

COMP4640/6460

Reinforcement Learning and Planning under Uncertainty

Assignment 3

Maximum marks	100
Weight	10% of final grade
Submission deadline	Wednesday, October 29, 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) Fun with decision diagrams:

- Prove that there exists a unique *reduced* ADD (for a fixed total variable ordering) for every possible function $\mathbb{B}^k \rightarrow \mathbb{R}$. Useful properties of canonical *reduced* ADDs are that (a) all paths from the root to the leaves obey the variable ordering, (b) leaf nodes are hashed to unique IDs based on the constant value, (c) internal nodes are hashed to unique IDs based on the decision variable and true and false branch IDs, and (d) no internal decision node can have true and false branches with the same ID (this decision node would have been removed during the reduce process). *Hints: You can prove this by induction on k .*
- Provide pseudocode for an algorithm using functions $Apply(F_1, F_2, op) \rightarrow F$, $GetLeafNode(value) \rightarrow F$, and $GetVarNode(var, value - false, value - true)$ (the latter function building single decision nodes with constant leaves only), to convert an unordered decision diagram to an ADD for some fixed variable ordering.

2 (30/100) Build a file in the SPUDD format used in lab to represent the Elevator problem for 1 elevator and 2 floors. You can use any formalization you want, but note that encoding time in the state is generally undesirable; you can use the reward function to accumulate penalties for waiting periods at each time step. Ignore variables like elevator direction, internal elevator buttons, or directionality of buttons at each floor that don't matter in a 2 floor problem.

Run your SPUDD file through a factored MDP solver (either the Java one used in class or the C-based version of SPUDD on the web if you need better scalability). Note: variable ordering at the top of the SPUDD file may affect performance.

Do the following:

- Provide a formatted printout of the SPUDD file with a key to variable naming.
- Provide a plot of the ADD value function size as a function of the number of iterations (until convergence within some epsilon).
- At convergence, has the ADD provided any compactness over a fully enumerated tabular representation? How can you tell?
- Provide one way that more compactness in the ADD could be achieved (at the expense of accuracy).
- After convergence within some tolerance, evaluate five different states of the value function and provide their state assignments and corresponding values in a table. If the values don't make sense, check your SPUDD file for errors!

3 (10/100) The transition function encoding of the Elevator domain can be used to enforce that an elevator cannot reverse direction if an occupant inside is headed in the current direction. If the transition function was *not* encoded with this restriction, could you think of another way to enforce this non-reversal constraint in the solution? Would this be more efficient, less efficient, or as efficient as encoding the constraints directly in the transition function?

4 (30/100) Formalize the following manufacturing problem for *100 machines operating in parallel* by specifying (S, A, O, T, Z, R) and an appropriate objective criterion:

- Each machine produces widgets and its operation is independent of the other machines.
- Each machine can be working or broken.
- Each machine breaks with probability 0.05 on every time step.
- A broken machine can only be diagnosed by direct inspection.
- Two zero-cost actions can be executed for each machine: a machine can be inspected or it can be used to produce 1 widget.
- For every widget produced by a correctly operating machine, a dollar is gained. A widget produced by a broken machine will be later discarded by quality-control and leads to no gain.
- With probability 0.0001 on every time-step, the company goes out of business.

As specified, how can you solve for an optimal policy for this problem efficiently (i.e., without enumerating the joint action space of 2^{100} actions)?

10 bonus points (to a maximum assignment score of 100 points): Use a POMDP solver¹ to find the optimal solution to this problem. Provide the solver input file and the output of the solver.

¹<http://www.pomdp.org/pomdp/code/index.shtml> or <http://www.cs.uwaterloo.ca/~ppoupart/software.html>.