Human-Robot Coordination for Disaster Relief Scenarios

Kevin Yu and Ryan Williams

Abstract— **This work looks at human-robot teams for disaster relief scenarios. Given a natural disaster first responders have to quickly and efficiently look at ways of cleaning up the disaster area as well as finding and rescuing victims that may be hurt. In a disaster scenario there are also many problems when dealing with pre-built maps because in any disaster scenario the area will have drastically changed based on the severity of the disaster.**

I. Introduction

Robotics have come a long way in the recent years, but there still leaves much to be desired when looking at human-robot teams. Robots are used in many different scenarios such as assembly lines in factories, agriculture monitoring, package delivery, search and rescue, and much more. All of these applications work with either only robotic teams or with a human as a high level operator. Many human-robot teams just look at using a human as a high level operator, treating a human as another type of agent with multiple constraints, or treating them as part of the environment. In my project I will look at how the paper written by Tair et al. [1] formalizes the creation of a human-robot team as well as talk about my implementation. I talk about what what I find interesting in the paper and also some aspects of the paper that I find are faults in their foundation of human-robot teams.

The rest of the paper will follow a structure that starts with related work in Section II, which will go over the contribution from the paper [1]. Then we will look at the differences that I make to the problem and how I formalize the human-robot team in Section III and Section IV. We will look at my implementation and the results in Section V and Section V-B. Finally we have the conclusion in Section VI with potential future work.

II. Related Work

In the paper written by Tair et al. [1] they look at formalizing a human-robot team with Decentralized Multi-agent Partially Observable Markov Decision Process (DEC-POMDP). This paper goes over just the problem formulation of a DEC-POMDP and how they formulate it for human-robot teams. The authors don't actually show any simulation results are real world experiments. In this section we will look over exactly what the authors do in this paper and some faults that I believe make their paper weak.

A. Problem Set-up

The authors of the paper look at formalizing the problem of human-robot teams with a DEC-POMDP. The authors state a Dec-POMDP as a tuple of $< n, S, A, T, R, O, o, h, b^0 >$. Below we will state exactly what each one of them means:

- n, finite set of agents;
- S, finite set of world states;
- A, joint set of actions available for the team of agents at each time step;
- T, transition function defined as the probability of going from state s to state s' taking action a;
- R, reward function defined as the reward given to the team;
- O, joint set of observations;
- o, observation function defined as the probability of an observation when in a state;
- h, time horizon;
- b^0, initial state distribution at time 0.

Using a multi-agent frame work they rely on a centralized agent to give a joint set of actions to the distributed agents. The authors then send the joint set of actions to all agents and have them execute them in a decentralized manner, Figure 1. These sets of actions are obtained from the joint policy that is created offline.

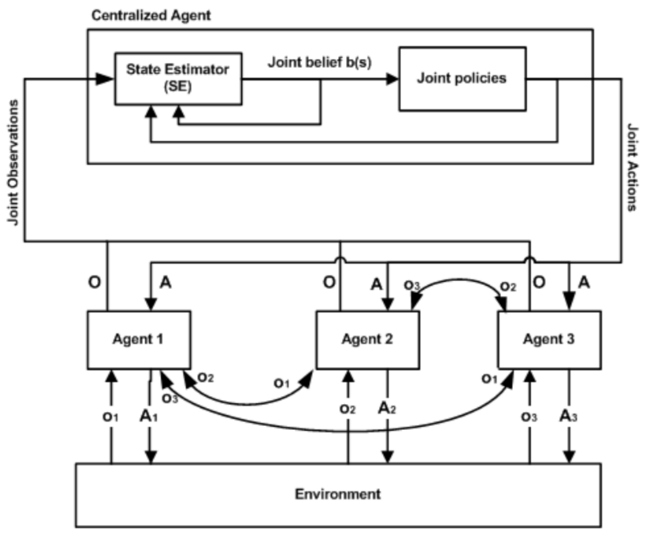


Fig. 1. Dynamics of multi-agent human-robot interactions [1].

The authors use a simple environment, that I also model for my implementation, for testing their problem formulation, Figure 2. The authors also show the frame work for their Dynamic Bayesian network (DBN), which is used for the update of the belief state of an agent 3. This DBN is needed because the robots have to share information and come to consensus on the joint belief state.

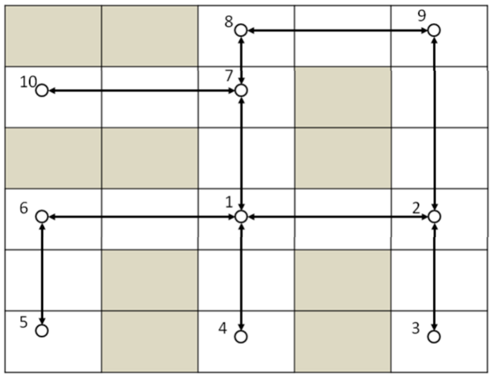


Fig. 2. 6x5 domain showing the discretization of the environment with 10 modes [1].

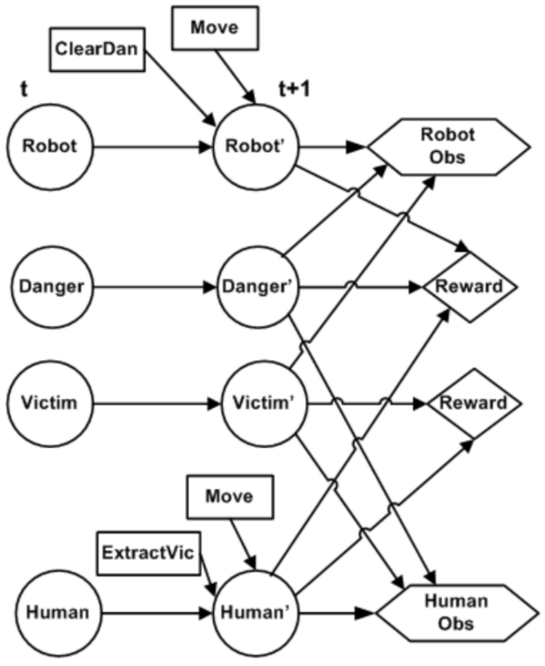


Fig. 3. POMDP representation using DBN [1].

The previous paper [1] states the actions as $< up$, $down$, $right$, $left$, $stop$, $clearDanger >$ for a robot agent and $< up$, $down$, $right$, $left$, $stop$, $extractVictim >$ for a human agent. The state space is the environment shown in Figure 2. The authors give the possible observations as $< victimanddanger$, $victimandnodanger$, $novictimanddanger$, $novictimandnodanger >$. These observations correlate to the types of rewards that can be obtained and allow us to formulate the problem as a POMDP instead of just a MDP. The observation function is deterministic, meaning that there is no probability of getting an incorrect observation. I believe the authors do this to make the initial formulation of the problem simpler and give room for expansion by formulating it as a DEC-POMDP. The rewards are -1 for every small step, -50 if a human enters a node that contains danger, $+50$ if a robot clears danger, and $+100$ if a human extracts a victim. Lastly the authors also make the transition functions deterministic as well. Similar to the observation function we believe that an agent will enter the state it expects to with 100% probability.

The authors only look at formulating the problem. Therefore they only state that they are able to use the Multi-agent Decision Process (MADP) toolbox to solve for optimal policies and don't actually show any implementation of the joint policy.

B. Drawbacks to this Paper

Some things that I didn't like about the problem formulation is that the authors assume the human as another type of agent. The authors do this by giving a transition function that is deterministic as well as observations deterministic. By doing this they convert the problem into a heterogeneous multi-agent problem, instead of directly addressing some of the nuances of dealing with humans. Something else that I didn't like about the paper was that the authors don't give any insight about the policies they made, potentially the time it took for this problem.

III. Problem Formulation

I went with duplicating the problem in [1], but changing a few things to make a more unique and interesting problem. I left the state space, action set, and observation set the same as the previous problem. I also decided to leave the transition function, observation function, and reward function the same. This meaning that the transition function and observation function are deterministic to allow for simplicity of this project and the reward function is -1 for every small step, -50 if a human enters a node that contains danger, $+50$ if a robot clears danger, and $+100$ if a human extracts a victim. The biggest change that I implemented was when I created the policy for the humans. Instead of believing that the humans will follow some optimal policy created by MADP I looked at creating a greedy policy for the humans and having robots mold their policies to the humans.

IV. Human-Robot Team Formalization

For a human-robot team there are many things that need to be taken into account, such as the unreliability of human movement, the unpredictability of human actions, and even the communication cost for humans and robots. Even though I did not directly address these problems in my implementation I believe to make a strong human-robot team you have to be able to characterize these aspects of humans and robots. For the unreliability of humans we can encode that in the belief update. We can't place that in the transition function because the transition function is based on the action you want to take not taking into account the intended action. We can also explore the ways of making the creation of a policy online to deal with the unpredictability of humans. There are methods like Dynamic Programming (DP) and auctioning [2] that allow for the re-mapping of policy to agents. By using some of these methods we can handle the unpredictability of humans.

Lastly communication cost is something that can be very taxing in disaster scenarios. In many cases there isn't even a structured way of communicating between teams. When looking at the communication between agents in a team we need to understand more on human first responders. The authors of the paper [1] actually state as one of their future works as looking into how first responders deal with disaster scenarios. I believe that we might also be able to leverage methods such as self-triggering [3]. The idea behind self-triggering is that agents will work in a collaborative manner to achieve some goal. When making movements agents follow some global collaborative movement assigned to it, but don't communicate it's information over to the other robots. Once an agent has hit a predefined time then the agent will communicate it's information to others. This is allows agents to come to consensus on when they should communicate next and stating that we won't communicate until then because we don't need to and it will help save communication costs. Overall, it would be beneficial to have teams work by themselves and only communicate when needed. Another method that I have thought about is having something like phone operators. The idea behind this would be that the phone operator could help connect two parties who need to talk to convey information, but they could also act as filters to hold information and relay it at potentially less stressful times to the other agent. Telephone operators also could make sure agents are sending information to the correct personnel and would have an overall picture of the disaster scenario allowing for overall updates to all agents as well. These methods may give a sub-optimal solution, but could reduce strain on human-robot teams and potentially bundle information to help reduce communication costs.

V. My Implementation

For my implementation I wrote it in Python and used only two agents. I created the same environment setting as the authors did in [1]. I then decided to look at how to create different policies for the different agents. I first decided to make a greedy policy for humans because they potentially won't follow an optimal policy at all and always go to what they see first. For the greedy policy the robot will try to go to nodes that are closest to it and always extract victims right away. I then looked at how robots should react based on those policies as well. I went with robots having a policy that tried to go the exact opposite direction of a human, unless they got the observation that had danger in it. If the robot got an observation of danger it would execute it's greedy policy of clearing danger. For this policy the robot might not always got the exact opposite way because there could be a wall there, but in principal the robot will always try to execute an action that is as opposite as possible to the human. By doing this we are able to cover the most area of an environment because the two agents will be executing opposite control functions. I wanted to look at the completion time of the environment if I implemented these two types of policies. We can see in Table I the number of steps it takes to fully explore a neutral environment. I compared 4 different combinations of agents and policies, one human with random movements, one human with a greedy policy, one human one robot with random policies, and one human one robot with the human taking a greedy policy and the robot taking an opposite policy. The results give are intuitive stating that more agents means faster times as well as smarter agents also mean faster times.

I then looked at making the environment more interesting by adding the ability to earn rewards, such as clearing danger and extracting victims. I decided to make nodes 2 and 9 as interesting nodes. Node 2 is a victim node and node 9 is a danger node. Due to the fact that we have one node that is a danger node we can not run a team of less than two agents because we need the uniqueness of each agent.

TABLE I
Number of steps and corresponding rewards for all nodes being empty 2.

Type of Agents	1 Random	1 Agent Smart	2 Agents Random	2 Agents Smart
Steps	301	33	87	20
Rewards	-300	-32	-172	-36

TABLE II
Number of steps and corresponding rewards for all node 2 being victim and 9 being danger 2.

Type of Agents	2 Agents Random	2 Agents Smart
Steps	251	21
Rewards	-447	12

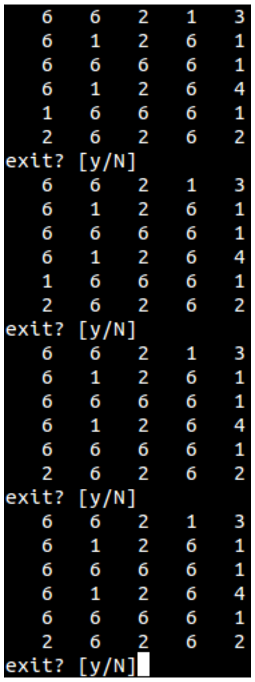
Even though this is not an actual implementation formulating the communication between agents within a human-robot team is important. When communicating I believe that there should be a cost function that directly incorporates the physical costs of communication as well as the mental costs of communication between humans. There is also a possibility of creating a Linear Temporal Logic (LTL) for communication. For instance we can create high level temporal logic that can help the flow of information. Some of the blocks in our communication logic would be; *Humans overwhelmed*, *Robots uncertainty is high*, and *explore*. With just these three types of blocks we can make a LTL formula that states: Eventually *explore* all states and Communicate if (not *Humans overwhelmed* and *Robots uncertainty is high*). Currently this is a very high level thought of how to deal with the communication, but if we were able to implement LTL in communication we could give guarantees on the LTL formula, as a high level planner, and use a low lever planner on optimizing the time it takes to explore a disaster scenario.

A. How to Run Everything

For my implementation I have created everything in Python. By doing this it is easy to run as well as it is easily expanded to Robotics Operating System (ROS). Being able to potentially migrate this algorithm to ROS I allow for an easier merging into real world experiments.

B. Results

You can see from Table I that the number of steps goes down as well as the reward increases. This is for a completely neutral environment that doesn't have any ability to gain rewards. In a sense this is the problem of coverage of an environment instead of search and rescue. In Table II you can see the affects of getting more agents and having smart agents for an environment where the agents can gain rewards. Since I did not run a an optimal policy maker I don't know what the optimal reward is, but I am able to see the affects of agent numbers and policies. I also show some snap shots of my code when running it, Figure 4.



```
   6   6   2   1   3
   6   1   2   6   1
   6   6   6   6   1
   6   1   2   6   4
   1   6   6   6   1
   2   6   2   6   2
exit? [y/N]
   6   6   2   1   3
   6   1   2   6   1
   6   6   6   6   1
   6   1   2   6   4
   1   6   6   6   1
   2   6   2   6   2
exit? [y/N]
   6   6   2   1   3
   6   1   2   6   1
   6   6   6   6   1
   6   1   2   6   4
   6   6   6   6   1
   2   6   2   6   2
exit? [y/N]
   6   6   2   1   3
   6   1   2   6   1
   6   6   6   6   1
   6   1   2   6   4
   6   6   6   6   1
   2   6   2   6   2
exit? [y/N]
```

Fig. 4. Screen shot of the code running in terminal. 6 is an occupied or already visited grid cell, 2 is a node grid cell, 1 is an empty grid cell, 3 is a danger node, and 4 is a victim node.

VI. Conclusion and Future work

In this report we look at formulating a human-robot team for disaster scenarios. We start off with how the authors of [1] formalize the problem and then we look at how to formulate the problem statement correctly. Then after creating a concrete formulation of a human-robot team we look at how robots have to change their policies to account for the partially random behavior of humans. I believe that there is a lot of expansion in the current state of the art in this area because of the lack of representing humans well as well as the minimal use of a DEC-POMDP.

I show my full implementation of the paper as well as possible formulations on how to incorporate communication in human-robot teams. Starting with high level thoughts of having a mitigator for the information (scheduler) and then going into possible usage of LTL to help give guarantees on the communication between agents.

References

[1] H. Al Tair, T. Taha, M. Al-Qutayri, and J. Dias, "Decentralized multi-agent pomdps framework for humans-robots teamwork coordination in search and rescue," in *Information and Communication Technology Research (ICTRC), 2015 International Conference on*. IEEE, 2015, pp. 210–213.

[2] J. Capitan, M. T. Spaan, L. Merino, and A. Ollero, "Decentralized multi-robot cooperation with auctioned pomdps," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 650–671, 2013.

[3] L. Zhou and P. Tokekar, "Active target tracking with self-triggered communications," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2117–2123.