tilde{u}_0$, and moves to a nonoptimal next state $\tilde{x}_1 = f_0(x_0, \tilde{u}_0)$.
- So without some safeguard, the rollout can deviate from an already available good trajectory.
- This suggests a possible remedy: Follow the currently best trajectory if rollout suggests following an inferior trajectory.

Fortified Rollout: Restores Cost Improvement for Base Heuristics that are not Sequentially Improving



- Upon reaching state x_k it stores the permanent trajectory

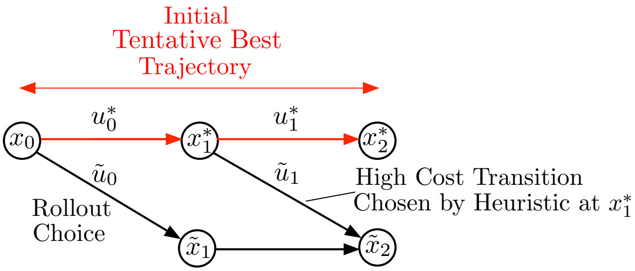
$$\overline{P}_k = \{x_0, u_0, \ldots, u_{k-1}, x_k\}$$

that has been constructed up to stage k, and also stores a tentative best trajectory

$$\overline{T}_k = \{x_k, \overline{u}_k, \overline{x}_{k+1}, \overline{u}_{k+1}, \ldots, \overline{u}_{N-1}, \overline{x}_N\}$$

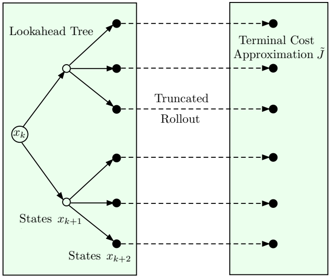
- The tentative best trajectory is such that $\overline{P}_k \cup \overline{T}_k$ is the best complete trajectory computed up to stage k of the algorithm.
- Initially, $\overline{P}_0 = \{x_0\}$ and $\overline{T}_0$ is the trajectory computed by the base heuristic from x_0.
- At each step, fortified rollout follows the best trajectory computed thus far.

Illustration of Fortified Algorithm



- 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 at x_0.
- Starting at x_0^* it runs the heuristic from x_1^* and $\tilde{x}_1$, and (despite that the heuristic prefers $\tilde{x}_1$ to x_1^*) it discards the control $\tilde{u}_0$ 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 the tentative best trajectory to (x_1^*, u_1^*, x_2^*).

Multistep Deterministic Rollout with Terminal Cost Approximation



- Terminal approximation saves computation but cost improvement is lost.
- We can prove cost improvement, assuming sequential consistency and a special property of the terminal cost function approximation that resembles sequential improvement (more on this when we discuss infinite horizon rollout).
- It is not necessarily true that longer lookahead leads to improved performance; but usually true (similar counterexamples as in the last lecture).
- It is not necessarily true that increasing the length of the rollout leads to improved performance (some examples indicate this). Moreover, long rollout is costly.
- Experimentation with length of rollout and terminal cost function approximation are recommended.

Stochastic Rollout [Uses $\tilde{J}_{k+\ell}(x_{k+\ell}) = J_{k+\ell,\pi}(x_{k+\ell})$ and MC Simulation]

At State x_k
DP minimization

First ℓ Steps

"Future"

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

Rollout Control $\tilde{u}_k$
Rollout Policy $\tilde{\mu}_k$

Lookahead Minimization

Base Policy Cost

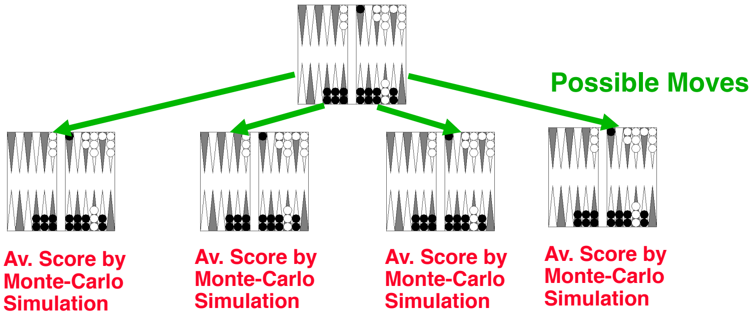
Consider the pure case (no truncation, no terminal cost approximation)

- Assume that the base heuristic is a legitimate policy $\pi = \{\mu_0, \dots, \mu_{N-1}\}$ (i.e., is sequentially consistent, in the context of deterministic problems).
- Let $\tilde{\pi} = \{\tilde{\mu}_0, \dots, \tilde{\mu}_{N-1}\}$ be the rollout policy. Then cost improvement is obtained

$$J_{k,\tilde{\pi}}(x_k) \le J_{k,\pi}(x_k), \qquad \text{for all } x_k \text{ and } k.$$

- Essentially identical induction proof as for the sequentially improving case (see the text).

Backgammon Example



Possible Moves
Av. Score by Monte-Carlo Simulation
Av. Score by Monte-Carlo Simulation
Av. Score by Monte-Carlo Simulation
Av. Score by Monte-Carlo Simulation

- Announced by Tesauro in 1996.
- Truncated rollout with cost function approximation provided by TD-Gammon (a 1991 famous program by Tesauro, involving a neural network trained by a form of approximate policy iteration that uses “Temporal Differences").
- Plays better than TD-Gammon, and better than any human.
- Too slow for real-time play (without parallel hardware), due to excessive simulation time.

Monte Carlo Tree Search - Motivation

We assumed equal effort for evaluation of Q-factors of all controls at a state x_k

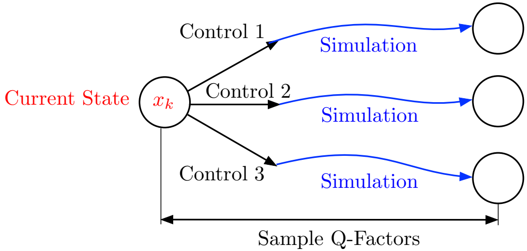
Drawbacks:

- The trajectories may be too long because the horizon length N is large (or infinite, in an infinite horizon context).
- Some controls u_k may be clearly inferior to others and may not be worth as much sampling effort.
- Some controls u_k that appear to be promising may be worth exploring better through multistep lookahead.

Monte Carlo tree search (MCTS) is a “randomized" form of lookahead

- MCTS aims to trade off computational economy with a hopefully small risk of degradation in performance.
- It involves adaptive simulation (simulation effort adapted to the perceived quality of different controls).
- Aims to balance exploitation (extra simulation effort on controls that look promising) and exploration (adequate exploration of the potential of all controls).

Monte Carlo Tree Search - Adaptive Simulation



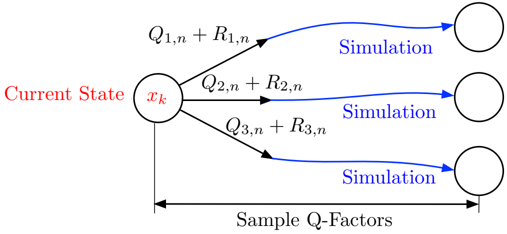
MCTS provides an economical sampling policy to estimate the Q-factors

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

Assume that $U_k(x_k)$ contains a finite number of elements, say $i = 1, \ldots, m$

- After the nth sampling period we have $Q_{i,n}$, the empirical mean of the Q-factor of each control i (total sample value divided by total number of samples corresponding to i). We view $Q_{i,n}$ as an exploitation index.
- How do we use the estimates $Q_{i,n}$ to select the control to sample next?

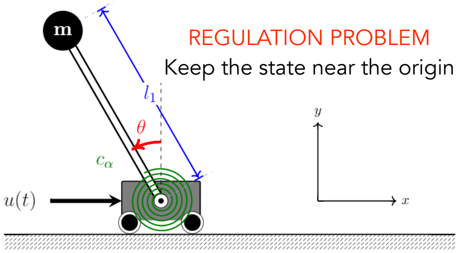
MCTS Based on Statistical Tests



A good sampling policy balances exploitation (sample controls that seem most promising, i.e., a small $Q_{i,n}$) and exploration (sample controls with small sample count).

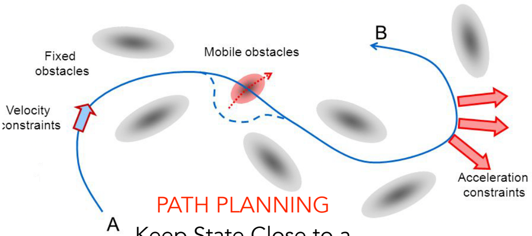
- A popular strategy: Sample next the control i that minimizes the sum $Q_{i,n} + R_{i,n}$ where $R_{i,n}$ is an exploration index.
- $R_{i,n}$ is based on a confidence interval formula and depends on the sample count s_i of control i (which comes from analysis of multiarmed bandit problems).
- The UCB rule (upper confidence bound) sets $R_{i,n} = -c\sqrt{\log n/s_i}$, where c is a positive constant, selected empirically (values $c \approx \sqrt{2}$ are suggested, assuming that $Q_{i,n}$ is normalized to take values in the range $[-1, 0]$).
- MCTS with UCB rule has been extended to multistep lookahead ... but AlphaZero has used a different (semi-heuristic) rule.

Classical Control Problems - Infinite Control Spaces



REGULATION PROBLEM

Keep the state near the origin

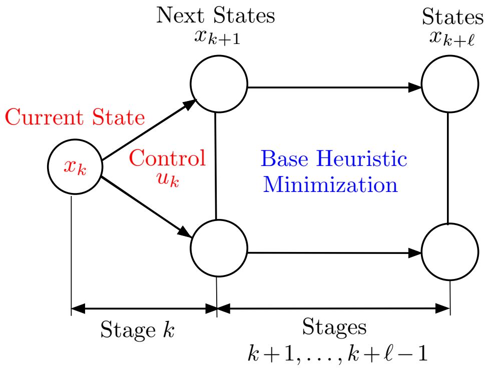


PATH PLANNING

Keep State Close to a Trajectory

Need to deal with state and control constraints; linear-quadratic is often inadequate

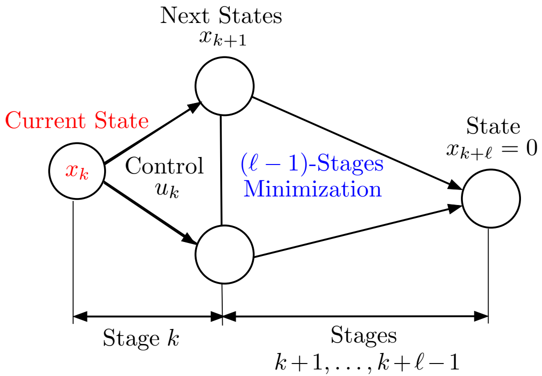
On-Line Rollout for Deterministic Infinite-Spaces Problems



Suppose the control space is infinite

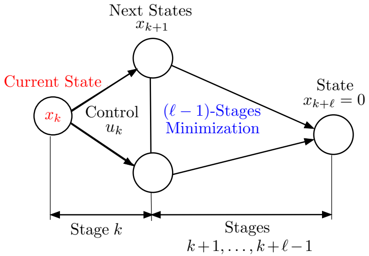
- One possibility is discretization of $U_k(x_k)$; but excessive number of Q-factors.
- Another possibility is to use optimization heuristics that look $(\ell - 1)$ steps ahead.
- Seemlessly combine the kth stage minimization and the optimization heuristic into a single ℓ-stage deterministic optimization.
- Can solve it by nonlinear programming/optimal control methods (e.g., quadratic programming, gradient-based).

Model Predictive Control for Regulation Problems



- System: $x_{k+1} = f_k(x_k, u_k)$
- Cost per stage: $g_k(x_k, u_k) \geq 0$, the origin 0 is cost-free and absorbing.
- State and control constraints: $x_k \in X_k$, $u_k \in U_k(x_k)$ for all k
- At x_k solve an ℓ-step lookahead version of the problem, requiring $x_{k+\ell} = 0$ while satisfying the state and control constraints.
- If $\{\tilde{u}_k, \ldots, \tilde{u}_{k+\ell-1}\}$ is the control sequence so obtained, apply $\tilde{u}_k$.

Relation to Rollout



- It is rollout with base heuristic the $(\ell - 1)$-step min (0 is cost-free and absorbing).
- This heuristic is sequentially improving (not sequentially consistent), i.e.,

$$\min_{u_k \in U_k(x_k)} \left[g_k(x_k, u_k) + H_{k+1}\big(f_k(x_k, u_k)\big) \right] \leq H_k(x_k)$$

  because (opt. cost to reach 0 in ℓ steps) $\leq$ (opt. cost to reach 0 in $\ell - 1$ steps)
- Sequential improvement implies “stability": $\sum_{k=0}^{\infty} g_k(x_k, u_k) \leq H_0(x_0) < \infty$, where $\{x_0, u_0, x_1, u_1, \ldots\}$ is the state and control sequence generated by MPC.
- Major issue: How do we know that the optimization of the base heuristic is solvable (e.g., there exists ℓ such that we can drive $x_{k+\ell}$ to 0 for all $x_k \in X_k$ while observing the state and control constraints). Methods of reachability of target tubes can be used for this (see the text).

About the Next Lecture

We will cover:

- Rollout for multiagent problems
- Constrained rollout
- Discrete optimization problems and rollout

PLEASE READ AS MUCH OF THE REMAINDER OF CHAPTER 2 AS YOU CAN

PLEASE DOWNLOAD THE LATEST VERSION