Temporal Logic Control in Dynamic Environments with Probabilistic Satisfaction Guarantees

A. I. Medina Ayala, S. B. Andersson, and C. Belta

Abstract— **Mobile robotic systems move in environments that are constantly changing due to the presence of dynamic obstacles. In this work we consider one such environment in which the dynamic nature comes from doors that can open or close during the robot's mission. We derive a solution to the automatic deployment of a robot from a temporal logic specification assuming three different levels of knowledge and sensing capabilities of the robot. Under each one of these settings, the motion of the robot is modeled either as a Markov decision process (MDP) or mixed observability MDP. The objective is to find a control strategy that maximizes the probability of satisfying a specification given in Probabilistic Computation Tree Logic (PCTL). We describe an optimal solution in one setting and sub-optimal, reactive solutions in the other two. We illustrate our methods with simulation results.**

I. INTRODUCTION

The need for finding better ways of planning under uncertainty has grown since the domains in which robots are being applied are becoming more complex. These domains include unknown environments with real time constraints arising from changes in the environment. Uncertainty in the motion of the robot arises from both noisy actuators and noisy sensors. Markov decision processes (MDPs) are adequate to model robotic systems that have unreliable controllers while maintaining the ability to accurately determine their position with respect to some description of the environment. In general, however, the motion of a robot in its environment is more accurately described using partially observable MDPs (POMDPs). Recently, a sub-class of POMDPs called *mixed observability* MDPs (MOMDPs) was proposed to treat POMDP problems in which some components of the state of the robot are fully observable. It has been shown that exploiting the full observability of these components improves the computational efficiency and speed of existing planning algorithms [1].

For the most part, solving a PO(MDP) implies finding a control policy that maximizes a value function that combines partial rewards over multiple steps. The value function is an intuitive high level representation of a specific motion task. It has been shown that the maximal probability of satisfying a temporal logic formula is the optimal value function corresponding to the task of maximum reachability in stochastic shortest path problems [2]. Algorithms based on model checking are used to find control policies for robotic tasks expressed as temporal logic specifications [3]-[6]. Such specifications can be given in, for example, Linear Temporal Logic (LTL) and Computation Tree Logic (CTL). The starting point of model checking algorithms is to represent the partitioned environment as a transition system, or Kripke structure [7]. The probabilistic counterpart of CTL, probabilistic CTL (PCTL) [2], allows one to perform synthesis of control strategies from specifications given as PCTL formulas. Previous work from the authors [5] suggested the use of PCTL model checking to deal with indoor robotic planning. This approach assumed the environment in which the robot is moving does not change during the robot's task.

In this paper, we consider an extension of the problem solved in [5] by allowing for a dynamic (changing) environment. Specifically, the environment includes doors that open and close during the robot's mission. We solve this problem under three settings that assume different levels of knowledge and sensing capabilities of the robot. In the first setting the robot is given *a priori* information about the states of the doors, but it can only learn their true states in a region adjacent to them. The second setting excludes the prior information about the states of the doors but retains the assumption that the exact state of any door is known when the robot is in a region adjacent to it. In the last setting, this assumption is also relaxed and we allow for possibly erroneous measurements as to the state of the doors observed by the robot. A Markov decision process (MDP) is used to model the system under the first setting. The second and third settings are cast as mixed observable Markov decision processes (MOMDPs). We consider specifications given as PCTL formulas and develop a framework for the synthesis of control strategies from such specifications. While this paper focuses for clarity on an indoor environment with doors, the problem and methods developed can easily be generalized to arbitrary environments with regions in which the transitions can be open or blocked. To illustrate these methods, we use the Robotic InDoor Environment (RIDE) Simulator [8] to generate the MDP and MOMDP models for a robot moving in a dynamic environment and to show the planning of the robot. To the best of our knowledge, the frameworks developed in this work for planning in non-static environments from a PCTL specification are novel and adaptable to the level of knowledge available.

The remainder of the paper is organized as follows. In Sec. II, we formulate the problem and outline our approach. Some definitions and notation are briefly described in Sec. III. The

This work is partially supported at Boston University by the NSF under grants CNS-0834260 and CMMI-0928776, the ARO under grant W911NF-09-1-0088, the AFOSR under grant FA9550-09-1-0209, and the ONR MURI under grant N00014-09-1051.

The authors are with the Department of Mechanical Engineering, Boston University, MA, USA, E-mail: duvinci@bu.edu, sanderss@bu.edu, cbelta@bu.edu

A. Medina Ayala is the corresponding author.

modeling technique for the robot motion and the solution algorithm for each of the proposed settings are presented in Secs. IV, V, and VI. The simulation platform and results are described in Sec. VII. The paper concludes with final remarks in Section VIII.

II. PROBLEM FORMULATION AND APPROACH

Consider a mobile robot moving in an indoor environment consisting of intersections, corridors, rooms, and doors as illustrated in Fig. 1. We assume that the robot is programmed with a set of primitives allowing it to move inside each region and from one region to an adjacent one provided that the region is not blocked by a closed door. Each primitive (or control *symbol*) is a feedback control law. We suppose that these controllers are not completely reliable. In other words, if in a given region a control designed to take the robot to a specific adjacent region is used, it is possible that the robot will instead transition to a different adjacent region. The success and failure rates of these controllers are assumed to be known. Such rates are given in the form of probabilities, i.e., $Pr(s'|s,\alpha)$, where s' represents the region to which the robot transitions after applying the primitive α from region s. The robot can determine its current region exactly and only gets information on whether a given door is open or closed when the current region contains that door. With this setting, we consider the following problem:

Problem 1. *Given a motion specification in the form of a temporal logic statement over a set of properties satisfied by the regions in a dynamic indoor environment with known topology, find a control strategy that maximizes the probability that the robot will satisfy the specification.* □

While this paper considers a highly structured indoor environment populated with doors, the methods developed can be applied to any environment consisting of arbitrary regions in which the motion of the robot can be blocked or obstructed by static or dynamic obstacles.

To capture the notion of the dynamic environment used in this paper, we introduce the following definition.

Definition 1. *A dynamic environment is a tuple* $\mathcal{E} = (R, D, A, B, H)$*, where*

- $R = \{r_1, r_2, \ldots, r_k\}$*, is a set of mutually disjoint regions;*
- $D = \{d_1, d_2, \ldots, d_N\}$*, is a set of doors;*
- $A \subseteq R \times R$*, is a binary relation representing the adjacency between two regions.* $(r_1, r_2) \in A$ *denotes that* r_1 *and* r_2 *are adjacent and there is no door in between;*
- $B = \{b_i : 1 \leq i \leq N\}$*, where* N *is the number of doors, is a set of Boolean variables indicating if a door is closed* ($b_i = 1$) *or open* ($b_i = 0$)*, and;*
- $H \subseteq D \times R$*, is a binary relation representing the adjacency between regions and doors. We say that a region* r *has a door* d *if* $(d, r) \in H$*.*

An example indoor environment is given in Fig. 1. This environment consists of 16 corridors (marked as C_1; ...; C_{16}),

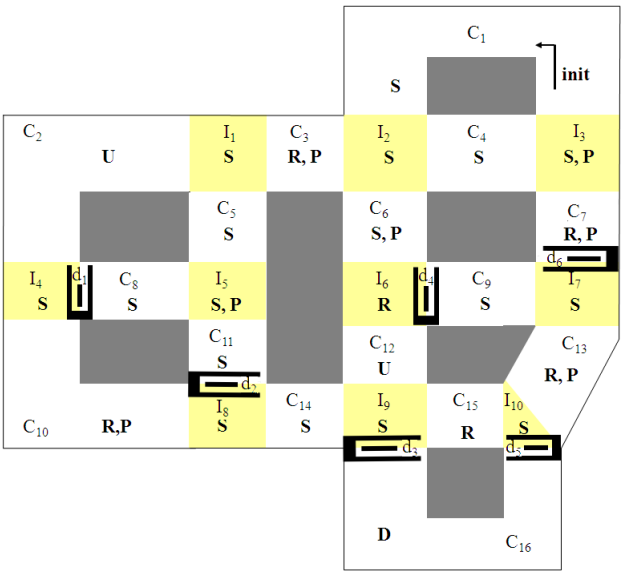

Fig. 1. Schematic representation of an example dynamic environment. $C_1, ..., C_{16}$ and $I_1, ..., I_{10}$ are identifiers representing corridors and intersections, respectively. The properties satisfied at each one of the regions are: **S** = Safe, **R** = Relatively safe, **U** = Unsafe, **P** = Power supply, and **D** = Destination.

ten intersections (I_1; ...; I_{10}), and six doors (d_1; ...;d_6). There are five properties of interest about these regions: *Safe* (the robot can safely drive through a region with this property), *Relatively safe* (the robot can pass through the region but should avoid it if possible), *Unsafe* (the corresponding region should be avoided), *Power supply* (there is a power supply station to charge the robot in the region), and *Destination* (a region the robot should visit). An example task based on these properties is: "Reach *Destination* while going only through either *Safe* or *Relatively safe* regions only if *Power supply* is available at the *Relatively Safe* regions".

In this paper we solve Problem 1 under three settings with different levels of assumed knowledge.

1) **Setting 1:** In the first setting, we assume that when the robot observes a door, it determines perfectly whether the door is open or closed. Further, we assume the robot knows *a priori* the probability that each door is either open or closed. Doors are allowed to switch between being open and closed, though in this work we make the simplifying assumption that such changes occur in synchronization with the transitions of the robot. Under these assumptions, the entire system can be modeled as an MDP. This structure then allows us to use the model checking approach developed in [5] to synthesize a control policy that computes the discrete plan satisfying a specification.
2) **Setting 2:** In the next setting, we remove the assumption that the robot has *a priori* knowledge about the state of the doors. In order to solve Problem 1 under this setting, we focus on a special class of POMDPs called MOMDPs (mixed observability Markov decision processes [1]). Using this framework, the state

space of the system is the Cartesian product of two components, the regions of the environment and the state of the doors. We assume the robot knows its current region perfectly and can only measure the state of a door when it is in a region that has a door. Thus the fully observable component corresponds to the regions while the hidden one corresponds to the states of the doors. Our approach starts by solving the problem for the fully observable state component offline, for each one of the possible *scenarios* the environment can be in. A *scenario* represents a particular configuration of the environment in which either all doors are open or only one is closed. Each *scenario* may require a distinctive control policy. Once the robot is deployed, it selects the policy corresponding to the current *scenario*; each time a door is encountered, the new *scenario* is determined and the policy is changed accordingly.

3) **Setting 3:** In the final setting, we also remove the assumption that the robot has perfect measurements as to the states of the doors. In this case, the robot may observe an open door as being closed or vice versa. We do assume, however, that the success and failure rates of these observations are known and are given in the form of the probabilities, i.e., $Pr(o|s, \alpha, x)$. Here o represents the observation made of the hidden state x from the fully observable state s after the controller α is performed. Following the same approach used to solve Setting 2, a sub-optimal strategy is generated.

Details for each one of these settings, including a discussion of the optimality and the complexity of the solution, are given in Secs. IV, V, and VI, respectively.

III. PRELIMINARIES

In this section, some concepts and notation used in the paper are introduced.

Definition 2. *A labeled Markov decision process $\mathcal{M}$ is a tuple (S, s_0, Act, T, AP, L), where*

- *S is a finite set of states;*
- *$s_0 \in S$ is the initial state;*
- *Act is a finite nonempty set of actions;*
- *$T : S \times Act \times S \to [0, 1]$ is a transition probability function such that for each $s \in S$ and $\alpha \in Act$, either $T(s, \alpha, .)$ is a probability distribution on S or $T(s, \alpha, .)$ is the null function (i.e. $T(s, \alpha, s') = 0$ for any $s' \in S$);*
- *AP is a finite set of atomic propositions;*
- *$L : S \to 2^{AP}$ is a labeling function assigning to each $s \in S$ possibly several elements of the set AP that are supposed to hold in s.*

Given $s \in S$, let $Act(s)$ denote the set of actions available in state s, i.e.

$$Act(s) = \{\alpha \in Act : T(s, \alpha, s') > 0 \text{ for some } s' \in S\}.$$

A path ω of an MDP, is an infinite sequence $s_i, s_{i+1}, s_{i+2}, ...$ of states. For each $i \geq 0$ such a sequence is constructed by iterating a two-phase selection process. First, an action $a \in Act(s_i)$ is selected nondeterministically; second, the successor state s_{i+1} is chosen according to the probability distribution $T(s_i, a, s_{i+1})$. The set of all non-empty finite sequences of states (paths) is denoted by $Path^{fin}$ and that of infinite ones by $Path^{inf}$.

Definition 3. *A policy (also called strategy, adversary, or scheduler) represents a particular resolution of nondeterminism only. Formally, a policy for an MDP $\mathcal{M} = (S, s_0, Act, T, AP, L)$ is defined as a function*

$$\pi : Path^{fin} \to Act,$$

such that for every finite path, a policy specifies the next action to be applied.

Given a policy π, the operational behavior of $\mathcal{M}$ under π can be represented by a Markov chain. This allows one to apply standard techniques for Markov chains to define a σ-algebra over infinite paths and the probability of path events [2].

PCTL is a probabilistic extension of CTL that includes a probabilistic operator, $\mathbb{P}$ [9]. The syntax of PCTL is defined as follows:

$$\Phi ::= true \mid p \mid \neg\Phi \mid \Phi \vee \Phi \mid \Phi \wedge \Phi \mid \mathbb{P}_{\sim\lambda}[\phi]$$
$$\phi ::= X\ \Phi \mid \mathbb{P}_{\sim\lambda}(\Phi\ \mathcal{U}^{\leq n}\ \Phi)$$

where p is an atomic proposition, $\sim \in \{<, \leq, >, \geq\}$ is a comparison operator, $\lambda \in [0, 1]$ is a probability threshold and $n \in \mathbb{N} \cup \{\infty\}$.

PCTL formulae are interpreted over the states of $\mathcal{M}$. For representing the syntax of PCTL, we distinguish between state formulae Φ and path formulae ϕ, which are evaluated over states and paths, respectively. The next (X) and bounded until ($\mathcal{U}^{\leq n}$) operators are considered as path formulae. The unbounded until operator is obtained by taking n equal to ∞.

In this work, we restrict ourselves to specifications of the form $P_{max=?}[\phi_1\ \mathcal{U}\ \phi_2]$, which is defined as the maximal probability for which there exists a policy π such that the formula "ϕ_1 until ϕ_2" is satisfied.

Definition 4. *A partially observable Markov decision process (POMDP) is a tuple $(S, Act, \Theta, T, O, R)$, where*

- *S is a finite set of states;*
- *Act is a finite set of actions;*
- *Θ is a finite set of observations;*
- *$T : S \times Act \times S \to [0, 1]$ is the state transition function, mapping from elements of $S \times Act$ into discrete probability distributions over S;*
- *$O : S \times Act \times \Theta \to [0, 1]$ is the observation function, mapping $S \times Act$ into discrete probability distributions over Θ;*
- *$R : S \times Act \to \mathbb{R}$ is the stationary reward function mapping from $S \times Act$ to $\mathbb{R}$.*

Definition 5. *A labeled mixed observability Markov decision process (MOMDP) is defined as a tuple $(S, s_0, X, \Theta, Act, T, O, AP, L)$, where:*

- *S is the set of fully observable states;*
- *$s_0 \in S$ is the initial fully observable state;*
- *X is the set of hidden states;*
- *Θ is a finite set of observations;*
- *Act is a finite set of possible actions;*
- *$T(s, x, \alpha, s') = Pr(s'|s, x, \alpha)$ represents the probability of making a transition to the fully observable state s' if action α is applied in the observable state s when the hidden state is x;*
- *$O(s', x', \alpha, o) = Pr(o|s', x', \alpha)$ is a set of observation probabilities that describe the probability of observing o from the fully observable state s when the hidden state is x after performing action α;*
- *AP is a finite set of atomic propositions;*
- *$L : S \to 2^{AP}$ is a labeling function assigning to each fully observable state possibly several elements of the set AP that are supposed to hold in s.*

IV. CONTROL POLICY SYNTHESIS UNDER SETTING 1

A. Construction of the MDP Model

In this section we discuss the construction of the MDP modeling the motion of the robot in $\mathcal{E}$. We assume that the transition probabilities of the robot in the environment depend only on its current region, i.e., $T(s, \alpha, s') = Pr(s'|s, \alpha)$. In practice, state augmentation may be needed to ensure this Markovianity property [5]. In order to model the state space of the MDP, we first define as R_2 the set of regions that have a door, i.e., $R_2 = \{r \in R | (d, r) \in H$, for some $d \in D\}$. We then denote by R_1 the set regions with no doors, i.e., $R_1 = R \setminus R_2$. Hence, the set of states of the MDP is the union of the set of regions with no doors with the union of the set of pairs of regions containing a door and the states of that door, i.e., $S = R_1 \cup \bigcup_{r \in R_2} \{(r, 0), (r, 1)\}$. Each state of the MDP is labeled with the property that is satisfied at the region representing such state, i.e., $L(s)$. The set of actions of the MDP is the set of motion primitives available at each one of the regions in the environment including the ones that allow the robot to make a decision at the regions adjacent to a door, i.e., $Act(s)$.

As an example, consider the environment depicted in Fig. 1. The controller FollowRoad is available at the corridors while the controllers GoRight, GoLeft, and GoStraight are available at the intersections. In regions adjacent to a door, the actions Wait and TurnAway are also available if the door is closed.

As already outlined, the *a priori* probability of the door being open or closed is known. We make the simplifying assumption that the state of the doors changes synchronously with the transitions of the robot. Thus these transitions define the only notion of "time" in the system. To ensure that the doors do not close while the robot is crossing, we assume an open door remains open so long as the robot is adjacent to it. To illustrate how these two assumptions are integrated into the system, a fragment of the MDP containing door d_2 between regions C_{11} and I_8 (from Fig. 1) is shown in Fig. 2. Suppose that door 2 has a probability of 0.4 of being open

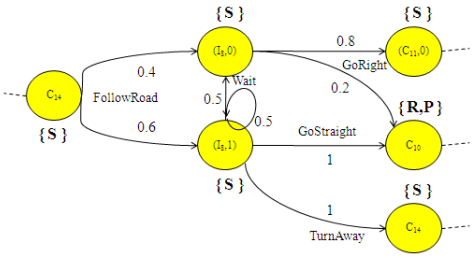

Fig. 2. Graphical representation of transition model after taking action FollowRoad at state C_{14}.

and 0.6 of being closed. As indicated in the figure, these probabilities multiply the initial probability of transitioning from C_{14} to I_8 (which in this case was one). As a result, the probability of applying action FollowRoad at region C_{14} and ending up in I_8 when the door is closed is 0.4 and 0.6 when it is open. The change in the state of the door from closed to open is associated to the transition probability under action Wait.

B. PCTL Control Synthesis

By constructing the MDP as described above, the problem can be solved using the tool developed by our research group [5]. This tool accepts an MDP and a PCTL specification and returns the maximum probability of satisfying this specification and the policy that gives rise to this probability. As described in [2], [5], [10], the complexity of this algorithm is polynomial in the size of the state space of the MDP. An example of this approach is given in Sec. VII.

V. CONTROL POLICY SYNTHESIS UNDER SETTING 2

A. Construction of the MOMDP Model

This section addresses the construction of the MOMDP to model the motion of the robot in the environment. The state space of the MOMDP is the Cartesian product of the set of regions, and the valuation of the set of Boolean variables representing the state of each door. The first represents the fully observable states, i.e., $S = R$, while the second represents the hidden ones, i.e., $X = \bigcup_{b_i \in B} b_i$. Each fully observable state component of the MOMDP is labeled with the property that is satisfied at the region representing the state, i.e, $L(s)$. The set of controllers of the MOMDP is the set of primitives available at the fully observable states, i.e., $Act(s)$.

The set of observations available is $\Theta = \{open, closed\}$, denoting the possible states of the doors. Under the assumption of perfect sensing, the exact state of the doors is known once the robot is in a region adjacent to them. Thus, the observation probabilities, $O(s', x', \alpha, o) = Pr(o|s', x', \alpha)$, are either 0 or 1 depending on the state of the doors when the robot sees them.

The transition probabilities from states that do not have a door are independent of the states of the doors, therefore $T(s, x, \alpha, s') = Pr(s'|s, \alpha)$. On the other hand, the transition

probabilities from states containing doors are conditioned by the states of the doors, thus $T(s, x, \alpha, s')$ can be constructed from the fully observable transition and observation functions:

$$T(s, x, \alpha, s') = \sum_{s \in S} Pr(s'|s, \alpha) Pr(o|s', x', \alpha). \quad (1)$$

However, assuming perfect sensing, (1) reduces to:

$$T(s, x, \alpha, s') = \begin{cases} Pr(s'|s, \alpha) & \text{if } o \text{ is open} \\ 0 & \text{if } o \text{ is closed} \end{cases}$$

B. PCTL Control synthesis

In general, POMDP problems attempt to find the optimal policy over the belief space of the system. In order to do this, the state of the system over this space must be updated. In our particular framework, the states of the doors can change at each transition of the robot. Thus, any information gained from a measurement is valid only until the next transition is made. This in turn implies that there is no need to update the belief state over the hidden states in the MOMDP.

Reactive policy search methods help to manage unpredictable changes in the environment. To solve Problem 1, we use an online policy search that proceeds as follows. By construction, only the state of the current door can be known; we arbitrarily assume that the other doors are open. If there are N doors in the environment, there are then $N+1$ possible configurations. We term each of these $N+1$ possibilities as a *scenario*. For each *scenario* there is an underlying MDP whose state space is comprised of the fully observable states of the system. As discussed in Sec. IV, the optimal policy for each MDP can be obtained using linear programming. This is attained by partitioning S into three subsets: S^{yes}, the states that satisfy the specification with probability exactly 1, S^{no}, the states that satisfy the specification with probability 0, and $S^{?}$, the remaining states. As a result, the policy that gives rise to the maximum probability of satisfying the specification is obtained off-line for each *scenario*. We denote by Π the set of policies for all possible *scenarios*.

Once the robot is deployed, it applies a reactive policy by choosing the optimal solution based on its current information. Thus, if it observes a closed door, it selects the previously calculated policy corresponding to that *scenario*. Otherwise, it selects the policy corresponding to all doors being open. The robot applies this policy until it encounters another closed door. The online algorithm is divided into a planning phase, and an execution phase, which is applied at each time step. These phases are presented in Off-line Planning Algorithm and Algorithm 1.

Producing the optimal policies involves solving one linear program for each *scenario*. Hence, the complexity of this online algorithm is $\mathcal{O}(|N|poly(|S|))$.

Off-line Planning Algorithm

Inputs: A PCTL specification, $\phi :: P_{max=?}[\phi_1 \; \mathcal{U} \; \phi_2]$
$1 + N$ MDPs, $\mathcal{M}_i = (S, s_0, Act, T_i, AP, L)$
Outputs: Set of policies for all possible scenarios,
$\Pi = \{\pi^0, ..., \pi^N\}$

for $i = 0 : N$ **do**
  $\pi^i \longleftarrow$ `SOLVE LINEAR PROGRAM` $(\mathcal{M}_i)$
end
return Π

Algorithm 1: Setting 2 execution phase

Inputs: Set of policies for all scenarios, $\Pi = \{\pi^0, ..., \pi^N\}$

$\pi^* = \pi^0$ `/* i = 0 is the index of the scenario in which all doors are open */`
  $Act(s) \leftarrow \pi^*(s)$
  s' = **execute** $Act(s)$
  $s = s'$
  while `IN EXECUTION` **do**
    if $s \in S^{yes} \vee s \in S^{no}$ **do**
      break
    else
      if $(Pr(closed|s', b_i, Act(s)) = 1)$ **do**
        $\pi^* = \pi^i$
        go to 3
      else
        go to 3
      end
    end
  end

VI. CONTROL POLICY SYNTHESIS UNDER SETTING 3

A. Construction of the MOMDP Model

The robot's motion under setting 3 is also modeled as a MOMDP [1]. The state, action, and observation space of the MOMDP under this setting are built in the same fashion as in Sec. V.

The main difference between Settings 2 and 3 is the notion of observability. Setting 3 describes a more realistic scenario in which the uncertainty in the robot's sensors is considered. Thus, the probability of correctly seeing a door open or closed depends on the reliability of the sensor measurements and is assumed to be known. Since the transition probabilities from states containing doors depend on the states of the doors, they are given by the full form in (1).

B. PCTL Control Synthesis

Following the approach used in the previous setting, a reactive policy search is developed. As before, the set of policies that generate the maximum probabilities of satisfying the specification for each *scenario* are computed offline. Assuming the initial region does not have a door, the robot starts by applying the policy that guarantees it will satisfy the specification with maximum probability under the assumption that all doors are open (otherwise, after the observation is made the policy is selected according to (2) as described below). This policy is executed until the robot crosses a region containing a door. At this region, based on the observation of the door, the robot can choose either to keep applying the current policy or to execute a new policy. The policy selected corresponds to the *scenario* that is most likely based on the observation and it is given by the following one-step lookahead DP

$$p_s = \max_{\pi \in \Pi} \Bigg\{ \sum_{s' \in S^?} T(s, x, \pi(s), s') \cdot p_{s'} + \sum_{s' \in S^{yes}} T(s, x, \pi(s), s') \Bigg\}. \quad (2)$$

Here p_s is the probability of satisfying the specification from state s, and S^{yes}, and $S^?$ represent the fully observable states that satisfy the specification with probability 1, and with probability strictly between 0 and 1, respectively. After making the decision that will allow it to satisfy the specification with maximum probability, the robot repeats the described procedure until eventually it achieves the destination with a sub-optimal policy. As in Setting 2, this approach involves a planning and an execution phases. The Off-line Planning Algorithm is the same as for the previous case. The execution phase is summarized in Algorithm 2.

Algorithm 2: Setting 3 execution phase

Inputs: Set of policies for all scenarios, $\Pi = \{\pi^0, ..., \pi^N\}$

$\pi^* = \pi^0$ /* $i = 0$ is the index of the scenario in which all doors are open */
$Act(s) \leftarrow \pi^*(s)$
s' = **execute** $Act(s)$
$s = s'$
while IN EXECUTION **do**
  if $s \in S^{yes} \vee s \in S^{no}$ **do**
    break
  else
    if $(s \in \{r \in R | (d_j, r) \in H)\})$ **do**
      $\pi^i \leftarrow$ SOLVE (2)
      $\pi^* = \pi^i$
      go to 3
    else
      go to 3
    end
  end
end

The complexity of this online algorithm comes from two sources: generation of optimal policies for each *scenario* and online selection of the appropriate policy to be applied after observing a door. Generating the optimal policies requires computing one linear program for each *scenario*. Once the robot is deployed, there are two choices available to select the policy to be executed after making an observation. Therefore, the overall complexity is $\mathcal{O}(2|N|poly(|S|))$.

VII. SIMULATIONS AND RESULTS

A. Simulation Tool

The RIDE simulator [8] is a recently developed real-time simulator that captures the motion capabilities of an iRobot iCreate platform equipped with a laser range finder and an RFID reader as it moves in an indoor environment. In order to capture the dynamic behavior of an indoor environment, the simulator was modified to integrate doors whose states randomly change. RIDE was utilized to generate the MDP and MOMDP models and to test the temporal logic-based control strategies based on these models.

As described in [5], the states of the MDP (and thus also the fully observable states of the MOMDP) are actually adjacent pairs of regions (e.g. I_2R_3 represents the state in which the robot was in I_2 and is now in R_3). The resulting models have 64 states.

As described in Sec. IV, for the first setting, states containing a region adjacent to a door were duplicated to account for the two possible states the doors could be in. Furthermore, in this case the robot was capable of performing actions FollowRoad, GoRight, GoLeft, GoStraight, TurnAway, and Wait. The transition probabilities associated to each action (with the exception of Wait) were computed through extensive simulation. In order to capture the relevant characteristics of the environment, the simulator was used to model the robot motion with and without doors. In each trial, the robot was initialized at the beginning of the region representing each state. If this region was a road, then the FollowRoad controller was applied until the system transitioned to the next state. On the other hand, if this region was an intersection, each one of the actions allowed at this state was applied and the resulting transition was recorded. The results were then integrated into the transition probabilities.

In the second and third settings, the primitives FollowRoad, GoRight, GoLeft, and GoStraight were available. The fully observable state transition probabilities were obtained in the same manner as in the first setting.

Since the third setting included partial observability, it was necessary to also determine the probability distribution over the set of observations. This was accomplished using the modified RIDE simulator and incorporating a function that generated the observations using the data from the simulated laser range finder. The probabilities were generated as follows. First, the robot was initialized in a region from which a transition to a region adjacent to a door was possible. Then, an action was selected and executed. If the robot moved to a region adjacent to a door then an observation was made. The process was repeated 500 times for each action and from each initial location. The number of times that an observation was made was saved in the form of probabilities.

B. Case Study

Consider the environment shown in Fig. 1 and the following motion specification:

"Reach *Destination* by going through either only *Safe* regions or through *Relatively safe* regions only if *Power supply* is available at the *Relatively safe* regions."

This specification can be translated to the following PCTL formula

$$\phi :: \mathbb{P}_{max=?}[\mathbf{S} \vee (\mathbf{R} \wedge \mathbf{P}) \; \mathcal{U} \; \mathbf{D}]. \quad (3)$$

Using the framework described in this paper, a control policy was determined under each of the three settings given in Secs. IV-VI. Fig. 3 depicts scenes from the simulation of the control strategy maximizing the probability of satisfying (3) under Setting 1. In the first scene, the robot is initially

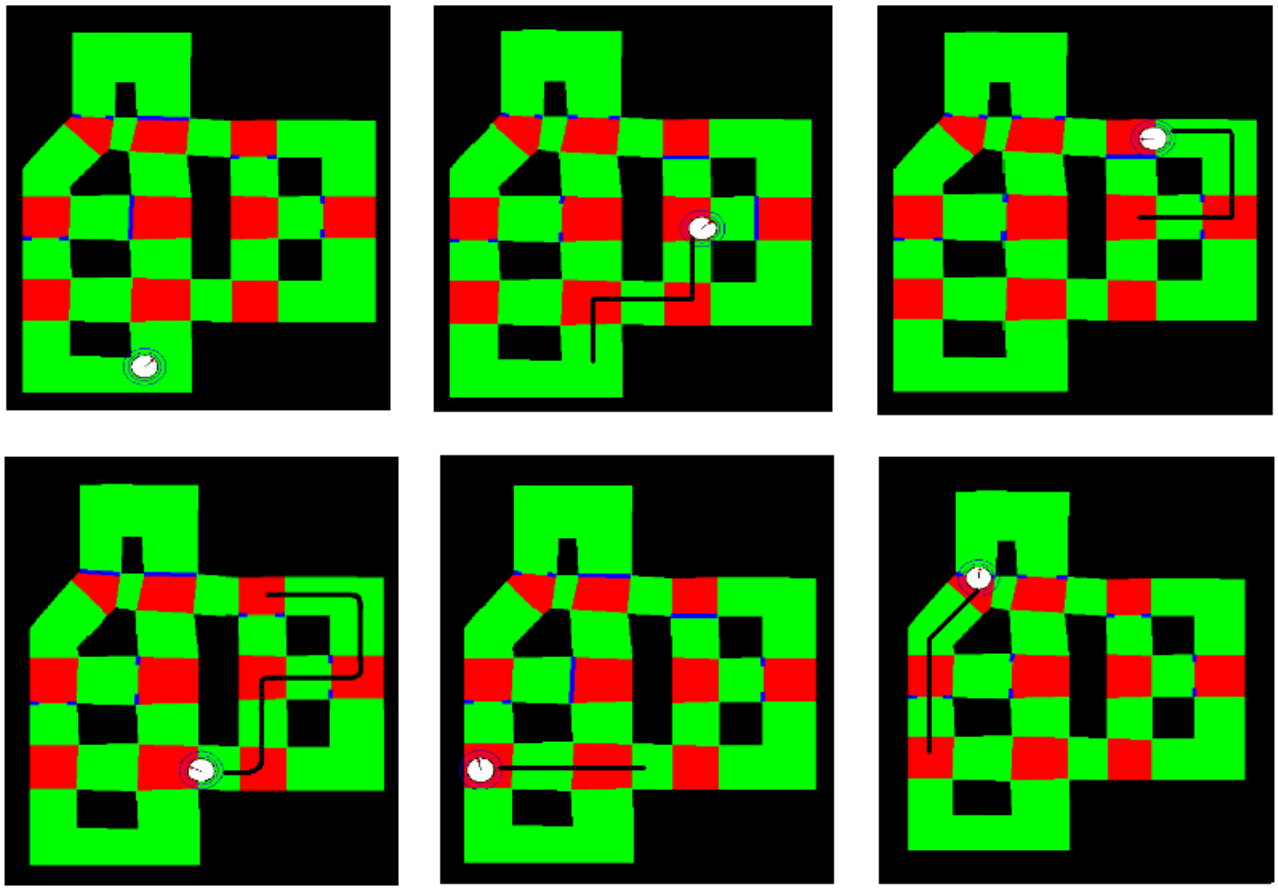

Fig. 3. Snapshots (to be read left-to-right and top-to-bottom) from a movie showing a robot motion produced by applying the control strategy maximizing the probability of satisfying ϕ.

at C_1. The robot follows the path $C_1 I_2 C_3 I_1 C_5 I_5 C_8 I_4 C_{10} I_8$. This route is partially shown in the second, and third scenes. Upon reaching I_8, the robot finds door 4 to be closed. It then pauses to see if the door will open. After 1 step, the door remains closed and the robot turns, following the route $I_8 C_{10} I_4 C_8 I_5 C_5 I_1 C_3 I_2 C_4 I_3 C_7 I_7 C_{13} I_{10}$. Scenes four, and five show part of this route. Finally, scene six demonstrates the efficacy of the strategy when the door separating the robot from the destination switches from closed to open, allowing the robot to achieve the task based on the specification.

Using the computational framework developed in [5], the maximum probability under Setting 1 was 0.425. To validate the computed probability, 500 simulations were performed. The simulations demonstrated that the probability of satisfying (3) was 0.416, reasonably close to the calculated probability. The maximum probabilities of satisfying (3) under Settings 2 and 3 were obtained performing 500 simulations. The resulting probabilities were 0.328 and 0.216, for the second and third settings, respectively.

These simulation results agree with an intuitive understanding of the role of information and sensing. The more information and accuracy the robot has about the states of the doors, the easier it is to choose and perform the best strategy. Moreover, having prior knowledge of the states of the doors, as in Setting 1, allows us to get an *a priori* estimate of the maximum probability of satisfaction. The lack of that *a priori* knowledge, depicted in Settings 2 and 3, does not allow us to obtain this measure. However, the assumption of perfect knowledge can be difficult to satisfy in practice, especially without environment modification.

VIII. CONCLUSIONS

We presented a solution to the automatic deployment of a mobile robot moving in a dynamic indoor environment according to a task specified as a temporal logic specification. Three different settings were considered, each assuming different levels of knowledge about the states of the doors and robot's sensing capabilities. Modeling these settings as MO(MDP)s allowed us to adapt and use PCTL model checking techniques to find the optimal or suboptimal control strategy that maximizes the probability of satisfying the specification as a PCTL formula. One of the key limitations of the proposed approaches is the assumption that the doors only change state in synchrony with the motion of the robot.

REFERENCES

[1] S. Ong, S. Png, D. Hsu and W. Lee, "POMDPs for robotic tasks with mixed observability", *The International Journal of Robotics Research*, 2010, vol. 29, no. 8, pp. 1053-1068.

[2] A. Bianco and L. de Alfaro, “Model checking of probabilistic and nondeterministic systems”, Springer-Verlag, 1995, pp. 499-513.
[3] G. E. Fainekos, H. Kress-Gazit and G. J. Pappas, “Hybrid controllers for path planning: a temporal logic approach”, *In IEEE Conference on Decision and Control*, 2005, pp. 4885-4890.
[4] H. K. Gazit, G. Fainekos and G. J. Pappas, “Wheres Waldo? sensorbased temporal logic motion planning”, *In IEEE International Conference on Robotics and Automation*, 2007, pp. 3116–3121.
[5] M. Lahijanian, J. Wasniewski, S. B. Andersson and C. Belta, “Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees”, *International Conference on Robotics and Automation*, 2010, pp. 3227-3232.
[6] S. G. Loizou and K. J. Kyriakopoulos, “Automatic synthesis of multiagent motion tasks based on LTL specifications”, *In 43rd IEEE Conference on Decision and Control*, 2004, vol. 26, pp. 153-158.
[7] E. M. M. Clarke, D. Peled and O. Grumberg, *Model Checking*. MIT Press, Cambridge, MA; 1999.
[8] ”Robotic indoor environment.” [Online]. Available: hyness.bu.edu/ride/
[9] H. Hansson and B. Jonsson, “A Logic for reasoning about time and reliability”, *In Formal Aspects of Computing*, 1994, vol. 6, pp. 102-111.
[10] M. Kwiatkowska, G. Norman and D. Parker, “Stochastic model checking”, *Lecture Notes in Computer Science*, Springer-Verlag, 2007, vol. 4486, pp. 220-270.
[11] D. Bertsekas, *Dynamic Programming and Optimal Control, Vol. 2*, 3rd Edition, Athena Scientific, Nashua, NH; 2007.
[12] C. Boutillier, T. Dean and S. Hanks, “Decision theoretic planning: structural assumptions and computational leverage”, *Journal of AI Research*, 1999, vol. 11, pp. 1-94.
[13] A. Cassandra, L. Kaelbling and M. Littman, “Acting optimally in partially observable stochastic domains”, *In Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1994, vol. 2, pp. 1023-1028.
[14] L. Kaelbling, M. Littman and A. Cassandra, “Planning and acting in partially observable stochastic domains” *Artificial Intelligence*, 1998, vol. 101, pp. 99-134.
[15] M. Kloetzer and C. Belta, “A fully automated framework for control of linear systems from temporal logic specifications”, *IEEE Transactions on Automatic Control*, 2008, vol. 53, no. 1, pp. 287297.
[16] M.Kloetzer and C. Belta, “Dealing with non-determinism in symbolic control”, *In Hybrid Systems: Computation and Control: 11th International Workshop, ser. Lecture Notes in Computer Science*, M. Egerstedt and B. Mishra, Eds. Springer Berlin, Heidelberg, 2008, pp. 287300.
[17] M. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York, NY; 1994.
[18] F. Ramponi, D. Chatterjee, S. Summers and J. Lygeros, “On the connections between PCTL and dynamic programming”, *In Proceedings of the 13th ACM international conference on Hybrid systems: computation and control*, 2010, pp. 253-262.