Updated POMDP Model

Overview

This model contains helper robots and random agents. Random agents are agents that are dependent on others for help, such as elderly people or toddlers. Helper robots seek to maximise the amount of help they provide to random agents according to a certain ethical framework.

State, Actions and Observations

A state for this problem comprises the Cartesian coordinates of each helper robot and each random agent, as well the condition of each random agent. The coordinates of each robot and agent are always known with 100% certainty. The condition of each random agent is partially observed.

An action for this problem is a pair of Cartesian coordinates, taken from all valid coordinates for the problem (that is, coordinates that are on a grid and do not contain a wall). Sending an action to a helper robot indicates that that robot should move towards those coordinates and help any random agent at those coordinates. It also signals to any random agent at those coordinates to wait and receive help.

An observation for this problem comprises the believed condition of each random agent. As mentioned before, conditions are partially observable.

GAMA

Grid

The grid in this model is of width 5 and height 5. Its layout is as follows:
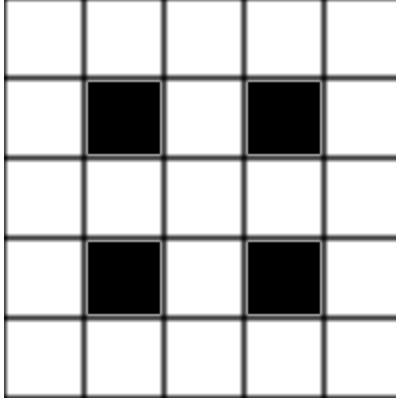
where white cells represent the floor and black cells represent walls.

Each cell in the grid has four neighbours, as defined by a <u>Von Neumann neighbourhood</u>.

GAMA also keeps track of which helper robots and which random agents are in a given cell, for easy access to this data. These data can be accessed by the fields `robots_in_cell` and `dependents_in_cell` in the `grid_cell` species.

Simulation Model

The simulator interacts with the OPPT solver by sending and receiving information related to the execution of the model. There are three pipes used to do this: these are `pipeToGama` (used by GAMA to receive information from the solver), `statePipeToSolver` (used by GAMA to send information relating to state to the solver) and `observationPipeToSolver` (used by GAMA to send information relating to observations to the solver).

When the simulation starts, it also sends some information to the solver needed to run the simulation using a series of files. These files encode information about the problem's grid and the details and condition change matrices for each random agent.

Note that GAMA represents matrices in (x, y) form as opposed to row-column form. An example is shown below, with the GAMA matrix representation shown on the left and the traditional matrix representation shown on the right.

```
[0.9, 0.0, 0.0, 0.5]
[0.1, 0.8, 0.0, 0.0]
 [0.0, 0.1, 1.0, 0.5]
 [0.0, 0.1, 0.0, 0.0]
```

$$\begin{bmatrix} 0.9 & 0.1 & 0.0 & 0.0 \\ 0.0 & 0.8 & 0.1 & 0.1 \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0.5 & 0.0 & 0.5 & 0.0 \end{bmatrix}$$

GAMA also takes data from the solver's configuration file, namely the initial state of the problem, so that it can initialise the locations and conditions of each helper robot and random agent.

Random Agents

Random agents are dependent on helper robots for help. Each random agent has a location, as well as a current condition. At present, there are four conditions in the problem. These are:

- `0` - "happy"

- `1` - "agitated"

- `2` - "sad"

- `3` - "angry"

Random agents move between these conditions stochastically. If a random agent is happy, it will move one cell on the grid with a 50% chance. If a random agent shares a cell with a helper robot, its condition becomes `0` with 100% certainty.

Each random agent has a type and an ID. A random agent's type determines what transition matrix it subscribes to with regards to changing conditions. At the moment, the two random agent types (elderly and toddler) have the same condition transition matrices, but this is subject to change. No two random agents can share the same type and ID.

Helper Robots

Helper robots serve to help random agents according to a certain ethical framework. This is defined in the OPPT solver. In the GAMA simulation, their state contains their location, their current action (as well as how to complete it), the location of each random agent in the problem, and their belief of the condition of each random agent in the problem.

While the method in which helper robots observe random agent conditions is still to be defined, it has previously been defined by a combination of probability and "sight". One agent "seeing" another is defined by the existence of a line `y = mx + c` existing between the two agents that does not cross a wall in the problem grid. If a helper robot can "see" a random agent, then it has a 0.8 probability of observing its condition.

If a helper robot has no action to complete, it will request a new action from the solver. This is done by listening to the named pipe `pipeToGama`. If it has made any observations while executing the previous action, it will send these prior to waiting for a new action. Once the helper robot has received an action, it will inform any random agents residing at that location to stay where they are while it comes to them for help.

If a helper robot has an action to complete, it will continue to follow a pre-set path towards the location encoded in this action.

If a helper robot happens to share the same cell as a random agent, it will immediately help that agent. This results in the random agent's condition being `0` (happy) and the helper robot observing this new condition with 100% probability.

OPPT Solver

Solver

At present, the solver for this problem uses the default ABT solver.

Reward Plugin

The reward plugin maximally rewards the solver if all random agents have condition `0`, that is, if all random agents in the problem are happy. If there are unhappy random agents (ones that do not have condition `0`), then the solver is penalised depending on how many random agents are unhappy and how far the robot is away from these unhappy agents.

Planning vs Execution

The OPPT solver has different transition and observation plugins depending on whether the solver is planning or executing a POMDP problem. The planning plugins must reflect the GAMA model as accurately as possible, otherwise there will be inconsistencies when executing the problem.