Topics in Reinforcement Learning: Rollout and Approximate Policy Iteration

ASU, CSE 691, Spring 2021

Links to Class Notes, Videolectures, and Slides at
http://web.mit.edu/dimitrib/www/RLbook.html

Dimitri P. Bertsekas
dbertsek@asu.edu

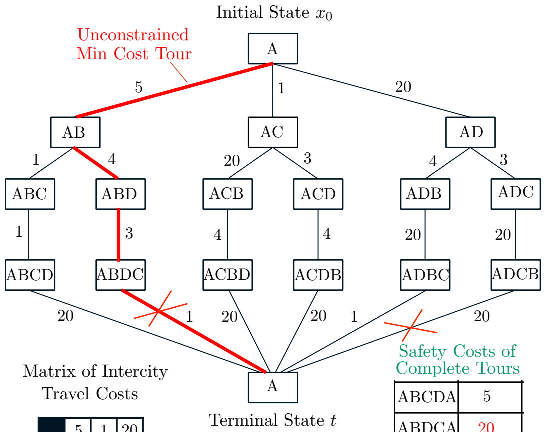
Lecture 7
Constrained Rollout, Rollout for Discrete Optimization, Minimax Rollout

Outline

1. Constrained Rollout for Deterministic Optimal Control

2. Discrete Optimization Applications

3. Rollout for Minimax Control



Matrix of Intercity Travel Costs

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

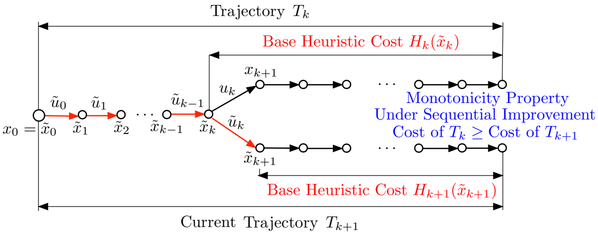
Constraint:
Tour Safety ≤ 10

Safety Costs of Complete Tours

ABCDA	5
ABDCA	20
ACBDA	4
ACDBA	3
ADBCA	1
ADCBA	15

Find a minimum cost tour subject to a safety constraint

Deterministic Rollout with Trajectory Constraint: Basic Idea



Review of the unconstrained rollout algorithm:

- Construct sequence of trajectories $\{T_0, T_1, \ldots, T_N\}$ with monotonically nonincreasing cost (assuming a sequential improvement condition).
- For each k, the trajectories $T_k, T_{k+1}, \ldots, T_N$ share the same initial portion $(x_0, \tilde{u}_0, \ldots, \tilde{u}_{k-1}, \tilde{x}_k)$.
- The base heuristic is used to generate candidate trajectories that correspond to the controls $u_k \in U_k(x_k)$.
- The next trajectory T_{k+1} is the candidate trajectory that has min cost.

To deal with a trajectory constraint $T \in C$, we discard all the candidate trajectories that violate the constraint, and we choose T_{k+1} to be the best of the remaining trajectories.

Deterministic Problems with Constraints: Definition

- Consider a deterministic optimal control problem with system $x_{k+1} = f_k(x_k, u_k)$.
- A complete trajectory is a sequence

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

- Problem:

$$\min_{T \in C} G(T)$$

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

State augmentation idea for rollout

- Redefine the state to be the partial trajectory

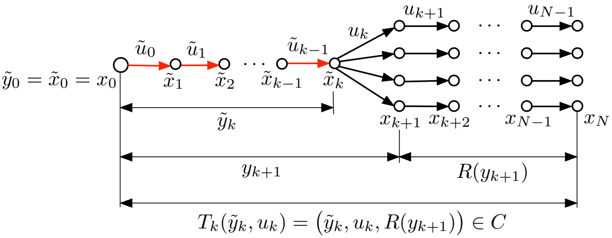
$$y_k = (x_0, u_0, x_1, \ldots, u_{k-1}, x_k)$$

- Partial trajectory evolves according to a redefined system equation:

$$y_{k+1} = \big(y_k, u_k, f_k(x_k, u_k)\big)$$

- The problem becomes to minimize $G(y_N)$ subject to the constraint $y_N \in C$.

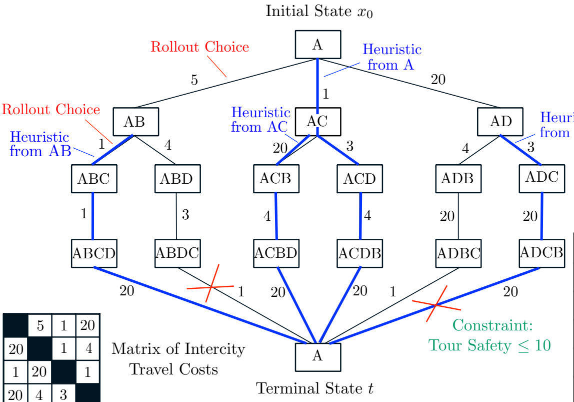
Rollout Algorithm - Partial Trajectory-Dependent Base Heuristic



- Given $\tilde{y}_k = \{\tilde{x}_0, \tilde{u}_0, \tilde{x}_1, \tilde{u}_1, \ldots, \tilde{u}_{k-1}, \tilde{x}_k\}$ consider all controls u_k and corresponding next states x_{k+1}.
- Extend $\tilde{y}_k$ to obtain the partial trajectories $y_{k+1} = \big(\tilde{y}_k, u_k, x_{k+1}\big)$, for $u_k \in U_k(x_k)$.
- 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 $T_k(\tilde{y}_k, u_k) = \big(\tilde{y}_k, u_k, R(y_{k+1})\big)$
- Find the set of controls $\tilde{U}_k(\tilde{y}_k)$ 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 $\tilde{u}_k \in \tilde{U}_k(\tilde{y}_k)$ according to the minimization

$$\tilde{u}_k \in \arg \min_{u_k \in \tilde{U}_k(\tilde{y}_k)} G\big(T_k(\tilde{y}_k, u_k)\big)$$

Constrained Traveling Salesman Example



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

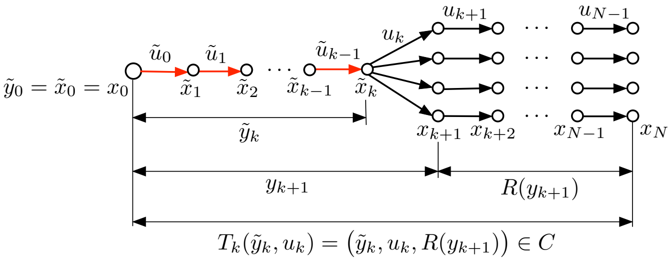
Matrix of Intercity Travel Costs

Safety Costs of Complete Tours

ABCDA	5
ABDCA	20
ACBDA	4
ACDBA	3
ADBCA	1
ADCBA	15

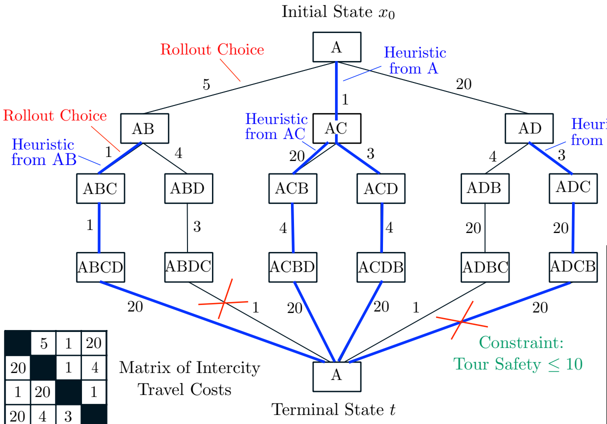
- Rollout at A: Considers partial tours AB, AC, and AD; Obtains the complete tours ABCDA, ACBDA, and ADCBA; Discards ADCBA as being infeasible; Compares ABCDA and ACBDA, finds ABCDA to have smaller cost, and selects AB.
- Rollout at AB: Considers the partial tours ABC and ABD; Obtains the complete tours ABCDA and ABDCA; Discards ABDCA as being infeasible; Selects the complete tour ABCDA.

Constrained Rollout Algorithm Properties



- The notions of sequential consistency and sequential improvement apply. Their definition includes that the set of "feasible" controls $\tilde{U}_k(\tilde{y}_k)$ is nonempty for all k.
- Sequential improvement condition: The min heuristic Q-factor over $\tilde{U}_k(\tilde{y}_k)$ is no larger than the heuristic cost at $\tilde{y}_k$ (see the notes).
- Fortified version (if sequential improvement does not hold; see the notes):
  - Maintains the "tentative best" trajectory, and follows it up to generating a better trajectory through rollout.
  - Has the cost improvement property, assuming the base heuristic generates a feasible trajectory starting from the initial condition $\tilde{y}_0 = x_0$.
- Multiagent version: Selects one-control-component-at-a-time (apply constrained rollout to the equivalent reformulation, i.e., the one with control space "unfolded").

Example of Sequential Consistency and Sequential Improvement



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

Matrix of Intercity Travel Costs

Safety Costs of Complete Tours

ABCDA	5
ABDCA	20
ACBDA	4
ACDBA	3
ADBCA	1
ADCBA	15

- The heuristic is not sequentially consistent at A, but it is sequentially improving.
- If we change the D→A cost to 25, the heuristic is not sequentially improving at A, and the cost improvement property is lost.
- If we change the D→A cost to 25 and we add fortification, the rollout algorithm at A sticks with the initial tentative best trajectory ACDBA, and rejects ABCDA.

A Retrospective Summary on Deterministic Constrained Rollout

Structural components

(1) Trajectories T consisting of a sequence of decisions defined by a layered/optimal control graph

(2) A cost function $G(T)$ to rank trajectories

(3) A constraint $T \in C$ to determine feasibility of trajectories

(4) A base heuristic that starts from a partial trajectory and generates a complete trajectory

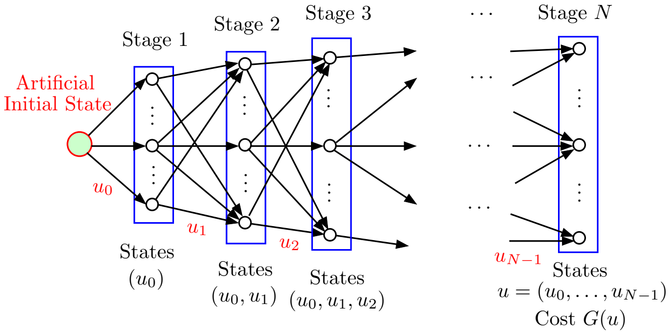
Given (1)

The choices of (2), (3), and (4) are independent of each other

In particular, given (1)-(3):

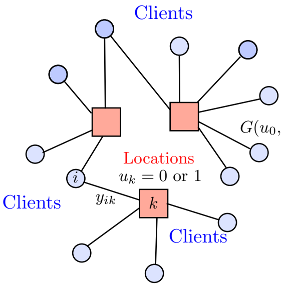
We can try several different base heuristics or a superheuristic

General Discrete Optimization Problem: Minimize $G(u)$ Subject to $u \in C$, where $u = (u_0, \ldots, u_{N-1})$



- This is a special case of the constrained deterministic optimal control problem where each state x_k can only take a single value, i.e., $x_k \equiv$ "artificial" x_0.
- A very broad range of problems, e.g., combinatorial, integer programming, etc.
- Solution by constrained rollout applies. Provides entry point to the use of RL ideas in discrete optimization through DP and approximation in value space.
- Competing methods: local/random search, genetic algorithms, integer programming/branch and bound, etc. Rollout is different.

Facility Location: A Prototype Integer Programming Problem



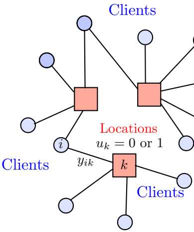
C : Set of $(u_0, \ldots, u_{N-1})$ such that $u_k \in \{0, 1\}$ and can satisfy the demand and other constraints (e.g., public policy constraints)

$$G(u_0, \ldots, u_{N-1}) = \min_{(y_{ik}, i, k) \in H(u_0, \ldots, u_{N-1})} \sum_{i=1}^{M} \sum_{k=0}^{N-1} a_{ik} y_{ik} + \sum_{k=0}^{N-1} b_k u_k$$

$H(u_0, \ldots, u_{N-1})$: Set of feasible demand allocations, i.e.
Set of $y_{ik} \geq 0$ such that
$\sum_k y_{ik} = d_i$ for all i,
$\sum_i y_{ik} \leq u_k c_k$ for all k

- Place facilities at some of the given candidate locations to serve M “clients."
- Client $i = 1, \ldots, M$ has a demand d_i for services that may be satisfied at a location $k = 0, \ldots, N-1$ at a cost a_{ik} per unit.
- A facility placed at location k has capacity c_k and cost b_k. Here $u_k \in \{0, 1\}$, with $u_k = 1$ if a facility is placed at k.
- Problem: Minimize $\sum_{i=1}^{M} \sum_{k=0}^{N-1} a_{ik} y_{ik} + \sum_{k=0}^{N-1} b_k u_k$ subject to total demand satisfaction constraints ($y_{ik} \geq 0$, $\sum_k y_{ik} = d_i$ for all i, and $\sum_i y_{ik} \leq u_k c_k$ for all k).
- There may be additional constraints on u, but we will ignore for the moment.
- Note: If the placement variables u_k are known, the remaining problem is easily solvable (it is a linear “transportation" problem).

Facility Location Problem: Formulation for Constrained Rollout



C : Set of $(u_0, \ldots, u_{N-1})$ such that $u_k \in \{0,1\}$ and can satisfy the demand and other constraints (e.g., public policy constraints)

$$G(u_0, \ldots, u_{N-1}) = \min_{(y_{ik}, i, k) \in H(u_0, \ldots, u_{N-1})} \sum_{i=1}^{M} \sum_{k=0}^{N-1} a_{ik} y_{ik} + \sum_{k=0}^{N-1} b_k u_k$$

$H(u_0, \ldots, u_{N-1})$: Set of feasible demand allocations, i.e.
Set of $y_{ik} \geq 0$ such that
$\sum_k y_{ik} = d_i$ for all i,
$\sum_i y_{ik} \leq u_k c_k$ for all k

- Consider placements one location at a time.
- Stage k = Placement decision $u_k \in \{0, 1\}$ at location k (N stages).
- Base heuristic: Having fixed $u_0, \ldots, u_k$, place a facility in all remaining locations.
- Rollout: Having fixed $u_0, \ldots, u_k$, compare two possibilities:
  - Set $u_{k+1} = 1$ (place facility at location $k + 1$), set $u_{k+2} = \cdots = u_{N-1} = 1$ (as per the base heuristic), and solve the remaining problem.
  - Set $u_{k+1} = 0$ (don't place facility at location $k + 1$), set $u_{k+2} = \cdots = u_{N-1} = 1$ (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.
- Sequential improvement is satisfied in the absence of additional constraints.
- Transportation problems are similar; solved efficiently with the auction algorithm (see literature on network optimization).

Let's Take a Working Break Before Discussing Minimax Control

A worst case point of view of the uncertainty

- The disturbances w_k are chosen by an adversarial and omniscient decision maker
- Instead of a probabilistic description of w_k, assume a set membership description $w_k \in W_k$; think of a minimax version of the principle of optimality

Minimax Control - Robust Control/Optimization - Games Against Nature

A worst case point of view of the uncertainty

The disturbances w_k are chosen by an adversarial and omniscient decision maker

Minimax Control Problems (Finite Horizon Case)

- Instead of a probabilistic description of w_k, assume a set membership description $w_k \in W_k(x_k, u_k)$ [it may depend on (x_k, u_k)]
- The minimax control problem is to find a policy $\pi = \{\mu_0, \ldots, \mu_{N-1}\}$ with $\mu_k(x_k) \in U_k(x_k)$ for all x_k and k, which minimizes the cost function

$$J_\pi(x_0) = \max_{\substack{w_k \in W_k(x_k, \mu_k(x_k)) \\ k=0,1,\ldots,N-1}} \left[g_N(x_N) + \sum_{k=0}^{N-1} g_k\big(x_k, \mu_k(x_k), w_k\big) \right]$$

- The DP algorithm (max in place of $E\{\cdot\}$): Starting with $J_N^*(x_N) = g_N(x_N)$,

$$J_k^*(x_k) = \min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \left[g_k(x_k, u_k, w_k) + J_{k+1}^*\big(f_k(x_k, u_k, w_k)\big) \right]$$

  Similar to the stochastic case ... but the max operation is nonlinear and Monte Carlo simulation is unavailable (this affects rollout/policy iteration)
- Approximation in value space with one-step lookahead applies at state x_k a control

$$\tilde{u}_k \in \arg\min_{u_k \in U(x_k)} \max_{w_k \in W_k(x_k, u_k)} \left[g_k(x_k, u_k, w_k) + \tilde{J}_{k+1}\big(f_k(x_k, u_k, w_k)\big) \right]$$

- Approximation in value space with multistep lookahead is similar

ℓ-Step Lookahead Approximation in Value Space for Minimax Control

At State x_k

First ℓ Steps

“Future”

$$\min_{u_k,\mu_{k+1},\ldots,\mu_{k+\ell-1}} \; \max_{w_k,\ldots,w_{k+\ell-1}} \left[\underbrace{g_k(x_k,u_k,w_k) + \sum_{t=k+1}^{k+\ell-1} g_t\big(x_t,\mu_t(x_t),w_t\big)}_{\text{First } \ell \text{ Steps}} + \underbrace{\tilde J_{k+\ell}(x_{k+\ell})}_{\text{“Future”}} \right]$$

Control $\tilde u_k$

Minimax Control Problem
with ℓ-Step Horizon

- Any cost function approximation $\tilde J_{k+\ell}(x_{k+\ell})$ is permissible
- Terminal cost approximation $\tilde J_{k+\ell}(x_{k+\ell})$ may be obtained by off-line training
- The “three approximations" view is valid (min approx, max approx, $\tilde J_{k+\ell}$ approx)
- The ℓ-step minimax control problem is solved by DP
- Its solution is facilitated by a special technique, called "alpha-beta pruning"
- There are variants with selective step lookahead
- This is the algorithm that most chess programs use for on-line play (including AlphaZero)

One-Step Rollout for Minimax Control in Discrete Spaces Problems

Cost of Base Policy
Corresponding to $x_{k+1} = f_k(x_k, u_k, w_k)$
and $w_{k+1}, \ldots, w_{N-1}$

At State x_k

$$\min_{u_k \in U_k(x_k)} \max_{w_k, \ldots, w_{N-1}} \Big[g_k(x_k, u_k, w_k) + \tilde{J}_{\pi,k+1}\big(f_k(x_k, u_k, w_k), w_{k+1}, \ldots, w_{N-1}\big) \Big]$$

Rollout Control $\tilde{u}_k$

LONGEST PATH PROBLEM
Optimal Control with
Control Variables $w_k, \ldots, w_{N-1}$

- At state x_k: For $u_k \in U_k(x_k)$, compute the Q-factor of the base policy π

$$\tilde{Q}_k(x_k, u_k) = \max_{w_k, \ldots, w_{N-1}} \Big[g_k(x_k, u_k, w_k) + \tilde{J}_{\pi,k+1}\big(f_k(x_k, u_k, w_k), w_{k+1}, \ldots, w_{N-1}\big) \Big]$$

  This is a longest path problem.
- Rollout control: $\tilde{u}_k \in \arg\min_{u_k \in U_k(x_k)} \tilde{Q}_k(x_k, u_k)$
- Any policy can be used as base policy (must be a legitimate policy, not a heuristic)
- Sequential consistency holds (assuming no terminal cost approximation)
- Sequential consistency implies cost improvement
- Variants: Terminal cost approx., extra constraints (no cost improvement guarantee)

Minimax Rollout Subject to Trajectory Constraint $(x_0, u_0, \ldots, u_{N-1}, x_N) \in C$

At trajectory
$\tilde{y}_k = (\tilde{x}_0, \tilde{u}_0, \ldots, \tilde{u}_{k-1}, \tilde{x}_k)$

Cost of Base Policy
Corresponding to $x_{k+1} = f_k(\tilde{x}_k, u_k, w_k)$
and $w_{k+1}, \ldots, w_{N-1}$

$$\min_{u_k \in \tilde{U}_k(\tilde{y}_k)} \max_{w_k, \ldots, w_{N-1}} \Big[g_k(\tilde{x}_k, u_k, w_k) + \tilde{J}_{\pi,k+1}\big(f_k(\tilde{x}_k, u_k, w_k), w_{k+1}, \ldots, w_{N-1}\big) \Big]$$

Rollout Control $\tilde{u}_k$

Q-factor $\tilde{Q}_k(\tilde{x}_k, u_k)$

LONGEST PATH PROBLEM
Optimal Control with
Control Variables $w_k, \ldots, w_{N-1}$

- At partial trajectory $\tilde{y}_k = (\tilde{x}_0, \tilde{u}_0, \ldots, \tilde{u}_{k-1}, \tilde{x}_k)$: Compute the Q-factor

$$\tilde{Q}_k(\tilde{x}_k, u_k) = \max_{w_k, \ldots, w_{N-1}} \Big[g_k(\tilde{x}_k, u_k, w_k) + \tilde{J}_{\pi,k+1}\big(f_k(\tilde{x}_k, u_k, w_k), w_{k+1}, \ldots, w_{N-1}\big) \Big]$$

  for each u_k in the set $\tilde{U}_k(\tilde{y}_k)$ that guarantee feasibility. A longest path problem.
- Once the set of “feasible controls" $\tilde{U}_k(\tilde{y}_k)$ is computed, we can obtain the rollout control: $\tilde{u}_k \in \arg\min_{u_k \in \tilde{U}_k(\tilde{y}_k)} \tilde{Q}_k(\tilde{x}_k, u_k)$
- Fortified version guarantees that the algorithm leads to a feasible cost-improved rollout policy, assuming the base heuristic at the initial state produces a trajectory that is feasible for all possible disturbance sequences

Relation of Minimax Control with Zero-Sum Game Theory

Zero-sum game problems involve two players and a cost function; one player aims to minimize the cost and the other aims to maximize the cost

- They involve TWO minimax control problems:
  - The min-max problem where the minimizer chooses policy first and the maximizer chooses policy second with knowledge of the minimizer's policy
  - The max-min problem where the maximizer chooses policy first and the minimizer chooses policy second with knowledge of the maximizer's policy
  - We have Max-Min optimal value $\leq$ Min-Max optimal value
- Game theory is particularly interested on conditions that guarantee that Max-Min value $=$ Min-Max value. This question is beyond the range of practical RL (but may still be of theoretical interest in many contexts).
- An interesting question: How do various algorithms work when approximations are used in the min-max and max-min problems?
- We can certainly improve either the minimizer's policy or the maximizer's policy by rollout, assuming a fixed policy for the opponent
- Can the policies be improved simultaneously? In practice this seems to work "often" ... but there is no reliable theory on this question ...
- In symmetric games like chess: What if both players train w/ a common policy?

About the Next Lecture

We will cover:

- Parametric approximation architectures.
- Neural networks and how we use them.
- Approximation in value space and policy space using neural nets.
- We will use material from videolecture 6 of the 2019 ASU class.