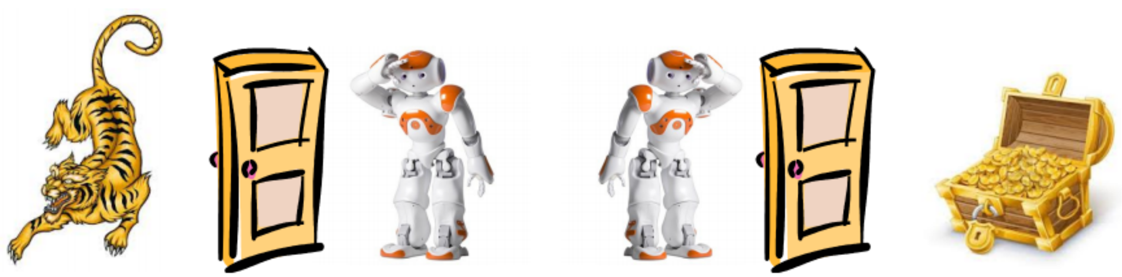
Sequential Decision Making for Cooperative Agents

Practical Part 1. An Exercise in Dec-POMDP Problem Specification

A common benchmark problem in the Dec-POMDP literature is the multiagent tiger problem, often called "Dec-Tiger". Paraphrasing from Nair *et al.* (*IJCAI*, 2003):

Two agents find themselves in a corridor facing two doors: "left" and "right". Behind one door lies a hungry tiger and behind the other lies untold riches but the agents do not know the position of either.   The agents can jointly or individually open either door. In addition, the agents can independently listen for the presence of the tiger. Every time either agent opens one of the doors, the state is randomly reset with uniform probability, regardless of the action of the other agent. However, if both agents listen, the state remains unchanged.



After every action, each agent receives a noisy observation about the new state: 85% of the time, the agent associates the sound of the tiger with the correct door, and 15% of the time with the incorrect door.

If either of them opens the door behind which the tiger is present, they are attacked by the tiger, thereby incurring a large penalty *(-100). However, the injury the agents sustain is less severe (-50) if they open the tiger's door jointly than if only one of them opens it. Similarly, they receive wealth (+10) when they open the door to the riches in proportion to the number of agents that opened that door. However, opening both doors simultaneously results in only a tiger attack and no wealth at all. The agents incur a small cost (-1) for jointly listening.*

On the other hand, acting jointly is beneficial because the agents gain more riches and sustain less damage by acting together. But because the agents receive independent observations (which they do not share), acting jointly requires that they each consider the potential histories of observation histories of the other agent and what action (s)he is likely to perform.

1. States, Actions, and Observation

a) Write down the state joint space S for the Dec-Tiger problem. Is there any notion of individual state here?

b) Write down the joint action space A for the Dec-Tiger problem. Is there any notion of individual action here?

c) Write down the joint observation space O for the Dec-Tiger problem. Is there any notion of individual observation here?

2. A Graphical Representation of the Dec-POMDP.

There is a nice open-source GUI for specifying graphical models called OpenMarkov: http://openmarkov.org/. We will use this software to encode Dec-Tiger as a Two-stage Dynamic Bayesian Network.

OpenMarkov can be downloaded and used on your computer/laptop/tablet. Simply access the JAR file at the following location: http://openmarkov.org/users.html. (Note: this application requires that Java 7+ is already installed on your machine.)

To create a Dec-POMDP in OpenMarkov, simply click the first icon below the menu bar, "Create a new network", then select "Network Type: Dec-POMDP", then click "OK".

a) Begin by creating nodes for the state, action, and observation variables that you identified in the previous question. In OpenMarkov, each state factor and observation variable takes the form of a "chance node" (see the icon with the yellow circle); each agent's action is encoded using a "decision node" (blue square); and the reward is encoded using a "utility node" (green diamond). Names and domains of your variables should be specified in the "Node Properties" dialog.

Note that, since you are specifying a two-stage bayesian network, state and action variables should be instantiated for time *t* (time slice "0" in OpenMarkov) and for time *t*+1 (time slice "1"). Right-click->"Create node in next slice" comes in handy here. Rewards, on the other hand, need only be instantiated for time *t*, and observations only at time *t*+1.

b) Connect your nodes appropriately so as to specify a standard Dec-POMDP. This involves using the "insert links" button (north-east-facing-arrow icon) below the menu bar.

Note that, in OpenMarkov, an observation variable <u>must</u> be connected to an action variable in the same slice, denoting that an agent bases its decisions on its observations.

c) Specify the transition function, observation function, and initial belief state by filling in conditional probability tables (CPTs) with probabilities that you derive from the text description of Dec-Tiger. For every chance node in OpenMarkov, right-click->”Edit Probability”, then select ”Table” as the relation type.

d) Specify a reward function, selecting numbers that appear appropriate according to the text description. Right-click->”Edit Utility”.

Lastly, be sure to save your file.

3. Policies.

a) With pencil and paper, draw a horizon h=3 policy tree for the Dec-Tiger Problem from the perspective of one agent.

b) Can you identify the optimal joint policy for horizon h=1? h=2? h=3?

c) For horizon h=3, how large is the joint policy space?

4. Solution Methods.

To compute solution policies for your Dec-Tiger Dec-POMP, you can use another piece of open-source software called the MADP (MultiAgent Decision Process) Toolbox:
http://staff.science.uva.nl/~faolieho/index.php?fuseaction=software.madp
A recent feature of the Toolbox is parsing model files that were created using OpenMarkov. A number of existing optimal and approximate solution algorithms can then be applied to solve the problem to a given horizon.

We welcome you to download the software and to try solving the model you just created. The installation of this software is somewhat involved. However, for simplicity here, we invite you to transfer your files to the tutors' laptop to be solved live.

a) Use the Generalized Multiagent A* algorithm to solve the Dec-Tiger problem for horizons 1, 2, 3, and 4. Are the results as you expected?
You can run GMAA* the following command:

```
src/solvers/GMAA [your-pgmx-file] -h [the-desired-horizon]
```

b) Experiment with other solution methods of your choice and with time horizons of your choice. (You can select infinite horizon if you like.) Compare the runtimes and solution qualities returned by the algorithms.