CoBot planning: Asking for human assistance

Armando Aguileta Mendoza

A robot asking for help is a topic that has recently started to be explored. The existent research mainly focuses on asking supervisors, bystanders or occupants for observations in order to reduce uncertainty. In this project we intend to model a planning problem in which a robot has to ask for help with actuation. In contrast to asking for observations, asking for actuation generally requires the person to move where the actuation-related task needs to be completed (i.e. getting the robot an object). Hence, we would prefer the robot to look for people available in such area. We took the approach of modeling the problem as a Markov Decision Process, and then analyzed its complexity and efficiency, resulting to be exponential in state space, but manageable for up to 5 people (about a quarter of a second computing time). With this baseline it is now possible to find heuristics to improve the model, and learn probabilities for getting help by using reinforcement learning.

1. INTRODUCTION

The CoBot robots are symbiotic agents that accomplish tasks both independently and with the assistance of humans. They are intended to operate autonomously until they are not able to perform a task given their physical, cognitive or perceptive limitations [1]. When this occurs, they ask for assistance.

This project intends to study; from a planning perspective, the situation in which a CoBot robot needs to choose between different humans to ask them for help in order to accomplish a task related to actuation. The goal is to maximize the probability of getting help while minimizing the costs incurred in navigation from one location to another, and the burden of asking people to leave their current activities to help the robot.

The current work models the problem as a Markov Decision Process, describes the complexity and presents test results.

2. RELATED WORK

This project models the situation in which a robot needs help specifically for completing an actuation related sub-task. Previous work has mainly modelled humans as observation providers, which the robot can ask for assistance in order to reduce uncertainty. For example, in Oracular Partially Observable Decision Processes (OPOMDPs) the agent can rely on a special type of supervisor; an Oracle, to provide the agent's exact state at anytime, at a fixed cost [2 ORACULAR POMDP]. Hence there is no precondition for the agent to ask, such as navigating to the Oracle's location, and it is assumed that it is always available. Similar approaches rely on asking bystanders; with the difference that they don't get any benefit from the robot, or don't have any affiliation with it [3 – BYSTANDERS].

More recently research has been done in asking for observations to occupants of the building, people with a fixed spatial location that benefit from the robot services. In the former framework; called Human Observation Provider POMDP (HOP-POMDP) [4], the concept of availability is taking into account, which is the probability of getting help from a specific person. Also the robot has to navigate to the person's location in order to get help.

Asking for help with actuation takes into account concepts from previous work such as the cost of asking, availability, and having to move to a person's location in order to receive help. Nevertheless it differs in the following aspects: the robot first navigates to the area where the actuation related sub task needs to be performed, for example it navigates to the kitchen when it needs to get a cup of water, then it senses the area looking for people, and creates a specific problem representation (using Markov Decision Processes) based on the sensor data. We assume that the tasks are usually performed in common, populated areas, and that the cost of asking one person already located in the area is less than asking an occupant in an office, or a supervisor. The model also includes a 'Give-Up' state, in which the robot fails to get help from people in the area and could proceed to ask the occupants or an oracle

3. MODELING THE PROBLEM

The problem is modelled using a Markov Decision Process (MDP) which is represented by a tuple {S, A, R, T}:

- S: States
- A(s): Actions for each state s
- T(s, a, s'): Transition function, probability of reaching state s' by taking action a in state s
- R(s): Reward, or cost (a negative reward) for each state s

The advantage of modelling the problem as a MDP, is that it can be solved by standard solvers and algorithms, such as value iteration; which was used for testing.

3.1 The states

There are three default states for this modelling approach, included in every problem representation):

- The Initial State (0), which represents the robot is in the initial position X, Y and assumes that all the people detected is available to help.
- The goal state (G), which represents a scenario in which the robot has completed the actuation related sub-task
- The 'Give Up' or 'Abort' state (A), which represents a scenario in which nobody is available to help, or the cost of asking for help was so high that the agent wouldn't take it.

The remaining states encode the following information:

- The robot is located in the coordinates X, Y corresponding to a specific person location

- The availability of all the people sensed before (not available can either represent that the robot arrived to the location and the person is not there anymore, or that the agent asked the person for help, and the person didn't help)

For example, a valid state could be: The robot is in (5,0); corresponding to person 1's (P1) location, P1 is not available, P2 is available and P3 is available. One reason for modelling the availability of all people in each state is to avoid asking the same person again.

3.2 The actions

There are three types of actions the agent can perform: move to a different location, ask for help and give up. The action function A(s) determines a list of actions available at each state. In the MDP problem representation the actions are no different from each other per se, what governs the outcome of each action is the Transition model.

3.2 The Transition Model

For the actions that represent moving from one location to another, the transition function returns a value which represents the probability of arriving to the destination and that the person is still there (available) or not. For example, consider in the following scenario, according to Fig 1:

- Suppose that agent is in the initial state S
- State 4 represents that the agent is in the location sensed for person 1, and person 1 is available (and person 2 is available also)
- State 3 represents the agent is in the location sensed for person 1, person 1 is not available

T(S, 0, 4) = .2 , means that there is an 20% probability of arriving to person's 1 location from the initial state using action 0, and that person 1 is available (also person 2). And the complimentary transition T(S, 0, 3) means that there is a 20% chance of arriving to person's 1 location from the initial state using action 0, and person 1 is not available (but person 1 is available).

For the *ask for help* action the transition function T(s, a, G) represents the probability of reaching the Goal state by asking a specific person for help with action *a*, being in state *s*. By applying action *a* in state *s*, there is also a probability of arriving to a state; in the same location, where the person is not available to help. For example in Fig. 1, if the agent is in state 5, and performs action 0, the is a .4 probability of arriving to the Goal state; as indicated by T(5, 0, G) = .4, and a .6 probability of staying in the same location with person 6 not available to help anymore; which is determined by T(5, 0, 6) = .6

For the *give up* action, there is always a 100% probability of arriving to the Give Up state.

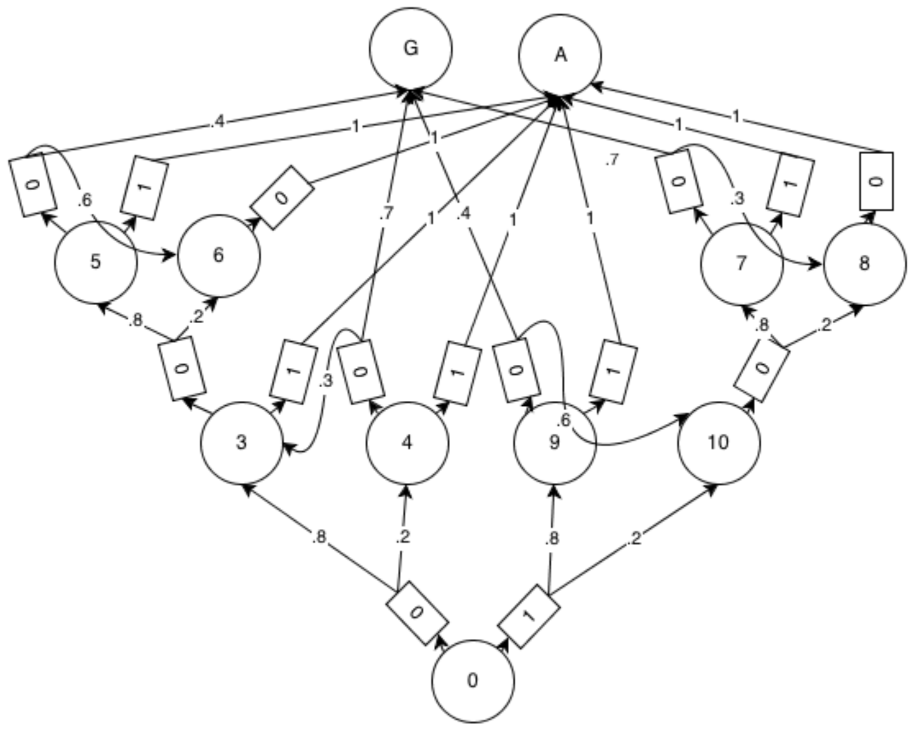
Fig.1 Problem Representation for two people

3.3 The reward function

The reward function for this problem works as follows:

- The Initial state (S), Goal state (G) and Give Up state (A) have a predefined reward, usually a low or null reward for the initial state, a high reward for the goal state, and a negative reward for the Give Up state. What is important to mention, is that the reward for the Give Up state could represent the expected reward for achieving the goal in other way (external from this problem model), for example, the expected reward for asking for help to a supervisor, or an occupant.
- The reward for the other states is the negative of the last immediate distance the robot had to travel to get to the state. For example in Fig. 1, for state 4, if person 1 is located in (5,0), and the initial location of the robot in state 0 was (0,0), then R(4) = 5. Similarly, if state 5 represents the robot is in (8, 0), R(5) = 3 since the robot arrived from state 3, which is located in (5, 0).

4. TESTING

For purposes of testing and evaluating the modelling of the problem, a system was developed consisting in two main components, as shown in Fig 2:

- A Problem Generator, which receives sensor data describing the location of each person in the area and generates a MDP Problem representation as outlined before

— A Standard MDP Solver implementing value iteration, which receives the problem representation as an input, and produces a Policy

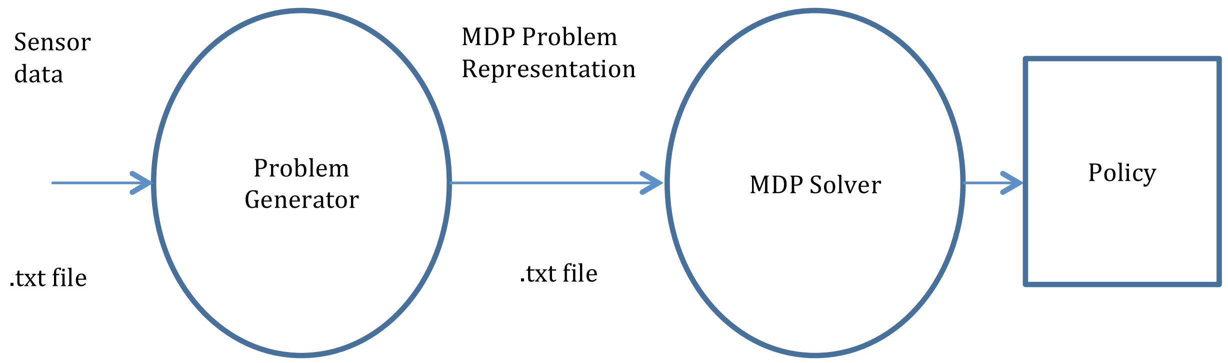

Fig. 2 System implementation

The first part of the testing consisted in generating problems for different amounts of people (1 to 10), verify correctness and calculate the number of nodes necessary to represent the problem. The Fig.3 shows that the number of states increases exponentially, this is because the structure of the MDP representation. If we look back at Fig. 2 it is possible to appreciate that from the initial state it is possible to reach 4 nodes (which we will call Level 0):

- State 3: The robot is in P1 location, P1 is not available, P2 is available
- State 4: The robot is in P1 location, P1 is available, P2 is available
- State 9: The robot is in P2 location, P1 is available, P2 is available
- State 10: The robot is in P1 location, P1 is available, P2 not is available

Then next level (non terminal states reached from 3, 4, 9 or 10) is reached when the robot fails to get help from a person in level 0, and moves to another one, there are 2 nodes for each remaining person. In this case two nodes for failing with person 1 in level zero:

- State 5: The robot is in P2 location, P1 is not available, P2 is available
- State 6: The robot is in P2 location, P1 is not available, P2 is not available

And two nodes after failing with person 2 in level zero:

- State 7: The robot is in P1 location, P1 is available, P2 is not available
- State 8: The robot is in P1 location, P1 is not available, P2 is not available

Following this logic it is possible to deduce that in a 3 people problem, there would be 3 * 2 states in the level zero, 3 * 2 * 2 in level one, and 3 * 2 * 2 in level three. Hence the equation that determines the total number of states for each problem representation is:

$$\sum_{k=0}^{x-1} \frac{2\,x!}{k!} + 3$$

Where x is the total number of people detected.

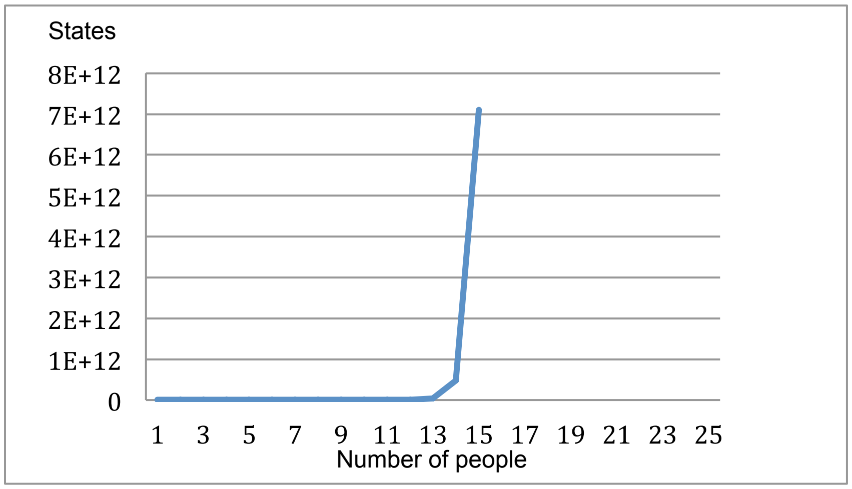

Figure 3. People vs. Nodes generated

The second part of testing consisted in generating 60 different scenarios, 10 per number of people (from 2 to 7) using a computer with a 3.16 GHz Intel Xeon processor (CMU Linux Timeshares), and measuring the time in which the MDP was solved it using value iteration. The following graph shows that the time is also exponential:

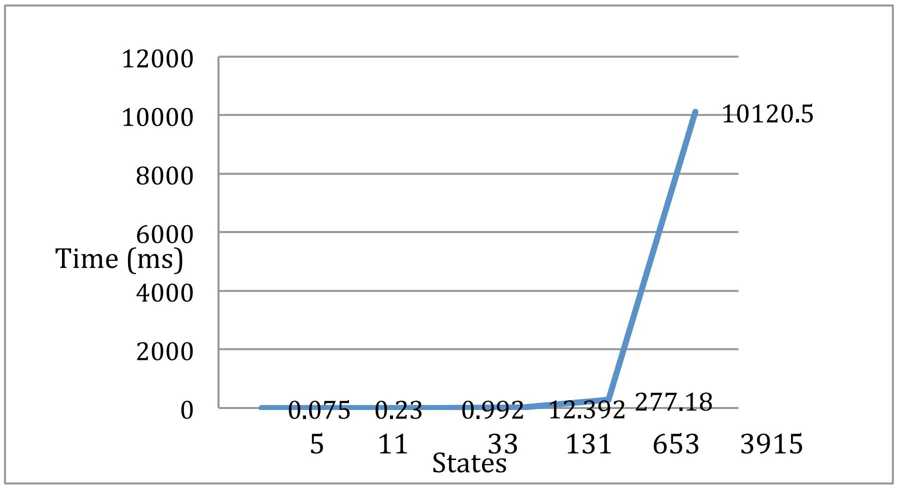

Fig 4. Average computing time

It is possible to see that for 5 states (corresponding to 2 people) the solving time was .000075 seconds, and even for 6 people (3915 states) the problem was tractable taking 10.12 seconds. For the problem with 7 people (27401 states) the system killed the process.

5. CONCLUSIONS AND FUTURE WORK

The modelling approach for this problem turned out to be exponential in state space and computing time. Nevertheless for up to 5 people, it could be computed in real time by taking .27 seconds, so one approach would be for the robot to generate the problem representation for the closest 5 people, and if it fails generate another with the remaining people in the area. Also given that the robot is supposed to operate indoors, typically the number of persons is expected to be in that order of magnitude.

As far as now, the probabilities for getting help were fixed, the next step would be to use reinforcement learning to learn those values, and then try to map them to the people characteristics, to use as a priori probability later. For example, the robot would learn a probability for a person seated and isolated several times, and we could try to determine if there is a correlation between this people characteristics and the probability of getting help from the person.

Another topic of future work would be to find heuristics or alternative methods to solve the MDP. One possibility would be not taking into account the cost of navigation to one location to another while generating the problem representation (which dramatically reduced the state space) and including it until the calculation of the policy, choosing the action with maximizes “Best expected value – Cost of navigating”.

5. REFERENCES

1. S. Rosenthal, J. Biswas and M. Veloso. An Effective Personal Mobile Robot Agent Through Symbiotic Human-Robot Interaction. Proceedings of AAMAS'10, the Ninth International Joint Conference on Autonomous Agents and Multi-Agent Systems, Toronto, Canada. Pages 915-922, 2010.
2. N. Armstrong-Crews and M. Veloso, "Oracular POMDPs: A very special case". ICRA '07, 2007
3. Weiss, A., Igelsböck, J., et al. Robots asking for directions: the willingness of passers-by to sup- port robots. In: HRI '10. Pages 23–30, 2010
4. Stephanie Rosenthal and Manuela Veloso. Modeling humans as observation providers using POMDPs. Proceedings of the 20th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN), Atlanta, GA, August 2011.
5. Stephanie Rosenthal, Manuela Veloso, and Anind Dey. Learning Accuracy and Availability of Humans who Help Mobile Robots. aaProceedings of AAAI'11, the Twenty-Fifth Conference on Artificial Intelligence, San Francisco, CA, August 2011.