Reinforcement Learning and Optimal Control

ASU, CSE 691, Winter 2020

Dimitri P. Bertsekas
dimitrib@mit.edu

Lecture 5
1. Multiagent Rollout

2. Deterministic Problem Rollout with Constraints

3. Combinatorial Applications - Examples
The Pure Form of Rollout: For a Suboptimal Base Policy $\pi$, Use

$$\tilde{J}_{k+\ell}(x_{k+\ell}) = J_{k+\ell,\pi}(x_{k+\ell})$$

Min Approximation

At $x_k$

First Step

\[ \min_{u_k} E \{ g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}(x_{k+1}) \} \]

“Future”

$E\{\cdot\}$ Approximation

Truncated Rollout Approximate Base Policy Cost

Policy improvement property (where no truncation is used):

$$J_{k,\tilde{\pi}}(x_k) \leq J_{k,\pi}(x_k), \quad \text{for all } x_k \text{ and } k$$

The key issue in the practical application of rollout: Too much computation

- If the problem is deterministic, the computation is greatly reduced (no Monte Carlo)
- Another computational bottleneck: Large control spaces, e.g., the multiagent case, where controls have many components,

$$u_k = (u_k^1, \ldots, u_k^m)$$
An equivalent reformulation - “Unfolding” the control action

- The control space is simplified at the expense of \( m - 1 \) additional layers of states, and corresponding \( m - 1 \) cost-to-go functions

\[
J_1^k(x_k, u_k^1), J_2^k(x_k, u_k^1, u_k^2), \ldots, J_{m-1}^k(x_k, u_k^1, \ldots, u_k^{m-1})
\]

- Multiagent or one-component-at-a-time rollout is just standard rollout for the reformulated problem.
- The increase in size of the state space does not adversely affect rollout.
- The cost improvement property is maintained.
- Complexity reduction: The one-step lookahead branching factor is reduced from \( n^m \) to \( nm \), where \( n \) is the number of possible choices for each component \( u_k^i \).
- Base policy for \( u_k^i \) may depend only on \( x_k \) or include dependence on \( u_k^{i-1}, u_k^{i-2}, \ldots \).
- $n$ vehicles move along the arcs of a given graph.
- Some of the nodes of the graph include a task to be performed by the vehicles. Each task will be performed only once.
- Find a route for each vehicle so that all the tasks are collectively performed by the vehicles in minimum time.
- Cost function choice: For each stage there is a cost of one unit for each task that is pending at the end of the stage.
- Total cost: The number of stages that a task is pending, summed over the tasks. What is the state? What is the control? Why is this good multiagent candidate?
Base heuristic:

- Each vehicle makes a move towards the pending task that is closest (in terms of number of moves needed to reach it).
- The vehicles make their selection independently one-at-a-time in the order 1, \ldots, n, and take into account the tasks that have already been performed.

What is the solution produced by the base heuristic?
What is the solution produced by the one-vehicle-at-a-time rollout?
Do we get cost improvement? What is the intuition?
Constrained Deterministic Rollout

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 \(x_{k+1}\)].

- Select \(u_k\) with minimal Q-factor, move to next state \(x_{k+1}\) and continue.
- **What if there are constraints**, i.e., future control choices are constrained by past control choices?
- **Base heuristic and rollout should be modified** (e.g., avoid controls that compromise feasibility of future controls).
Examples of constraints

- **Vehicle capacity constraints** (limit on how many tasks some vehicles can perform).
- **Vehicle specializations** (some tasks can be performed only by some of the vehicles).
- **Time windows** (some tasks must be performed within specified time intervals).
Consider a deterministic optimal control problem with system \( x_{k+1} = f_k(x_k, u_k) \).

A complete trajectory is a sequence

\[
T = (x_0, u_0, x_1, u_1, \ldots, u_{N-1}, x_N)
\]

We consider the (very general) problem

\[
\min_{T \in C} G(T)
\]

where \( G \) is a given cost function and \( C \) is a given constraint set of trajectories.

We transform to an unconstrained problem in order to apply rollout ideas

- Redefine the state to be the partial trajectory

  \[
y_k = (x_0, u_0, x_1, \ldots, u_{k-1}, x_k)
  \]

- Partial trajectory evolves according to a redefined system equation:

  \[
y_{k+1} = (y_k, u_k, f_k(x_k, u_k))
  \]

The problem becomes to minimize \( G(y_N) \) subject to the constraint \( y_N \in C \).
Given \( \tilde{y}_k = \{\tilde{x}_0, \tilde{y}_0, \tilde{x}_1, \tilde{u}_1, \ldots, \tilde{u}_{k-1}, \tilde{x}_k \} \) consider all controls \( u_k \) and corresponding next states \( x_{k+1} \).

Extend \( \tilde{y}_k \) to obtain the partial trajectories \( y_{k+1} = (\tilde{y}_k, u_k, x_{k+1}) \).

Run the base heuristic from each \( y_{k+1} \) to obtain the partial trajectory \( R(y_{k+1}) \).

Join the partial trajectories \( y_{k+1} \) and \( R(y_{k+1}) \) to obtain complete trajectories denoted by \( T_k(\tilde{y}_k, u_k) = (\tilde{y}_k, u_k, R(y_{k+1})) \).

Find the set of controls \( U_k(\tilde{y}_k) \) for which \( T_k(\tilde{y}_k, u_k) \) is feasible, i.e., \( T_k(\tilde{y}_k, u_k) \in C \).

Choose the control \( \tilde{u}_k \in U_k(\tilde{y}_k) \) according to the minimization

\[
\tilde{u}_k \in \arg \min_{u_k \in U_k(\tilde{y}_k)} G(T_k(\tilde{y}_k, u_k))
\]
The notions of sequential consistency and sequential improvement apply. Part of their definition includes that the set of “feasible" controls $U_k(\tilde{y}_k)$ is nonempty for all $k$; see the notes.

There is a fortified version (follows the best feasible trajectory). Has the cost improvement property, assuming the base heuristic generates a feasible trajectory starting from the initial condition $\tilde{y}_0 = x_0$.

There is a multiagent version that uses one-control-component-at-a-time minimization.

Additional variants are possible; see the notes.
- Base heuristic moves each vehicle (one-at-a-time) to the closest pending task.
- What is the first move of vehicle 1 chosen by base heuristic and by rollout?
- What is the solution found by base heuristic?
- What is the solution found by rollout?
This is a special case of the constrained deterministic optimal control problem where each state $x_k$ can only take a single value.

A very broad range of problems, e.g., combinatorial, integer programming, etc.

Solution by constrained rollout applies. Provides entry point to the use of neural nets in discrete optimization through approximation in policy space.

Other methods: local/random search, genetic algorithms, etc. Rollout is different.
Place facilities at some locations to serve the needs of $M$ “clients.”

Client $i = 1, \ldots, M$ has a demand $d_i$ for services that may be satisfied at a location $j = 1, \ldots, N$ at a cost $a_{ij}$ per unit.

A facility placed at location $j$ has capacity $c_j$ and costs $b_j$. Here $u_j \in \{0, 1\}$, with $u_j = 1$ if a facility is placed at $j$.

Problem: Minimize $\sum_{i=1}^{M} \sum_{j=1}^{N} a_{ij} x_{ij} + \sum_{j=1}^{N} b_j u_j$ subject to total demand satisfaction constraints.

Note: If the placement variables $u_j$ are known, the remaining problem is easily solvable (it is a linear “transportation" problem).
Consider placements one location at a time.

**Stage** = Placement decision at a single location ($N$ stages). **State** at time $k$ = The placement choices at locations $1, \ldots, k$. **Control** = Choice between 1 (place) or 0 (not place) for the next facility. **What are the constraints? Is this multiagent?**

**Base heuristic:** Having fixed $u_1, \ldots, u_k$, place a facility in every remaining location.

**Rollout:** Having fixed $u_1, \ldots, u_k$, compare two possibilities:

- Set $u_{k+1} = 1$ (place facility at location $k + 1$), set $u_{k+2} = \cdots = u_N = 1$ (as per the base heuristic), and solve the remaining problem.
- Set $u_{k+1} = 0$ (don’t place facility at location $k + 1$), set $u_{k+2} = \cdots = u_N = 1$ (as per the base heuristic), and solve the remaining problem.

Select $u_{k+1} = 1$ or $u_{k+1} = 0$ depending on which yields feasibility and min cost.

**Linear transportation problems** can be solved with the auction algorithm.
Given a sequence of nucleotides, “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 **critic software**.

**Note:** There is no explicit cost function here (it is internal to the critic software).
The performance of a job $j$ requires a single machine $\ell$ and a single worker $w$.

There is a given cost $a_{j\ell w}$ corresponding to the triplet $(j, \ell, w)$.

Given a set of $m$ jobs, a set of $m$ machines, and a set of $m$ workers, we want to find a collection of $m$ job-machine-worker triplets that has minimum total cost.

A favorable case is when the cost has separable form $a_{j\ell w} = \beta_{j\ell} + \gamma_{\ell w}$.

Enforced separation heuristic:

- First solve an artificial 2-dimensional machines-to-workers assignment problem with costs $\gamma_{\ell w}$ derived from $a_{j\ell w}$, e.g., $\gamma_{\ell w} = \min_j a_{j\ell w}$ (the “optimistic” assignment costs).
- Next solve the 2-dimensional jobs-to-machines assignment problem with costs $\beta_{j\ell}$ specified by the machines-to-workers assignment and $a_{j\ell w}$.

2-D assignment problems are easy (using the auction algorithm; see the notes).
**Three-Dimensional Assignment Problem: Use of Rollout**

- **View as a constrained multistage problem.**
- Two stages; control at stage 1 = the jobs-to-machines assignment; control at stage 2 = the machines-to-workers assignment
- We view stage 1 assignment as “multiagent”: Assign one job at a time.
- We view stage 2 assignment as “single-agent”: Assign all machines at once optimally (given the stage 1 assignment).
- **Base heuristic:** Having fixed some stage 1 assignments, use enforced separation heuristic for the remaining stage 1 assignments, and the stage 2 assignment.
- **Rollout:** Fix each job assignment one-at-a-time using the base heuristic to compare all machine options.
About the Next Two Lectures

We will cover:

- Parametric approximation architectures.
- Neural networks and how we use them.
- Approximation in value space and policy space using neural nets.

PLEASE READ AS MUCH OF CHAPTER 3 AS YOU CAN
PLEASE DOWNLOAD THE LATEST VERSION