Reinforcement Learning for POMDP: Rollout and Policy Iteration with Application to Sequential Repair

Sushmita Bhattacharya, Thomas Wheeler
advised by Stephanie Gil, Dimitri P. Bertsekas
Arizona State University

Abstract

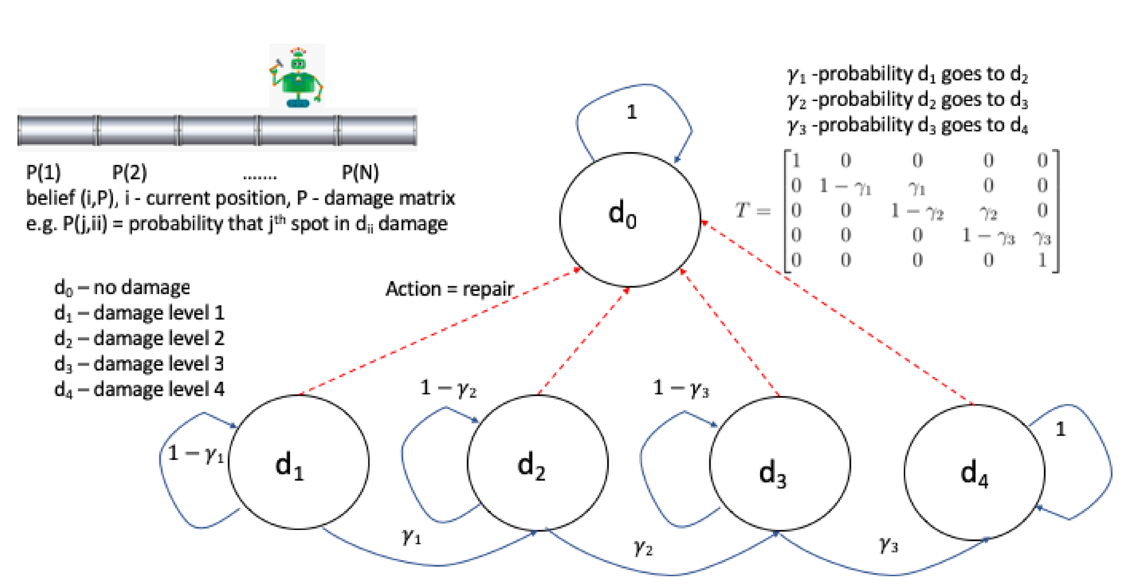
We study rollout algorithms which combine limited lookahead and terminal cost function approximation in the context of POMDP. We demonstrate their effectiveness in the context of a sequential pipeline repair problem, which also arises in other contexts of search and rescue. We provide performance bounds and empirical validation of the methodology, in both cases of a single rollout iteration, and multiple iterations with intermediate policy space approximations.

Problem Definition

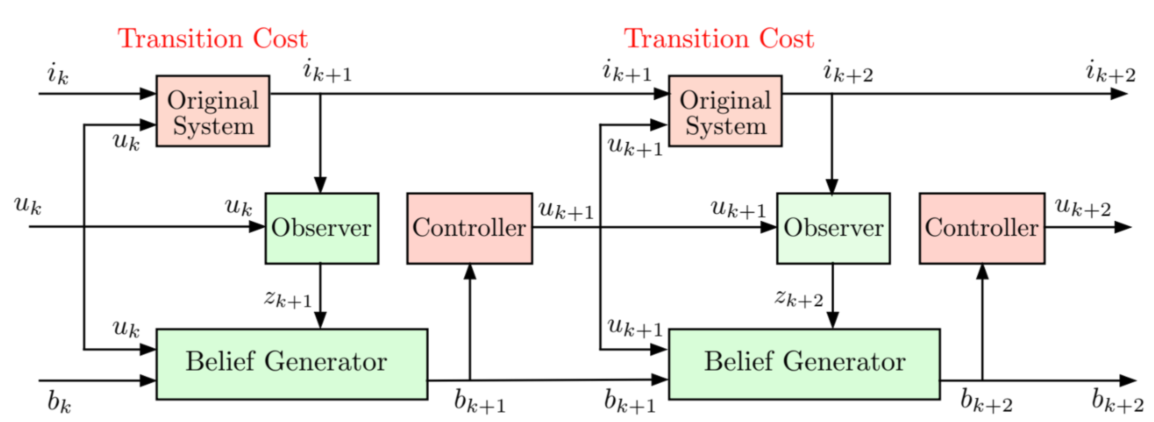
- Consider a partially observable, α-discounted Markov decision problem, where the goal is to optimize repair of a partially observed damaged pipeline.
- Underlying system state, $i = (x, f_1, f_2 \cdots f_N)$, where N is the number of segments in the pipeline, $x \in \{1, \cdots, N\}$ is the current location of the agent, and $f_j \in \{0 \cdots 4\}$ $\forall j \in [1, N]$ is the segment's damage level
- Transition probabilities $p_{ij}(u)$
- Belief state $b = (b(1), \ldots, b(n))$, where $b(i)$ is the conditional probability that the state is i, given all the observations and controls up to the current time.
- Control space $U = \{\text{go left, go right, fix damage}\}$
- The damage status $z(x)$ of the current position x is observed exactly.
- Expected stage cost at i, $g(i, u)$ depends on the damage level of each pipeline segment.
- $\phi(b, u, z)$ is the next belief state after applying action u and observing z on current belief b:
- Bellman's equation for our problem:

$$J^*(b) = \min_{u \in U} \sum_{i=1}^{n} b(i) g(i, u) + \alpha E_z\{J^*(\phi(b, u, z))\}$$

- Damage level of each segment of the pipeline becomes progressively worse according to a Markov chain



COMPOSITE SYSTEM SIMULATOR FOR POMDP



Approach

- We use limited lookahead in combination with truncated rollout to obtain an approximation to J^* and a corresponding suboptimal policy.

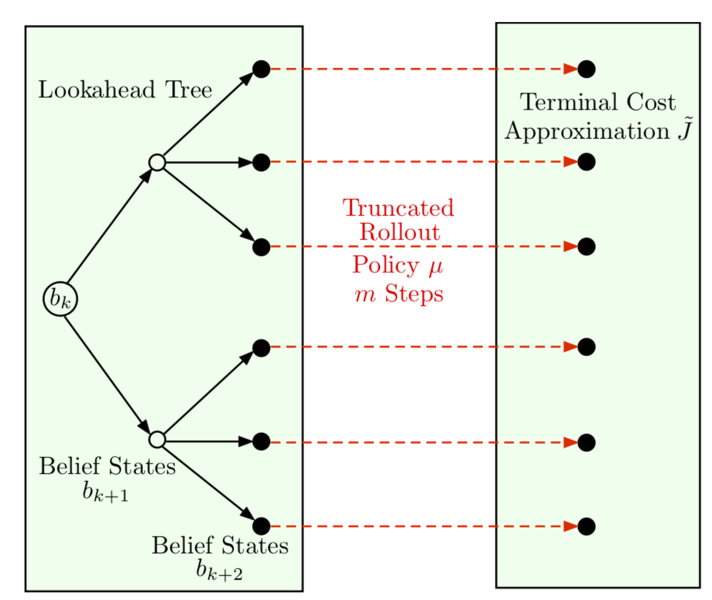


Figure 1: One-step lookahead, rollout with policy μ and terminal cost approximation $\tilde{J}$

Pipeline Repair Problem as a POMDP:

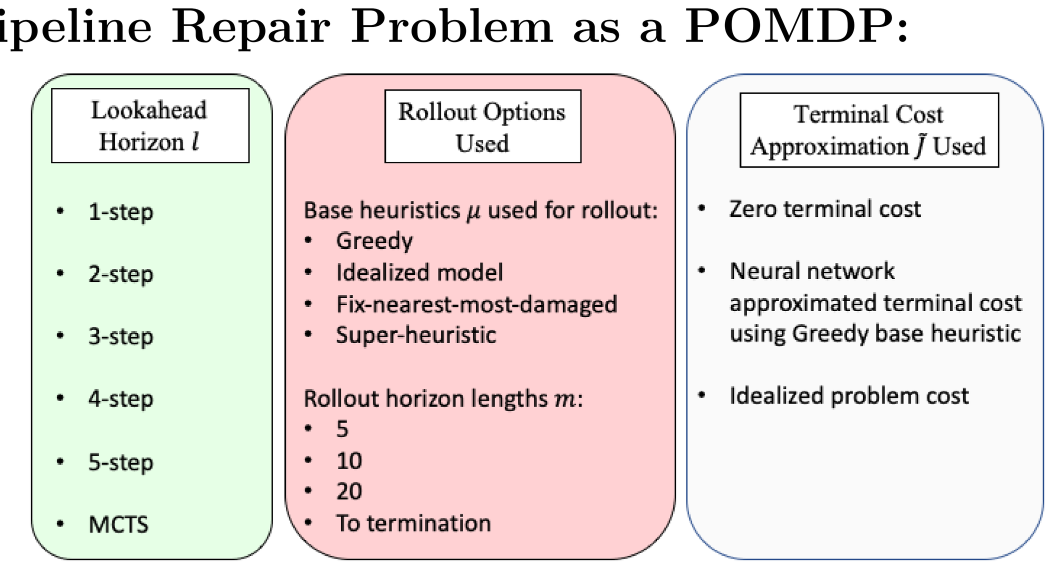
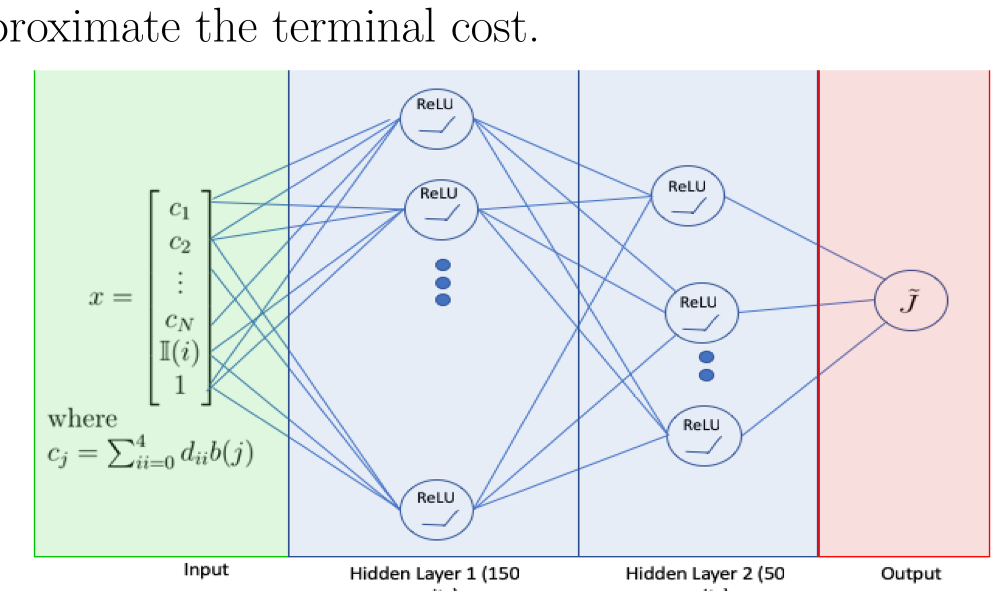


Figure 2: Parameters of proposed solution framework

Cost Approximation using Neural Net $\tilde{J}$: We used one million rollout samples from a greedy policy applied to random initial states, to train a 2-layer neural network to approximate the terminal cost.



Rollout Policy and Approximation using an Idealized Pipeline Model: This model assumes,

- At most one spot is damaged.
- Belief (x, P'); x-current position $P' = [p'_1, p'_2, \ldots p'_N]$, vector of probabilities indicating likelihood of damage, mapped from original system state (x, P), $k \in [1, N]$
- At each time t, the agent moves in the direction that minimizes the expected shortest path to the terminal state in the idealized problem.
- We use a good policy based on this model as a heuristic to select actions in the full problem.

Results on Performance Bound

Performance Bound: The theoretical bound on generated policy $\tilde{\mu}$ with limited l-step lookahead, m-step policy μ rollout followed by cost approximation $\tilde{J}$ is given by,

$$\left\| J_{\tilde{\mu}} - J^* \right\| \leq \frac{2\alpha^l}{1-\alpha} \left\| T_\mu^m \tilde{J} - J^* \right\| \quad (1)$$

where, T_μ^m is the Bellman operator corresponding to μ applied m times and $J_{\tilde{\mu}}$ is the cost function of the generated policy $\tilde{\mu}$. We study a related performance bound in simulation given below:

$$J_{\tilde{\mu}}(b) - \tilde{J}(b) \leq \frac{\max_{b'}(T_\mu \tilde{J}(b') - \tilde{J}(b'))}{1-\alpha}, \text{ for all } b \quad (2)$$

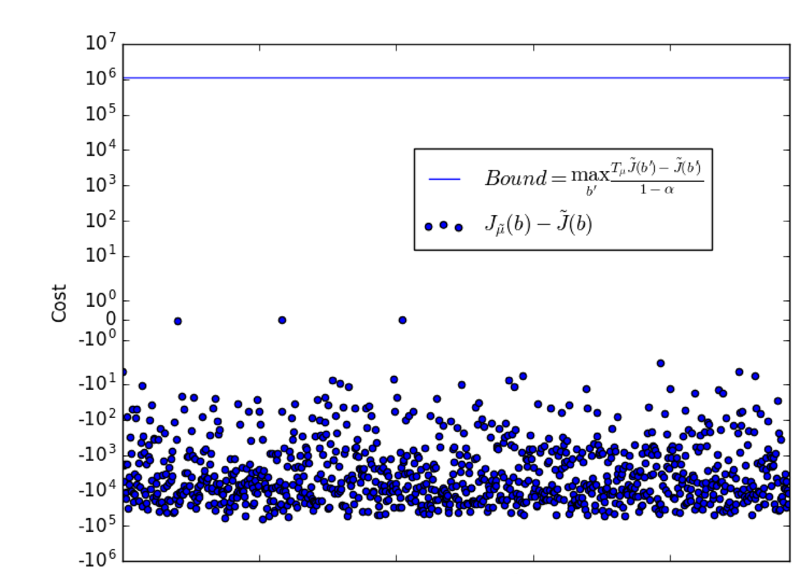
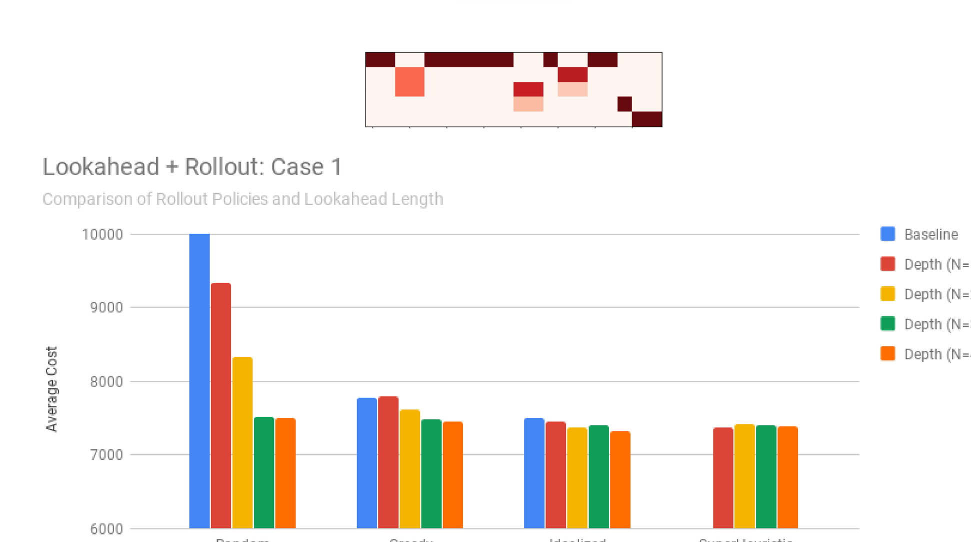


Figure 3: 1-step lookahead performance bound of rollout and terminal cost approximation using idealized pipeline model

Results on Rollout Methods



Results on Rollout Methods

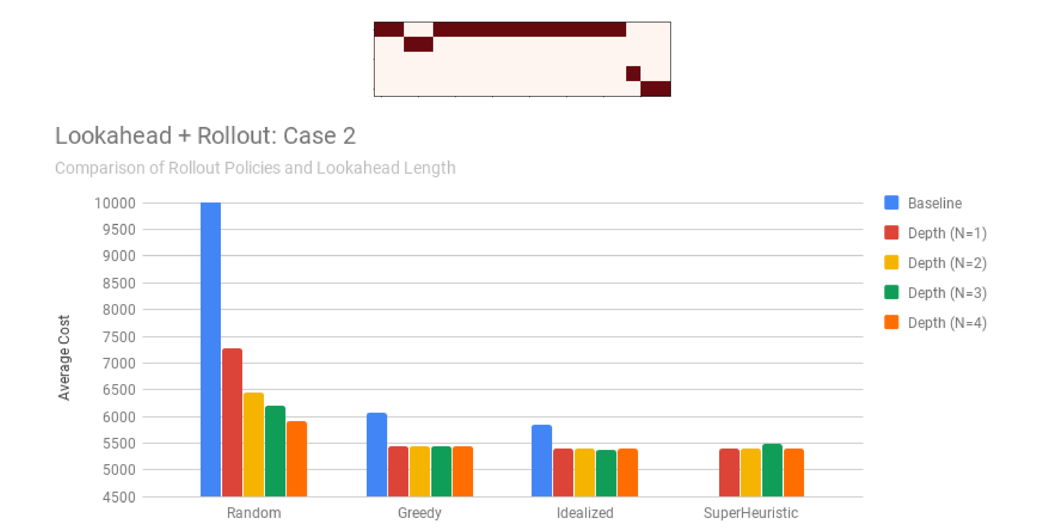


Figure 4: Performance evalution of rollout policies with 1-4 step lookahead and different heuristic policies.

Results of Approximate PI

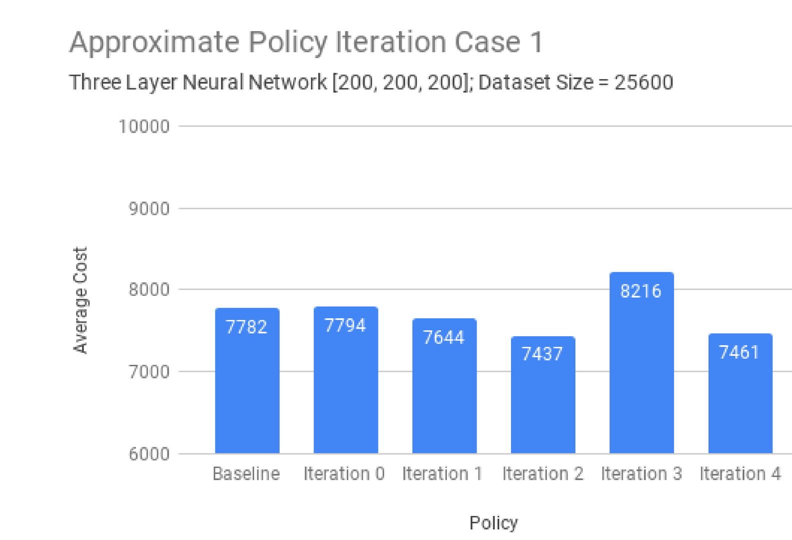


Figure 5: Approximate policy iteration with a greedy initial policy trained for 4 iterations, to solve case 1 of the Pipeline Problem.

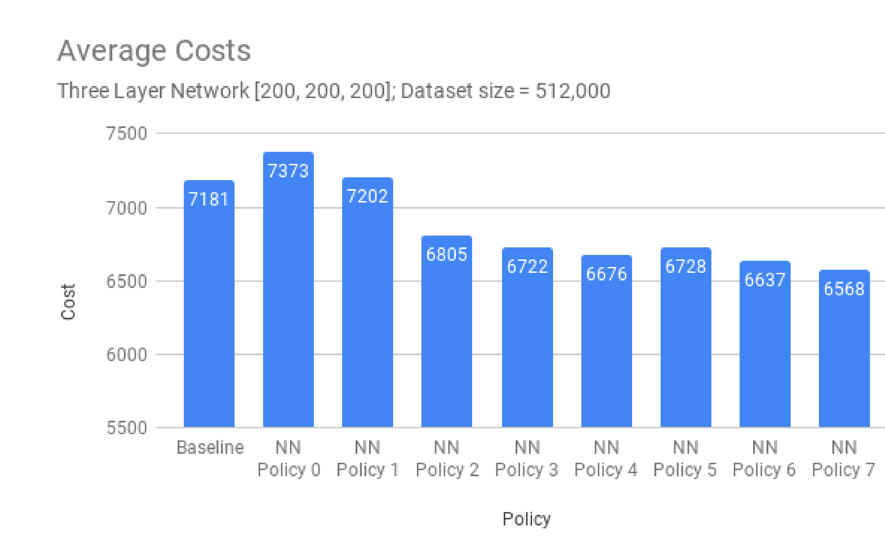


Figure 6: Approximate policy iteration with a neural network which is trained to classify actions taken by a lookahead rollout policy, which uses the policy from the previous iteration as a base heuristic.

Related Work

1. Bertsekas, D.P.,'19. "Reinforcement Learning and Optimal Control," Athena Scientific, Belmont, MA.
2. Bertsekas, D.P.,'17. "Dynamic Programming and Optimal Control," Vol. I, 4th Edition, Athena Scientific.
3. Lagoudakis, M.G., and Parr, R.,'03. "Reinforcement Learning as Classification: Leveraging Modern Classifiers," in Proc. of ICML, pp. 424-431.