From APPL

Main: FAQ

Frequently Asked Questions

The most up-to-date version of this document is available at **http://bigbird.comp.nus.edu.sg/pmwiki/farm/appl/index.php?n=Main.FAQ**.

1. APPL Software

1.1. What are the limitations of the SARSOP algorithm?

The SARSOP algorithm is a point-based POMDP solver. Just like other point-based solvers, it outputs a policy that approximates the optimal policy for a given problem. The longer the solver is run, the better the approximation. For more information about SARSOP and the MOMDP representation, see [**1**,**2**].

1.2. What are the limitations of the current APPL release?

This release (version 0.92) does not yet support models involving dependency of variables within the same time slice. The same also holds true for state transition functions. For initial belief, this means that only the keyword `null` is allowed within the `<Parent>` tags. For state transition functions, only identifiers which had been declared as the `vnamePrev` attributes of state variables or identifiers which had been declared as the `vname` attribute of action variables, are allowed as conditioning variables which

appear within the <Parent> tags. We are currently improving the APPL implementation to remove these restrictions and hope to have it ready by June or July, 2010.

1.3. What is the difference between the simulator and the evaluator?

To get a simulation path for a policy, use the simulator pomdpsim. To estimate the expected total reward (ETR) for a policy, use the evaluator pomdpeval. It is also possible to use the simulator to estimate ETR by generating many simulation paths and computing the average reward of these paths. This is much slower than using the evaluator, but uses less memory.

1.4. What does the "--randomization" compilation flag for pomdpsol do?

SARSOP chooses the next belief point to sample based on the current upper bounds. The "--randomization" flag turns on randomization for this choice, when there is a tie between the upper bounds. The flag is off by default.

1.5. What do the "-msse2 -mfpmath=sse" compilation flags do?

These flags control the code generation for floating point processing. They are not strictly necessary, but seem to improve performance, based on our experiences.

1.6. Why does pomdpsol fail when my input POMDPX file contains <DD> (decision diagram) tag?

APPL parser does not support decision diagrams currently. However, decision diagrams are officially part of POMDPX, and we plan to support it in the future.

1.7. Why does pomdpsol give a "bad_alloc()" error when I attempt to solve my problem?

Very likely your problem is too large to be loaded. Try to reduce the size of your model, or ask the system administrator to increase the memory limit for your system.

1.8. Why does the policy graph generator crash with a seg fault?

The most likely reason is that the out.policy you used was for another problem. Try running pomdpsol first to generate the pomdp policy and then re-run policy graph generator with the just generated policy file.

1.9. I am lazy but I want the best of both worlds. Is there a way to speed up the entire code without modifying the source?

Actually there is, in the Makefile, add the option: "-march=native". This takes advantage of the specific architecture of your machine to give a speed-up of 3-5% and about 5-7% reduction in executable size. Note that this feature is only available in gcc version 4.2 and above.

2. POMDP/MOMDP Modeling

2.1. How to convert Tony Cassandra's POMDP file format to POMDPX format?

We are currently working on a POMDP to POMDPX converter. It is going to be released soon.

2.2. Why does TagAvoid.pomdpx's fully observed state variable have a uniform initial belief, rather than a single state initial belief?

TagAvoid.pomdpx is an example of a problem where one of the state variables is not fully observed at the first time step (it has a uniform initial belief) but is fully observed in subsequent time steps. APPL's solver and simulator/evaluator recognize this special case and take into account both the uniform initial belief of the state variable as well as its fully observability in subsequent time steps.

2.3. What if the fully observed state variable in the model is not really fully observed in reality?

Generate a policy with the fully observed state variable in the model. For robustness, the generated policy should be evaluated with a model where the state variable is not indicated as fully observed. To use the generated policy with this model, a modified policy based on the value function associated with the generated policy must be used. Refer to [2] for details on how this can be done.

2.4. What if the model is a Markov decision process (MDP), i.e. all the state variables in the model are fully observed?

If the POMDPX file specifies all the state variables as fully observed, then the model is an MDP. The Appl solver will generate a policy file which is the solution to the MDP problem. The policy file can then be used to simulate/evaluate the MDP problem.

3. References

[1] H. Kurniawati, D. Hsu, and W.S. Lee. **SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces**. In *Proc. Robotics: Science and Systems*, 2008.

[2] S.C.W. Ong, S.W. Png, D. Hsu, and W.S. Lee. **POMDPs for robotic tasks with mixed observability**. In *Proc. Robotics: Science and Systems*, 2009.