Planning under Uncertainty for Search and Rescue

Ana Rita Mendes, Matthijs T.J. Spaan and Pedro U. Lima

Abstract—Partially observable Markov decision processes (POMDPs) provide a framework for planning under uncertainty. We present an application of POMDPs in a realistic search and rescue situation. Specifically, the aim of the problem is to find victims in a disaster environment. We will define, implement and test a POMDP model that suits the problem and its characteristics. The environment and its features will be learned (by letting the robot interact with it) and taken into account when building the model. It can be concluded that POMDP based solutions, and this solution in particular, work well in these search and rescue scenarios.

I. INTRODUCTION

Disasters, and rescues from disasters represent a worrying social issue. Fire scenes, major traffic accidents or building explosions are just a few examples of situations that represent very hazardous environments, where many human lives are endangered (not only victims, but rescue teams also). More often now, robotics come to help in these environments, even though most tasks are still the responsibility of humans. In order to be helpful, robots must learn how to behave in these environments. However, catastrophes and unplanned situations are not available for training and a reliable simulator, USARsim (Unified System for Automation and Robot Simulation), is used. Due to the existence of RoboCup Rescue Competition [1], many maps for rescue situation are available for this simulator, as well as stable code built by the participating teams in order to act in these environments.

In a simulated disaster environment, robots should be able to look for the victims in an optimal way even though their positions are unknown. Due to the uncertainty of victim positions, as well as other common uncertainty sources in robotics (sensors, actuators, etc.), the problem will be modeled as a Partially observable Markov decision process (POMDP) [4]. POMDPs form a powerful framework for planning under uncertainty and have been gaining in popularity in robotics. A topological map of the site is known, however, it should be kept in mind that the map represents a building in a rescue situation and as a result some of its structure might have changed. As such, we will learn some of the models of the POMDP, in particular how the robot can learn the parameters of the topological map.

We will build our work on both code and map from the competition of 2008 [1]. The code belongs to the Jacobs University team [6] and it will be responsible for all the low level control. The map represents the interior of an office building with several injured victims for the robot to find. A POMDP model will be built, learned and tested in this environment, interacting with the controller code, giving the robot instructions on what to do, while receiving observations on the environment (only on some of its features).

II. RELATED WORK

Applications of POMDPs in real world situations have been studied for very different fields over time [3]. One of the most common tasks which can be of interest for applying POMDPs is robot navigation. In [10] planning navigation under uncertainty is studied for long term autonomous navigation in office environments, where long corridors can be found and scenarios that are very hard to differentiate from each other predominate. In [2] Dynamic Bayesian Networks (DBNs) are used to model the uncertainty and POMDPs are used to plan navigation. Also on-line planning algorithms for solving POMDPs have been proposed [8], appropriate when some characteristics of the environment are not known for robot navigation.

In [9], it is demonstrated that POMDPs can be used for a health care situation where an agent should find and guide people to specific places. They conclude that POMDPs do indeed scale tractably to this real world problem, calculating policies within reasonable amounts of time. Still in the area of health care, in [5] the feasibility of assisting elderly people with limitations through interactive robotic devices is put to test. For that an implemented robot system is used, which relies heavily on probabilistic artificial intelligence (AI) techniques for acting under uncertainty. It is shown that human robot interaction with elderly people can be successfully achieved.

III. BACKGROUND

Automated decision making is desirable in many day to day situations (for example, flight controls, power network control, etc). Deciding how to behave in a particular way in a particular situation can be very important, as well as a challenging task. In this perspective, decision making has always been an area of interest for the scientific community. Planning problems aim to find a way (preferably an optimal way) to behave over time. Given different situations at different time steps, a specific action has to be taken, and the main goal of planning problems is to define these actions for each specific possible situation.

As planning implies, at some level, prediction, it can be done only with deep knowledge of the problem and the factors it depends on. As so, for a planning task to be able to foresee the possible occurrences in a given circumstance,
the problem has to be modeled, that is, reality has to be simulated. Moreover, with some environments, the agent cannot understand the state with full certainty from its sensor readings. This can be due to their features (when multiple states look alike) or due to the agent’s characteristics (when sensors can only sense a limited part of the environment). A partially observable Markov decision process, POMDP, is a framework for acting optimally when in such environments, where states are not fully observable and must be estimated from the observations.

A POMDP models an agent acting synchronously with the environment, aiming to find the best way to act at every time step. In order to do so, at every time step a reward (or a penalty) is given to the agent, so it can understand its goal, what it should do and what it should avoid doing. POMDPs can be described by:

- \( S \) a discrete and finite space state;
- \( A \) a finite set of possible actions, where \( A(s) \) represents the actions available at state \( s \in S \);
- \( T_{sa} \) the transitions probabilities (probability of going to state \( s' \) when in state \( s \) and given the action taken \( a \), \( p(s'|s,a) \));
- \( R_{sa} \) a reward function, giving the immediate reward for taking the action \( a \) when in state \( s \);
- \( \Omega \) the set of all possible observations;
- \( O_{oa} \) the observation model, representing the probability of being at state \( s' \) after taking action \( a \) and observing \( o \), \( p(o|s,a) \).

Since the state is not known, to be able to make decisions, some kind of memory would be necessary to keep track of (possibly) the entire history of the process. However, using a probability distribution over all of the states has the same information as keeping track of the complete history. That probability distribution is called a belief and the POMDP is simply an MDP with belief states instead of nominal states.

When an action is performed and an observation taken, the distribution has to be updated to reflect the new knowledge. More formally, let \( b(s) \) denote the probability assigned to world state \( s \) by belief state \( b \). Updating the belief is computing \( b' \) which can be obtained given the previous belief state \( b \), the action executed \( a \) and the observation \( o \) through:

\[
b'(s') = P(s'|a,b) = \frac{O(s',a,o) \sum_{s \in \Omega} T(s,a,s')b(s)}{P(o|a,b)}
\]

Where \( P(o|a,b) \) can be seen as a normalizing factor in order for \( b' \) to sum to one [4].

To represent and solve models in this paper we will use Symbolic Perseus [7], which is a point-based value iteration algorithm for POMDPs. Moreover, to represent transition, observation and reward functions in a compact way they are represented as Dynamic Bayesian networks (DBNs). A DBN is an acyclic graph that allows representation at the variable level instead of the state level [7]. Their graphs represent two consecutive time steps, with nodes as the variables (state, observation and reward variables) and connections as the probabilistic dependencies. Furthermore, Symbolic Perseus allows for exploiting context-specific independence by representing the conditional probabilities as Algebraic Decision Diagrams (ADDs). A DBN representing a simple POMDP is shown in Figure 1.

## IV. LEARNING ROBOT TRANSITION MODELS

In general, POMDP solvers require the transition, observation, and reward models to be known a priori. In this paper, we focus on methods for defining the transition model for a robot, by learning the transitions between nodes in a topological map. Figure 2 shows a real map from USARsim and the corresponding topological map representation.

Defining the transition model is one of the most important tasks when creating a POMDP model. The closer the transition model is to reality the better the model will work when in real situations. In this paper the environment is available for testing and learning the transition model, allowing the
we choose the time step to be used for the real tests. We arriving at the destination node, for all paths, based on which paths were executed successfully. Moreover, with no time limit all the $s$ have some uncertainty. With smaller time steps such as 20 and different time steps. It is very clear to see the difference in some paths for which 30 did not allow the robot to leave the node but with 50s it does indeed get to its destination with some probability. Moreover, with no time limit all the paths were executed successfully.

Figure 5 represents the average of the probability of arriving at the destination node, for all paths, based on which we choose the time step to be used for the real tests. We define the time step to be 50s as it allows most paths to have some uncertainty. With smaller time steps such as 20s or 30s for most paths the robot would never get to the goal while with 60s many paths were successfully completed. As the main goal of this problem is dealing with uncertainty, choosing a 50s time step seems the best option.

With the probabilities defined they have to be integrated in the generation of the model. However, probabilities that are 0 or 1 will not be used, as this would be a too strong statement and because we are dealing with uncertainty this would be limiting the problem. Saying that a path could never be done would be saying the robot should never choose to do that or a path with such a transition. Likewise, saying a certain path always leads arriving at the desired destination does not contemplate the possibility that, if something in the environment changes for some reason and the robot does not reach that destination, the model would not be prepared. So, keeping in mind that these environments always carry a great deal of uncertainty all probabilities that equal 1 will be set to a slightly smaller value and 0 probabilities will be set to a very small value, but bigger than 0.
V. POMDP MODELS FOR SEARCH AND RESCUE

In this section we will develop POMDP models for the victim localization task referred to in the introduction, which are based on the learned transition model as described in Section IV. Every problem can be seen from different perspectives and therefore defined and designed through different logical approaches. As such, for this specific planning problem, two models will be developed and explored.

A. Model A - victims as variables

First we consider a model in which the number of victims present in the environment is assumed to be known. A Dynamic Bayesian network representing the model is shown in Figure 6(a). Considering the map presented in Figure 2 with one victim in one of the nodes (any node) the aim of the robot is to find the victim. With this setup, the state variables will be the location of the robot and each victim (only one for one victim). Moreover, a third state variable, seen, will be needed (one for each victim) to keep track of which victims have been detected and allow a correct distribution of rewards. Victim variables and the robot variable each have 9 possible values (as many as the number of nodes). Seen variables have 3 possible values, no, yes and done.

The transition model for each seen variable can be described by: it starts with with value no, once the corresponding victim is detected the value changes to yes and in the next time step it changes to done. A victim is considered to be detected once the robot and the victim are in the same node. Victim variables have the identity matrix as transition model, as they never change their state (i.e., victims remain in the same node for the entire problem). The transition model for the robot has to be learned from interacting with the environment as detailed in Section IV. Regarding observations, the robot is considered to localize itself without any error (taking into account the controller used for testing, that is a valid assumption). For the victims, each has an observation model according to which if the robot and the victim are in the same node the victim is detected at that node with 99% chance, otherwise, the victim is not detected.

The reward function depends only on seen variables, and is the sum of a reward function per seen variable, as they are independent. When the variable is in state no or done the robot receives a small negative reward, when it is in state yes, the robot receives a large positive reward. Due to the fact that these variables only remain in state yes for one time step and can never go back to it, only one positive reward can be received per victim. In this way, we are preventing the robot from finding the same victim more than once, encouraging it to look for other victims.

To sum up, the model is described by its variables and their possible states (shown in Figure 6(b)), the transition model (for variables Robot and Seen1), the observation model (for variables Robot, Victim1) and the reward function (for variable Seen1).

B. Model B - Nodes as Variables

Another approach to the same problem with the same map (Figure 2), is to consider the nodes as variables, instead of the victims. Each node is represented by a variable, e.g., \textit{victimAtNode01} for node one. This variable can assume two values, yes and no, representing the presence or absence (respectively) of a victim in that specific node. This representation is very useful for when the number of victims present is unknown. The previous model assumed the number of victims were known from the start and that is not necessarily the case.

The seen variables are still needed to keep track of what has been seen before and what is new (allowing the correct distribution of rewards), representing which nodes have been
seen. As there are 9 victimAtNode variables, 9 seen variables are needed, with the same 3 possible values, yes, no and done. This leads to 19 state variables, one from the robot, 9 from victimAtNode variables plus the 9 seen variables.

The transition model of the robot is the same as for model A (explained in Section IV). The presence or absence of victims at a specific node does not change over time, as victims do not move. As a result, variables concerning victims (victimAtNode) do not have transition models. As the seen variables keep track of visited nodes, the transition model is slightly different now. All nodes start with seen in the no state. Once a node is visited if there is a victim in it Seen variable will switch to yes and in the next state it will switch to done. If a victim is not present, the node is most likely considered visited and without a victim, so with 99% chance seen changes to done (staying in no with 1% to contemplate the small possibility that the victim was there but the robot did not see it).

Regarding observations, the robot is once more considered to localize itself without any error. For the victim nodes the idea is the same as with victim variables in model A. If the robot is in node x and a victim is there too (victimAtNode magic yes) the observation for victimPresent, will be yes with 99% chance (and no with 1% chance). The reward function is the same as for model A. Each seen variable is responsible for a reward. A negative reward for states no or done and a big positive reward for state yes, which happens only once (or never) per variable.

To sum up, the model is described by the variables and their possible states (shown in Table 7(b)), the transition models (for variables Robot, Seen1, Seen2, ..., Seen9), the observation model (for variables Robot, victimAtNode1, victimAtNode2, ..., victimAtNode9) and the reward function (for variables Seen1, Seen2, ..., Seen9). A Dynamic Bayesian network representing the model is shown in Figure 7(a).

VI. EXPERIMENTS

The basic structure of the whole system needed for the experiments is represented in Figure 8. The models defined in the previous section are generated by the generatePOMDP function. Then the model is solved (through solvePOMDP), which runs Symbolic Perseus [7] to compute the policy. There are two ways for testing a POMDP policy. The POMDPsimulation function, which simulates the environment, i.e., all variables are simulated according to their specifications and the POMDP policy is tested within its own model. The USARsim function, on the other hand, is ready to deal with real environments, that is, this function interacts with the Jacobs controller, which in turn operates on the USARsim simulation.

With these two functions it is possible to observe and compare the estimated behavior with the simulated real behavior. The more similar their behaviors are the better the model describes the environment. That is, with more realistic and accurate transition and observation models reflecting the behavior of the system it is possible to obtain better plans.

For using the models with the USARsim environment, the basic structure of the whole system is represented in Figure 8. Both the USARsim function and the controller code were adapting in order to be able to communicate with each other.

In this section the models will be tested for their behavior. Moreover, model B, due to its large number of variables is also studied regarding the model complexity.

A. Experiments with model A

Model A, as described in Section V, is solved through the solvePOMDP function. Given its reduced number of

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>goTo9</td>
<td>09</td>
<td>No</td>
<td>-0.65</td>
<td>-0.65</td>
</tr>
<tr>
<td>2</td>
<td>goTo8</td>
<td>08</td>
<td>No</td>
<td>-0.59</td>
<td>-1.24</td>
</tr>
<tr>
<td>3</td>
<td>goTo5</td>
<td>05</td>
<td>No</td>
<td>-0.52</td>
<td>-1.76</td>
</tr>
<tr>
<td>4</td>
<td>goTo6</td>
<td>06</td>
<td>No</td>
<td>-0.43</td>
<td>-2.19</td>
</tr>
<tr>
<td>5</td>
<td>goTo5</td>
<td>05</td>
<td>No</td>
<td>-0.97</td>
<td>-3.16</td>
</tr>
<tr>
<td>6</td>
<td>goTo7</td>
<td>07</td>
<td>No</td>
<td>-0.29</td>
<td>-4.42</td>
</tr>
<tr>
<td>7</td>
<td>goTo5</td>
<td>05</td>
<td>No</td>
<td>-0.95</td>
<td>-5.37</td>
</tr>
<tr>
<td>8</td>
<td>goTo1</td>
<td>05</td>
<td>No</td>
<td>-0.95</td>
<td>-6.32</td>
</tr>
<tr>
<td>9</td>
<td>goTo1</td>
<td>01</td>
<td>No</td>
<td>-0.07</td>
<td>-6.39</td>
</tr>
<tr>
<td>10</td>
<td>goTo4</td>
<td>01</td>
<td>No</td>
<td>-0.93</td>
<td>-7.32</td>
</tr>
<tr>
<td>11</td>
<td>goTo4</td>
<td>04</td>
<td>No</td>
<td>-0.34</td>
<td>-7.66</td>
</tr>
<tr>
<td>12</td>
<td>goTo1</td>
<td>04</td>
<td>No</td>
<td>-0.90</td>
<td>-8.56</td>
</tr>
<tr>
<td>13</td>
<td>goTo1</td>
<td>01</td>
<td>No</td>
<td>-0.90</td>
<td>-8.46</td>
</tr>
<tr>
<td>14</td>
<td>goTo2</td>
<td>02</td>
<td>02</td>
<td>50.000</td>
<td>40.54</td>
</tr>
</tbody>
</table>

(a) Solve POMDP. (b) Evaluate using POMDP models. (c) Evaluate using USARsim.

Fig. 8. Experimental setup, showing the different components of the evaluation setup.

TABLE I

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>goTo9</td>
<td>-1.00</td>
<td>09</td>
<td>No</td>
<td>-1.00</td>
</tr>
<tr>
<td>2</td>
<td>goTo8</td>
<td>-0.65</td>
<td>08</td>
<td>No</td>
<td>-1.65</td>
</tr>
<tr>
<td>3</td>
<td>goTo5</td>
<td>-0.59</td>
<td>05</td>
<td>No</td>
<td>-2.24</td>
</tr>
<tr>
<td>4</td>
<td>goTo6</td>
<td>-0.52</td>
<td>06</td>
<td>No</td>
<td>-2.76</td>
</tr>
<tr>
<td>5</td>
<td>goTo5</td>
<td>-0.43</td>
<td>05</td>
<td>No</td>
<td>-3.19</td>
</tr>
<tr>
<td>6</td>
<td>goTo7</td>
<td>-0.97</td>
<td>07</td>
<td>No</td>
<td>-4.16</td>
</tr>
<tr>
<td>7</td>
<td>goTo5</td>
<td>-1.00</td>
<td>05</td>
<td>No</td>
<td>-5.16</td>
</tr>
<tr>
<td>8</td>
<td>goTo2</td>
<td>-1.00</td>
<td>05</td>
<td>No</td>
<td>-6.16</td>
</tr>
<tr>
<td>9</td>
<td>goTo2</td>
<td>-1.00</td>
<td>05</td>
<td>No</td>
<td>-7.16</td>
</tr>
<tr>
<td>10</td>
<td>goTo2</td>
<td>-1.00</td>
<td>05</td>
<td>No</td>
<td>-8.16</td>
</tr>
<tr>
<td>11</td>
<td>goTo2</td>
<td>-1.00</td>
<td>02</td>
<td>02</td>
<td>-9.16</td>
</tr>
<tr>
<td>12</td>
<td>goTo4</td>
<td>50.00</td>
<td>02</td>
<td>02</td>
<td>40.84</td>
</tr>
</tbody>
</table>

(b) POMDPsimulation environment.
variables its complexity is low and a policy is computed within a couple of minutes. The model is then submitted to a few tests, in order to check its behavior in the USARsim simulated environment. Moreover, this behavior is compared with simulations with POMDPsim where the position of the victims is given as an input, so both situations can be compared. Table I shows the behavior of the system for one test situation compared with the simulated behavior.

A small difference between USARsim and POMDPsim functions should be mentioned. With POMDPsim the reward is given to the robot right after the action in chosen, i.e., before the robot acts and observes while with USARsim the reward is given when the robot acts and observes. This difference results in the robot receiving the reward one iteration later in POMDPsim that with USARsim. This has to be taken into account when comparing the number of iterations and the total reward of both runs (simulated with POMDPsim and tested in USARsim).

Moreover, it is possible to observe some differences between the plans executed in USARsim and in POMDPsim functions should be mentioned. With POMDPsim the reward is given to the robot right after the action in chosen, i.e., before the robot acts and observes while with USARsim the reward is given when the robot acts and observes. This difference results in the robot receiving the reward one iteration later in POMDPsim that with USARsim. This has to be taken into account when comparing the number of iterations and the total reward of both runs (simulated with POMDPsim and tested in USARsim).

B. Experiments with model B

Model B, due to the high number of variables (and states), is too complex to solve with the specified settings in Section V. The computational resources (RAM memory) needed to run the function solvePOMDP on this model are not available. In model B, as the number of variables grows with the number of nodes (at a rate of two state variables and one observation variable per node), large maps become impossible to solve, due to memory limitations. As such, the model was tested in simulation (with POMDPsim) and in USARsim (with USARsim) with a smaller version of the problem, for only 5 nodes, instead of 9 (Figure 9(a)).

The robot started in node 3 and victims could be found in nodes 4 and 2. Table II shows both runs, simulated with POMDPsim and tested in USARsim with USARsim. Figure 9(b) shows the path the robot did during the USARsim test. It has to be reminded that the transition model used for this model is the same as for model A, as the environment is the same. Once more, the model seems to represent reality well enough for the policy to execute a good plan.

Reducing complexity can also be accomplished by lowering the number of observation variables and/or number of observations. This solution implies that some changes are made to the model. Two situations are studied in order to decrease the complexity of the model.

In the first situation (Model5) the number of observation variables is decreased from as many as the number of nodes to only one. This variable represents observations for all nodes, thus the number of possible observations increases as it represents observing the victim in any of the nodes and not observing the victim. This solution is slightly limiting the problem, as it means in each time step no more than one observation is possible.

The second situation (Model6) is accomplished simplifying the previous one (Model5). Still only one observation variable is used to observe any victim in any of the nodes, however, that is achieved only through two possible observations, hence reducing considerably the number of observations. The possible observations are seeing or not seeing a victim. As such, information on the victim’s position is not so clear. The robot is assumed only to detect a victim when in the same node as the robot and seeing a victim in a close node would be, in this model, interpreted as seeing it in the close node the robot is in, as the observation for seeing victim does not allow any information regarding the position.

The complexity was measured through the time taken to
solve the POMDP. Figure 10 allows the computation times for Model5 and Model6 to be compared with Model1. It is possible to observe that decreasing the number of observations represents a much lower complexity that cannot be achieved simply through altering the order of the variables. However, the models representing the whole desired area (9 nodes) are still uncomputable given the available resources.

VII. CONCLUSIONS

In this paper POMDP models were successfully developed for planning under uncertainty in search and rescue situations, in which a robot has to locate victims. Dealing with uncertainty is an area of great scientific interest, and can have a positive effect on socially relevant planning problems such as search and rescue.

We showed that the POMDP approach does work well in realistic situations. Moreover, it provides some major advantages that need to be mentioned. A POMDP framework allows models to be compact and simple to perceive due to its factored representation. It enables the state space, observations, reward functions, etc. to be represented by a set of variables. This also enhances the process of solving the POMDP, as the independence between the variables can be exploited.

The main goal of this project was to build a POMDP model from scratch, specifically thought for the problem. Two different models were developed, contemplating two different situations, and at completely different costs. One model allows for a much faster computation but with the downside of needing some previous knowledge regarding the number of victims present at site. The other model, on the other hand, allows for a completely arbitrary number of victims but with computational time growing very fast with the size of the map (more accurately, with the number of nodes).

Moreover, transition probabilities were learnt, allowing the POMDP models to take into account environment features and the characteristics of both robot and controller. This is crucial for planning problems and it represented a major concern in this paper, as it models the way a plan would be executed by the system.

In future work, we would like to take into account the fact that in real rescue situations the exact same environment would never be available for learning the transition model previously. Even if the transition model was learned for the same area, some walls could be down, some new paths open and some hazardous obstacles along the way. Designing POMDP algorithms that can efficiently changing transition models remains an important and relevant challenge.

REFERENCES


