Optimal Control of Markov Decision Processes with Linear Temporal Logic Constraints

Xuchu (Dennis) Ding   Stephen L. Smith   Calin Belta   Daniela Rus

***Abstract*—In this paper, we develop a method to automatically generate a control policy for a dynamical system modeled as a Markov Decision Process (MDP). The control specification is given as a Linear Temporal Logic (LTL) formula over a set of propositions defined on the states of the MDP. Motivated by robotic applications requiring persistent tasks, such as environmental monitoring and data gathering, we synthesize a control policy that minimizes the expected cost between satisfying instances of a particular proposition over all policies that maximize the probability of satisfying the given LTL specification. Our approach is based on the definition of a novel optimization problem that extends the existing average cost per stage problem. We propose a sufficient condition for a policy to be optimal, and develop a dynamic programming algorithm that synthesizes a policy that is optimal for a set of LTL specifications.**

I. Introduction

In this paper, we consider the problem of controlling a (finite state) Markov Decision Process (MDP). Such models are widely used in many areas encompassing engineering, biology, and economics. They have been successfully used to model and control autonomous robots subject to uncertainty in their sensing and actuation, such as ground robots [3], unmanned aircraft [4] and surgical steering needles [5]. In these studies, the underlying motion of the system cannot be predicted with certainty, but it can be obtained from the sensing and the actuation model through a simulator or empirical trials. Standard textbooks in this area include [6], [7].

Several authors [8]–[11] have proposed using temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL), as rich and expressive specification languages for control systems. As their syntax and semantics are well-defined and deeply explored [12], [13], these logics can be easily used to specify complex system behaviors. In particular, LTL is well-suited for specifying persistent tasks, *e.g.,* "Visit regions A, then B, and then C, infinitely often. Never enter B unless coming directly from D." Off-the-shelf model checking algorithms [12] and temporal logic game strategies [14] can be used to verify the correctness of system trajectories.

Existing works focusing on LTL assume that a finite model of the system is available and the current state can be precisely determined. If the control model is deterministic (*i.e.,* at each state, an available control enables exactly one transition), control strategies from specifications given as LTL formulas can be found through simple adaptations of off-the-shelf model checking algorithms [15]. If the control is non-deterministic (an available control at a state enables one of several transitions, and their probabilities are not known), the control problem from an LTL specification can be mapped to the solution of a Rabin game [16] in general, and a Büchi [17] or GR(1) game [8] if the specification is restricted to certain fragments of LTL. If the probabilities of the enabled transitions at each state are known, then the control problem reduces to finding a control policy for an MDP such that a probabilistic temporal logic formula is satisfied. Solutions can be found by adapting methods from probabilistic model-checking [13], [18], [19]. In our recent work [1] we addressed the problem of synthesizing control policies for MDPs subject to LTL satisfaction constraints. A similar approach was taken in a number of other studies [20], [21].

In all of the above results, a control policy is designed to maximize the probability of satisfying a given LTL formula. Such policies that maximize the satisfaction probability are generally not unique, but rather belong to a family of policies with the same "transient response" that maximizes the probability of reaching a specific set of accepting states. So far, no attempt has been made to optimize the long-term behavior of the system after the accepting states have been reached, while enforcing LTL satisfaction guarantees.

In this paper, we compute an optimal control policy over the infinite time horizon for a dynamical system modeled as an MDP under temporal logic constraints. This work aims to bridge the gap between preliminary work on MDP control policies with LTL probabilistic guarantees [1], and optimal control synthesis under LTL constraints for deterministic systems [22]. In particular, we consider LTL formulas defined over a set of propositions assigned to the states of the MDP. We synthesize a control policy that minimizes the expected cost between satisfying instances of an "optimizing proposition" over all policies that maximize the probability of satisfying the given LTL specification. Such an objective is often critical in many applications, such as surveillance, persistent monitoring, and pickup-delivery tasks, where an agent must repeatedly visit some areas in an environment and the time in between revisits should be minimized. It can also be relevant to certain persistent tasks (e.g. cell growth and division) in biological systems.

Preliminary versions of this work appeared as [1] and [2].

This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, NSF CNS-0834260, and the Singapore-MIT Alliance for Research and Technology (SMART) under the Future of Urban Mobility project.

X. Ding is with Embedded Systems and Networks, United Technologies Research Center, East Hartford, CT 06108, USA (email:dingx@utrc.utc.com) (note that this research was conducted while at Boston University). S. L. Smith is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo ON, N2L 3G1 Canada (email: stephen.smith@uwaterloo.ca). C. Belta is with the Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA (email: belta@bu.edu). D. Rus is with the Computer Science and Artificial Intelligence Laboratory, Electrical Engineering and Computer Science Department, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (email: rus@csail.mit.edu).

The main contribution of this paper is two-fold. First, we formulate the above MDP optimization problem in terms of minimizing the average cost per cycle, where cycles are defined by successive satisfactions of the optimizing proposition. We present a novel connection between this problem and the well-known average cost per stage problem, and the condition of optimality for average cost per cycle problems is proposed. Second, we incorporate the LTL constraints and obtain a sufficient condition for optimal policies. We present a dynamic programming algorithm that under some conditions synthesizes an optimal control policy, and a sub-optimal policy otherwise. We show that the given algorithm produces the optimal policy for a fragment of LTL formulas. We demonstrate that the proposed framework is useful for a variety of persistent tasks and demonstrate our approaches with case studies in motion planning. This work combines and extends preliminary results [1], [2] by addressing a more general formulation of the problem, examining the optimality of the proposed approach, and including all proofs which were omitted in [1], [2].

The organization of this paper is as follows. In Sec. II we provide some preliminaries on LTL and MDPs. We formulate the problem in Sec. III and we formalize the connection between the average cost per cycle and the average cost per stage problems in Sec. IV. In Sec. V, we provide a method for incorporating LTL constraints. We present a case study illustrating our framework in Sec. VI and we conclude in Sec. VII

II. Preliminaries

In this section we provide background material on linear temporal logic and Markov decision processes.

Notation: For a set S, we use 2^S and $|S|$ to denote its power set and cardinality, respectively. A sequence over S will be referred as a word over S. For a matrix A, we denote the element at the ith row and jth column by $A(i,j)$. For a vector g, we denote its ith element by $g(i)$. We let $\mathbf{1} \in \mathbb{R}^n$ denote the vector of all 1s. For two vectors a and b, $a \leq b$ means $a(i) \leq b(i)$ for all i.

A. Linear Temporal Logic

We employ Linear Temporal Logic (LTL) to describe MDP control specifications. A detailed description of the syntax and semantics of LTL is beyond the scope of this paper and can be found in [12], [13]. Roughly, an LTL formula is built up from a set of atomic propositions Π, standard Boolean operators $\neg$ (negation), $\vee$ (disjunction), $\wedge$ (conjunction), and temporal operators X (next), U (until), F (eventually), G (always). The semantics of LTL formulas are given over infinite words in 2^Π. A word satisfies an LTL formula ϕ if ϕ is true at the first position of the word; $\mathsf{G}\,\phi$ means that ϕ is true at all positions of the word; $\mathsf{F}\,\phi$ means that ϕ eventually becomes true in the word; $\phi_1 \,\mathsf{U}\, \phi_2$ means that ϕ_1 has to hold at each position in the word, at least until ϕ_2 is true. More expressivity can be achieved by combining the above temporal and Boolean operators. As an example, the formula $\mathsf{G}\,\mathsf{F}\,\phi$ means that at all positions of the word, ϕ must eventually become true at some later position in the word. This implies that ϕ must be satisfied infinitely often in the word. We use $o \vDash \phi$ to denote that word o satisfies the LTL formula ϕ. An LTL formula can be represented by a deterministic *Rabin automaton*, which is defined as follows.

Definition II.1 (Deterministic Rabin Automaton)**.** *A deterministic Rabin automaton (DRA) is a tuple $\mathcal{R} = (Q, \Sigma, \delta, q_0, F)$, where (i) Q is a finite set of states; (ii) Σ is a set of inputs (alphabet); (iii) $\delta : Q \times \Sigma \to Q$ is the transition function; (iv) $q_0 \in Q$ is the initial state; and (v) $F = \{(L(1), K(1)), \ldots, (L(M), K(M))\}$, with M being a positive integer, is a set of pairs of sets of states such that $L(i), K(i) \subseteq Q$ for all $i = 1, \ldots, M$.*

A run of a Rabin automaton $\mathcal{R}$, denoted by $r_\mathcal{R} = q_0 q_1 \ldots$, is an infinite sequence of states in $\mathcal{R}$ such that for each $i \geq 0$, $q_{i+1} \in \delta(q_i, \alpha)$ for some $\alpha \in \Sigma$. A run $r_\mathcal{R}$ is *accepting* if there exists a pair $(L, K) \in F$ such that 1) there exists $n \geq 0$, such that for all $m \geq n$, we have $q_m \notin L$, and 2) there exist infinitely many indices k where $q_k \in K$. This acceptance conditions means that $r_\mathcal{R}$ is accepting if for a pair $(L, K) \in F$, $r_\mathcal{R}$ intersects with L finitely many times and K infinitely many times. Note that for a given pair (L, K), L can be an empty set, but K is not empty. This acceptance condition will be used later in the paper to enforce the satisfaction of LTL formulae by words produced by MDP executions.

For any LTL formula ϕ over Π, one can construct a DRA with input alphabet $\Sigma = 2^\Pi$ accepting all and only words over Π that satisfy ϕ (see [23]). We refer readers to [24] and references therein for algorithms and to freely available implementations, such as [25], to translate a LTL formula over Π to a corresponding DRA.

B. Markov Decision Process and Probability Measure

Definition II.2 (Markov Decision Process)**.** *A (weighted, labeled) Markov decision process (MDP) is a tuple $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$, where $S = \{1, 2, \ldots, n\}$ is a finite set of states; U is a finite set of controls (actions) (with a slight abuse of notation $U(i) \subseteq U$ denotes the controls available at state $i \in S$); $P : S \times U \times S \to [0, 1]$ is the transition probability function such that for all $i \in S$, $\sum_{j \in S} P(i, u, j) = 1$ if $u \in U(i)$, and $P(i, u, j) = 0$ if $u \notin U(i)$; $s_0 \in S$ is the initial state; Π is a set of atomic propositions; $\mathcal{L} : S \to 2^\Pi$ is a labeling function and $g : S \times U \to \mathbb{R}^+$ is such that $g(i, u)$ is the (non-negative) cost when control $u \in U(i)$ is taken at state i.*

We define a control function $\mu : S \to U$ such that $\mu(i) \in U(i)$ for all $i \in S$. A infinite sequence of control functions $M = \{\mu_0, \mu_1, \ldots\}$ is called a *policy*. One can use a policy to resolve all non-deterministic choices in an MDP by applying the action $\mu_k(s_k)$ at state s_k [1]. Given an initial state s_0, an infinite sequence $r^M_\mathcal{M} = s_0 s_1 \ldots$ on $\mathcal{M}$ generated under a policy M is called a *path* on $\mathcal{M}$ if $P(s_k, \mu_k(s_k), s_{k+1}) > 0$ for all k. The index k of a path is called *stage* k. If $\mu_k = \mu$ for all k, then we call it a *stationary* policy and we denote it

[1]Throughout the paper, we use $i \in \{1, 2, \ldots, n\}$ to refer to a state of the MDP and $s_i \in \{1, 2, \ldots, n\}$, $i \in \{0, 1, 2, \ldots\}$ to refer to a position in a word over $\{1, 2, \ldots, n\}$.

simply as μ. A stationary policy μ induces a Markov chain where its set of states is S and the transition probability from state i to j is $P(i, \mu(i), j)$.

We define $\mathrm{Paths}_{\mathcal{M}}^{M}$ and $\mathrm{FPaths}_{\mathcal{M}}^{M}$ as the set of all infinite and finite paths of $\mathcal{M}$ under a policy M, respectively. We can then define a probability measure over the set $\mathrm{Paths}_{\mathcal{M}}^{M}$. For a path $r_{\mathcal{M}}^{M} = s_0 s_1 \ldots s_m s_{m+1} \ldots \in \mathrm{Paths}_{\mathcal{M}}^{M}$, the *prefix* of length m of $r_{\mathcal{M}}^{M}$ is the finite subsequence $s_0 s_1 \ldots s_m$. Let $\mathrm{Paths}_{\mathcal{M}}^{M}(s_0 s_1 \ldots s_m)$ denote the set of all paths in $\mathrm{Paths}_{\mathcal{M}}^{M}$ with the prefix $s_0 s_1 \ldots s_m$. (Note that $s_0 s_1 \ldots s_m$ is a finite path in $\mathrm{FPaths}_{\mathcal{M}}^{M}$.) Then, the probability measure $\mathrm{Pr}_{\mathcal{M}}^{M}$ on the smallest σ-algebra over $\mathrm{Paths}_{\mathcal{M}}^{M}$ containing $\mathrm{Paths}_{\mathcal{M}}^{M}(s_0 s_1 \ldots s_m)$ for all $s_0 s_1 \ldots s_m \in \mathrm{FPaths}_{\mathcal{M}}^{M}$ is the unique measure satisfying

$$\mathrm{Pr}_{\mathcal{M}}^{M}\{\mathrm{Paths}_{\mathcal{M}}^{M}(s_0 s_1 \ldots s_m)\} = \prod_{0 \leq k < m} P(s_k, \mu_k(s_k), s_{k+1}).$$

Finally, we can define the probability that an MDP $\mathcal{M}$ under a policy M satisfies an LTL formula ϕ. A path $r_{\mathcal{M}}^{M} = s_0 s_1 \ldots$ deterministically generates a word $o = o_0 o_1 \ldots$, where $o_i = \mathcal{L}(s_i)$ for all i. With a slight abuse of notation, we denote $\mathcal{L}(r_{\mathcal{M}}^{M})$ as the word generated by $r_{\mathcal{M}}^{M}$. Given an LTL formula ϕ, one can show that the set $\{r_{\mathcal{M}}^{M} \in \mathrm{Paths}_{\mathcal{M}}^{M} : \mathcal{L}(r_{\mathcal{M}}^{M}) \vDash \phi\}$ is measurable [13]. We define:

$$\mathrm{Pr}_{\mathcal{M}}^{M}(\phi) := \mathrm{Pr}_{\mathcal{M}}^{M}\{r_{\mathcal{M}}^{M} \in \mathrm{Paths}_{\mathcal{M}}^{M} : \mathcal{L}(r_{\mathcal{M}}^{M}) \vDash \phi\} \tag{1}$$

as the probability of satisfying ϕ for $\mathcal{M}$ under M. As a special case, given a set of states $A \subseteq S$, we let $\mathrm{Pr}_{\mathcal{M}}^{M}(\mathsf{F}\, A)$ denote the probability of eventually reaching the set of states A under policy M. For more details about probability measures on MDPs under a policy and measurability of LTL formulas, we refer readers to a text in probabilistic model checking, such as [13].

III. Problem Formulation

Consider a MDP $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$ and an LTL formula ϕ over Π. We assume that formula ϕ is of the form:

$$\phi = \mathsf{G}\,\mathsf{F}\,\pi \wedge \psi, \tag{2}$$

where the atomic proposition $\pi \in \Pi$ is called the *optimizing proposition* and ψ is an arbitrary LTL formula. In other words, ϕ requires that ψ be satisfied and π be satisfied infinitely often.

Let $\mathbb{M}$ be the set of all policies $M = \{\mu_0, \mu_1, \ldots\}$. We let $\mathbb{M}_\phi$ denote the set of all policies that maximize the probability of satisfying ϕ. Thus, if $M^\star \in \mathbb{M}_\phi$, we have that

$$\mathrm{Pr}_{\mathcal{M}}^{M^\star}(\phi) = \max_{M \in \mathbb{M}} \mathrm{Pr}_{\mathcal{M}}^{M}(\phi). \tag{3}$$

It has been shown that the maximization in (3) is well-defined for arbitrary LTL formula ϕ [18], [19]. Note that there typically exist many (possibly an infinite number of) policies in $\mathbb{M}_\phi$. We assume that $\mathrm{Pr}_{\mathcal{M}}^{M^\star}(\phi) \neq 0$ to avoid triviality.

We would like to obtain the optimal policy M such that the probability of satisfying ϕ is maximized (*i.e.,* $M \in \mathbb{M}_\phi$), and the expected cost in between visiting states satisfying π is minimized. To formalize the latter objective, we first denote $S_\pi = \{i \in S, \pi \in \mathcal{L}(i)\}$ (*i.e.,* the states where atomic proposition π is true). We say that each visit to set S_π *completes a cycle*. Thus, starting at the initial state, the finite path reaching S_π for the first time is the first cycle; the finite path that starts after the completion of the first cycle and ends with revisiting S_π for the second time is the second cycle, and so on. Given a path $r_{\mathcal{M}}^{M} = s_0 s_1 \ldots$, we use $C(r_{\mathcal{M}}^{M}, N)$ to denote the cycle index up to stage N, which is defined as the total number of cycles completed at stage N plus 1 (*i.e.,* the cycle index starts with 1 at the initial state).

The main problem that we consider in this paper is to find a policy that minimizes the average cost per cycle (ACPC) starting from the initial state s_0. Formally, this problem can be stated as follows:

Problem III.1. *Given a MDP $\mathcal{M}$ and an LTL formula ϕ of the form* (2)*, find a policy $M = \{\mu_0, \mu_1, \ldots\}$, $M \in \mathbb{M}_\phi$ that minimizes*

$$J(s_0) = \limsup_{N \to \infty} E\left\{ \frac{\sum_{k=0}^{N} g(s_k, \mu_k(s_k))}{C(r_{\mathcal{M}}^{M}, N)} \,\middle|\, \mathcal{L}(r_{\mathcal{M}}^{M}) \vDash \phi \right\}, \tag{4}$$

where $E\{\cdot\}$ denotes the expected value (i.e., we find the policy with the minimal average cost per cycle over all policies that maximize the probability of satisfying ϕ).

Prob. III.1 is related to the standard average cost per stage (ACPS) problem, which consist of minimizing

$$J^s(s_0) = \limsup_{N \to \infty} E\left\{ \frac{\sum_{k=0}^{N} g(s_k, \mu_k(s_k))}{N} \right\}, \tag{5}$$

over $\mathbb{M}$, with the noted difference that the right-hand-side (RHS) of (5) is divided by the index of stages instead of cycles. The ACPS problem has been widely studied in the dynamic programming community, without the constraint of satisfying temporal logic formulas [6].

The ACPC cost function we consider in this paper is relevant for probabilistic abstractions and practical applications, where the cost of controls can represent the time, energy, or fuel required to apply controls at each state. In particular, it is a suitable performance measure for persistent tasks, which can be specified by LTL formulas. In these applications, the ACPS cost function does not usually translate to a meaningful performance criterion. For example, in a data gathering mission [22], an agent is required to repeatedly gather and upload data. We can assign π to the data upload locations and a solution to Prob. III.1 minimizes the expected cost in between data uploads. In this case, the average cost per stage problem formulation is not meaningful as it does not in general minimize the cost towards satisfying the LTL specification. Nevertheless, we will make a connection between the ACPS and the ACPC problems in Sec. IV as a starting point in finding a solution.

Remark III.2 (Optimization Criterion)**.** *The conditional expectation and the form* (2) *of LTL formula we consider are chosen such that Prob. III.1 is sufficiently well-posed. Part of the formula* ($\mathsf{G}\,\mathsf{F}\,\pi$) *ensures that π is satisfied infinitely often, which is a necessary condition for Prob. III.1 to have a finite solution.*

IV. Solving the average cost per cycle problem

In this section we develop a solution to the average cost per cycle problem in the absence of a LTL specification (*i.e.,* Prob. III.1 without the requirement that $M \in M_\phi$). We extend the results from this section to LTL specifications in Sec. V.

A. Optimality conditions for ACPS problems

In this section, we recall some known results on the ACPS problem, namely finding a policy over $\mathbb{M}$ that minimizes J^s in (5). The reader interested in more details is referred to [6], [7] and references therein.

Definition IV.1 (Classification of MDPs)**.** *An MDP $\mathcal{M}$ is said to satisfy the Weak Accessibility (WA) condition if there exists a set of states $S_r \subseteq S$, such that (i) there exists a stationary policy where j is reachable from i for any $i, j \in S_r$, and (ii) states in $S \setminus S_r$ are transient under all stationary policies. An MDP $\mathcal{M}$ is called single-chain (or weakly-communicating) if it satisfies the WA condition. If $\mathcal{M}$ satisfies the WA condition with $S_r = S$, then $\mathcal{M}$ is called communicating.*

A stationary policy induces a Markov chain with a set of recurrent classes. A state that does not belong to any recurrent class is called *transient.* A stationary policy μ is called *unichain* if the Markov chain induced by μ contains one recurrent class (and a possible set of transient states). If every stationary policy of $\mathcal{M}$ is unichain, $\mathcal{M}$ is called unichain. A unichain MDP is single-chain, but not vice versa. The unichain condition requires a set of "recurrent" states to be mutually reachable under *all* stationary policies, while the single chain condition requires them to be mutually reachable only under *some* stationary policy.

For each stationary policy μ, we use P_μ to denote the transition probability matrix: $P_\mu(i,j) = P(i,\mu(i),j)$. Define the vector g_μ as $g_\mu(i) = g(i,\mu(i))$ for each $i \in S$. For each stationary policy μ, we can obtain a so-called gain-bias pair (J^s_μ, h^s_μ),[2] where

$$J^s_\mu = P^*_\mu g_\mu, \qquad h^s_\mu = H^s_\mu g_\mu \tag{6}$$

with

$$P^*_\mu = \lim_{N\to\infty} \frac{1}{N}\sum_{k=0}^{N-1} P^k_\mu, \qquad H^s_\mu = (I - P_\mu + P^*_\mu)^{-1} - P^*_\mu. \tag{7}$$

The vector J^s_μ is such that $J^s_\mu(i)$ is the ACPS starting at initial state i under policy μ. Note that the limit in (7) exists for any stochastic matrix P_μ, and P^*_μ is stochastic. Therefore, the $\limsup$ in (5) can be replaced by the limit for a stationary policy. A key result for the ACPS problem is that, (J^s_μ, h^s_μ) satisfies

$$J^s_\mu = P_\mu J^s_\mu, \qquad J^s_\mu + h^s_\mu = g_\mu + P_\mu h^s_\mu. \tag{8}$$

Moreover,

$$h^s_\mu + v^s_\mu = P_\mu v^s_\mu, \tag{9}$$

for some vector v^s_μ. The triple $(J^s_\mu, h^s_\mu, v^s_\mu)$ can be found as the solution of $3n$ linear equations with $3n$ unknowns.

[2]Note, here we use the superscript s to remind the reader that this gain-bias pair is for the average cost per stage problem. We will then extend this notion to average cost per cycle problems.

It has been shown that there exists a stationary optimal policy $\mu^\star$ minimizing (5) over all policies, such that its gain-bias pair, denoted by (J^s, h^s), satisfies the Bellman's equations for ACPS problems:

$$J^s(i) = \min_{u\in U(i)} \sum_{j=1}^{n} P(i,u,j) J^s(j) \tag{10}$$

and

$$J^s(i) + h^s(i) = \min_{u\in \bar{U}(i)} \left[g(i,u) + \sum_{j=1}^{n} P(i,u,j) h^s(j) \right], \tag{11}$$

for all $i = 1, \ldots, n$, where $\bar{U}_i$ is the set of controls attaining the minimum in (10). Furthermore, if $\mathcal{M}$ is single-chain, the optimal average cost does not depend on the initial state, *i.e.,* $J^s(i) = \lambda$ for all $i \in S$. In this case, (10) is trivially satisfied and $\bar{U}_i$ in (11) can be replaced by $U(i)$. Hence, $\mu^\star$ with gain-bias pair $(\lambda\mathbf{1}, h)$ is optimal over all polices if for all stationary policies μ, we have

$$\lambda\mathbf{1} + h \leq g_\mu + P_\mu h. \tag{12}$$

Therefore, the optimal solution for ACPS problems can be found by policy iteration algorithms using (10) and (11).

B. Optimality conditions for ACPC problems

Now we derive equations similar to (10) and (11) for ACPC problems, without considering the satisfaction constraint, *i.e.,* we do not limit the set of polices to $\mathbb{M}_\phi$ at the moment. We consider the following problem.

Problem IV.2. *Given a communicating MDP $\mathcal{M}$ and a set S_π, find a policy $\mu \in \mathbb{M}$ that minimizes* (4).

Note that, for reasons that will become clear in Sec. V, we assume in Prob. IV.2 that the MDP is communicating.

We limit our attention to stationary policies. We will show that, as in many dynamic programming problems, there exist optimal policies that are stationary, thus it is sufficient to consider only stationary policies. For such policies, we use the following notion of *proper policies*, which is the same as the one used in stochastic shortest path problems (see [6]).

Definition IV.3 (Proper Polices)**.** *We say a stationary policy μ is proper if, under μ, all initial states have positive probability to reach the set S_π in a finite number of stages.*

We define J_μ such that $J_\mu(i)$ is the ACPC in (4) starting from state i under policy μ. If policy μ is improper, then there exist some states $i \in S$ that can never reach S_π. In this case, since $g(i,u)$ is positive for all i, u, we can immediately see that $J_\mu(i) = \infty$. Thus, we will first consider only proper policies.

Without loss of generality, we assume that $S_\pi = \{1, \ldots, m\}$ (*i.e.,* states $m+1, \ldots, n$ are not in S_π). Given a proper policy μ, we obtain its transition matrix P_μ as $P_\mu(i,j) = P(i,\mu(i),j)$. Our goal is to express J_μ in terms of P_μ, similar to (6) in the ACPS case. To achieve this, we first compute the probability that $j \in S_\pi$ is the first state visited in S_π after leaving from a state $i \in S$ by applying policy μ. We denote this probability by $\widetilde{P}(i,\mu,j)$. We can obtain this probability for all $i \in S$ and $j \in S_\pi$ by the following proposition:

Proposition IV.4. *For a proper policy μ, the probability $\widetilde{P}(i,\mu,j)$ satisfies*

$$\widetilde{P}(i,\mu,j) = \sum_{k=m+1}^{n} P(i,\mu(i),k)\widetilde{P}(k,\mu(k),j) + P(i,\mu(i),j). \tag{13}$$

Proof: From i, the next state can either be in S_π or not. The first term in the RHS of (13) is the probability of reaching S_π and the first state is j, given that the next state is not in S_π. Adding it with the probability of next step is in S_π and the state is j gives the desired result. ■

We now define a $n \times n$ matrix $\widetilde{P}_\mu$ such that

$$\widetilde{P}_\mu(i,j) = \begin{cases} \widetilde{P}(i,\mu,j) & \text{if } j \in S_\pi \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

We can immediately see that $\widetilde{P}_\mu$ is a stochastic matrix, *i.e.,* all its rows sum to 1, since $\sum_{j=1}^{n} \widetilde{P}(i,\mu,j) = 1$. More precisely, $\sum_{j=1}^{m} \widetilde{P}(i,\mu,j) = 1$ since $\widetilde{P}(i,\mu,j) = 0$ for all $j = m+1,\ldots,n$.

Using (13), we can express $\widetilde{P}_\mu$ in a matrix equation in terms of P_μ. To do this, we need to first define two $n \times n$ matrices from P_μ as follows:

$$\overleftarrow{P}_\mu(i,j) = \begin{cases} P_\mu(i,j) & \text{if } j \in S_\pi \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

$$\overrightarrow{P}_\mu(i,j) = \begin{cases} P_\mu(i,j) & \text{if } j \notin S_\pi \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

From Fig. 1, we can see that matrix P_μ is "split" into $\overleftarrow{P}_\mu$ and $\overrightarrow{P}_\mu$, *i.e.,* $P_\mu = \overleftarrow{P}_\mu + \overrightarrow{P}_\mu$.

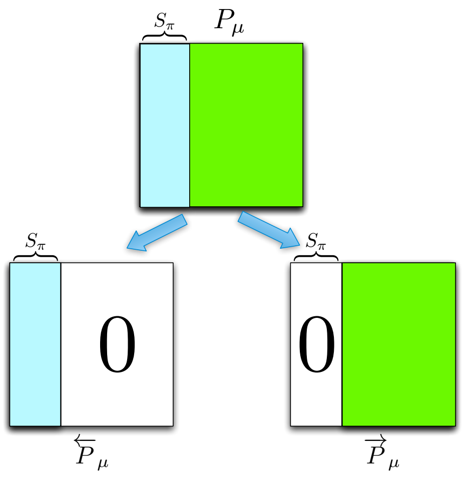


Fig. 1. The constructions of $\overleftarrow{P}_\mu$ and $\overrightarrow{P}_\mu$ from P_μ.

Proposition IV.5. *If a policy μ is proper, then matrix $I - \overrightarrow{P}_\mu$ is non-singular.*

Proof: Since μ is proper, for every initial state $i \in S$, the set S_π is eventually reached. Because of this, and since $\overrightarrow{P}_\mu(i,j) = 0$ if $j \in S_\pi$, the matrix $\overrightarrow{P}_\mu$ is transient, *i.e.,* $\lim_{k\to\infty} \overrightarrow{P}_\mu^k = 0$. From matrix analysis (see, *e.g.,* Ch. 9.4 of [26]), since $\overrightarrow{P}_\mu$ is transient and sub-stochastic, $I - \overrightarrow{P}_\mu$ is non-singular. ■

We can then write (13) as the following matrix equation:

$$\widetilde{P}_\mu = \overrightarrow{P}_\mu \widetilde{P}_\mu + \overleftarrow{P}_\mu. \tag{17}$$

Since $I - \overrightarrow{P}_\mu$ is invertible, we have

$$\widetilde{P}_\mu = (I - \overrightarrow{P}_\mu)^{-1} \overleftarrow{P}_\mu. \tag{18}$$

Next, we give an expression for the expected cost of reaching S_π from $i \in S$ under μ (if $i \in S_\pi$, this is the expected cost of reaching S_π again), and denote it as $\tilde{g}(i,\mu)$.

Proposition IV.6. *$\tilde{g}(i,\mu)$ satisfies*

$$\tilde{g}(i,\mu) = \sum_{k=m+1}^{n} P(i,\mu(i),k)\tilde{g}(k,\mu) + g(i,\mu(i)). \tag{19}$$

Proof: The first term of the RHS of (19) is the expected cost from the next state if the next state is not in S_π (if the next state is in S_π then no extra cost is incurred), and the second term is the one-step cost, which is incurred regardless of the next state. ■

We define $\tilde{g}_\mu$ such that $\tilde{g}_\mu(i) = \tilde{g}(i,\mu)$, and note that (19) can be written as

$$\begin{aligned} \tilde{g}_\mu &= \overrightarrow{P}_\mu \tilde{g}_\mu + g_\mu \\ \tilde{g}_\mu &= (I - \overrightarrow{P}_\mu)^{-1} g_\mu, \end{aligned} \tag{20}$$

where g_μ is defined in Sec. IV-A.

We can now express the ACPC J_μ in terms of $\widetilde{P}_\mu$ and $\tilde{g}_\mu$. Observe that, starting from i, the expected cost of the first cycle is $\tilde{g}_\mu(i)$; the expected cost of the second cycle is $\sum_{j=1}^{m} \widetilde{P}_\mu(i,\mu,j)\tilde{g}_\mu(j)$; the expected cost of the third cycle is $\sum_{j=1}^{m}\sum_{k=1}^{m} \widetilde{P}_\mu(i,\mu,j)\widetilde{P}_\mu(j,\mu,k)\tilde{g}_\mu(k)$; and so on. Therefore,

$$J_\mu = \limsup_{C\to\infty} \frac{1}{C} \sum_{k=0}^{C-1} \widetilde{P}_\mu^k \tilde{g}_\mu, \tag{21}$$

where C represents the cycle count. Since $\widetilde{P}_\mu$ is a stochastic matrix, the $\limsup$ in (21) can be replaced by the limit, and we have

$$J_\mu = \lim_{C\to\infty} \frac{1}{C} \sum_{k=0}^{C-1} \widetilde{P}_\mu^k \tilde{g}_\mu = \widetilde{P}_\mu^* \tilde{g}_\mu, \tag{22}$$

where P^* for a stochastic matrix P is defined in (7).

We can now make a connection between Prob. IV.2 and the ACPS problem. Each proper policy μ of $\mathcal{M}$ can be mapped via (18) and (20) to a policy $\tilde{\mu}$ with transition matrix $P_{\tilde{\mu}} := \widetilde{P}_\mu$ and vector of costs $g_{\tilde{\mu}} := \tilde{g}_\mu$, and we have

$$J_\mu = J^s_{\tilde{\mu}}. \tag{23}$$

Moreover, we define $h_\mu := h^s_{\tilde{\mu}}$. Together with J_μ, pair (J_μ, h_μ) can be seen as the gain-bias pair for the ACPC problem. We denote the set of all polices that can be mapped from a proper policy as $\mathbb{M}_{\tilde{\mu}}$. We see that a proper policy μ minimizing the ACPC maps to a policy over $\mathbb{M}_{\tilde{\mu}}$ minimizing the ACPS.

A by-product of the above analysis is that, if μ is proper, then $J_\mu(i)$ is finite for all i, since $\widetilde{P}_\mu^*$ is a stochastic matrix and $g_\mu(i)$ is finite. We now show that, among stationary policies, it is sufficient to consider only proper policies.

Proposition IV.7. *Assume μ to be an improper policy. If $\mathcal{M}$ is communicating, then there exists a proper policy μ' such that $J_{\mu'}(i) \leq J_\mu(i)$ for all $i = 1, \dots, n$, with strict inequality for at least one i.*

Proof: We partition S into two sets of states: $S_{\nrightarrow\pi}$ is the set of states in S that cannot reach S_π and $S_{\rightarrow\pi}$ as the set of states that can reach S_π with positive probability. Since μ is improper and $g(i,u)$ is postive-valued, $S_{\nrightarrow\pi}$ is not empty and $J_\mu(i) = \infty$ for all $i \in S_{\nrightarrow\pi}$. Moreover, states in $S_{\nrightarrow\pi}$ cannot visit $S_{\rightarrow\pi}$ by definition. Since $\mathcal{M}$ is communicating, there exists some actions at some states in $S_{\nrightarrow\pi}$ such that, if applied, all states in $S_{\nrightarrow\pi}$ can now visit S_π with positive probability and this policy is now proper (all states can now reach S_π). We denote this new policy as μ'. Note that this does not increase $J_\mu(i)$ if $i \in S_{\rightarrow\pi}$ since controls at these states are not changed. Moreover, since μ' is proper, $J_{\mu'}(i) < \infty$ for all $i \in S_{\nrightarrow\pi}$. Therefore $J_{\mu'}(i) < J_\mu(i)$ for all $i \in S_{\nrightarrow\pi}$. ■

Corollary IV.8. *If μ is optimal over the set of proper policies, it is optimal over the set of stationary policies.*

Proof: Follows directly from Prop. IV.7. ■

Using the connection to the ACPS problem, we have the following result.

Proposition IV.9. *The optimal ACPC policy over stationary policies is independent of the initial state.*

Proof: We first consider the optimal ACPC over proper policies. As mentioned before, if all stationary policies of an MDP satisfies the WA condition (see Def. IV.1), then the ACPS is equal for all initial states. Thus, we need to show that the WA condition is satisfied for all $\tilde{\mu}$. We will use S_π as set S_r. Since $\mathcal{M}$ is communicating, then for each pair $i, j \in S_\pi$, $P(i, \mu, j)$ is positive for some μ, therefore from (13), $\widetilde{P}_\mu(i,j)$ is positive for some μ (*i.e.,* $P_{\tilde{\mu}}(i,j)$ is positive for some $\tilde{\mu}$), and the first condition of Def. IV.1 is satisfied. Since μ is proper, the set S_π can be reached from all $i \in S$. In addition, $P_{\tilde{\mu}}(i,j) = 0$ for all $j \notin S_\pi$. Thus, all states $i \notin S_\pi$ are transient under all policies $\tilde{\mu} \in \mathbb{M}_{\tilde{\mu}}$, and the second condition is satisfied. Therefore WA condition is satisfied and the optimal ACPS over $\mathbb{M}_{\tilde{\mu}}$ is equal for all initial state. Hence, the optimal ACPC is the same for all initial states over proper policies. Using Prop. IV.7, we can conclude the same statement over stationary policies. ■

The above result is not surprising, as it mirrors the result for communicating MDPs in the ACPS problem. Essentially, transient behavior does not matter in the long run so the optimal cost is the same for any initial state.

Using Bellman's equation (10) and (11), and in the particular case when the optimal cost is the same for all initial states (12), policy $\tilde{\mu}^\star$ with the ACPS gain-bias pair $(\lambda\mathbf{1}, h)$ satisfying for all $\tilde{\mu} \in \mathbb{M}_{\tilde{\mu}}$:

$$\lambda\mathbf{1} + h \leq g_{\tilde{\mu}} + P_{\tilde{\mu}} h \tag{24}$$

is optimal. Equivalently, $\mu^\star$ that maps to $\tilde{\mu}^\star$ is optimal over all proper policies (and from Prop. IV.7, all stationary policies).

Remark IV.10. *Eq.* (23) *and* (24) *relate the ACPS and ACPC costs for the same MDP. However, they do not directly allow one to solve the ACPC problem using techniques from the ACPS problem. Indeed, it is not clear how to find the optimal policy from* (24) *except for by searching through all policies in $\mathbb{M}_{\tilde{\mu}}$. This exhaustive search is not feasible for reasonably large problems. Instead, we would like equations in the form of Bellman's equations* (10) *and* (11)*, so that computations can be carried out independently at each state.*

To circumvent this difficulty, we need to express the gain-bias pair (J_μ, h_μ), which is equal to $(J^s_{\tilde{\mu}}, h^s_{\tilde{\mu}})$, in terms of μ. From (8), we have

$$J_\mu = P_{\tilde{\mu}} J_\mu, \qquad J_\mu + h_\mu = g_{\tilde{\mu}} + P_{\tilde{\mu}} h_\mu.$$

By manipulating the above equations, we can now show that J_μ and h_μ can be expressed in terms of μ (analogous to (8)) instead of $\tilde{\mu}$.

Proposition IV.11. *The gain-bias pair (J_μ, h_μ) for the average cost per cycle problem satisfies*

$$J_\mu = P_\mu J_\mu, \qquad J_\mu + h_\mu = g_\mu + P_\mu h_\mu + \overrightarrow{P}_\mu J_\mu. \tag{25}$$

Moreover, we have

$$(I - \overrightarrow{P}_\mu) h_\mu + v_\mu = P_\mu v_\mu, \tag{26}$$

for some vector v_μ.

Proof: We start from (8):

$$J_\mu = P_{\tilde{\mu}} J_\mu, \qquad J_\mu + h_\mu = g_{\tilde{\mu}} + P_{\tilde{\mu}} h_\mu. \tag{27}$$

For the first equation in (27), using (18), we have

$$\begin{aligned}
J_\mu &= P_{\tilde{\mu}} J_\mu \\
J_\mu &= (I - \overrightarrow{P}_\mu)^{-1} \overleftarrow{P}_\mu J_\mu \\
(I - \overrightarrow{P}_\mu) J_\mu &= \overleftarrow{P}_\mu J_\mu \\
J_\mu - \overrightarrow{P}_\mu J_\mu &= \overleftarrow{P}_\mu J_\mu \\
J_\mu &= (\overrightarrow{P}_\mu + \overleftarrow{P}_\mu) J_\mu \\
J_\mu &= P_\mu J_\mu.
\end{aligned}$$

For the second equation in (27), using (18) and (20), we have

$$\begin{aligned}
J_\mu + h_\mu &= g_{\tilde{\mu}} + P_{\tilde{\mu}} h_\mu \\
J_\mu + h_\mu &= (I - \overrightarrow{P}_\mu)^{-1} (g_\mu + \overleftarrow{P}_\mu h_\mu) \\
(I - \overrightarrow{P}_\mu)(J_\mu + h_\mu) &= g_\mu + \overleftarrow{P}_\mu h_\mu \\
J_\mu - \overrightarrow{P}_\mu J_\mu + h_\mu - \overrightarrow{P}_\mu h_\mu &= g_\mu + \overleftarrow{P}_\mu h_\mu \\
J_\mu + h_\mu - \overrightarrow{P}_\mu J_\mu &= g_\mu + (\overrightarrow{P}_\mu + \overleftarrow{P}_\mu) h_\mu \\
J_\mu + h_\mu &= g_\mu + P_\mu h_\mu + \overrightarrow{P}_\mu J_\mu.
\end{aligned}$$

Thus, (27) can be expressed in terms of μ as:

$$J_\mu = P_\mu J_\mu, \qquad J_\mu + h_\mu = g_\mu + P_\mu h_\mu + \overrightarrow{P}_\mu J_\mu.$$

To compute J_μ and h_μ, we need an extra equation similar to (9). Using (9), we have

$$\begin{aligned}
h_\mu + v_\mu &= P_{\tilde{\mu}} v_\mu \\
h_\mu + v_\mu &= (I - \overrightarrow{P}_\mu)^{-1} \overleftarrow{P}_\mu v_\mu \\
(I - \overrightarrow{P}_\mu) h_\mu + v_\mu &= P_\mu v_\mu,
\end{aligned}$$

which completes the proof. ■

From Prop. IV.11, similar to the ACPS problem, (J_μ, h_μ, v_μ) can be solved as a linear system of $3n$ equations and $3n$ unknowns. The insight gained when simplifying J_μ and h_μ in terms of μ motivate us to propose the following optimality condition for an optimal policy.

Proposition IV.12. *The policy $\mu^\star$ with gain-bias pair $(\lambda\mathbf{1}, h)$ satisfying*

$$\lambda + h(i) = \min_{u\in U(i)}\left[g(i,u) + \sum_{j=1}^{n} P(i,u,j)h(j) + \lambda\sum_{j=m+1}^{n} P(i,u,j)\right], \tag{28}$$

for all $i = 1, \ldots, n$, is the optimal policy, minimizing (4) *over all stationary policies in $\mathbb{M}$.*

Proof: The optimality condition (28) can be written as:

$$\lambda\mathbf{1} + h \leq g_\mu + P_\mu h + \overrightarrow{P}_\mu \lambda\mathbf{1}, \tag{29}$$

for all stationary policies μ.

Note that, given $a, b \in \mathbb{R}^n$ and $a \leq b$, if A is an $n\times n$ matrix with all non-negative entries, then $Aa \leq Ab$. Moreover, given $c \in \mathbb{R}^n$, we have $a + c \leq b + c$.

From (29) we have

$$\begin{aligned}
\lambda\mathbf{1} + h &\leq g_\mu + P_\mu h + \overrightarrow{P}_\mu\lambda\mathbf{1} \\
\lambda\mathbf{1} - \overrightarrow{P}_\mu\lambda\mathbf{1} + h &\leq g_\mu + P_\mu h \\
\lambda\mathbf{1} - \overrightarrow{P}_\mu\lambda\mathbf{1} + h &\leq g_\mu + (\overleftarrow{P}_\mu + \overrightarrow{P}_\mu)h \\
\lambda\mathbf{1} - \overrightarrow{P}_\mu\lambda\mathbf{1} + h - \overrightarrow{P}_\mu h &\leq g_\mu + \overleftarrow{P}_\mu h \\
(I - \overrightarrow{P}_\mu)(\lambda\mathbf{1} + h) &\leq g_\mu + \overleftarrow{P}_\mu h
\end{aligned} \tag{30}$$

If μ is proper, then $\overrightarrow{P}_\mu$ is a transient matrix (see proof of Prop. IV.5), and all of its eigenvalues are strictly inside the unit circle. Therefore, we have

$$(I - \overrightarrow{P}_\mu)^{-1} = I + \overrightarrow{P}_\mu + \overrightarrow{P}_\mu^2 + \ldots.$$

Therefore, since all entries of $\overrightarrow{P}_\mu$ are non-negative, all entries of $(I - \overrightarrow{P}_\mu)^{-1}$ are non-negative. Thus, continuing from (30), we have

$$\begin{aligned}
(I - \overrightarrow{P}_\mu)(\lambda\mathbf{1} + h) &\leq g_\mu + \overleftarrow{P}_\mu h \\
\lambda\mathbf{1} + h &\leq (I - \overrightarrow{P}_\mu)^{-1}(g_\mu + \overleftarrow{P}_\mu h) \\
\lambda\mathbf{1} + h &\leq g_{\tilde{\mu}} + P_{\tilde{\mu}} h
\end{aligned}$$

for all proper policies μ and all $\tilde{\mu} \in \mathbb{M}_{\tilde{\mu}}$. Hence, $\tilde{\mu}^\star$ satisfies (24) and $\mu^\star$ is optimal over all proper policies. This completes the proof. ∎

In the next section, we will present an algorithm that uses Prop. IV.12 to find the optimal policy. Note that, unlike (24), the equations in (28) can be checked for any policy μ by finding the minimum for all states $i = 1, \ldots, n$, which is significantly easier than searching over all proper policies.

Finally, we show that the optimal stationary policy is optimal over all policies in $\mathbb{M}$.

Proposition IV.13. *The proper policy $\mu^\star$ that maps to $\tilde{\mu}^\star$ satisfying* (24) *is optimal over all policies in $\mathbb{M}$.*

Proof: Define an operator $T_{\tilde{\mu}}h$ as

$$(T_{\tilde{\mu}}h)(i) = g_{\tilde{\mu}}(i) + \sum_{i=1}^{n} P_{\tilde{\mu}}(i,j)h(j)$$

for all $i = 1\ldots, n$. Since $\tilde{\mu}^\star$ is ACPS optimal, we have (see [6]) that $T_{\tilde{\mu}_k}h \geq \lambda\mathbf{1} + h$ for all λ, h.

Consider a $M = \{\mu_1, \mu_2, \ldots\}$ and assume it to be optimal. We first consider that M is stationary for each cycle, but different for different cycles, and the policy is μ_k for the k-th cycle. Among this type of polices, from Prop. IV.7, we see that if M is optimal, then μ_k is proper for all k. In addition, the ACPC of policy M is the ACPS with policy $\{\tilde{\mu}_1, \tilde{\mu}_2, \ldots\}$. Since the optimal policy of the ACPS is $\tilde{\mu}^\star$ (stationary). Then we can conclude that if M is stationary in between cycles, then optimal policy for each cycle is $\mu^\star$ and thus $M = \mu^\star$.

Now we assume that M is not stationary for each cycle. For any cycle k, assume now we solve a stochastic shortest path problem with S_π as the target state, with no cost on controls, and terminal cost $g'(i) = J_M(i)$, where $J_M(i)$ is the optimal ACPC cost for M. Since there exists at least one proper policy and terminal costs are positive, the stochastic shortest path problem for S_π admits an optimal stationary policy as a solution [6]. If we replace the policy for cycle k with this stationary policy, we see that it clearly yields the same ACPC cost. Therefore, if we know M

Since $g(i,u) > 0$ for all i, u, and there exists at least one proper policy, the stochastic shortest path problem for S_π admits an optimal stationary policy as a solution [6]. Hence, for each cycle k, the cycle cost can be minimized if a stationary policy is used for the cycle. Therefore, a policy which is stationary in between cycles is optimal over $\mathbb{M}$, which is in turn, optimal if $M = \mu^\star$. This completes the proof. ∎

V. Synthesizing the optimal policy under LTL constraints

In this section we present an approach for Prob. III.1. We aim for a computational framework that produces a policy that both maximizes the satisfaction probability and optimizes the long-term performance of the system, using results from Sec. IV.

A. Automata-theoretical approach to LTL control synthesis

Our approach proceeds by converting the LTL formula ϕ to a DRA as defined in Def. II.1. We denote the resulting DRA by $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ with $F = \{(L(1), K(1)), \ldots, (L(M), K(M))\}$ where $L(i), K(i) \subseteq Q$ for all $i = 1, \ldots, M$. We now obtain an MDP as the product of a MDP $\mathcal{M}$ and a DRA $\mathcal{R}_\phi$, which captures all paths of $\mathcal{M}$ satisfying ϕ.

Definition V.1 (Product MDP)**.** *The product MDP $\mathcal{M} \times \mathcal{R}_\phi$ between a MDP $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$ and a DRA $\mathcal{R}_\phi = (Q, 2^\Pi, \delta, q_0, F)$ is a tuple $\mathcal{P} = (S_\mathcal{P}, U_\mathcal{P}, P_\mathcal{P}, s_{\mathcal{P}0}, F_\mathcal{P}, S_{\mathcal{P}\pi}, g_\mathcal{P})$, where*

(i) $S_\mathcal{P} = S \times Q$ is a set of states;
(ii) $U_\mathcal{P}$ is a set of controls inherited from $\mathcal{M}$ and we define $U_\mathcal{P}((s,q)) = U(s)$;
(iii) $P_\mathcal{P}$ gives the transition probabilities:

$$P_\mathcal{P}((s,q), u, (s',q')) = \begin{cases} P(s,u,s') & \textit{if } q' = \delta(q, \mathcal{L}(s)) \\ 0 & \text{otherwise;} \end{cases}$$

(iv) $s_{\mathcal{P}0} = (s_0, q_0)$ is the initial state;

(v) $F_{\mathcal{P}} = \{(L_{\mathcal{P}}(1), K_{\mathcal{P}}(1)), \ldots, (L_{\mathcal{P}}(M), K_{\mathcal{P}}(M))\}$ *where* $L_{\mathcal{P}}(i) = S \times L(i)$, $K_{\mathcal{P}}(i) = S \times K(i)$, *for* $i = 1, \ldots, M$;

(vi) $S_{\mathcal{P}\pi}$ *is the set of states in* $S_{\mathcal{P}}$ *for which proposition* π *is satisfied. Thus,* $S_{\mathcal{P}\pi} = S_\pi \times Q$;

(vii) $g_{\mathcal{P}}((s,q),u) = g(s,u)$ *for all* $(s,q) \in S_{\mathcal{P}}$;

Note that some states of $S_{\mathcal{P}}$ may be unreachable and therefore have no control available. To maintain a valid MDP satisfying Def. II.2, the unreachable states should be removed from $\mathcal{P}$ (via a simple graph search). With a slight abuse of notation we always assume that unreachable states are removed and still denote the resulting product MDP as $\mathcal{P}$. An example of a product MDP between a MDP and a DRA corresponding to the LTL formula $\phi = \mathsf{G}\,\mathsf{F}\,\pi \wedge \mathsf{G}\,\mathsf{F}\,a$ is shown in Fig. 2.

There is an one-to-one correspondence between a path $s_0 s_1, \ldots$ on $\mathcal{M}$ and a path $(s_0, q_0)(s_1, q_1)\ldots$ on $\mathcal{P}$. Moreover, from the definition of $g_{\mathcal{P}}$, the costs along these two paths are the same. The product MDP is constructed so that, given a path $(s_0, q_0)(s_1, q_1)\ldots$, the corresponding path $s_0 s_1 \ldots$ on $\mathcal{M}$ generates a word satisfying ϕ if and only if, there exists $(L_{\mathcal{P}}, K_{\mathcal{P}}) \in F_{\mathcal{P}}$ such that the set $K_{\mathcal{P}}$ is visited infinitely often and $L_{\mathcal{P}}$ finitely often.

A similar one-to-one correspondence exists for policies.

Definition V.2 (Inducing a policy from $\mathcal{P}$). *A policy* $M_{\mathcal{P}} = \{\mu_0^{\mathcal{P}}, \mu_1^{\mathcal{P}}, \ldots\}$ *on* $\mathcal{P}$*, where* $\mu_k^{\mathcal{P}}((s,q)) \in U_{\mathcal{P}}((s,q))$ *induces a policy* $M = \{\mu_0, \mu_1, \ldots\}$ *on* $\mathcal{M}$ *by setting* $\mu_k(s_k) = \mu_k^{\mathcal{P}}((s_k, q_k))$ *for all* k*. We denote* $M_{\mathcal{P}}|_{\mathcal{M}}$ *as the policy on* $\mathcal{M}$ *induced by* $M_{\mathcal{P}}$*, and we use the same notation for a set of policies.*

An induced policy can be implemented on $\mathcal{M}$ by simply keeping track of its current state on $\mathcal{P}$. Note that a stationary policy on $\mathcal{P}$ induces a non-stationary policy on $\mathcal{M}$. From the one-to-one correspondence between paths and the equivalence of their costs, the expected cost in (4) from initial state s_0 under $M_{\mathcal{P}}|_{\mathcal{M}}$ is equal to the expected cost from initial state (s_0, q_0) under $M_{\mathcal{P}}$.

For each pair of states $(L_{\mathcal{P}}, K_{\mathcal{P}}) \in F_{\mathcal{P}}$, we can obtain a set of accepting maximal end components (AMEC):

Definition V.3 (Accepting Maximal End Components). *Given* $(L_{\mathcal{P}}, K_{\mathcal{P}}) \in F_{\mathcal{P}}$*, an end component* $\mathcal{C}$ *is a communicating MDP*[3] $(S_{\mathcal{C}}, U_{\mathcal{C}}, P_{\mathcal{C}}, K_{\mathcal{C}}, S_{\mathcal{C}\pi}, g_{\mathcal{C}})$ *such that* $S_{\mathcal{C}} \subseteq S_{\mathcal{P}}$*,* $U_{\mathcal{C}} \subseteq U_{\mathcal{P}}$*,* $U_{\mathcal{C}}(i) \subseteq U(i)$ *for all* $i \in S_{\mathcal{C}}$*,* $K_{\mathcal{C}} = S_{\mathcal{C}} \cap K_{\mathcal{P}}$*,* $S_{\mathcal{C}\pi} = S_{\mathcal{C}} \cap S_{\mathcal{P}\pi}$*, and* $g_{\mathcal{C}}(i,u) = g_{\mathcal{P}}(i,u)$ *if* $i \in S_{\mathcal{C}}$*,* $u \in U_{\mathcal{C}}(i)$*. If* $P(i,u,j) > 0$ *for any* $i \in S_{\mathcal{C}}$ *and* $u \in U_{\mathcal{C}}(i)$*, then* $j \in S_{\mathcal{C}}$ *and* $P_{\mathcal{C}}(i,u,j) = P(i,u,j)$*. An accepting maximal end component (AMEC) is the largest such end component such that* $K_{\mathcal{C}} \neq \emptyset$ *and* $S_{\mathcal{C}} \cap L_{\mathcal{P}} = \emptyset$*. We denote the set of all AMECs for* $\mathcal{P}$ *as* $\mathcal{C}(\mathcal{P})$ *and* $S_{\mathcal{C}(\mathcal{P})} := \bigcup_{\mathcal{C} \in \mathcal{C}(\mathcal{P})} S_{\mathcal{C}}$.

Note that for a given pair $(L_{\mathcal{P}}, K_{\mathcal{P}})$, states of different AMECs are pairwise disjoint. For simplicity of notation, given $i \in S_{\mathcal{C}(\mathcal{P})}$, we let $\mathcal{C}(i)$ denote the AMEC that state i belongs to. An AMEC is a communicating MDP (and a subset of $\mathcal{P}$) containing at least one state in $K_{\mathcal{P}}$ and no state in $L_{\mathcal{P}}$. In the example shown in Fig. 2, there exists only one AMEC corresponding to $(L_{\mathcal{P}}, K_{\mathcal{P}})$, which is the only pair of states in $F_{\mathcal{P}}$.

[3]with a slight abuse of notation, we call $\mathcal{C}$ an MDP even though it is not initalized.

A procedure to obtain all AMECs of an MDP was provided in [13]. Note that there exists at least one AMEC. From probabilistic model checking [18], [19], a policy M maximizes the probability of satisfying ϕ (*i.e.,* $M \in \mathbb{M}_\phi$ as defined in (3)) if and only if $M = M^\star_{\mathcal{P}}|_{\mathcal{M}}$ and $M^\star_{\mathcal{P}}$ is such that the probability of reaching a state $i \in S_{\mathcal{C}(\mathcal{P})}$ from the initial state (s_0, q_0) is maximized on $\mathcal{P}$, *i.e.,* $M^\star_{\mathcal{P}} = \arg\max_{M \in \mathbb{M}} \Pr^M_{\mathcal{P}}(\mathsf{F}\, S_{\mathcal{C}(\mathcal{P})})$. A stationary optimal policy for the problem maximizing the satisfaction probability can be obtained by a value iteration algorithm or via a linear program (see [13]). We denote this optimal stationary policy as $\mu^\star_{\to\mathcal{C}(\mathcal{P})}$. Note that $\mu^\star_{\to\mathcal{C}(\mathcal{P})}$ is defined only for states outside of $S_{\mathcal{C}(\mathcal{P})}$.

On the product MDP $\mathcal{P}$, once an AMEC $\mathcal{C}$ is reached, since $\mathcal{C}$ itself is a communicating MDP, a policy can be constructed such that a state in $K_{\mathcal{C}}$ is visited infinitely often, satisfying the acceptance condition.

Remark V.4 (Maximal Probability Policy). *One can always construct at least one stationary policy* μ *from* $\mu^\star_{\to\mathcal{C}(\mathcal{P})}$ *such that* $\mu|_{\mathcal{M}}$ *maximizes the probability of satisfying* ϕ*. Define* μ *such that* $\mu(i) = \mu^\star_{\to\mathcal{C}(\mathcal{P})}(i)$ *for all* $i \notin S_{\mathcal{C}(\mathcal{P})}$*. Given an* (L, K) *pair, for any AMEC* $\mathcal{C} \in \mathcal{C}(\mathcal{P})$*, assume that the recurrent class of the Markov chain induced by a stationary policy does not intersect* $K_{\mathcal{C}}$*. Since* $\mathcal{C}$ *is communicating, one can always alter the action for a state in* $K_{\mathcal{C}}$ *and such that the recurrent class induced by the policy now intersects* $K_{\mathcal{C}}$*. The set* $\mathbb{M}_\phi$ *is, in general not unique since this construction can be used to generate a set of maximal probability stationary policies on* $\mathcal{P}$.

B. Optimizing the long-term performance of the MDP

For a control policy designed to satisfy an LTL formula, the system behavior outside an AMEC is transient, while the behavior inside an AMEC is long-term. The policy obtained from finding the maximal probability policy is not sufficient for an optimal policy, since it disregards the behavior inside an AMEC, as long as an accepting state is visited infinitely often. We now aim to optimize the long-term behavior of the MDP with respect to the ACPC cost function, while enforcing the satisfaction constraint. Since each AMEC is a communicating MDP, we can use results in Sec. IV-B to obtain a solution. Our approach consists of the following steps:

(i) Convert formula ϕ to a DRA $\mathcal{R}_\phi$ and obtain the product MDP $\mathcal{P}$ between $\mathcal{M}$ and $\mathcal{R}_\phi$;

(ii) Obtain the set of reachable AMECs, $\mathcal{C}(\mathcal{P})$;

(iii) For each $\mathcal{C} \in \mathcal{C}(\mathcal{P})$: Find a stationary policy $\mu^\star_{\to\mathcal{C}}(i)$, defined for $i \in S \setminus S_{\mathcal{C}}$, that reaches $S_{\mathcal{C}}$ with maximal probability $\max_{M \in \mathbb{M}} \Pr^M_{\mathcal{M}}(\phi)$; Find a stationary policy $\mu^\star_{\circlearrowright\mathcal{C}}(i)$, defined for $i \in S_{\mathcal{C}}$ minimizing (4) for MDP $\mathcal{C}$ and set $S_{\mathcal{C}\pi}$ while satisfying the LTL constraint; Define $\mu^\star_{\mathcal{C}}$ to be:

$$\mu^\star_{\mathcal{C}} = \begin{cases} \mu^\star_{\to\mathcal{C}}(i) & \text{if } i \notin S_{\mathcal{C}} \\ \mu^\star_{\circlearrowright\mathcal{C}}(i) & \text{if } i \in S_{\mathcal{C}} \end{cases}, \tag{31}$$

and denote the ACPC of $\mu^\star_{\circlearrowright\mathcal{C}}$ as $\lambda_{\mathcal{C}}$;

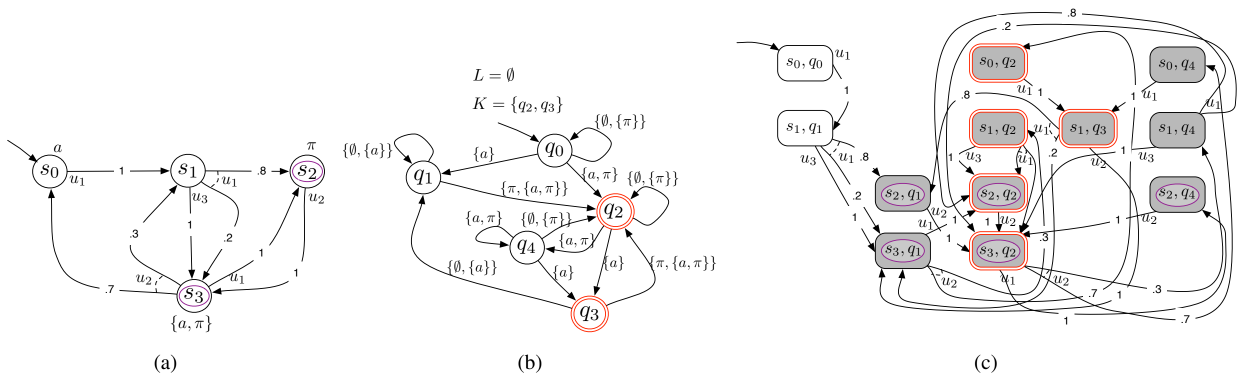


Fig. 2. The construction of the product MDP between a MDP and a DRA. In this example, the set of atomic proposition is $\{a,\pi\}$. **(a)**: A MDP where the label next to a state denotes the set of atomic propositions assigned to the state. The number on top of an arrow pointing from a state s to s' is the probability $P(s,u,s')$ associated with a control $u\in U(s)$. The set of states marked by ovals is S_π. **(b)**: The DRA $\mathcal{R}_\phi$ corresponding to the LTL formula $\phi=\mathsf{G}\,\mathsf{F}\,\pi\wedge\mathsf{G}\,\mathsf{F}\,a$. In this example, there is one set of accepting states $F=\{(L,K)\}$ where $L=\emptyset$ and $K=\{q_2,q_3\}$ (marked by double-strokes). Thus, the accepting runs of this DRA must visit q_2 or q_3 (or both) infinitely often. **(c)**: The product MDP $\mathcal{P}=\mathcal{M}\times\mathcal{R}_\phi$, where the states of $K_\mathcal{P}$ are marked by double-strokes and the states of $S_{\mathcal{P}\pi}$ are marked by ovals. The states with dashed borders are unreachable, and they are removed from $S_\mathcal{P}$. The states inside the AMEC are shown in grey.

(iv) Find the solution to Prob. III.1 by

$$J^\star(s_0)=\min_{\mathcal{C}\in\mathcal{C}(\mathcal{P})}\lambda_\mathcal{C}. \tag{32}$$

The optimal policy is $\mu^\star_{\mathcal{C}^\star}|_\mathcal{M}$, where $\mathcal{C}^\star$ is the AMEC attaining the minimum in (32).

We now provide the sufficient conditions for a policy $\mu^\star_{\circlearrowright\mathcal{C}}$ to be optimal. Moreover, if an optimal policy $\mu^\star_{\circlearrowright\mathcal{C}}$ can be obtained for each $\mathcal{C}$, we show that the above procedure indeed gives the optimal solution to Prob. III.1.

Proposition V.5. *For each $\mathcal{C}\in\mathcal{C}(\mathcal{P})$, let $\mu^\star_\mathcal{C}$ to be constructed as in* (31)*, where $\mu^\star_{\circlearrowright\mathcal{C}}$ is a stationary policy satisfying two optimality conditions: (i) its ACPC gain-bias pair is $(\lambda_\mathcal{C}\mathbf{1},h)$, where*

$$\begin{aligned}&\lambda_\mathcal{C}+h(i)\\ =\;&\min_{u\in U_\mathcal{C}(i)}\Big[g_\mathcal{C}(i,u)+\sum_{j\in S_\mathcal{C}}P(i,u,j)h(j)\\ &+\lambda_\mathcal{C}\sum_{j\notin S_{\mathcal{C}\pi}}P(i,u,j)\Big],\end{aligned} \tag{33}$$

for all $i\in S_\mathcal{C}$, and (ii) there exists a state of $K_\mathcal{C}$ in each recurrent class of $\mu^\star_{\circlearrowright\mathcal{C}}$. Then the optimal cost for Prob. III.1 is $J^\star(s_0)=\min_{\mathcal{C}\in\mathcal{C}(\mathcal{P})}\lambda_\mathcal{C}$, and the optimal policy is $\mu^\star_{\mathcal{C}^\star}|_\mathcal{M}$, where $\mathcal{C}^\star$ is the AMEC attaining this minimum.

Proof: Given $\mathcal{C}\in\mathcal{C}(\mathcal{P})$, define a set of policies $\mathbb{M}_\mathcal{C}$, such that for each policy in $\mathbb{M}_\mathcal{C}$: from initial state (s_0,q_0), (i) $S_\mathcal{C}$ is reached with probability $\max_{M\in\mathbb{M}}\Pr^M_\mathcal{M}(\phi)$, (ii) $S\setminus S_\mathcal{C}$ is not visited thereafter, and (iii) $K_\mathcal{C}$ is visited infinitely often. We see that, by the definition of AMECs, a policy almost surely satisfying ϕ belongs to $\mathbb{M}_\mathcal{C}|_\mathcal{M}$ for a $\mathcal{C}\in\mathcal{C}(\mathcal{P})$. Thus, $\mathbb{M}_\phi=\cup_{\mathcal{C}\in\mathcal{C}(\mathcal{P})}\mathbb{M}_\mathcal{C}|_\mathcal{M}$.

Since $\mu^\star_\mathcal{C}(i)=\mu^\star_{\rightarrow\mathcal{C}}(i)$ if $i\notin S_\mathcal{C}$, the state reaches $S_\mathcal{C}$ with probability $\max_{M\in\mathbb{M}}\Pr^M_\mathcal{M}(\phi)$ and in a finite number of stages. We denote the probability that $j\in S_\mathcal{C}$ is the first state visited in $S_\mathcal{C}$ when $\mathcal{C}$ is reached from initial state $s_{\mathcal{P}0}$ as $\widetilde{P}_\mathcal{C}(j,\mu^\star_{\rightarrow\mathcal{C}},s_{\mathcal{P}0})$. Since the ACPC for the finite path from the initial state to a state $j\in S_\mathcal{C}$ is 0 as the cycle index goes to ∞, the ACPC from initial state $s_{\mathcal{P}0}$ under policy $\mu^\star_\mathcal{C}$ is

$$J(s_0)=\sum_{j\in S_\mathcal{C}}\widetilde{P}_\mathcal{C}(j,\mu^\star_{\rightarrow\mathcal{C}},s_{\mathcal{P}0})J_{\mu^\star_{\circlearrowright\mathcal{C}}}(j). \tag{34}$$

Since $\mathcal{C}$ is communicating, the optimal cost is the same for all states of $S_\mathcal{C}$ (and thus it does not matter which state in $S_\mathcal{C}$ is first visited when $S_\mathcal{C}$ is reached). We have

$$\begin{aligned}J(s_0)&=\sum_{j\in S_\mathcal{C}}\widetilde{P}_\mathcal{C}(j,\mu^\star_{\rightarrow\mathcal{C}},(s_0,q_0))\lambda_\mathcal{C}\\&=\lambda_\mathcal{C}.\end{aligned} \tag{35}$$

Applying Prop. IV.12, we see that $\mu^\star_{\circlearrowright\mathcal{C}}$ satisfies the optimality condition for MDP $\mathcal{C}$ with respect to set $S_{\mathcal{C}\pi}$. Since there exists a state of $K_\mathcal{C}$ is in each recurrent class of $\mu^\star_{\circlearrowright\mathcal{C}}$, a state in $K_\mathcal{C}$ is visited infinitely often and it satisfies the LTL constraint. Therefore, $\mu^\star_\mathcal{C}$ as constructed in (31) is optimal over $\mathbb{M}_\mathcal{C}$ and $\mu^\star_\mathcal{C}|_\mathcal{M}$ is optimal over $\mathbb{M}_\mathcal{C}|_\mathcal{M}$ (due to equivalence of expected costs between $M_\mathcal{P}$ and $M_\mathcal{P}|_\mathcal{M}$). Since $\mathbb{M}_\phi=\cup_{\mathcal{C}\in\mathcal{C}(\mathcal{P})}\mathbb{M}_\mathcal{C}|_\mathcal{M}$, we have that $J^\star(s_0)=\min_{\mathcal{C}\in\mathcal{C}(\mathcal{P})}\lambda_\mathcal{C}$ and the policy corresponding to $\mathcal{C}^\star$ attaining this minimum is the optimal policy. ■

We can relax the optimality conditions for $\mu^\star_{\circlearrowright\mathcal{C}}$ in Prop. V.5 and require that there exist a state $i\in K_\mathcal{C}$ in one recurrent class of $\mu^\star_{\circlearrowright\mathcal{C}}$. For such a policy, we can construct a policy such that it has one recurrent class containing state i, with the same ACPC cost at each state. This construction is identical to a similar procedure for ACPS problems when the MDP is communicating (see [6, p. 203]). We can then use (31) to obtain the optimal policy $\mu^\star_\mathcal{C}$ for $\mathcal{C}$.

We now present an algorithm (see Alg. 1) that iteratively updates the policy in an attempt to find one that satisfies the optimality conditions given in Prop. V.5, for a given $\mathcal{C}\in\mathcal{C}(\mathcal{P})$. Note that Alg. 1 is similar in nature to policy iteration algorithms for ACPS problems. Moreover, Alg.1 always returns a stationary policy, and the cost of the policy returned by the algorithm does not depend on the initial state.

Algorithm 1 : Policy iteration algorithm for ACPC

```
Input: $\mathcal{C} = (S_\mathcal{C}, U_\mathcal{C}, P_\mathcal{C}, K_\mathcal{C}, S_{\mathcal{C}\pi}, g_\mathcal{C})$
Output: Policy $\mu_{\circlearrowright\mathcal{C}}$
1: Initialize $\mu^0$ to a proper policy containing $K_\mathcal{C}$ in its recurrent classes (such a policy can always be constructed since $\mathcal{C}$ is communicating)
2: repeat
3:   Given $\mu^k$, compute $J_{\mu^k}$ and $h_{\mu^k}$ with (25) and (26)
4:   Compute for all $i \in S_\mathcal{C}$:
       $\bar{U}(i) = \arg\min_{u \in U_\mathcal{C}(i)} \sum_{j \in S_\mathcal{C}} P(i,u,j) J_{\mu^k}(j)$   (36)
5:   if $\mu^k(i) \in \bar{U}(i)$ for all $i \in S_\mathcal{C}$ then
6:     Compute, for all $i \in S_\mathcal{C}$:
         $\bar{M}(i) = \arg\min_{u \in \bar{U}(i)} \Big[ g_\mathcal{C}(i,u) + \sum_{j \in S_\mathcal{C}} P(i,u,j) h_{\mu^k}(j) + \sum_{j \notin S_{\mathcal{C}\pi}} P(i,u,j) J_{\mu^k}(j) \Big]$   (37)
7:     Find $\mu^{k+1}$ such that $\mu^{k+1}(i) \in \bar{M}(i)$ for all $i \in S_\mathcal{C}$, and contains a state of $K_\mathcal{C}$ in its recurrent classes. If one does not exist. Return: $\mu^k$ with "not optimal"
8:   else
9:     Find $\mu^{k+1}$ such that $\mu^{k+1}(i) \in \bar{U}(i)$ for all $i \in S_\mathcal{C}$, and contains a state of $K_\mathcal{C}$ in its recurrent classes. If one does not exist, Return: $\mu^k$ with "not optimal"
10:  end if
11:  Set $k \leftarrow k + 1$
12: until $\mu^k$ with gain-bias pair satisfying (33) and Return: $\mu^k$ with "optimal"
```

Theorem V.6. *Given an accepting maximal end component $\mathcal{C}$, Alg. 1 terminates in a finite number of iterations. If it returns policy $\mu_{\circlearrowright\mathcal{C}}$ with "optimal", then $\mu_{\circlearrowright\mathcal{C}}$ satisfies the optimality conditions in Prop. V.5. If $\mathcal{C}$ is unichain (i.e., each stationary policy of $\mathcal{C}$ contains one recurrent class), then Alg. 1 is guaranteed to return the optimal policy $\mu^\star_{\circlearrowright\mathcal{C}}$.*

Proof: If $\mathcal{C}$ is unichain, then since it is also communicating, $\mu^\star_{\circlearrowright\mathcal{C}}$ contains a single recurrent class (and no transient state). In this case, since $K_\mathcal{C}$ is not empty, states in $K_\mathcal{C}$ are recurrent and the LTL constraint is always satisfied at step 7 and 9 of Alg. 1. The rest of the proof (for the general case and not assuming C to be unichain) is similar to the proof of convergence for the policy iteration algorithm for the ACPS problem (see [6, pp. 237-239]). Note that the proof is the same except that when the algorithm terminates at step 11 in Alg. 1, μ^k satisfies (33) instead of the optimality conditions for the ACPS problem ((10) and (11)). ■

If we obtain the optimal policy for each $\mathcal{C} \in \mathcal{C}(\mathcal{P})$, then we use (32) to obtain the optimal solution for Prob. III.1. If for some $\mathcal{C}$, Alg. 1 returns "not optimal", then the policy returned by Alg. 1 is only sub-optimal. We can then apply this algorithm to each AMEC in $\mathcal{C}(\mathcal{P})$ and use (32) to obtain a sub-optimal solution for Prob. III.1. Note that similar to policy iteration algorithms for ACPS problems, either the gain or the bias strictly decreases every time when μ is updated, so policy μ is improved in each iteration. In both cases, the satisfaction constraint is always enforced.

C. Properties of the proposed approach

1) Complexity: The complexity of our proposed algorithm is dictated by the size of the generated MDPs. We use $|\cdot|$ to denote cardinality of a set. The size of the DRA ($|Q|$) is in the worst case, doubly exponential with respect to $|\Sigma|$. However, empirical studies such as [24] have shown that in practice, the sizes of the DRAs for many LTL formulas are generally much lower and manageable. The size of product MDP $\mathcal{P}$ is at most $|\mathcal{S}| \times |Q|$. The complexity for the algorithm generating AMECs is at most quadratic in the size of $\mathcal{P}$ [13]. The complexity of Alg. 1 depends on the size of $\mathcal{C}$. The policy evaluation (step 3) requires solving a system of $3 \times |S_\mathcal{C}|$ linear equation with $3 \times |S_\mathcal{C}|$ unknowns. The optimization step (step 4 and 6) each requires at most $|U_\mathcal{C}| \times |S_\mathcal{C}|$ evaluations. Checking the recurrent classes of μ is linear in $|S_\mathcal{C}|$. Therefore, assuming that $|U_\mathcal{C}|$ is dominated by $|S_\mathcal{C}|^2$ (which is usually true) and the number of policies satisfying (36) and (37) for all i is also dominated by $|S_\mathcal{C}|^2$, for each iteration, the computational complexity is $O(|S_\mathcal{C}|^3)$.

2) Optimality: Even though the presented algorithm produces policies that are optimal only under a strict condition, it can be proved that for some fragments of LTL, the produced policy is optimal. One such fragment that is particularly relevant to persistent robotic tasks is in the form of

$$\phi = \mathsf{G}\,\mathsf{F}\,\pi \bigwedge_{\psi \in \Phi_{\mathsf{GF}}} \mathsf{G}\,\mathsf{F}\,\psi \bigwedge_{i=1}^{N} \mathsf{G}\left(\psi_i \Rightarrow \mathsf{X}\,\big(\bigwedge_{\psi \in \Phi_{\mathsf{GF}} \setminus \psi_{i+1}} \neg\psi\,\mathsf{U}\,\psi_{i+1}\big)\right), \quad (38)$$

where $\{\psi_1, \ldots, \psi_N\} = \Phi_{\mathsf{GF}}$ is a set of propositions such that $\pi \in \Phi_{\mathsf{GF}}$, and $\psi_{N+1} = \psi_1$ for convenience of notation. The first part of (38), $\bigwedge_{\psi \in \Phi_{\mathsf{GF}}} \mathsf{G}\,\mathsf{F}\,\psi$, ensures that all propositions in Φ_{GF} are satisfied infinitely many times; the second part of (38), $\bigwedge_{i=1}^{N} \mathsf{G}\,(\psi_i \Rightarrow \neg\psi_i\,\mathsf{U}\,\psi_{i+1})$, ensures that proposition ψ_{i+1} is always satisfied after ψ_i, and not before ψ_i is satisfied again.

Proposition V.7. *If the LTL formula ϕ is in the form of* (38)*, then Alg. 1 always returns an optimal solution.*

Proof: If $\mathsf{G}\,\mathsf{F}\,\pi$ is satisfied, then the last portion of (38), $\bigwedge_{i=1}^{N} \mathsf{G}\left(\psi_i \Rightarrow \mathsf{X}\,(\neg\psi_i\,\mathsf{U}\,\psi_{i+1})\right)$ ensures that all propositions in the sequence $\psi_1, \ldots, \psi_N$ are visited infinitely often. Therefore, $\bigwedge_{\psi \in \Phi_{\mathsf{GF}}} \mathsf{G}\,\mathsf{F}\,\psi$ is always satisfied. In this case, any policy in which π is visited infinitely often satisfies the LTL constraint, and therefore Alg. 1 returns the optimal solution. ■

VI. Case study

The algorithmic framework developed in this paper is implemented in MATLAB, and is available at http://hyness.bu.edu/Optimal-LTL.html. In this section, we examine two case studies. The first case study is aimed at demonstrating the ACPC algorithm without additional LTL constraints. The second case study is an example where we show that the ACPC algorithm can be used with LTL constraints.

A. Case Study 1

Consider a robot moving in a discrete environment modeled as an undirected graph $G = (V, E, c)$, where V is the set of vertices, E is a set of undirected edges, where each edge $e \in E$ is a two-element subset of V, and $c : E \to \mathbb{R}_{\geq 0}$ gives the travel cost between vertices. We can define the neighbors of a vertex $i \in V$ to be $\mathcal{N}_i := \{j \in V \mid \{i, j\} \in E\}$. The robot can deterministically choose among the edges available at a vertex and make a transition to one of its neighbors.

Events occur on the vertices of this graph, and the robot's goal is to repeatedly detect these events by visiting the associated vertices. Formally, each vertex $i \in V$ has a time-varying *event status* $x_i(k) \in \{0, 1\}$, where $k \in \mathbb{N}$ (i.e., the event status may change at each stage). The goal for the robot is to visit vertices while they have a status $x_i(k) = 1$ (an event is observed). This can be viewed as an average cost per cycle metric in which the robot seeks to minimize the expected time between observing the event. If the robot visits a vertex $i \in V$ at stage k, it views whether or not there is an event at i, and thus observes $x_i(k)$. We assume the probability of observing the event at vertex i is independent of k, and thus we denote it as $P_{\texttt{Obs}}(i)$. An example graph with its corresponding $P_{\texttt{Obs}}(i)$ is shown in Fig. 3a.

This problem can be converted into an ACPC problem with an MDP constructed as follows.

- We define the states as $S := V \times \{0, 1\}$. That is, we create two copies of each vertex, one for the status 0, and one for the status 1.
- For each state $s = (i, x_i) \in S$ we define $U(s) := \mathcal{N}_i$.
- For two states $s = (i, x_i)$ and $s' = (j, x_j)$ with $j \in U(s)$, we define the transition probability as

$$P(s, j, s') := \begin{cases} P_{\texttt{Obs}}(j), & \text{if } s' = (j, 1), \\ 1 - P_{\texttt{Obs}}(j), & \text{if } s' = (j, 0), \\ 0, & \text{otherwise.} \end{cases}$$

  We set $P(s, j, s') = 0$ if $s = (i, x_i), s' = (j, x_j)$ and $j \notin U(s)$.
- We define $\Pi := \{\pi\}$ and for a state $s = (i, x_i)$

$$\mathcal{L}(s) := \begin{cases} \pi, & \text{if } x_i = 1, \\ \emptyset, & \text{otherwise.} \end{cases}$$

- Finally, for a state $s = (i, x_i)$, and a control $j \in \mathcal{N}_i$ we define $g(s, j) := c\big((i, j)\big)$.

The problem of minimizing the expected time between observing an event is an average cost per cycle problem in which we minimize the time between visits to a state with π. We assume that in this case study, the probability of observing an event is highest at the top left and the bottom right corners of the environment, and decays exponentially with respect to the distance to these two corners. We first set s_0 to the state at the lower-left corner in the graph as shown in Fig. 3a. After running the algorithm as outlined in Alg. 1 (without the LTL constraints), we obtain the optimal ACPC cost $J^{\star}(s_0) = 15.737$ after 6 iterations. Note that the ACPC costs from all states are the same. The plot of ACPC cost $J^{\star}(s_0)$ versus the iteration count is shown in Fig. 3b. Finally, in Fig. 3c we show 2 sample paths, each containing 300 stages under the optimal policy from two different initial states. To verify that the average cycle costs are correct, we also computed the ACPC (up to stage 300) for 10 sample paths and catalogued the costs in Table I.

The mean cost over these sample paths is 15.86. On a laptop computer with a 2.2GHz Core i7 processor, Alg. 1 took less than 3 seconds to generate the optimal policy.

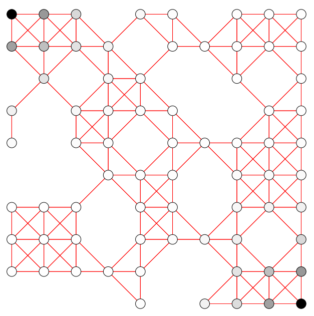
B. Case Study 2

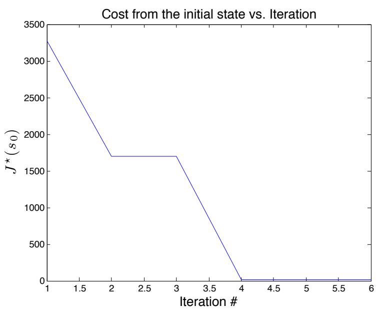
We now consider a case study, where a robot navigates in a graph-like environment $G = (V, E, c)$ similar to the first case study, while being required to satisfy an LTL specification. The environmental graph considered in the case study is shown in Fig. 4(a).

The robot is required to continuously pick up items (or customers) in the environment, and deliver them to designated drop-off sites (in this case study, either `DROPOFFA` or `DROPOFFB`, as shown in Fig. 6a). Note that `DROPOFFA` or `DROPOFFB` can each be a set of vertices. The robot can find out the destination of the item only after it is picked up. We assume that there is a pick-up probability $\texttt{PU}(v)$ associated with each vertex in the graph (see Fig. 4b). However, the robot is allowed to pick up items only if it is not currently delivering an item. Moreover, we assume that once picked-up at a vertex v, the probability that the item is destined for `DROPOFFA` depends on the vertex v, and is given as $\texttt{DOA}(v)$ (*i.e.,* the probability that the item is destined for `DROPOFFB` will be $1 - \texttt{DOA}(v)$). This probability is shown in Fig. 4c. As shown in Fig. 4b and Fig. 4c, the highest value of $\texttt{PU}(v)$ occurs at two vertices on the graph, while the highest value of $\texttt{DOA}(v)$ occurs at the `DROPOFFB` location. The problem we aim to solve for the robot is to design a control policy that minimizes the expected cost in between pick-ups, while satisfying the temporal constraints imposed by the pick-up delivery task.

We start by constructing the MDP capturing the motion of the robot in the graph as well as the state of an item being picked up.

- We define the states as $S := V \times \{\texttt{NONE}, \texttt{A}, \texttt{B}\}$. That is, we create three copies of each vertex, one for the state that no item is currently picked up ($x_i = \texttt{NONE}$), one for the state that an item destined for `DROPOFFA` is just picked up at i ($x_i = \texttt{A}$), and finally one for the state that an item destined for `DROPOFFB` is just picked up at i ($x_i = \texttt{B}$).
- For each state $s = (i, x_i) \in S$ we define $U(s) := \mathcal{N}_i \times \{\texttt{PICK}, \texttt{NOPICK}\}$, where an action $(j, \texttt{PICK})$ indicates that the robot moves to vertex j allowing an item to be picked-up at j, and $(j, \texttt{NOPICK})$ indicates that the robot does not pick up anything at j, regardless of $\texttt{PU}(j)$. The action $(j, \texttt{PICK})$ can only be enabled at states where $x_i = \texttt{NONE}$.
- For two states $s = (i, x_i)$ and $s' = (j, x_j)$ with $u \in U(s)$, we define the transition probability as follows:


(a)



(b)

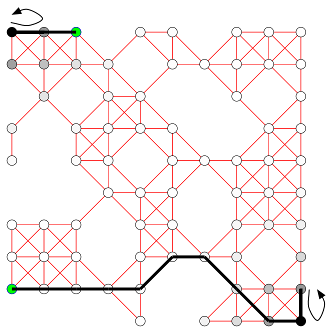
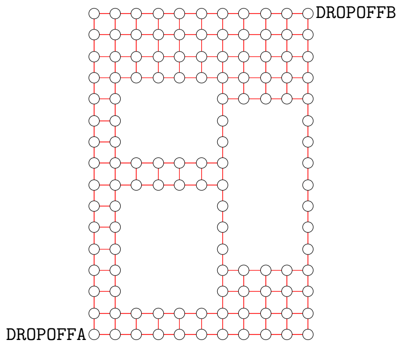

(c)

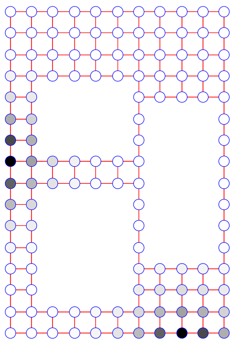
Fig. 3. (a) An example of a graph G modeling the motion of a robot in a grid-like environment. Its set of 74 vertices is a subset of the set of vertices of a rectangular grid with cell size 1. The weight function c is the Euclidean distance between vertices, and two vertices are neighbors if the Euclidean distance between them is less than 15. The color of the vertex corresponds to the probability that some event can be observed at the vertex, with black being 1 and white being 0. (b) The plot of $J^\star(s_0)$ versus iteration count. (c) Two sample paths (colored in black) under the optimal policy with two different initial states (colored in green) are shown. The sample paths converge either to the vertex at top left or at bottom right, where the probability of observing events is highest. It then cycles between that vertex and its neighbor vertex indefinitely.

Sample path no.	1	2	3	4	5	6	7	8	9	10
ACPC	15.85	15.96	15.55	15.79	15.91	15.53	16.73	15.98	15.73	15.56

TABLE I
ACPC (UP TO STAGE 300) FOR 10 SAMPLE PATHS



(a)


(b)

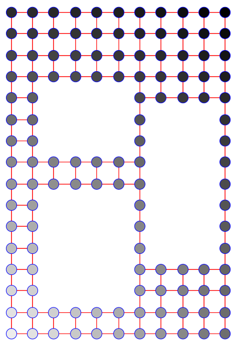

(c)

Fig. 4. (a) A graph G modeling the motion of a robot in a grid-like environment. Neighbohoods and costs are similar to the ones from Fig. 3. Delivery destination DROPOFFA is located at the bottom-left corner and DROPOFFB is located at top-right corner. (b) The shade of gray of the vertex corresponds to the probability $\mathtt{PU}(v)$ that some item will be picked up at the vertex, with black being probability 1 and white being probability 0. (c) The shade of gray of the vertex corresponds to the probability $\mathtt{DOA}(v)$ that after picking an item, it is to be dropped off at DROPOFFA, with black being probability 1 and white being probability 0 (*i.e.,* if an item is picked up near DROPOFFA, then it is more likely to be delivered to DROPOFFB and vice versa).

(i) If $u = (j, \mathtt{NOPICK})$ then $P(s, u, s') = 1$ if $s = (i, x)$, $s' = (j, x)$ and $j \in \mathcal{N}_i$;

(ii) If $u = (j, \mathtt{PICK})$, $x_i = \mathtt{NONE}$, and $j \in \mathcal{N}_i$

$$P(s,u,s') := \begin{cases} 1 - \mathtt{PU}(j), & \text{if } x_j = \mathtt{NONE}, \\ \mathtt{PU}(j)\mathtt{DOA}(j) & \text{if } x_j = \mathtt{A}, \\ \mathtt{PU}(j)(1-\mathtt{DOA}(j)) & \text{if } x_j = \mathtt{B}. \end{cases};$$

(iii) $P(s, u, s') = 0$ otherwise.

- We define $\Pi := \{\mathtt{PICKUP}, \mathtt{DROPOFFA}, \mathtt{DROPOFFB}, \mathtt{GOTOA}\}$. For a state $s = (i, x_i)$, we set $\mathcal{L}(s) := \{\mathtt{PICKUP}, \mathtt{GOTOA}\}$ if an item is to be picked up at i and to be dropped off at DROPOFFA, and $\mathcal{L}(s) := \{\mathtt{PICKUP}\}$ if an item is to be picked up at i and to be dropped off at DROPOFFB ; we set $\mathcal{L}(s) := \{\mathtt{DROPOFFA}\}$ if i is at the DROPOFFA location and $x_i = \mathtt{A}$; and $\mathcal{L}(s) := \{\mathtt{DROPOFFB}\}$ if i is at the DROPOFFB location and $x_i = \mathtt{B}$.
- We set the optimizing proposition π to be PICKUP.
- Finally, for a state $s = (i, x_i)$, and a control $u = (j, \alpha) \in U(s)$, we define $g(s, u) := c\big((i, j)\big)$ (recall that this is the edge weight for (i, j)).

The pick-up delivery task can be described by a set of temporal constraints, which is captured by the following LTL formula:

$$\begin{aligned} \phi \quad = \quad & \mathsf{G}\,\mathsf{F}\,\mathtt{PICKUP} \\ & \wedge \mathsf{G}\,(\mathtt{PICKUP} \Rightarrow \mathsf{X}\,(\neg\mathtt{PICKUP}\,\mathsf{U}\,\mathtt{DROPOFFA} \vee \mathtt{DROPOFFB})) \\ & \wedge \mathsf{G}\,(\mathtt{PICKUP} \wedge \neg\mathtt{GOTOA} \Rightarrow \mathsf{X}\,(\neg\mathtt{DROPOFFA}\,\mathsf{U}\,\mathtt{DROPOFFB})) \\ & \wedge \mathsf{G}\,(\mathtt{PICKUP} \wedge \mathtt{GOTOA} \Rightarrow \mathsf{X}\,(\neg\mathtt{DROPOFFB}\,\mathsf{U}\,\mathtt{DROPOFFA})) \end{aligned} \quad (39)$$

In (39), the first line enforces that the robot repeatedly pick up items. The second line ensures that new items cannot be picked up until the current items are dropped off. The last two lines enforce that, if an item is picked up at state with proposition GOTOA, then the item is destined for DROPOFFA, or DROPOFFB otherwise.

We generated the DRA $\mathcal{R}$ using the ltl2dstar tool [25] with 101 states and 2 pairs $(L, K) \in F$. The product MDP $\mathcal{P}$ contains 37269 states. There is one AMEC $\mathcal{C}$ corresponding to the first accepting state pair, and none for the second pair. The size of the AMEC is 1980. After picking an initial proper policy, Alg. 1 converged in 14 iterations to an optimal policy for this case study and the optimal cost is 269.48. The ACPC versus iteration count is plotted in Fig. 5. Some segments of a sample path are shown in Fig. 6. In these sample paths, the robot is driven to mid-center of the environment for an increased chance of picking-up items. For each of these sample paths, once an item is picked up, it is delivered to DROPOFFB.

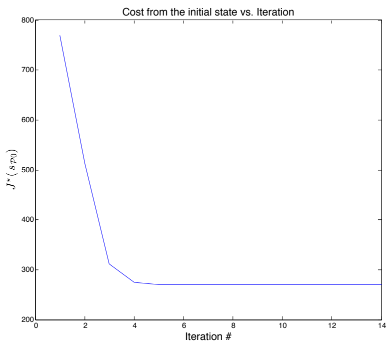


Fig. 5. The plot of $J^\star(s_0)$ versus iteration count.

The entire case study took about 20 minutes of computation time on a MacBook Pro with 2.2GHz Intel Core i7 CPU and 4GB of RAM.

VII. Conclusions

We developed a method to automatically generate a control policy for a Markov Decision Process (MDP), in order to satisfy specifications given as Linear Temporal Logic formulas. The control policy maximizes the probability that the MDP satisfies the given specification, and in addition, the policy optimizes the expected cost between satisfying instances of an "optimizing proposition", under some conditions. The problem is motivated by robotic applications requiring persistent tasks to be performed such as environmental monitoring or data gathering.

We are working on several extensions of this work. First, we are interested in characterizing the class of LTL formulas for which our approach finds the optimal solution. Second, we would like to apply the optimization criterion of average cost per cycle to more complex models such as Partially Observable MDPs (POMDPs) and semi-MDPs. Furthermore, in this paper we assume that the transition probabilities are known. In the case that they are not, we would like to address the robust formulation of the same problem. A potential direction is to convert the proposed algorithm into a linear program (LP) and take advantage of literature on robust solutions to LPs [27] and sensitivity analysis for LPs [28]. We are also interested in combining formal synthesis techniques with incremental learning of the MDP, as suggested in [29]. Finally, we are interested in determining more general classes of cost functions for which it is possible to determine optimal MDP policies, subject to LTL constraints.

References

[1] X. Ding, S. L. Smith, C. Belta, and D. Rus, "LTL control in uncertain environments with probabilistic satisfaction guarantees," in *IFAC World Congress*, Milan, Italy, Aug. 2011.

[2] ——, "MDP optimal control under temporal logic constraints," in *IEEE Conf. on Decision and Control*, Orlando, FL, December 2011, pp. 532–538.

[3] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transaction on Robotics*, vol. 28, pp. 396–409, 2011.

[4] S. Temizer, M. J. Kochenderfer, L. P. Kaelbling, T. Lozano-Pérez, and J. K. Kuchar, "Collision avoidance for unmanned aircraft using Markov decision processes," in *AIAA Conf. on Guidance, Navigation and Control*, Toronto, Canada, Aug. 2010.

[5] R. Alterovitz, T. Siméon, and K. Goldberg, "The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty," in *Robotics: Science and Systems*, Atlanta, GA, Jun. 2007.

[6] D. Bertsekas, *Dynamic programming and optimal control, vol. II*. Athena Scientific, 2007.

[7] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley, 1994.

[8] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, "Where's Waldo? Sensor-based temporal logic motion planning," in *IEEE Int. Conf. on Robotics and Automation*, Rome, Italy, 2007, pp. 3116–3121.

[9] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *IEEE Conf. on Decision and Control*, Shanghai, China, 2009, pp. 2222–2229.

[10] S. G. Loizou and K. J. Kyriakopoulos, "Automatic synthesis of multiagent motion tasks based on LTL specifications," in *IEEE Conf. on Decision and Control*, Paradise Island, Bahamas, 2004, pp. 153–158.

[11] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, vol. 57, no. 11, pp. 2817–2830, 2012.

[12] E. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.

[13] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.

[14] N. Piterman, A. Pnueli, and Y. Saar, "Synthesis of reactive(1) designs," in *International Conference on Verification, Model Checking, and Abstract Interpretation*, Charleston, SC, 2006, pp. 364–380.

[15] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Transactions on Automatic Control*, vol. 53, no. 1, pp. 287–297, 2008.

[16] B. Yordanov, J. Tumova, I. Cerna, J. Barnat, and C. Belta, "Temporal logic control of discrete-time piecewise affine systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 6, pp. 1491–1504, 2012.

[17] M. Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Hybrid Systems: Computation and Control*, ser. Lecture Notes in Computer Science, M. Egerstedt and B. Mishra, Eds. Springer, 2008, pp. 287–300.

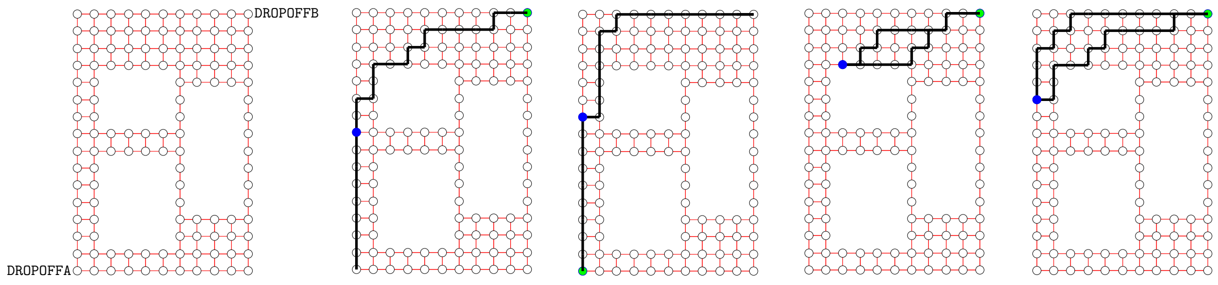
[18] L. De Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1997.

[19] M. Vardi, "Probabilistic linear-time model checking: An overview of the automata-theoretic approach," *Formal Methods for Real-Time and Probabilistic Systems*, pp. 265–276, 1999.

[20] C. Courcoubetis and M. Yannakakis, "Markov decision processes and regular events," *IEEE Transactions on Automatic Control*, vol. 43, no. 10, pp. 1399–1418, 1998.

[21] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, "Controller synthesis for probabilistic systems," in *Proceedings of IFIP TCS'2004*. Kluwer Academic Publishers, 2004.

[22] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning for surveillance with temporal logic constraints," *International Journal of Robotics Research*, vol. 30, no. 14, pp. 1695–1708, 2011.



(a) (b) (c) (d) (e)

Fig. 6. Four segments within one sample path for case study 2 where each segment completes one pickup-delivery cycle. The initial vertex of the cycle is shown in green. The vertex where the item is picked up is shown in blue.

[23] E. Gradel, W. Thomas, and T. Wilke, *Automata, logics, and infinite games: A guide to current research*, ser. Lecture Notes in Computer Science. Springer, 2002, vol. 2500.

[24] J. Klein and C. Baier, "Experiments with deterministic ω-automata for formulas of linear temporal logic," *Theoretical Computer Science*, vol. 363, no. 2, pp. 182–195, 2006.

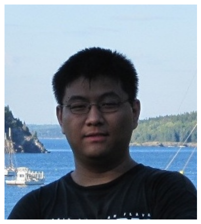
[25] J. Klein, "ltl2dstar - LTL to deterministic Streett and Rabin automata," http://www.ltl2dstar.de/, 2007, viewed September 2010.

[26] L. Hogben, *Handbook of linear algebra*. CRC Press, 2007.

[27] A. Ben-Tal and A. Nemirovski, "Robust solutions of uncertain linear programs," *Operations research letters*, vol. 25, no. 1, pp. 1–13, 1999.

[28] J. E. Ward and R. E. Wendell, "Approaches to sensitivity analysis in linear programming," *Annals of Operations Research*, vol. 27, no. 1, pp. 3–38, 1990.

[29] Y. Chen, J. Tumová, and C. Belta, "Ltl robot motion control based on automata learning of environmental dynamics," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 5177–5182.



Xu Chu (Dennis) Ding received his B.S., M.S. and Ph.D. degree in Electrical and Computer Engineering from the Georgia Institute of Technology, Atlanta, in 2004, 2007 and 2009, respectively. During 2010 to 2011 he was a postdoctoral research fellow at Boston University. In 2011 he joined United Technologies Research Center as a Senior Research Scientist. His research interests include hierarchical mission planning with formal guarantees under dynamic and rich environments, optimal control of hybrid systems, and coordination of multi-agent networked systems.

Calin Belta is an Associate Professor in the Department of Mechanical Engineering, Department of Electrical and Computer Engineering, and the Division of Systems Engineering at Boston University, where he is also affiliated with the Center for Information and Systems Engineering (CISE), the Bioinformatics Program, and the Center for Biodynamics (CBD). His research focuses on dynamics and control theory, with particular emphasis on hybrid and cyber-physical systems, formal synthesis and verification, and applications in robotics and systems biology. Calin Belta is a Senior Member of the IEEE and an Associate Editor for the SIAM Journal on Control and Optimization (SICON). He received the Air Force Office of Scientific Research Young Investigator Award and the National Science Foundation CAREER Award.

Stephen L. Smith (S'05–M'09) received the B.Sc. degree in engineering physics from Queens University, Canada, in 2003, the M.A.Sc. degree in electrical and computer engineering from the University of Toronto, Canada, in 2005, and the Ph.D. degree in mechanical engineering from the University of California, Santa Barbara in 2009. From 2009 to 2011 he was a Postdoctoral Associate in the Computer Science and Artificial Intelligence Laboratory at the Massachusetts Institute of Technology. He is currently an Assistant Professor in electrical and computer engineering at the University of Waterloo, Canada. His main research interests lie in control and optimization for networked systems, with an emphasis on autonomy, transportation, and robotic motion planning.

Daniela Rus is the Andrew (1956) and Erna Viterbi Professor of Electrical Engineering and Computer Science and Director of the Computer Science and Artificial Intelligence Laboratory (CSAIL) at MIT. Prior to her appointment as Director, she served as Associate Director of CSAIL from 2008 to 2011, and as the Co-Director of CSAILs Center for Robotics from 2005 to 2012. She also leads CSAILs Distributed Robotics Laboratory. Rus is the first woman to serve as director of CSAIL. Rus research interests include distributed robotics, mobile computing and programmable matter. At CSAIL she has led numerous groundbreaking research projects in the areas of transportation, security, environmental modeling and monitoring, underwater exploration, and agriculture.