51st IEEE Conference on Decision and Control
December 10-13, 2012. Maui, Hawaii, USA

# Multi-Agent Persistent Monitoring in Stochastic Environments with Temporal Logic Constraints

Yushan Chen, Kun Deng and Calin Belta

*Abstract*— **In this paper, we consider the problem of generating control policies for a team of robots moving in an environment containing elements with probabilistic behaviors. The team is required to achieve an optimal surveillance mission, in which a certain proposition needs to be satisfied infinitely often. The goal is to minimize the average time between satisfying instances of the proposition, while ensuring that the mission is accomplished. By modeling the robots as Transition Systems and the environmental elements as Markov Chains, the problem reduces to finding an optimal control policy satisfying a temporal logic specification on a Markov Decision Process. The existing approaches for this problem are computational intensive and therefore not feasible for a large environment or a large number of robots. To address this issue, we propose an approximate dynamic programming framework. Specifically, we choose a set of basis functions to approximate the optimal cost and find the best parameters for these functions based on the least-square approximation. We develop an approximate policy iteration algorithm to implement our framework. We provide illustrative case studies and evaluate our method through simulations.**

## I. Introduction

Recently there has been an increasing interest in using temporal logics, such as Linear Temporal Logic (LTL), Computation Tree Logic (CTL), and $\mu$-calculus as task specifications for mobile robotics [1]–[3]. These logics are appealing because they allow for high-level, expressive specifications. In particular, LTL can be used to specify persistent monitoring *e.g.,* "Observe regions A and then B infinitely often. Never enter D unless coming directly from C."

Most of the existing works using LTL assume that a finite model of the robot motion in the environment is available. If this is deterministic, control strategies from specifications given as LTL formulae can be found through adaption of off-the-shelf model checking algorithms [4]. If the model is nondeterministic, the control problem can be mapped to a Rabin game [5], and to a Büchi [6] or GR(1) game if the specifications are restricted to fragments of LTL [1], [2]. If the model of the system is probabilistic, the control problem reduces to synthesizing a control policy for a Markov Decision Process (MDP) subject to LTL satisfaction constraints [7].

In this paper, we consider the problem of controlling a team of robots in an environment containing elements with probabilistic behaviors from a temporal logic persistent monitoring task given in Linear Temporal Logic (LTL). In addition, we optimize the long-term behavior of the team by minimizing the time between consecutive satisfactions of a "persistence" property. An example of a mission that we can accommodate with the proposed computational framework is "Download data from A or B and then upload it in C infinitely often. Minimize the expected value of the time between consecutive uploads."

The motivation for this work is to bridge the gap between the (deterministic) multi-agent optimal control problems [8] and MDP optimal control problems [9] subject to temporal logic constraints. Similar to [8], we assign a clock to each robot to measure its travel time between the regions of the environment. However, since we consider probabilistic systems, we model the motion of team as a labeled and weighted Markov Decision Process (MDP), rather than a timed automaton as in [8]. Our original problem translates to finding an optimal policy enforcing the satisfaction of an LTL formula on the team MDP. Starting from the observation that the algorithm proposed in [9] does not scale with the large size of our multi-agent system, we propose an approximate dynamic programming framework, which points to a trade-off between optimality and complexity. A set of basis functions is carefully chosen based on the Krylov space method [10] to approximate the optimal solution, and then the best approximation is achieved by minimizing the least-square error [10]. In summary, the main contributions of this paper are to extend the results in [9] to multi-agent systems, and to propose a sub-optimal solution based on approximate dynamic programming.

## II. Problem Formulation

*Notation:* For a finite set $\Pi$, we use $|\Pi|$ and $2^{\Pi}$ to denote its cardinality and power set, respectively. An infinite (finite) word $\alpha^0\alpha^1\ldots$ ($\alpha^0\alpha^1\ldots\alpha^n$) over a set $\Pi$ is an infinite (finite) sequence of elements from $\Pi$. $\Pi^\omega$ ($\Pi^*$) is the set of all infinite (finite) words over $\Pi$. $\Pi^k$ denotes the set of all words over $\Pi$ with length $k$.

### A. Environment, Door, and Robot Models

In this paper, we consider a team of robots moving in an environment consisting of both static and stochastically changing elements. To keep the discussion focused, we consider an indoor-like environment consisting of rooms (static) and doors (changing and stochastic). Formally, such an environment can be modeled as a tuple:

$$\mathcal{E} = (\mathcal{V}, \rightarrow_{\mathcal{E}}, \Pi, L_{\mathcal{E}}) \tag{II.1}$$

where (i) $\mathcal{V}$ is a set of labels for the rooms in the environment; (ii) $\rightarrow_{\mathcal{E}} \subseteq \mathcal{V}\times\mathcal{V}$ is the adjacency relation of the rooms;

Y. Chen and C. Belta are with Boston University, {yushanc, cbelta}@bu.edu, and K. Deng is with University of Illinois at Urbana-Champaign, kundeng2@illinois.edu. This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088 and AFOSR YIP FA9550-09-1-020 at Boston University.

978-1-4673-2066-5/12/$31.00 ©2012 IEEE

(iii) $\Pi$ is a set of atomic propositions; and (iv) $L_{\mathcal{E}} : \mathcal{V} \to 2^{\Pi}$ is a labeling function over the set of rooms, where $L_{\mathcal{E}}(v)$ represents the set of atomic propositions that hold true in room $v$. An atomic proposition $\alpha \in \Pi$ can be used to represent a service request occurring in the environment, or a property of a location (*e.g.,* $v$ is unsafe).

Assume that there is a set $\mathcal{D}$ of doors located in the environment. Two adjacent rooms in $\mathcal{E}$ may be separated by a door, which can be open or closed. To capture the door locations, we define a partial function $\mathcal{F}_{\mathcal{D}} : \mathcal{V} \times \mathcal{V} \to \mathcal{D} \cup \emptyset$, where $\mathcal{F}_{\mathcal{D}}$ is defined for all $(v, v') \in \to_{\mathcal{E}}$; $\mathcal{F}_{\mathcal{D}}(v, v') = i$ means that the adjacent rooms $v$ and $v'$ are separated by door $i$, whereas $\mathcal{F}_{\mathcal{D}}(v, v') = \emptyset$ means that there exists no door in between $v$ and $v'$. We assume that the doors behave independently from the robots, and the status of each door $i \in \mathcal{D}$ (*i.e.,* open or closed) evolves according to a finite discrete time Markov chain. Formally, we have

**Definition II.1 (Door Model)** *Each door $i \in \mathcal{D}$ is modeled as a discrete-time labeled Markov chain $\mathcal{C}_i = (S_i, \iota_i, P_i, \Omega_i, L_i^{\mathcal{C}})$ where (i) $S_i$ is a set of states; (ii) $\iota_i : S_i \to [0, 1]$ is an initial distribution with $\sum_{s \in S_i} \iota_i(s) = 1$; (iii) $P_i : S_i \times S_i \to [0, 1]$ is a transition probability function such that $\forall s \in S_i$, $\sum_{s' \in S_i} P_i(s, s') = 1$; (iv) $\Omega_i = \{o, c\}$ is a status set, where $o$ and $c$ stand for open and closed, respectively; (v) $L_i^{\mathcal{C}} : S_i \to \Omega_i$ is a labeling function.*

We assume that the time is uniformly discretized and we use $t$ to denote the $t$th time instance. We assume that the time domain is $\mathbb{N}$ and initially, $t = 0$. For the sake of simplicity, we assume that the door models (*i.e.,* Markov chains) take a transition every time instance. The *runs* of $\mathcal{C}_i$ are defined as infinite state sequences $r_i = s^0 s^1 \ldots \in S_i^{\omega}$, such that $\iota_i(s^0) > 0$, $s^t \in S_i$ and $P_i(s^t, s^{t+1}) > 0$, $\forall t \geq 0$. A run $s^0 s^1 \ldots$ generates an *output word* $L_i^{\mathcal{C}}(s^0) L_i^{\mathcal{C}}(s^1) \ldots \in \Omega_i^{\omega}$. In this paper, an output word of $\mathcal{C}_i$ is also referred to as a behavior of door $i \in \mathcal{D}$.

We consider a team of robots moving in environment $\mathcal{E}$, whose motions are restricted by the doors $\mathcal{D}$. The robots are assumed to have a negligible size. We denote $\mathcal{R}$ as an index set for the robots. To capture the interaction between the robots and the doors, we model each robot $k \in \mathcal{R}$ as a game transition system (see Def. II-A below), denoted by $\mathcal{T}_k$, $k \in \mathcal{R}$. There are two players in the game: the robot (player) and the doors (adversary). The set of states of $\mathcal{T}_k$ is partitioned in two sets: robot set $\mathcal{V}$, at which the robot takes control, and door set $Q_{\mathcal{D}}$, at which the doors decide the next transitions. Each door $i \in \mathcal{D}$, which separates rooms $v$ and $v'$, is represented by two states (one for each room), denoted by $q_i^v$ and $q_i^{v'}$. Thus, $Q_{\mathcal{D}} = \cup_{i \in \mathcal{D}} \{q_i^v, q_i^{v'} \mid i = \mathcal{F}_{\mathcal{D}}(v, v'),\ v, v' \in \mathcal{V}\}$.

**Definition II.2 (Robot Model)** *We model each robot $k \in \mathcal{R}$ as a game transition system $\mathcal{T}_k = (Q_k, v_k^{in}, \to_k, \Pi_k, L_k, g_k)$, where (i) $Q_k = \mathcal{V} \cup Q_{\mathcal{D}}$ is a finite set of states; (ii) $v_k^{in} \in \mathcal{V}$ is an initial state; (iii) $\to_k \subseteq Q_k \times Q_k$ is a transition relation where $\forall (v, v') \in \to_{\mathcal{E}}$, we have $(v, v') \in \to_k$ iff $F_{\mathcal{D}}(v, v') = \emptyset$, and $(v, q_i^v), (q_i^v, v), (q_i^v, v') \in \to_k$ iff $F_{\mathcal{D}}(v, v') = i$; (iv) $\Pi_k \subseteq \Pi$ is a set of atomic propositions; (v) $L_k : \mathcal{V} \to 2^{\Pi_k}$ is a labeling function over the states of the robot; (vi) $g_k : \to_k \to \mathbb{N}$[1] is a weight function that assigns a non-negative integer to each transition.*

A transition $(q, q') \in \to_k$ is also denoted by $q \to_k q'$. At a robot state $v \in \mathcal{V}$, the robot chooses its next location $v'$, where $(v, v') \in \to_{\mathcal{E}}$. If $v$ and $v'$ are not separated by a door, the robot starts moving towards $v'$ and will reach $v'$ after a certain period of time. Thus, we have $(v, v') \in \to_k$, and $g_k((v, v'))$ captures the travel time for the robot to go from $v$ to $v'$. For robot $k$, $\forall k \in \mathcal{R}$, staying at the same location is assumed to take one time interval, *i.e.,* $\forall (v, v) \in \to_k$, $g_k(v, v) = 1$. If door $i$ separates $v$ and $v'$, the robot stays at $v$ if door $i$ is closed, and moves to $v'$ otherwise. We use $(v, q_i^v) \in \to_k$ to represent that the robot plans to go through door $i$, and $(q_i^v, v), (q_i^v, v') \in \to_k$ to capture that the door decides the next location of the robot. In addition, we have $g_k((v, q_i^v)) = 0$, $g_k((q_i^v, v)) = 1$ (*i.e.,* the robot is forced to stay at $v$ for one time interval); $g_k((q_i^v, v')) \in \mathbb{N}$ captures the travel time between $v$ and $v'$. We assume that to move from $v$ to $v'$, which are separated by door $i$, each robot first takes one time interval to go through the door, and then travels $g_k((q_i^v, v')) - 1$ time intervals to reach $v'$. By specifying different $\Pi_k \subseteq \Pi$, we can assign different atomic propositions to robot $k$. For example, a service request is contained in $\Pi_k$ means that robot $k$, $k \in \mathcal{R}$, is capable of servicing the request. An example of $\mathcal{T}_k$ is shown in Fig. 1.

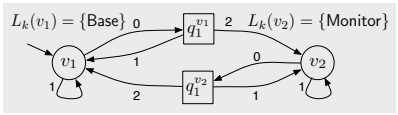


Fig. 1. An example of a game transition system $\mathcal{T}_k$, $k \in \mathcal{R}$. The robot and door states are represented by circles and squares, respectively.

A *run* of $\mathcal{T}_k$ is an infinite sequence $q^0 q^1 \ldots \in Q_k^{\omega}$ such that $q^0 = v_l^{in}$ and $q^n \to_k q^{n+1}$, $\forall n \geq 0$. By removing all the door states from the run of $\mathcal{T}_k$, we can obtain an infinite path of the robot, denoted by $\mathbb{P}_k = v_k^0 v_k^1 \ldots \in \mathcal{V}^{\omega}$. A path of the robot $\mathbb{P}_k = v_k^0 v_k^1 \ldots$ generates an *output word* of the robot, denoted by $\mathbb{O}_k = L_k(v_k^0) L_k(v_k^1) \ldots$, and an infinite sequence of time instances $\mathbb{T}_k := \mathbb{T}_k^0 \mathbb{T}_k^1 \ldots$ such that $L_k(v_k^n)$ is satisfied at time $\mathbb{T}_k^n$.

**Definition II.3 (Team Output Word (Behavior))** *Given a set of robot paths $\{\mathbb{P}_k, k \in \mathcal{R}\}$, and the corresponding event time sequences $\{\mathbb{T}_k, k \in \mathcal{R}\}$, a time sequence for the team, denoted by $\mathbb{T}_{team} = \mathbb{T}^0 \mathbb{T}^1 \ldots$, is obtained by taking the union $\cup_{k \in \mathcal{R}} \mathbb{T}_k$ and ordering this set in an ascending order. The team output word (behavior) produced by $\{\mathbb{P}_k, k \in \mathcal{R}\}$ is defined as $\mathbb{O}_{team} = o^0 o^1 \ldots$, where $o^n \in \cup_{k \in \mathcal{R}} \Pi_k$ ($n \geq 0$) is the union of all propositions satisfied at $\mathbb{T}^n$.*

### *B. Task Specification*

In this paper, we consider robot missions requiring infinite executions, such as surveillance, persistent monitoring, and pickup-delivery tasks. Linear Temporal Logic (LTL) [11]

[1] In general, a weight function assigns real numbers to the transitions. In this paper, since $g_k$ is used to capture the travel time of the robot and the time is discretized, we use natural numbers.

offers a formal framework for describing such missions. A detailed description of the syntax and semantics of LTL is beyond the scope of this paper and can be found in [11]. Roughly, an LTL formula is built up from a set of atomic propositions $\Pi$, standard Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), $\Rightarrow$ (implication), and temporal operators $\mathsf{X}$ (next), $\mathsf{U}$ (until), $\mathsf{F}$ (eventually), $\mathsf{G}$ (always). The semantics of LTL formulae are given over infinite words over $2^{\Pi}$, where $\Pi = \cup_{k\in\mathcal{R}}\Pi_k$, such as the team output words (see Def. II.3). As an example, let us consider a persistent monitoring task for one robot, whose model is shown in Fig. 1. The specification "monitor infinitely many times, and come back to base every time after monitoring" translates to formula $\phi = \mathsf{GF}\ \mathsf{Monitor} \wedge \mathsf{G}(\mathsf{Monitor} \Rightarrow \mathsf{X}\ \mathsf{Base})$.

In addition, we define a special task, called "optimizing" task, which is required to be executed infinitely often. We want to minimize the time in between two consecutive executions of this task. Specifically, our specification is an LTL formula of the form

$$\phi := \varphi \wedge \mathsf{GF}\psi, \tag{II.2}$$

where $\varphi$ can be any LTL formula over $\Pi = \cup_{k\in\mathcal{R}}\Pi_k$, and $\psi$ is a boolean combination of atomic propositions in $\Pi$. We use $\psi$ to capture the optimizing task and $\phi$ to specify other missions or rules that must be obeyed. Given a team output word $\mathbb{O}_{team} = o^0o^1\ldots$, we say the optimizing task $\psi$ is executed by the team at time $\mathbb{T}^n$, $n \geq 0$, if $o^n$ satisfies $\psi$.

### C. Problem Formulation

Our goal is to control the robots to accomplish a mission (Eqn. (II.2)), and also minimize the time in between two consecutive executions of the optimizing task $\psi$.

**Definition II.4 (Robot Control Policy)** *A history dependent control policy for robot $k \in \mathcal{R}$ is defined as an infinite sequence $\pi_k = \{\mu_k^0, \mu_k^1, \ldots\}$, where $\mu_k^n : \prod_{k\in\mathcal{R}}\mathcal{V}^* \times \prod_{i\in\mathcal{D}} S_i \rightarrow \{v \in \mathcal{V} \mid (v_k^n, v) \in \rightarrow_{\mathcal{E}}\}$.*

At time $\mathbb{T}_k^n$, given the sequence of the rooms $v_l^0 v_l^1 \ldots v_l^n$ visited by robot $l$, $\forall l \in \mathcal{R}$ ($n$ may vary for different robots), and the current state of door $i$, $\forall i \in \mathcal{D}$, the policy $\mu_k^n$ returns the next target location $v \in \mathcal{V}$, where $(v_k^n, v) \in \rightarrow_{\mathcal{E}}$, for robot $k$. Given the initial locations of the robots $\{v_k^{in}, k \in \mathcal{R}\}$, the control policies $\{\pi_k, k \in \mathcal{R}\}$, and the behaviors of the doors, we can produce a set of robot paths $\{\mathbb{P}_k = v_k^0 v_k^1 \ldots, k \in \mathcal{R}\}$, where at $v_k^n$, the next room $v_k^{n+1}$ is determined by both control $\mu_k^n$ and the status of the doors at time $\mathbb{T}_k^n$. Note that given $\{v_k^{in}, k \in \mathcal{R}\}$ and $\{\pi_k, k \in \mathcal{R}\}$, the resultant robot paths $\{\mathbb{P}_k, k \in \mathcal{R}\}$ and the corresponding team behavior $\mathbb{O}_{team}$ are not unique due to the stochasticity in the behaviors of the doors. The probability of the set of team behaviors satisfying an LTL formula is well defined since the team behaviors can be modeled as an Markov Decision Process (MDP) (the construction of this MDP is described later in the paper). Now we are ready to formulate the problem:

**Problem II.1 Given** *a partitioned environment $\mathcal{E}$ (Eqn. (II.1)), a team of robots modeled as game transition systems $\mathcal{T}_k$, $k \in \mathcal{R}$ (Def. (II-A)), a set of doors modeled as Markov Chains $\mathcal{C}_i$, $i \in \mathcal{D}$ (Def. (II-A)), and a specification in the form of an LTL formula $\phi$ (Eqn. (II.2)),* **Synthesize** *a set of robot control policies $\{\pi_k, k \in \mathcal{R}\}$ for the team, such that the team behaviors (Def. II.3) generated by $\{\pi_k, k \in \mathcal{R}\}$ satisfy $\phi$ with probability 1, and*

$$J(\{v_k^{in}, k \in \mathcal{R}\}) = \limsup_{n\rightarrow\infty} \mathsf{E}((\mathbb{T}_{team}^{\psi}(n+1) - \mathbb{T}_{team}^{\psi}(n))$$

*is minimized, where $\mathsf{E}(\cdot)$ denotes the expectation operator and $\mathbb{T}_{team}^{\psi}(n)$ stands for the time instance when the optimizing task $\psi$ is executed for the $n$th time.*

**Remark 1** *We restrict our attention to sets of policies, which produce team behaviors satisfying $\phi$ with probability 1, because the optimality problem is well posed (see [9]) only when there exists at least one such set of policies.*

Our approach to Prob. II.1 proceeds as follows. We first define the parallel composition of the game transition systems modeling the robots and the Markov chains capturing the door behaviors in the form of a weighted and labeled Markov Decision Process (MDP). We show that Prob. II.1 reduces to finding an optimal policy enforcing the satisfaction of an LTL formula on this MDP. Even though the latter problem was recently solved in [9], the resulting algorithm cannot be readily applied to our multi-agent problem due to the large size of the obtained MDP. To deal with this issue, we propose an approximate dynamic programming framework.

**Remark 2** *Throughout this paper we assume that the robots can deterministically choose their transitions. This assumption, which is made for simplicity of presentation, can be easily relaxed by allowing nondeterminism and probabilities in their transitions. The resulting model of the team would be a similar MDP.*

## III. MDP Construction

We start by equipping each robot $k \in \mathcal{R}$ with a clock, which keeps track of the amount of time that the robot has been traveling between robot states in the game transition system $\mathcal{T}_k$. The values of these clocks are non-negative integers and can be reset to zero. We initiate all the clocks at zeros. Given two robot states $(v, v')$ such that $v \rightarrow_k v'$, robot $k$ can transit from $v$ to $v'$ only when the current clock value plus 1 is equal to the travel time of this transition. After taking the transition, the clock will be reset to 0. When the clock value is smaller than the required travel time, the robot is in an intermediate state, which means that the robot has left $v$ and is moving towards its target location $v'$. Thus, for each robot, we insert a new state denoted by $v^{\rightarrow v'}$, $\forall (v, v') \in \rightarrow_{\mathcal{E}}$, such that the travel time between $v$ and $v'$ is greater than 1, to represent the intermediate state between $v$ and $v'$.

In addition, the MDP captures how the doors affect the motion of the robotic team. If robot $k$ is at $v$ and plans to move to $v'$, where $\exists i \in \mathcal{D}$ such that $v \rightarrow_k q_i^v \rightarrow_k v'$ (*i.e.,* $v$ and $v'$ are separated by a door), the next state of robot $k$ is decided by the current status of door $i$. If door $i$ is open, the robot starts moving from state $v$ to $v'$, and similarly, if $g_k(q_i^v, v') > 1$, the robot transits to the intermediate state $v^{\rightarrow v'}$. If the door is closed, the robot stays at the same state.

Therefore, the MDP can be seen as a special product of the game transition systems $\mathcal{T}_k$ with the set of clocks and the Markov Chains $\mathcal{C}_i$. Formally, it is defined as follows:

$$\mathcal{M}_\mathcal{G} = (Q_\mathcal{G}, \iota_\mathcal{G}, U_\mathcal{G}, P_\mathcal{G}, \Pi, L_\mathcal{G}, g_\mathcal{G}), \text{where} \quad \text{(III.1)}$$

(i) $Q_\mathcal{G} \subset \prod_{k\in\mathcal{R}}\{\mathcal{V}\cup\mathcal{V}_k^{\mathsf{mid}}\}\times\prod_{i\in\mathcal{D}} S_i \times \mathbb{N}^{|R|}$ is a set of states, where $\mathcal{V}_k^{\mathsf{mid}}$ denotes the set of intermediate states; for a more intuitive notation, we use $(\mathsf{st}_r, \mathsf{st}_d, \mathsf{clk})$ to represent a state in $Q_\mathcal{G}$, where $\mathsf{st}_r = (\overline{v}_1, \ldots, \overline{v}_{|\mathcal{R}|})$ is the states of the robots, $\mathsf{st}_d = (s_1, \ldots, s_{|\mathcal{D}|})$ is the states of the doors, and $\mathsf{clk} = (\mathsf{clk}_1, \ldots, \mathsf{clk}_{|\mathcal{R}|})$ is the clock values, one for each robot;

(ii) $\iota$ is an initial distribution such that $\iota_\mathcal{G}(\mathsf{st}_r, \mathsf{st}_d, \mathsf{clk}) = \prod_{i\in\mathcal{D}}\iota_i(\mathsf{st}_d[i])$, iff $\mathsf{st}_r = (v_1^{in}, \ldots, v_{|\mathcal{R}|}^{in})$ and $\mathsf{clk} = (0, 0, \ldots, 0)$, and $\iota_\mathcal{G}(\mathsf{st}_r, \mathsf{st}_d, \mathsf{clk}) = 0$ otherwise;

(iii) $U_\mathcal{G} \subseteq \prod_{k\in\mathcal{R}}\{\mathcal{V}\cup\{\epsilon_k\}\}$ is a set of controls, where $\epsilon_k$ is a dummy control $\forall k \in \mathcal{R}$;

(iv) $P_\mathcal{G} : Q_\mathcal{G}\times U_\mathcal{G}\times Q_\mathcal{G} \to [0,1]$ is a transition probability function such that $P_\mathcal{G}((\mathsf{st}_r, \mathsf{st}_d, \mathsf{clk}), (\overline{u}_1, \ldots, \overline{u}_{|\mathcal{R}|}), (\mathsf{st}'_r, \mathsf{st}'_d, \mathsf{clk}')) = \prod_{i\in\mathcal{D}} P_i(\mathsf{st}_d[i], \mathsf{st}'_d[i])$, iff $\forall k \in \mathcal{R}$, one of the following conditions holds:

1) $\mathsf{st}_r[k] = v, \mathsf{st}'_r[k] = v'$, $\overline{u}_k = v'$, $\mathsf{clk}_k = 0$, $\mathsf{clk}'_k = 1$, and, a) $\mathcal{F}_\mathcal{D}(v, v') = \emptyset$, $g_k(v, v') = 1$, or b) $\mathcal{F}_\mathcal{D}(v, v') = i$, $L_i^\mathcal{C}(\mathsf{st}_d[i]) = o$, $g_k(q_i^v, o) = 1$
2) $\mathsf{st}_r[k] = \mathsf{st}'_r[k] = v$, $\overline{u}_k = v$, $\mathsf{clk}_k = 0$, $\mathsf{clk}'_k = 0$, $\exists v', i,$ s.t. $\mathcal{F}_\mathcal{D}(v, v') = i$, $L_i^\mathcal{C}(\mathsf{st}_d[i]) = c$
3) $\mathsf{st}_r[k] = v$, $\mathsf{st}'_r[k] = v^{\to v'}$, $\overline{u}_k = v'$, $\mathsf{clk}_k = 0$, $\mathsf{clk}'_k = 1$, and a) $\mathcal{F}_\mathcal{D}(v, v') = \emptyset$, $g_k(v, v') > 1$, or b) $\mathcal{F}_\mathcal{D}(v, v') = i$, $L_i^\mathcal{C}(\mathsf{st}_d[i]) = o$, $g_k(q_i^v, v') > 1$
4) $\mathsf{st}_r[k] = \mathsf{st}'_r[k] = v^{\to v'}$, $\overline{u}_k = \epsilon_k$, $1 \le \mathsf{clk}_k < g_k(v, v') - 1$, and $\mathsf{clk}'_k = \mathsf{clk}_k + 1$
5) $\mathsf{st}_r[k] = v^{\to v'}$, $\mathsf{st}'_r[k] = v'$, $\overline{u}_k = \epsilon_k$, $\mathsf{clk}'_k = 0$, and a) $\mathcal{F}_\mathcal{D}(v, v') = \emptyset$, $g_k(v, v') = \mathsf{clk}_k + 1$, or b) $\mathcal{F}_\mathcal{D}(v, v') = i$, $g_k(q_i^v, v') = \mathsf{clk}_k + 1$

and $P_\mathcal{G}((\mathsf{st}_r, \mathsf{st}_d, \mathsf{clk}), (\overline{u}_1, \ldots, \overline{u}_{|\mathcal{R}|}), (\mathsf{st}'_r, \mathsf{st}'_d, \mathsf{clk}')) = 0$ otherwise;

(v) $\Pi = \cup_{k\in\mathcal{R}}\Pi_k$ is a set of atomic propositions;

(vi) $L_\mathcal{G} : Q_\mathcal{G} \to 2^\Pi$ such that $L_\mathcal{G}(\mathsf{st}_r, \mathsf{st}_d, \mathsf{clk}) = \cup_{k\in\mathcal{R}}\{L_k(\mathsf{st}_r[k]) \mid \mathsf{st}_r[k] \in \mathcal{V}\}$.

(vii) $g_\mathcal{G} : Q_\mathcal{G}\times U_\mathcal{G} \to 1$ is the trivial weight function that assign 1 to all transitions with probability larger than 0.

At an intermediate state $v^{\to v'}$, only a dummy control $\epsilon_k$ is enabled. The weight function $g_\mathcal{G}$ assigns 1 to all transitions since the states of the MDP evolve every time step (*i.e.,* this is because the states of the MCs evolve every time step).

**Remark 3** *The number of states $|Q_\mathcal{G}|$ of $\mathcal{M}_\mathcal{G}$ is bounded above by $\prod_{k\in\mathcal{R}}(|\mathcal{V}| + \sum_{\substack{(q,q')\in\to_k \\ g_k(q,q')>1}}(g_k(q,q') - 1)) \times \prod_{i\in\mathcal{D}}|S_i|$. The size of $\mathcal{M}_G$ can be reduced by removing states with only dummy actions $(\epsilon_1, \epsilon_2, \ldots, \epsilon_{|\mathcal{R}|})$ as inputs, and then adjusting the relative transitions accordingly. This direction will be considered in our future work.*

We define a control function $\mu_\mathcal{G} : Q_\mathcal{G} \to U_\mathcal{G}$, $\forall q_\mathcal{G} \in Q_\mathcal{G}$. An infinite sequence of control functions $\{\mu_\mathcal{G}^0, \mu_\mathcal{G}^1, \ldots\}$ is called an MDP policy. If $\mu_\mathcal{G}^t = \mu_\mathcal{G}, \forall t \ge 0$, we call it a *stationary* MDP policy and we denote it simply as $\mu_\mathcal{G}$. Given an initial state, an infinite sequence $q_\mathcal{G}^0 q_\mathcal{G}^1 \ldots$ on $\mathcal{M}$ generated under $\{\mu_\mathcal{G}^0, \mu_\mathcal{G}^1, \ldots\}$ is called a path on $\mathcal{M}$ if $\iota_\mathcal{G}(q_\mathcal{G}^0) > 0$, and $P_\mathcal{G}(q_\mathcal{G}^t, \mu^n(q_\mathcal{G}^t), q_\mathcal{G}^{t+1}) > 0$, $\forall t \ge 0$. A path $q_\mathcal{G}^0 q_\mathcal{G}^1 \ldots$ on $\mathcal{M}$ generates an *output* word $L_\mathcal{G}(q_\mathcal{G}^0)L_\mathcal{G}(q_\mathcal{G}^1)\ldots \in \Pi^\omega$ on $\mathcal{M}$. A path on $\mathcal{M}$ satisfies an LTL formula if and only if its corresponding output word satisfies the LTL formula. Given a path $q_\mathcal{G}^0 q_\mathcal{G}^1 \ldots$ of $\mathcal{M}_\mathcal{G}$ with $q_\mathcal{G}^t = (\mathsf{st}_r^t, \mathsf{st}_d^t, \mathsf{clk}^t)$, we can obtain a path $\mathbb{P}_k = v_k^0 v_k^1 \ldots$ of robot $k \in \mathcal{R}$, by removing all the intermediate states from $\mathsf{st}_r^0[k]\mathsf{st}_r^1[k]\ldots$. In addition, given $q_\mathcal{G}^0 q_\mathcal{G}^1 \ldots$, the corresponding set of paths $\{\mathbb{P}_k, k \in \mathcal{R}\}$, the team output word $\mathbb{O}_{team}$, and time sequence $\mathbb{T}_{team}$, it holds that a) $\forall t = \mathbb{T}^n \in \cup_{k\in\mathcal{R}}\mathbb{T}_k, L_\mathcal{G}(q_\mathcal{G}^t) = o^n$ and b) $\forall t \in \mathbb{N}, t \notin \cup_{k\in\mathcal{R}}\mathbb{T}_k$, $L_\mathcal{G}(q_\mathcal{G}^t) = \emptyset$.

**Definition III.1 (Inducing a policy from $\mathcal{M}_\mathcal{G}$)** *A policy $\{\mu_\mathcal{G}^0, \mu_\mathcal{G}^1, \ldots\}$ on $\mathcal{M}_\mathcal{G}$ induces a control policy $\{\mu_k^0, \mu_k^1, \ldots\}$ for each robot $k \in \mathcal{R}$, by setting $\mu_k^n(v_k^n) = \mu_\mathcal{G}^{\mathbb{T}_k^n}((\mathsf{st}_r^{\mathbb{T}_k^n}, \mathsf{st}_d^{\mathbb{T}_k^n}, \mathsf{clk}^{\mathbb{T}_k^n}), \forall n \ge 0$.*

Note that $\mathsf{st}_r^{\mathbb{T}_k^n}$ and $\mathsf{clk}^{\mathbb{T}_k^n}$ can be decided by keeping track of the motion of the robots. Therefore, our optimal control synthesis problem reduces to the problem of finding a control policy $\{\mu_\mathcal{G}^0, \mu_\mathcal{G}^1, \ldots\}$ on $\mathcal{M}_\mathcal{G}$, such that a) $\{\mu_\mathcal{G}^0, \mu_\mathcal{G}^1, \ldots\}$ satisfies the LTL formula $\phi$ with probability 1, and b) the expected time in between visiting states in $Q_\mathcal{G}$ satisfying $\psi$ is minimized when $t \to \infty$ (*i.e.,* optimize the long-term behavior of the robotic team). To formalize this, we let $M_\phi$ denote the set of policies satisfying the LTL formula $\phi$ with probability 1, and $Q_\psi$ denote the set of states where $\psi$ (*i.e.,* the optimizing task) holds true. We say that each visit of the MDP path to the set $Q_\psi$ completes a cycle. Given an MDP sample path $q_\mathcal{G}^0 q_\mathcal{G}^1 \ldots$, we use $C(q_\mathcal{G}^0, \ldots, q_\mathcal{G}^N)$ to denote the number of cycles completed at stage $N+1$. Then, we are interested in finding an optimal policy in $M_\phi$ to minimize the *average cost per cycle* (ACPC) given the initial distribution. Therefore, Prob. II.1 reduces to:

**Problem III.1** *Find a policy $\{\mu_\mathcal{G}^0, \mu_\mathcal{G}^1, \ldots\} \in M_\phi$ on $\mathcal{M}_\mathcal{G}$ that minimizes*

$$J_\pi(\iota_\mathcal{G}) = \limsup_{N\to\infty} \mathsf{E}\left\{\frac{\sum_{n=0}^N g_\mathcal{G}(q_\mathcal{G}^n, \mu_\mathcal{G}^n(q_\mathcal{G}^n))}{C(q_\mathcal{G}^0 q_\mathcal{G}^1 \ldots q_\mathcal{G}^N)}\right\}. \quad \text{(III.2)}$$

*where $\mathsf{E}\{\cdot\}$ denotes the expectation operator.*

## IV. Approximate Dynamic Programming Approach

### A. Existing Approach to Prob. III.1

The existing approach to Prob. III.1 can be divided into two parts: 1) LTL synthesis part, where $M_\phi$ is computed, and 2) optimizing part, where a policy in $M_\phi$ minimizing Eqn. (III.2) is found. The LTL synthesis part proceeds with constructing a product MDP $\mathcal{M}_\mathcal{P} = \mathcal{M}_\mathcal{G} \times \mathcal{A}_\phi$, where $\mathcal{A}_\phi$ is a Deterministic Rabin Automaton accepting all and only words satisfying $\phi$ [11]. Then, a set of accepting maximum end components (AMECs) of $\mathcal{M}_\mathcal{P}$ is computed. These AMECs are sub-MDPs of $\mathcal{M}_\mathcal{P}$ that are *communicating*. As stated in [11], a stationary policy on $\mathcal{M}_\mathcal{P}$ that can reach an AMEC given the initial distribution with probability 1 induces a policy on $\mathcal{M}$ that satisfies the LTL formula $\phi$ with probability 1. In the optimizing part, a stationary policy minimizing the ACPC cost defined in (III.2) is computed for each obtained AMEC. Since we only aim to optimize the

long-term behavior of the system, we only need to solve the optimization problem within each (reachable) AMEC. The ACPC optimization problem is defined as follows:

**Problem IV.1** *Given a communicating MDP, denoted by $\mathcal{M} = (Q, \iota, U, P, \Pi, L, g)$ and the optimizing task $\psi$, find a stationary policy $\mu$, that minimizes the cost from Eqn.* (III.2).

A stationary policy $\mu$ on $\mathcal{M}$ induces a finite-state Markov chain where its set of states is $Q$ and the transition probability from state $q$ to $q'$ is $P(q, \mu(q), q')$. We use $P_\mu$ to denote the transition probability matrix: $P_\mu(q, q') := P(q, \mu(q), q')$. We define a cost per stage vector $g_\mu$, where $g_\mu(q) := g(q, \mu(q))$. A stationary policy $\mu$ is said to be *proper* if, under $\mu$, all initial states have positive probabilities to reach the set $Q_\psi$ in a finite number of stages. It is shown in [9] that Prob. IV.1 can be converted to the traditional average cost per stage (ACPS) problem (see [10]). To achieve this, another MDP is constructed such that solving the ACPS problem in the new MDP is equivalent to finding a solution to Prob. IV.1. To obtain a new MDP, we first introduce two $|Q| \times |Q|$ matrices:

$$\overleftarrow{P}_\mu(q, q') = \begin{cases} P_\mu(q, q') & \text{if } q' \in Q_\psi \\ 0 & \text{otherwise} \end{cases} \quad \text{(IV.1)}$$

$$\overrightarrow{P}_\mu(q, q') = \begin{cases} P_\mu(q, q') & \text{if } q' \notin Q_\psi \\ 0 & \text{otherwise} \end{cases} \quad \text{(IV.2)}$$

The matrix $(I - \overrightarrow{P}_\mu)$ is shown to be non-singular for any proper policy. Given $\overleftarrow{P}_\mu$ and $\overrightarrow{P}_\mu$, we obtain the transition probability matrix, denoted by $\widetilde{P}_\mu$, and the cost per stage vector, denoted by $\widetilde{g}_\mu$, of the new MDP. Formally, we have $\widetilde{P}_\mu := (I - \overrightarrow{P}_\mu)^{-1}\overleftarrow{P}_\mu$, and $\widetilde{g}_\mu := (I - \overrightarrow{P}_\mu)^{-1} g_\mu$. The matrix $\widetilde{P}_\mu$ is also a stochastic matrix. Given the new MDP with $\widetilde{P}_\mu$ and $\widetilde{g}_\mu$, Eqn. (III.2) in Prob. IV.1 is proven to be equal to the average cost per stage (ACPS) for the new MDP: $J_\mu(\iota) = \lim_{N\to\infty} \mathsf{E}\left\{\frac{\sum_{n=0}^{N} \widetilde{g}(q_n, \mu(q_n))}{N+1}\right\}$. As shown in [9], $J_\mu(\iota)$ does not depend on the initial states. Thus, we have $J_\mu(q) = \lambda_\mu$, $\forall q \in Q$, where $\lambda_\mu$ is a scalar. In addition, a *relative cost* vector $h_\mu := \lim_{N\to\infty} \sum_{k=0}^{N} (\widetilde{P}_\mu^k \widetilde{g}_\mu - \lambda_\mu \mathsf{1})$ is defined to quantify the total deviation from the average cost.

*Dynamic Programming (DP) Approach to Prob. IV.1:* As shown in Prop. IV.10 of [10], for any policy $\mu$, the average cost $\lambda_\mu$ associated with the relative cost vector $h_\mu$ satisfies the Bellman's equation $\lambda_\mu \mathsf{1} + h_\mu = \widetilde{g}_\mu + \widetilde{P}_\mu h_\mu$. The solution to this equation can be made unique by eliminating one degree of freedom [10], such as adding one more linear equation for $h_\mu$: $\mathsf{1}^T h_\mu = 0$. We denote $\mu^\star$ as the stationary policy $\mu^\star$ minimizing (III.2) over all policies on $\mathcal{M}$. Let $\lambda^\star$ and $h^\star$ denote the average cost and relative cost vector corresponding to the policy $\mu^\star$, respectively. One of common methods to find the optimal policy $\mu^\star$ uses the policy iteration algorithm (PIA) [10]. However, for large state spaces, PIA is computationally intensive. At each iteration, the computational complexity is of order $O(|Q|^3)$.

### B. Approximate DP Approach to Prob. IV.1

Instead of searching for an optimal solution to Prob. IV.1, we employ the function approximation method to compute sub-optimal solutions with less computational complexity [10].

*1) Linear Parametric Function Approximation:* We employ the function approximation method to approximate the solution to the Bellman's equation $\lambda_\mu \mathsf{1} + h_\mu = \widetilde{g}_\mu + \widetilde{P}_\mu h_\mu$. According to Prop IV.10 of [9], the Bellman's equation can be represented in the following equivalent form

$$\lambda_\mu \mathsf{1} + h_\mu = g_\mu + P_\mu h_\mu + \lambda_\mu \overrightarrow{P}_\mu \mathsf{1}. \quad \text{(IV.3)}$$

Together with $\mathsf{1}^T h_\mu = 0$, Eqn. (IV.3) can be expressed compactly as $A_\mu x_\mu = b_\mu$ where

$$A_\mu := \begin{bmatrix} I - P_\mu & \mathsf{1} - \overrightarrow{P}_\mu \mathsf{1} \\ \mathsf{1}^T & 0 \end{bmatrix},\ x_\mu := \begin{bmatrix} h_\mu \\ \lambda_\mu \end{bmatrix},\ b_\mu := \begin{bmatrix} g_\mu \\ 0 \end{bmatrix}.$$

To reduce the high computational complexity of solving $x_\mu$, we aim to find a lower dimensional approximation for $h_\mu$, and a scalar $\lambda \in \mathbb{R}$ to approximate $\lambda_\mu$. Formally, we approximate $h_\mu$ using a linear parametric form: $h(r) = \sum_{k=1}^{m} r_k \phi_k$, where $r_k$ is a tunable parameter, $\phi_k$ is called a *basis function*, and $m$ is a user-defined number to trade-off between optimality and computational complexity. For a given policy $\mu$, we define a *basis matrix* $\Phi_\mu := [\phi_1 | \cdots | \phi_m]$. Then $h(r)$ can be expressed compactly as a linear combination $h(r) = \Phi_\mu r$, where $r = [r_1, \ldots, r_m]^T$. We assume that the basis functions $\{\phi_1, \ldots, \phi_m\}$ together with the unit vector $\mathsf{1}$ are linearly independent.

*2) Automatic generation of basis functions:* We employ the Krylov subspace method to automatically generate basis functions [10]. For any finite-state Markov chain, the limiting matrix $P_\mu^* := \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} P_\mu^k$ is well-defined. Moreover, it is easy to check that $P_\mu^* = P_\mu^* P_\mu = P_\mu P_\mu^* = P_\mu^* P_\mu^*$, and the matrix $(I - P_\mu + P_\mu^*)$ is non-singular [10]. Given $\mathcal{M}$, the limiting matrix $P_\mu^*$ has identical rows, *i.e.,* $P_\mu^* = \mathsf{1}\rho_\mu^T$ (this is due to the fact that $\mathcal{M}$ is communicating [10]). Consequently, we can represent Eqn. (IV.3) equivalently as the following: $\lambda_\mu \mathsf{1} + h_\mu = g_\mu + P_\mu h_\mu + \lambda_\mu \overrightarrow{P}_\mu \mathsf{1} - P_\mu^* h_\mu + \mathsf{1}\rho_\mu^T h_\mu$. Since $(I - P_\mu + P_\mu^*)$ is non-singular, then

$$h_\mu = \sum_{k=0}^{\infty} \left( P_\mu^k - \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} P_\mu^k \right) (g_\mu + \widetilde{\lambda}_\mu \mathsf{1} + \lambda_\mu p_\mu)$$

where $\widetilde{\lambda}_\mu$ is a scalar defined as $\widetilde{\lambda}_\mu := \rho_\mu^T h_\mu - \lambda_\mu$ and $p_\mu$ is a vector defined as $p_\mu := \overrightarrow{P}_\mu \mathsf{1}$. The expansion form of $h_\mu$ implies that $h_\mu$ can be approximated by the basis functions of the form: $P_\mu^k g_\mu, P_\mu^k \mathsf{1} \equiv \mathsf{1}$, and $P_\mu^k p_\mu$. Thus, a candidate set of basis functions is taken as $\mathcal{B}_\mu = \{\mathsf{1}, g_\mu, P_\mu g_\mu, \ldots, P_\mu^{\bar{m}} g_\mu, p_\mu, P_\mu p_\mu, \ldots, P_\mu^{\bar{m}} p_\mu\}$. We usually choose an integer $\bar{m}$ large enough, such that we can obtain $m$ independent basis functions by eliminating the dependencies of vectors in the set $\mathcal{B}_\mu$.

*3) Selection of optimal parameters:* Given a set of basis functions chosen from $\mathcal{B}_\mu$, we approximate $h_\mu$ with $h(r) = \Phi_\mu r$, and $\lambda_\mu$ with a scalar $\lambda \in \mathbb{R}$, respectively. We denote $\Psi_\mu := \begin{bmatrix} \Phi_\mu & \mathsf{0} \\ \mathsf{0}^T & 1 \end{bmatrix}$, $x(r, \lambda) := \Psi_\mu \begin{bmatrix} r \\ \lambda \end{bmatrix}$. We use the least-square method to approximate the solution of the Bellman's equation $A_\mu x_\mu = b_\mu$ [10], which leads to solving the following least-square approximation problem:

$$\min_{r\in\mathbb{R}^m, \lambda\in\mathbb{R}} \|A_\mu x(r, \lambda) - b_\mu\|^2 \quad \text{(IV.4)}$$

where $\|\cdot\|$ denotes Euclidean norm. Under independence assumption of basis functions, the solution to (IV.4) is unique

$$\begin{bmatrix} r^* \\ \lambda^* \end{bmatrix} = (\Psi_\mu^T A_\mu^T A_\mu \Psi_\mu)^{-1} \Psi_\mu^T A_\mu^T b_\mu. \quad \text{(IV.5)}$$

Therefore, the optimal approximation of $h_\mu$ is $h^* = \Phi_\mu r^*$. The optimal approximation to the average cost $\lambda_\mu$ is $\lambda^*$.

*4) Approximate dynamic programming algorithm:* We propose an approximate policy iteration algorithm (APIA) in Alg. 1 to compute the sub-optimal solution to Prob. IV.1.

**Algorithm 1** : Approximate Policy Iteration Algorithm

**Input:** $\mathcal{M} = (Q, \iota, U, P, \Pi, L, g)$, $\pi$, and $m$
**Output:** Sub-optimal policy $\mu^*$
1: Initialize a proper policy $\mu^0$ and set $k = 0$
2: **repeat**
3: Construct basis matrix $\Phi_{\mu^k}$ through the set $\mathcal{B}_{\mu^k}$
4: Compute $r_k^*$ and $\lambda_k^*$ using (IV.5) and obtain $h_k^* = (\Phi_{\mu^k}) r_k^*$
5: Find $\mu^{k+1}$ through $\mu^{k+1} \in \arg\min_\mu [g_\mu + P_\mu h_k^* + \lambda_k^* \overrightarrow{P}_\mu \mathbf{1}]$
6: Set $k \leftarrow k+1$
7: **until** $\mu^{k+1} = \mu^k$
8: **return** $\mu^k$ as the sub-optimal policy $\mu^*$

**Remark 4 (Complexity)** *During each iteration of Alg. 1, the computational complexity is of order $O(m|Q|^2)$, where $m$ is the number of selected basis function. In most cases, we have $m \ll |Q|$. Given the special structure of $\mathcal{M}$ (since $\mathcal{M}$ is an AMEC of $\mathcal{M}_\mathcal{P}$, $P_\mu$ is often a sparse matrix), the computational complexity is of order $O(m\bar{n}+m^2|Q|)$, where $\bar{n}$ denotes the number of non-zero entries of $P_\mu$.*

## V. Case Study and Simulation Results

The algorithmic framework developed in this paper was implemented in MATLAB, and used in conjunction with a simulator to demonstrate the motion of a robotic team in a partitioned environment. A computer at 1.30 GHz and with 2GB RAM was used to generate the simulation results. A movie of the simulation is available at `http://hyness.bu.edu/CDC2012/`.

We consider two robots and assume that Robot 1 moves twice as fast as Robot 2. We consider a persistent surveillance task, where they are required to monitor rooms V1 and V3 and then return to Base to report the collected information. In other words, the robots should occupy rooms V1 and V3 at the same time and then return to Base together, infinitely often. Robots 1 and 2 should always avoid unsafe regions. To specify this task, we define a set of atomic propositions in the form $\Pi = \{\mathsf{Base1}, \mathsf{Base2}, \mathsf{M\_V1}, \mathsf{M\_V3}, \mathsf{Unsafe1}, \mathsf{Unsafe2}\}$. and assign the atomic propositions to the robots as follows: $\Pi_1 = \{\mathsf{Base1}, \mathsf{M\_V1}, \mathsf{M\_V3}, \mathsf{Unsafe1}\}, \Pi_2 = \{\mathsf{Base2}, \mathsf{M\_V1}, \mathsf{M\_V3}, \mathsf{Unsafe2}\}$. The labeling functions for Robot 1 and 2 are defined as follows: $L_1(\mathsf{V1}) = L_2(\mathsf{V1}) = \{\mathsf{M\_V1}\}, L_1(\mathsf{V3}) = L_2(\mathsf{V3}) = \{\mathsf{M\_V3}\}, L_1(\mathsf{V6}) = \{\mathsf{Unsafe1}\}, L_2(\mathsf{V5}) = \{\mathsf{Unsafe2}\}, L_1(\mathsf{V7}) = \{\mathsf{Base1}\}, L_2(\mathsf{V7}) = \{\mathsf{Base2}\}$.

Our goal is to minimize the expected time in between the robots' simultaneous visits to V1 and V3. Therefore, the optimizing task $\psi$ is $\mathsf{M\_V1} \wedge \mathsf{M\_V3}$. The specification

$$\begin{aligned}\phi \;=\; & \mathsf{G}(\neg\mathsf{Unsafe1}) \wedge \mathsf{G}(\neg\mathsf{Unsafe2}) \wedge \mathsf{GF}(\mathsf{Base1} \wedge \mathsf{Base2}) \wedge \\ & \mathsf{G}((\mathsf{M\_V1} \wedge \mathsf{M\_V3}) \rightarrow \mathsf{X}\,((\neg\mathsf{M\_V1} \wedge \neg\mathsf{M\_V3}) \\ & \mathsf{U}\,(\mathsf{Base1} \wedge \mathsf{Base2}))) \wedge \mathsf{GF}(\mathsf{M\_V1} \wedge \mathsf{M\_V3}).\end{aligned}$$

$\mathsf{GF}(\mathsf{Base1} \wedge \mathsf{Base2})$ ensures that both robots visit the base simultaneously. $\mathsf{G}(\neg\mathsf{Unsafe1})$ and $\mathsf{G}(\neg\mathsf{Unsafe2})$ specifies that Robots 1 and 2 should always avoid regions Unsafe1 and Unsafe2, respectively. $\mathsf{G}((\mathsf{M\_V1} \wedge \mathsf{M\_V3}) \rightarrow \mathsf{X}\,((\neg\mathsf{M\_V1} \wedge \neg\mathsf{M\_V3})\,\mathsf{U}\,(\mathsf{Base1} \wedge \mathsf{Base2})))$ ensures that after monitoring both V1 and V3, none of them are visited again before the robots visit the base together.

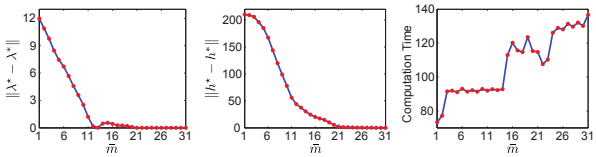

Fig. 2. left: average cost error, middle: relative cost error, and right: computation time (seconds) for different values of $\bar{m}$.

To compute the optimal policies $\{\pi_k, k \in \mathcal{R}\}$ satisfying $\phi$ with probability 1, we first constructed the MDP $\mathcal{M}_\mathcal{G}$ (Eqn. (III.1)), and then computed the product automaton $\mathcal{M}_\mathcal{P} = \mathcal{M}_\mathcal{G} \times \mathcal{A}_\phi$ ($\mathcal{A}_\phi$ is computed using the software tool [12]). The constructed $\mathcal{M}_\mathcal{G}$ and $\mathcal{M}_\mathcal{P}$ have 2575 and 33475 states, respectively. We found one AMEC with 4102 states in the product automaton $\mathcal{M}_\mathcal{P}$. Finally, we applied Alg. 1 to find the desired robot policies given the AMEC.

Using the policy iteration algorithm (PIA) proposed in [9] (see Sec. IV-A), we obtained the optimal control policy $\mu^\star$ and the optimal ACPC cost $\lambda^\star = 12.9794$. For the approximate policy iteration algorithm (APIA) described in Alg. 1, we select different values of basis functions. The error plots and computation times of APIA are shown in Fig. 2. We observe from Fig. 2 that both the average and relative cost errors nearly approach zero when $\bar{m}$ is greater than 22. The computation time of employing APIA with $\bar{m} = 22$ is around 107.62 seconds, while the computation time of employing PIA is around 809.63 seconds.

## References


[1] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Proc CDC and CCC*, Shanghai, China, December 2009, pp. 5997–6004.
[2] H. Kress-Gazit, D. C. Conner, H. Choset, A. A. Rizzi, and G. J. Pappas, "Courteous cars," *IEEE RAM*, vol. 15, no. 1, pp. 30–38, 2008.
[3] S. Karaman and E. Frazzoli, "Complex mission optimization for multiple-UAVs using linear temporal logic," in *Proc ACC*, Seattle, US, 2008, pp. 2003–2009.
[4] M. Antoniotti and B. Mishra, "Discrete event models + temporal logic = supervisory controller: Automatic synthesis of locomotion controllers," in *Proc ICRA*, 1995.
[5] J. Tumova, B. Yordanov, C. Belta, I. Cerna, and J. Barnat, "A symbolic approach to controlling piecewise affine systems," in *Proc CDC*, Atlanta, GA, 2010.
[6] M. Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Proc HSCC*, ser. LNCS. Spinger Verlag, 2008, pp. 287–300.
[7] X. Ding, S. L. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *Proc IFAC World C*, Milan, Italy, 2011.
[8] A. Ulusoy, S. L. Smith, X. C. Ding, and C. Belta, "Robust multi-robot optimal path planning with temporal logic constraints," in *Proc ICRA*, St. Paul, USA, 2012 (to appear).
[9] X. Ding, S. L. Smith, C. Belta, and D. Rus, "MDP optimal control under temporal logic constraints," in *Proc CDC*, Orlando, FL, USA, 2011.
[10] D. P. Bertsekas, *Dynamic Programming and Optimal Control, 3rd Ed.* Athena Scientific, 2007.
[11] C. Baier and J. P. Katoen, *Principles of Model Checking*. MIT Press, 2008.
[12] J. Klein. (2007) ltl2dstar-ltl to deterministic streett and rabin automata. [Online]. Available: http://www.ltl2dstar.de/