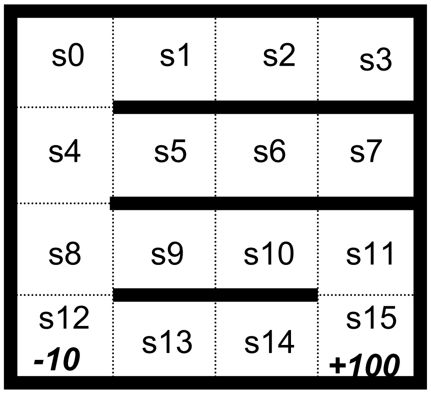
Homework 2

COMP-765: Decision-making under uncertainty
Due in class Monday March 6 2006

The goal of this assignment is to experiment with POMDP planning problems and algorithms. In the first part of the assignment, you will implement a simple maze navigation task using standard POMDP syntax. In the second part, you will investigate the use of different methods (exact, approximate and heuristic) to solve this problem. In the third part, you will implement an additional heuristic and test its performance.



<u>Domain description:</u>
The domain is a 4x4 grid world. The walls are indicated by thick lines. The agent has access to four motion actions: {*North, South, East, West*}, each of which has the expected deterministic effect , e.g. *T(*s0*, south, s4)=1.0*. The observation produced at each state indicates whether there is a wall in each cardinal directions (e.g. observation at state *s0* is *NW*, indicating there is a wall to the North and a wall to the West). Observations are deterministic (i.e. every time the agent is in *s0*, it perceives observation *NW*), but there is state aliasing since many states have the same wall configurations (e.g. *s1* and *s2* both perceive *NS*). The reward is a function of the state only, and therefore the same for any action. It is 0 for all states, except states *s12* and *s15*, where it is *-10* and *+100*. The agent can start (with equal probability) in any state except *s12* and *s15*. Whenever the agent reaches *s15*, the task is reset, and so the agent transitions to any of the possible start states (with equal probability). Assume a discount factor of $\gamma=0.95$.

<u>Planning description:</u>
Throughout this assignment, you will use the POMDP software developed by Tony Cassandra. Download the software from: *http://www.cs.mcgill.ca/~jpineau/comp765/pomdp.tgz*
There is more information on this software package and POMDP-related matters at:
http://www.cs.brown.edu/research/ai/pomdp
(but don't download the software package from there, since I've modified it).
To compile:

```
tar xzvf pomdp.tgz
cd pomdp
make
```

This will produce an executable:

```
./pomdp-solve
```

To see examples of POMDP domains in this syntax, see the files in sub-directory:

```
examples/
```

To try running one of these examples using the incremental pruning algorithm:

```
./pomdp-solve –p examples/ejs1.POMDP –method incprune
```

To understand the syntax used to specify POMDP domains for this package, see:

```
docs/pomdp-file-spec.txt
```

To understand the command line options associated with this package, see:

```
docs/pomdp-solve.txt
```

Alternately, just use the command line help option:

```
./pomdp-solve -h
```

Problem 1 (50%):

a) Represent the maze domain above using the appropriate POMDP syntax.

i) Submit a printout of this file.

ii) Email me a copy called grid.POMDP

iii) Explain any non-obvious design choices used to build your model.

b) Solve the problem using the MDP method. This provides an upper-bound on the results since it assumes that the state is observable. Now solve the problem using the greedy heuristics (most-likely, voting, qmdp) and using one exact method (enumeration, witness, incremental pruning).

i) Compare (table/graph) the average reward of the various methods.

ii) Discuss which methods perform better/worse than others and explain why.

c) Do the same as in (b), but assuming that the reward at state *s12* is *-100*.

i) Report on the results.

ii) Summarize differences with the results in (b) and explain why.

d) Do the same as in (b), but assuming that the reward at state *s12* is *-1000*.

i) Report on the results.

ii) Summarize differences with the results in (b) and (c) and explain why.

Problem 2 (20%):

a) Use the same maze world as in Problem 1, but make the following modifications. Add an action called {*look*}. Assume that it generates the wall observations as explained in the domain description above. Also assume that all other actions now only generate a new observation called {*none*} for every state. Assume once again that the reward for *s12* is *-10*. Implement this new domain using the correct POMDP syntax.

i) Submit a printout of this file.

ii) Email me a copy called grid2.POMDP

iii) Explain any non-obvious design choices used to build your model.

b) Solve this new problem using the same set of methods as in Problem 1(b).

i) Report on the results, both in terms of average reward and planning time.

ii) Summarize differences with the results in Problem 1(b) and explain why.

Problem 3 (30%):

a) Extend the software to include Cassandra's entropy heuristic as presented in class (see slides from Feb.14). In main.c there is a function called entropy_select_action(). This is the only function you need to implement. The infrastructure to call this function has been provided.

i) Submit a printout of this function (don't need the entire main.c).

ii) Email me a copy of main.c

iii) Report on the results of solving grid.POMDP and grid2.POMDP with this heuristic, assuming different entropy thresholds (e.g. $\delta=1$, $\delta=2$, $\delta=3$).

ii) Summarize differences with the results in Problems 1 and 2 for the other heuristics, and explain why.

The following functions may be useful to you for this problem:

newBeliefState()	> Allocate memory for a new belief state.
transformBeliefState(b,b',a,z)	> Update posterior $b'(s')=\eta O(s',a,z)\sum_{s\in S} T(s,a,s')b(s)$
getEntryMatrix(R[a],s,z)	> Get the observation probability O(s, a, z)
getEntryMatrix(P[a],s,s')	> Get the transition probability T(s, a, s')

The following (global) variables may also be useful to you:

gNumStates, gNumActions, gNumObservations