

COMP4640/6460

Reinforcement Learning and Planning under Uncertainty

Assignment 1

Maximum marks	100
Weight	10% of final grade
Submission deadline	Wednesday, August 27, 2008, 23:59
Submission	Email to Scott.Sanner (@nicta.com.au)
Questions to	Scott Sanner (preferably after class)
Late Penalty	10% 1 day, 20% 2 days, 100% 3+ days
Collaboration	Permitted (but each student must submit their own code / writeup)

Each question (and how to approach it) will be discussed in-depth during tutorial.

Written Questions

1 (30/100) Model the following three problems by specifying $\langle S, O, A, Z, T, R \rangle$ and an appropriate objective criterion (horizon, average or discounted reward):

1. The game of Tic-Tac-Toe (3×3).
2. An elevator scheduling problem with three elevators, all serving the same 20 floors of a building.
3. Traffic light control for a single intersection of two single lane one-way roads (no special turning lanes) where the lights must cycle in a given sequence.

This question is inherently open-ended and under-specified — similar to a problem that you might encounter in a real-world scenario. If you think part of the problem can be modelled in different ways, state the assumptions that led to *your* particular model.

The model should be consistent with what is technically feasible. So in an elevator setting, you may know that someone is waiting at floor 5 to go down (i.e., they pushed the down button on floor 5), but you would not likely know that 3 people are waiting at floor 5 to go down and the specific destination of each person prior to them entering the elevator.

You need only formalize the problem with information relevant to achieving the objective, you do not need to specify how the problem should be solved.

2 (30/100) Provide a proof that *policy iteration* will converge to the optimal policy $\pi^*(s)$ in a *finite* number of iterations when initialized with any arbitrary policy $\pi^0(s)$ in the infinite-horizon discounted reward setting. *Hint: do your reading!*

You may use the following known properties of MDPs:

- There exists an optimal value function $V^*(s)$ s.t. $\forall s, \pi V^*(s) \geq V_\pi(s)$ and $V^*(s) = \max_a \{R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^*(s')\}$.
- There exists an optimal policy $\pi^*(s)$ satisfying $\pi^*(s) = \arg \max_a \{R(s, a) + \gamma \sum_{s'} P(s'|s, a)V^*(s')\}$.

3 (40/100) Based on the discussion in lab, make the following modifications to your `RTDP` class and put them in a class called `ModRTDP`:

- Maintain lower (as well as upper bounds) on the value function. Note that `Racetrack.State` already has an admissibly initialized lower bound, it simply is not (yet) updated during RTDP trials.
- Prioritize exploration by biasing the outcome of action selection during RTDP trial simulations.

Submit the following:

1. Given the true transition distribution $T(s, a, s')$, provide the weakest conditions on a biased distribution $\hat{T}(s, a, s')$ used for RTDP *outcome selection* (but not the Bellman backup!) such that RTDP will still converge to the optimal policy. Provide a (very short) proof using the fact vanilla RTDP converges when using unbiased outcome selection since all relevant states are updated with non-zero probability.
2. State the method you used for biasing outcome selection in RTDP. Justify why you expect this form of biased outcome selection to speedup convergence. Did it? (Refer to the plots in step 4.)
3. The code for the `ModRTDP` class and the *modified* portions of the `Racetrack` and member subclasses.
4. Three plots for `RTDP` vs. `ModRTDP` comparing the average reward per trial (`ERROR`), average Bellman error on the initial states (`INI ERR`), and the number of states expanded (`NUM STATES`) on `large-b.racetrack`. Use `Solver.NUM_TRIALS=5000` and `Solver.DISPLAY_INTERVAL=250` (comment out visualization of the policy — `_racetrack.runPolicy(.)` — in `Solver` to allow the experiments to complete faster). Average the results for each algorithm over 30 runs.
5. Provide at least one other modification to RTDP that you would expect to speedup convergence. You are not required to implement this modification, but you should justify why it might speedup convergence.