Continual Planning for Search and Rescue Robots

Luis Pineda, Takeshi Takahashi, Hee-Tae Jung, Shlomo Zilberstein, Roderic Grupen
College of Information and Computer Sciences,
University of Massachusetts Amherst, MA 01003, USA
{lpineda,ttakahas,hjung,shlomo,grupen}@cics.umass.edu

Abstract— **The deployment of robots for emergency response tasks such as search and rescue is a promising application of robotics with growing importance. Given the perilous nature of these tasks, autonomous robot operation is highly desirable in order to reduce the risk imposed on the human rescue team. While much work has been done on creating robotic systems that can be deployed for search and rescue, limited work has been devoted to devise efficient real-time automated planning algorithms for these tasks. In this work, we present REDHI, an efficient algorithm for solving probabilistic models of complex problems such as search and rescue. We evaluate our algorithm on the search and rescue problem using both an abstract domain representation and a semi-realistic simulator of an existing robot system. The results show that REDHI can obtain near optimal performance with negligible planning time.**

I. Introduction

We are concerned with the deployment of robots for emergency response tasks such as urban search and rescue (USAR) [1], wilderness search and rescue (WiSAR) [2] and natural disasters [3]. Robots can be of immense values on these tasks by helping the human rescue team to explore potentially dangerous territory, neutralize environmental threats and assist in the evacuation or treatment of victims.

Given the dangerous nature of emergency response tasks such as search and rescue, autonomous robot operation is highly desirable to reduce the risk of the humans involved in the task. In this sense, a crucial component to achieve autonomous operation is the planning algorithm used by the robot, which must take into account uncertainty about the robot's observations, its actions and the underlying state of the environment. Although much work has been devoted to developing robotic systems that can be used for search and rescue [4], [5], [6], [7], [8], limited work has been done on creating automated planning algorithms for this problem.

A popular model for probabilistic planning is the Markov Decision Process (MDP) [9]. MDPs offer a highly expressible representation that can model uncertainty about an agent's operation and its knowledge of the surrounding environment. Moreover, the model can be extended to handle partial observability [10] and multiple cooperating agents [11]. However, solving MDPs is computationally intractable in general, a problem that has limited their application to time-sensitive tasks such as search and rescue robots.

The contribution of our work is an efficient approach to solve MDPs based on the combination of *reduced models* and *hindsight optimization*, integrated into a continual (online) planning framework for solving MDPs. The resulting algorithm, REDHI, is able to solve MDPs extremely fast and with close to optimal performance. By using reduced models, REDHI makes it possible to increase the scalability of hindsight optimization approaches. We evaluate the use of our algorithm for search and rescue robots using a semi-realistic simulation of the uBot [12] robotic system.

II. Related Work

The use of reduced models to solve MDPs has received considerable attention in the planning community. Determinization-based approaches have gained popularity in recent years, after the surprising success of a deterministic online planner, FF-Replan [13], in the first international probabilistic planning competition [14]. More robust versions of this idea have been developed afterward and are now some of the state-of-the-art planners for goal oriented MDPs [15], [16]. Some of these are closely related to our work due to the use of hindsight optimization [17], [18].

Additionally, our formulation of the search and rescue problem is a cost-based variant of the *conformant probabilistic planning* (CPP) problem [19]. State-of-the-art approaches for CPPs use heuristic search and weighted model counting [20], compilation into metric-planning problems [21], and relevance-analysis combined with a regular conformant planner [22]. The last two approaches have some similarities to our work, in that they use a form of model reduction to make the problem more tractable—albeit a different type of reduction than the one we propose. Moreover, our formulation attempts to maximize some notion of utility, while typical CPPs focus on guaranteeing goal reachability.

III. Problem Formulation

In this section, we describe the mathematical framework used in this work: a special type of Markov Decision Process (MDPs) called Stochastic Shortest Path Problems (SSPs). We then give a high-level description of the search and rescue scenario that is the focus of this work and its formal description as an SSPs.

A. Stochastic Shortest Path Problems

An SSP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, s_0, \mathcal{G} \rangle$, where $\mathcal{S}$ is a finite set of states; $\mathcal{A}$ is a finite set of actions; $\mathcal{T}(s'|s,a)$ is a stationary transition function specifying the probability of outcome state s' when action a is executed in state s; $\mathcal{C}(s,a)$ is the positive cost of executing action a in state s; $s_0 \in \mathcal{S}$ is a given start state; and $\mathcal{G} \subset \mathcal{S}$ is a set of absorbing goal states. Starting in state s_0, the objective is to reach one of the goal states while minimizing the expected cumulative cost.

An optimal solution to an SSP can be represented as a policy π^* that maps each (reachable) state to an action, $\pi^* : \mathcal{S} \rightarrow \mathcal{A}$. The well-known Bellman equation defines a value function over states, $V^*(s)$, from which the optimal policy π^* can be extracted:

$$V^*(s) = \min_a [\mathcal{C}(s,a) + \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s,a) V^*(s')] \qquad (1)$$

B. Search and Rescue Scenario

In the search and rescue problem a robot must explore an environment described by a known map, with the goal of finding and rescuing an unknown number of victims within some time limit T_{max}. The robot is given a set of L locations that potentially contain victims, and it starts execution at another special location called *BASE*. Two types of victims can be present: ambulatory victims are those that can move with the robot's assistance, while non-ambulatory victims cannot be moved and they must receive treatment on the spot. Additionally, the environment can contain hazards, which must be cleared by taking them to a designated location.

This problem can be formulated as an SSP. States are defined as tuples $(l, t, \langle b_1, b_2, ..., b_L \rangle)$, where $l \in \{BASE, 1, ..., L\}$ represents the robot's location, t is the time elapsed since the robot started execution (measured in some discrete units) and $\langle b_1, b_2, ..., b_L \rangle$ is a vector representing the robot's knowledge about objects present in the environment (victims or hazards). Every b_i can have one of five possible values: *unknown*, *vict-amb*, *vict-non-amb*, *hazard* and *nothing*; we denote the set of these values as $\mathcal{O}$. The initial state s_0 is defined by $l = BASE$, $t = 0$ and $\forall_{1 \leq i \leq L} b_i = unknown$.

The robot can choose between 4 different types of actions:

- *MOVE* to any location l such that $b_l \neq nothing$.
- *EVACUATE* an ambulatory victim at the robot's location by moving the victim to *BASE*.
- *TREAT* a non-ambulatory victim at a specified location. The robot moves to *BASE*, picks up medicine and brings the medicine back to the victim at the specified location.
- *DECONTAMINATE* a hazard at the robot's location by moving it to *BASE*.

With the exception of *MOVE* actions, all actions are deterministic and, after execution, the object that they act on (victim/hazard) is ignored for the remaining of execution. On the other hand, in the case of *MOVE* actions, assuming perfect sensing, there is uncertainty about what the robot will observe upon reaching the target location. This uncertainty is described by a size $L \times 4$ matrix P_{obs} specifying, for each location, the probability that it contains an ambulatory victim, a non-ambulatory victim, a hazard or nothing. The underlying assumption is that the probability of location l containing a victim/hazard is independent from the probabilities of other locations containing victim/hazards. Accordingly, the transition function for moving to location l is defined so that, for each $x \in \mathcal{O}$ it generates a successor with probability $P_{obs}[l, y]$, where y is the column in matrix P_{obs} corresponding to observation x.

Regarding the time increments, each action a has some time length ΔT_a associated with distance traveled as well as the time invested in picking up victims/hazards. That is, for any action a taken at time step t, the resulting state will have time $t' = t + \Delta T_a$. After time $t' > T_{max}$ the robot is not allowed to take any more of its normal actions. To avoid the resulting planning dead-end, the robot is given a special action, *END*, that leads to an absorbing goal state at no cost.

The cost of all actions is directly related to the time elapsed since the robot started operation. We reward the robot for rescuing victims and clearing hazards as quickly as possible and there is no penalty directly associated to moving; that is, there are no costs associated with factors such as energy consumption. Therefore, the cost of *MOVE* actions is set to 0, while the cost of any other action a taken at time t is $(t + \Delta T_a) - T_{max} - R_a$ if $(t + \Delta T_a) \leq T_{max}$, and 0 otherwise; R_a is a constant reward associated with the type of action the robot executed. In other words, the later the robot completes a rescue/decontamination action, the higher the cost incurred for that action. Note that under this formulation action costs are always negative, so the robot has an incentive to take rescuing/decontamination actions in order to minimize the cost incurred before it ends operation (the negative-cost formulation is equivalent to maximizing the reward obtained by rescuing victims early).

IV. Solution Approach

Solving SSP is generally intractable, since the size of the state space increases exponentially with the size of any compact problem representation. This hinders the use of SSPs for the highly time-sensitive search and rescue problem. To address this challenge, in this work we adopt a *continual planning* paradigm. Continual planning refers to a variety of methods for interleaving planning and plan execution in situations where complete offline planning cannot address all runtime contingencies [23], [24]. Typically, a continual planning method does planning on a reduced version of the original problem so that planning can be done much faster. If execution progresses beyond this reduced model, the plan is updated to account for previously unforeseen events, and execution proceeds using the new plan.

A. Reduced Model

Our continual planning approach is based on the $\mathcal{M}_l^k$-reduction for MDPs [25]. An $\mathcal{M}_l^k$-reduction divides the sets of outcomes for each action into two types: *primary* and *exceptional*. Primary outcomes are assumed to occur an unbounded number of times, while exceptions are assumed to occur up to some predetermined bound. The state space is augmented with a counter of how many exceptions have occurred so far, and the transition function is modified so that the counter is increased by one if an only if a transition to an exception occurs. When the counter reaches a predetermined bound, k, the transition function only allows transitions to primary outcomes, and the full set of probabilities is redistributed among these. The value l determines the maximum number of primary outcomes that can be considered for each action. A more detailed explanation can be found in [25].

At first sight, an $\mathcal{M}_l^k$-reduction might seem more complex than the original MDP. However, a well-chosen reduction can significantly cut down the size of the reachable state space (i.e., the states that can be reached from s_0), which is particularly advantageous when using heuristic search MDP solvers such as LAO* [26] and LRTDP [27].

Note that different MDP reductions are characterized by two choices: the set of outcomes to label as primary and the choice of k. In the search and rescue problem, the set of primary outcomes consists of all outcomes in which the robot observes *nothing* after *MOVE*, while observing a victim/hazard is considered an exception. Therefore, this reduced model simplifies the problem by assuming that at most k victims/hazards are present in the environment.

A common approach in continual planning is to compute a complete optimal plan for the reduced model and then apply the resulting plan to the original problem. Execution of this plan continues until a goal is reached, or until a state that was outside the reduced model is encountered. In this case, a new reduced model is created starting at the current state of execution, this new model is then solved optimally to produce an updated plan, and execution is resumed. However, the size of the state space in the search and rescue scenario is intractably large, even when using an $\mathcal{M}_l^k$-reduction. Concretely, with L locations, T time steps and a bound k on the number of victims/hazards, the state space has size $O(TL^k 2^L)$. For a modest problem size of 10 locations, 20 time steps and $k = 1$ this is already in the order of 10^5 states. Therefore, complete optimal planning, even with an $\mathcal{M}_l^1$-reduction, is still not practical.

B. Hindsight Optimization

We can take advantage of the reduced model by noting that it drastically reduces the number of possible environment configurations (usually referred as "worlds") that the robot could be working on. In particular, under the assumption of k victims/hazards the number of possible worlds is in the order of $O(TL^{k+1})$, one for each combination of time, location and set of k locations where victims/hazards can be. This is drastically smaller than the $O(4^L TL)$ worlds that are possible under the original SSP model, and also much smaller than the state space of the reduced model. Note the difference between the number of possible worlds the robot can be in and the number of possible *beliefs* about the world the robot could observe. The large number of beliefs is the reason why planning optimally for the reduced model is still intractable; each state in the reduced model represents a belief about the current world configuration.

The small number of possible worlds resulting from the reduced model can be leveraged by using hindsight optimization (HOP) [17]. In HOP, the expected cost of taking an action in some state is estimated by sampling worlds that are consistent with the current belief, solving the resulting deterministic planning problems and averaging over the results. We can express this approach through the following equations:

$$\hat{V}(s) = \sum_{w \in \mathcal{W}} P(w|s) V^*(w) \tag{2}$$

$$\pi(s) = \arg\min_a [\mathcal{C}(s,a) + \sum_{s'} \mathcal{T}(s'|s,a)\hat{V}(s')] \tag{3}$$

where $\mathcal{W}$ is the set of possible worlds that are also consistent with the current belief state, $P(w|s)$ represents the probability of being in world w given that the current state is s and action a was taken, and $V^*(w)$ is the cost of an optimal plan for world w. Note that the values $V^*(w)$ can be computed very easily because the resulting problem is deterministic. Moreover, the advantage of using a reduced model before applying HOP is that for small values of k, we don't really need to use sampling, since we can quickly enumerate all worlds consistent with the current state, and therefore efficiently compute the average in (2) exactly.

There is one caveat with this method. Hindsight optimization assumes that the world becomes completely observable to the robot immediately after taking an action. This is an overly optimistic assumption that can, in general, lead to arbitrarily bad plans. Intuitively, the reason is that the approach ignores the impact of actions that gather information. However, as our experiments show, this algorithm performs very well for the search and rescue scenario, possibly because every action in this problem must be preceded by actions that gather information (e.g., before rescuing a victim at location A, the robot must move to location A and gather information about the objects in that location).

To compute the probabilities $P(w|s)$ we rely on the independence assumption regarding the object observation probabilities. Note that the value $P(w)$ depends solely on the types of objects it specifies at each location. Let $\langle w.x_1, w.x_2, ..., w.x_L \rangle$ be the vector describing these objects and $w.y_i$ the column index corresponding to entry $w.x_i$ on P_{obs}. Then, by the independence assumption we have:

$$P(w) = \prod_{i=1}^{L} P_{obs}[i, w.y_i] \tag{4}$$

A state s specifies some knowledge about the objects present in the world. Let $\langle b_1, b_2, ..., b_L \rangle$ the belief vector corresponding to state s and $\mathcal{K} = \{i : b_i \neq \textit{unknown}\}$. A world w is said to be consistent with state s iff $\forall_{i \in \mathcal{K}}, w.x_i = b_i$. We can compute $P(w|s)$ when w is consistent with s as:

$$P(w|s) = \frac{\prod_{i=1}^{L} P_{obs}[i, w.y_i]}{\sum_{w' \in \mathcal{W}} \prod_{i=1}^{L} P_{obs}[i, w'.y_i]} \tag{5}$$

where $\mathcal{W}$ is the set of worlds that are consistent with s.

C. Continual Planning

Our complete method, REDHI (Reduction+Hindsight), is summarized by Algorithm 1. The main idea is to increase the value of k for the $\mathcal{M}_l^k$-reductions every time an exception

Algorithm 1: REDHI: A continual planning and execution strategy for solving belief MDPs

```
input: SSP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{C}, s_0, \mathcal{G} \rangle$, $k$, $l$ and $\mathcal{P}_a$
$s \leftarrow s_0$ while $s \notin \mathcal{G}$ do
1     Obtain action $a$ using (2), where $\mathcal{W}$ is the set of worlds
      consistent with $s$ under an $\mathcal{M}_l^k$-reduction
2     $s \leftarrow$ ExecuteAction($s, a$)
3     if exception occurs at $s'$ then
          $k \leftarrow k + 1$
```

occurs during execution. For the search and rescue problem, this amounts to planning for k more victims/hazards than what have been observed so far. The set $\mathcal{P}_a$ given as part of the input represents the choice of primary outcomes for the $\mathcal{M}_l^k$-reduction. As mentioned before, in this work the primary outcomes are those in which the robot observes *nothing* after moving to an unexplored location.

V. Experimental Results

In this section we present an experimental evaluation of our approach. We first evaluate it on an abstract domain representation of the search and rescue problem, which makes it easier to test different environmental configurations and compare with existing MDP solvers. We then evaluate the planner on a semi-realistic simulation of the search and rescue scenario implemented using ROS.

A. Abstract Domain Representation

We implemented an abstract representation of the search and rescue scenario in C++. A search and rescue problem is described by a set of locations on the 2D plane, a matrix of probabilities P_{obs}, and a value T_{max} representing the maximum number of time steps for robot operation. Since there is no underlying map, locations are arbitrarily specified and movement occurs following a straight line between locations. We assume the robot moves at a constant velocity $v = 5$ (with time measured in time steps), so that the time it takes to move between two locations at distance d is d/v time steps; we thus have $\Delta T_a = \lceil d/v \rceil$ time steps for all *MOVE* actions. Additionally, we assume a value of $\Delta T_a = 4$ time steps for all other actions. The rewards are 0 for EVACUATE, 50 for DECONTAMINATE, and 100 for TREAT.

To evaluate the performance of our approach, we experimented on a set of small problems that could be solved off-line using an optimal MDP solver (LAO*). We solved problems ranging from 2 to 6 locations placed randomly on a square of size 10, with random probabilities for P_{obs} and a time horizon of 40 time steps. We generated 10 different problems for each number of locations, and the plans obtained for each problem were simulated 10,000 times. Table I shows the relative difference between the expected cost of the policy obtained with REDHI and the expected cost of the optimal policy, averaged over problems of the same size. Note that the expected cost of the policies generated by REDHI are relatively close to the optimal

TABLE I
Performance of our continual planning approach on randomly generated search and rescue problems.

# Locations	2	3	4	5	6
Rel. $\Delta_{optimal}$	0.058	0.137	0.159	0.206	0.282

TABLE II
Running time (in seconds) of both solvers on randomly generated search and rescue problems.

# Locations	2	3	4	5	6
# of states	837	1.07E4	7.07E4	4.15E5	2.42E6
Time LAO*	0.005	0.136	2.643	35.22	336.66
Time REDHI	0.006	0.057	0.182	0.487	1.022

($< 30\%$ in all cases), although they seem to deteriorate as the number of locations increases.

The benefit of using REDHI becomes more apparent once we take into account the run time required to optimally solve the search and rescue problem. Table II shows the average run time for an optimal MDP solver to compute a plan for the same random problems described above. Note the drastic increase in run time as the size of the problem increases. The reason is due to the exponential increase in the size of the state space (also shown in Table II), which increases by approximately a factor of 7 when the number of locations increases by 1. However, the running time of REDHI in all of these case is less than 2 seconds and seems to increase quadratically with the number of locations.

B. Search and Rescue Robot Simulator

We implemented a semi-realistic virtual simulation of a simple robotic search and rescue scenario using ROS (Robot Operating System). The scenario is based on the uBot-5 robot [12], a small light-weight bimanual mobile manipulator (see Figure 1). Each arm has four degrees of freedom—two in the shoulder and two in the elbow. The robot is equipped with an RGB-D camera with a tilt axis to see the ground in front of the robot. The simulator is able to model the robot's operation using a kinematic model for wheels and arms motion and a simple vision model for a RGB-D sensor.

We model victims, hazards and medicine as ARcubes—28 cm cubes with unique combinations of Augmented Reality tags. Then, EVACUATE, DECONTAMINATE and TREAT actions are equivalent to picking up and placing down an ARcube. The objects are perfectly observed when present in the field of view of the robot.

The experimental environment has size 8.7 m $\times$ 10.8 m, as is illustrated in Figure 2; the robot is able to move between locations using a path planner. The robot is given a maximum of $T_{max} = 40$, where each time step represents 15 seconds for a total of 10 minutes. Time is discretized as $\lfloor t_{real}/15 \rfloor$, where t_{real} is the current time measured in seconds. There are 10 possible locations for victims/hazards and in every scenario we include 3 non-ambulatory victims, 3 ambulatory victims and 2 hazards. Note that this violates the independence assumption for the observation

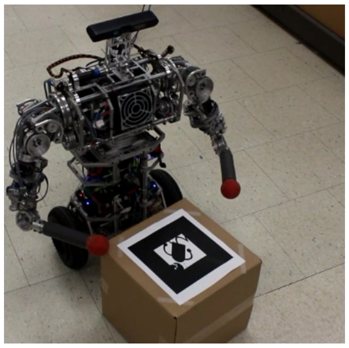

Fig. 1. The uBot-5 mobile manipulator.

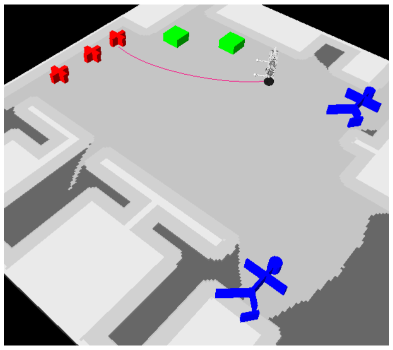

Fig. 2. Illustration of the simulation environment used.

probabilities; however, our experiments suggest the algorithm performs well even if this assumption is broken. We chose the observation probabilities so that the expected number of victims of each type matched the actual number of victims. That is, we assigned, for all locations, a probability of 0.3 of observing a non-ambulatory victim, 0.3 of observing an ambulatory victim, 0.2 of observing a hazard and 0.2 of observing nothing.

The state space for the complete model has more than 4 billion states, thus solving it optimally is clearly impractical. In our experiments we used an initial value of $k = 1$ for the reduced model, as it greatly decreases the size of the state space and works well in practice, even though this significantly underestimates the number of objects present. To see why this value works well, recall that whenever a victim or hazard is observed, k is increased, allowing REDHI to plan ahead for additional objects in the environment.

We compute the time taken by each action as $\Delta t_a = \lceil \frac{d}{v} \rceil + t_h$, where d is the distance traveled in meters, v is the robot's speed in meters per step and t_h is the number of time steps it takes for the robot to pick up and place down an ARcube (this is ignored for MOVE actions). We set the values $v = 5$ (0.33 m/s) and $t_h = 4$ (1 min.) based on empirical observation of the robot's behavior. The rewards of all actions are the same as in the abstract domain representation.

In contrast to the simplified abstract domain representation and the definition of the transition function for the SSP, movement between two locations in the simulator can lead to observe objects at other locations besides the target. Given the difficulty in modeling this behavior properly—due to the large number of geometrical factors involved— this supports the idea that a continual planning approach is the most appropriate planning choice for this problem.

We compared the performance of REDHI with a greedy baseline planner. This baseline tries to perform a clearing action at the location closest to the robot, and if there is no object that can be interacted with at that location, the robot moves to the nearest location where the belief is different than *nothing*. We performed experiments using 10 different placements of the 8 objects on the environment. Table III shows the result of these experiments in terms of the total reward obtained in each scenario.

The results show that REDHI significantly outperformed the greedy baseline in 5 out of the 10 scenarios, and was only outperformed in 1 scenario. The average improvement over the greedy approach is around 13% and the median is 9.5%. However, it is worth mentioning that in the only scenario where REDHI was outperformed, the robot was actually able to finish treatment on a non-ambulatory victim less than 2 seconds after the 10 minute limit. This suggests that the robot was overly confident about the time it would take to perform actions leading, in turn, to a less adequate prioritization. This is most likely due to inaccuracies in estimated speed and handling time. On the other hand, out of the 5 cases where the greedy baseline was outperformed, there was only one in which the robot could have cleared an additional object with less than 30 seconds after the 10 minute limit.

In terms of observed behavior trends, the use of REDHI led the robot to explore several locations before starting to act on the objects observed. In fact, early in some of the scenarios the robot chose to explore nearby locations even in the presence of non-ambulatory victims. The intuition for this behavior is that with high probability (50%) the robot expects to see a non-ambulatory victim or hazard at the next location. In that case, moving the next object to the base, then picking up the medicine and then treating the non-ambulatory victim would be faster than treating the victim first (requiring moving to the base and back) and then moving to clear the object afterward. However, when the time is close to the limit REDHI leads to greedier behavior, favoring rescue actions prioritized according to their immediate rewards.

VI. Conclusions

In this work we introduced REDHI, an efficient planning algorithm to solve complex MDPs such as those required for search and rescue robotics problems. The algorithm is based on a continual planning framework that combines $\mathcal{M}_l^k$-reductions for MDPs with a hindsight optimization approach. Experimental results show that REDHI can obtain near-optimal performance with negligible planning time and it compares favorably with a greedy baseline on a simulated search and rescue robot scenario.

We show how the use of an $\mathcal{M}_l^k$-reduction allows for all possible configuration to be quickly enumerable, and thus a hindsight optimization can use the full probability

TABLE III

TOTAL REWARD ACCRUED BY REDHI AND THE GREEDY PLANNER IN 10 DIFFERENT CONFIGURATIONS OF THE SEARCH AND RESCUE SCENARIO.

REDHI	320.55	**330.96**	372.13	**328.49**	323.78	**322.96**	**324.41**	325.14	274.36	**325.81**
Greedy	322.96	273.28	372.60	275.66	328.84	223.16	272.31	325.53	**326.96**	226.00

distribution of world realizations, instead of resorting to sampling as it is usually done in practice. Moreover, our continual planning approach incorporates knowledge about the environment during planning and always plans ahead for additional exceptions. This increases the coverage of the plan during execution, without needing lengthy periods of off-line planning.

Our approach relies on certain independence assumptions about the joint probability distribution of world configurations. However, we can get around this assumption by resorting back to sampling-based hindsight optimization. Note that even in this case, using an $\mathcal{M}_l^k$-reduction is still appealing, as the number of samples required to solve Eq. (3) can be reduced significantly.

In future work we would like to extend our approach to handle human-rescue teams; this extension is non trivial, as the robot must reason about the human's behavior and must make sure that it doesn't interfere with their operation. Additionally, we want to extend our approach to handle the presence of non-perfect observations and localization problems, which would convert the problem into a Partially Observable MDP (POMDP), and thus much harder to solve.

VII. ACKNOWLEDGMENTS

This research was supported in part by NSF grants IIS-1405550 and IIS-1524797 and NASA grant NASA-GCT-NNX12AR16A. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Aeronautics and Space Administration.

REFERENCES

[1] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 33, no. 3, pp. 367–385, 2003.

[2] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey, "Supporting wilderness search and rescue using a camera-equipped mini UAV," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 89–110, 2008.

[3] R. R. Murphy and S. Stover, "Rescue robots for mudslides: A descriptive study of the 2005 La Conchita mudslide response," *Journal of Field Robotics*, vol. 25, no. 1-2, pp. 3–16, 2008.

[4] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, "Distributed search and rescue with robot and sensor teams," in *Field and Service Robotics*. Springer, 2006, pp. 529–538.

[5] N. Ruangpayoongsak, H. Roth, and J. Chudoba, "Mobile robots for search and rescue," in *Safety, Security and Rescue Robotics, Workshop, 2005 IEEE International*. IEEE, 2005, pp. 212–217.

[6] I. Nourbakhsh, K. Sycara, M. Koes, M. Yong, M. Lewis, and S. Burion, "Human-robot teaming for search and rescue," *Pervasive Computing, IEEE*, vol. 4, no. 1, pp. 72–79, 2005.

[7] M. Pfingsthorn, B. Slamet, A. Visser, and N. Vlassis, "UvA rescue team 2006; RoboCup rescue-simulation league," in *Proceedings of the 10th RoboCup International Symposium*, 2006.

[8] B. Li, S. Ma, J. Liu, M. Wang, T. Liu, and Y. Wang, "AMOEBA-I: a shape-shifting modular robot for urban search and rescue," *Advanced Robotics*, vol. 23, no. 9, pp. 1057–1083, 2009.

[9] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York, NY, USA: John Wiley & Sons, Inc., 1994.

[10] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman, "Acting optimally in partially observable stochastic domains," in *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'94)*, 1994.

[11] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein, "The complexity of decentralized control of Markov decision processes," *Mathematics of Operations Research*, vol. 27, pp. 819–840, 2002.

[12] P. Deegan, B. J. Thibodeau, and R. Grupen, "Designing a self-stabilizing robot for dynamic mobile manipulation," DTIC Document, Tech. Rep., 2006.

[13] S. W. Yoon, A. Fern, and R. Givan, "FF-Replan: A baseline for probabilistic planning," in *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS'07)*, 2007, pp. 352–359.

[14] H. L. S. Younes, M. L. Littman, D. Weissman, and J. Asmuth, "The first probabilistic track of the International Planning Competition," *Journal of Artificial Intelligence Research*, vol. 24, no. 1, pp. 851–887, 2005.

[15] F. Teichteil-Königsbuch, U. Kuter, and G. Infantes, "Incremental plan aggregation for generating policies in MDPs," in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, 2010, pp. 1231–1238.

[16] E. Keyder and H. Geffner, "The HMDPP planner for planning with probabilities," *The ICAPS 3rd International Probabilistic Planning Competition (IPPC'08)*, 2008.

[17] S. Yoon, A. Fern, R. Givan, and S. Kambhampati, "Probabilistic planning via determinization in hindsight," in *Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI'08)*, 2008, pp. 1010–1016.

[18] S. Yoon, W. Ruml, J. Benton, and M. B. Do, "Improving determinization in hindsight for online probabilistic planning," in *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS'10)*, 2010, pp. 209–216.

[19] N. Hyafil and F. Bacchus, "Conformant probabilistic planning via CSPs." in *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, vol. 98, 2003, pp. 205–214.

[20] C. Domshlak and J. Hoffmann, "Probabilistic planning via heuristic forward search and weighted model counting." *Journal of Artificial Intelligence Research (JAIR)*, vol. 30, pp. 565–620, 2007.

[21] R. I. Brafman and R. Taig, "A translation based approach to probabilistic conformant planning," in *Algorithmic Decision Theory*. Springer, 2011, pp. 16–27.

[22] R. Taig and R. I. Brafman, "A relevance-based compilation method for conformant probabilistic planning," *Models and Paradigms for Planning under Uncertainty: a Broad Perspective*, p. 49, 2014.

[23] M. Brenner and B. Nebel, "Continual planning and acting in dynamic multiagent environments," *Autonomous Agents and Multi-Agent Systems*, vol. 19, no. 3, pp. 297–331, 2009.

[24] M. E. desJardins, E. H. Durfee, C. L. Ortiz, and M. J. Wolverton, "Continual planning and acting in dynamic multiagent environments," *AI Magazine*, vol. 20, no. 4, pp. 13–22, 1999.

[25] L. Pineda and S. Zilberstein, "Planning under uncertainty using reduced models: Revisiting determinization," in *Proceedings of the 24th International Conference on Automated Planning and Scheduling*, 2014, pp. 217–225.

[26] E. A. Hansen and S. Zilberstein, "LAO*: A heuristic search algorithm that finds solutions with loops," *Artificial Intelligence*, vol. 129, no. 1-2, pp. 35–62, 2001.

[27] B. Bonet and H. Geffner, "Labeled RTDP: Improving the convergence of real-time dynamic programming," in *Proceedings of the 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*, 2003, pp. 12–21.