

MDP Optimal Control under Temporal Logic Constraints

- Technical Report -

Xu Chu 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. We synthesize a control policy such that the MDP satisfies the given specification almost surely, if such a policy exists. In addition, we designate an “optimizing proposition” to be repeatedly satisfied, and we formulate a novel optimization criterion in terms of minimizing the expected cost in between satisfactions of this proposition. We propose a sufficient condition for a policy to be optimal, and develop a dynamic programming algorithm that synthesizes a policy that is optimal under some conditions, and sub-optimal otherwise. This problem is motivated by robotic applications requiring persistent tasks, such as environmental monitoring or data gathering, to be performed.

I. INTRODUCTION

In this paper, we consider the problem of controlling a (finite state) Markov Decision Process (MDP). Such models are widely used in various areas including engineering, biology, and economics. In particular, in recent results, they have been successfully used to model and control autonomous robots subject to uncertainty in their sensing and actuation, such as for ground robots [1], unmanned aircraft [2] and surgical steering needles [3].

Several authors [4]–[7] have proposed using temporal logics, such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL) [8], as specification languages for control systems. Such logics are appealing because they have well defined syntax and semantics, which can be easily used to specify complex behavior. In particular, in LTL, it is possible to specify persistent tasks, *e.g.*, “Visit regions A , then B , and then C , infinitely often. Never enter B unless coming directly from D .” In addition, off-the-shelf model checking algorithms [8] and temporal logic game strategies [9] can be used to verify the correctness of system trajectories and to synthesize provably correct control strategies.

The 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 [10]. 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 Büchi or GR(1) game if the specification is restricted to a fragment of LTL [4], [11]. If the probabilities of the enabled transitions at each state are known, the control problem reduces to finding a control policy for an MDP such that a probabilistic temporal logic formula is satisfied [12].

By adapting methods from probabilistic model-checking [12]–[14], we have recently developed frameworks for deriving MDP control policies from LTL formulas [15], which is related to a number of other approaches [16], [17] that address the problem of synthesizing control policies for MDPs subject to LTL satisfaction constraints. In all of the above approaches, a control policy is designed to maximize the probability of satisfying a given LTL formula. However, no attempt has been made so far to optimize the long-term behavior of the system, while enforcing LTL satisfaction guarantees. Such an objective is often critical in many applications, such as surveillance, persistent monitoring, and pickup delivery tasks, where a robot must repeatedly visit some areas in an environment and the time in between revisits should be minimized.

As far as we know, this paper is the first attempt to compute an optimal control policy for a dynamical system modeled as an MDP, while satisfying temporal logic constraints. This work begins to bridge the gap between our prior work on MDP control policies maximizing the probability of satisfying an LTL formula [15] and optimal path planning under LTL constraints [18]. We consider LTL formulas defined over a set of propositions assigned to the states of the MDP. We synthesize a control policy such that the MDP satisfies the specification almost surely, if such a policy exists. In addition, we minimize the expected cost between satisfying instances of an “optimizing proposition.”

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 a cycle is defined between successive satisfaction of the optimizing proposition. We present a novel connection between this problem and the well-known average cost per stage problem. Second, we incorporate the LTL constraints and obtain a sufficient condition for a policy to be optimal. We present a dynamic programming algorithm that under some (heavy) restrictions synthesizes an optimal control policy, and a sub-optimal policy otherwise.

This work was supported in part by ONR-MURI N00014-09-1051, ARO W911NF-09-1-0088, AFOSR YIP FA9550-09-1-020, and NSF CNS-0834260.

X. C. Ding and C. Belta are with Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA (email: {xcding; cbelta}@bu.edu). 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). D. Rus is with the Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA (email: rus@csail.mit.edu).

The organization of this paper is as follows. In Sec. II we provide some preliminaries. We formulate the problem in Sec. III and we formalize the connection between the average cost per cycle and the average cost per stage problem 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

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 [8], [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 \bigcirc (next), \mathcal{U} (until), \diamond (eventually), \square (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; $\square\phi$ means that ϕ is true at all positions of the word; $\diamond\phi$ means that ϕ eventually becomes true in the word; $\phi_1 \mathcal{U} \phi_2$ means that ϕ_1 has to hold at least until ϕ_2 is true. More expressivity can be achieved by combining the above temporal and Boolean operators (more examples are given later). 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) $\delta : Q \times \Sigma \rightarrow Q$ is the transition function; (iv) $q_0 \in Q$ is the initial state; and (v) $F = \{(L(1), K(1)), \dots, (L(M), K(M))\}$ is a set of pairs of sets of states such that $L(i), K(i) \subseteq Q$ for all $i = 1, \dots, M$.*

A run of a Rabin automaton \mathcal{R} , denoted by $r_{\mathcal{R}} = q_0 q_1 \dots$, 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.

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 [19]). We refer readers to [20] and references therein for algorithms and to freely available implementations, such as [21], to translate a LTL formula over Π to a corresponding DRA.

B. Markov Decision Process and probability measure

Definition II.2 (Labeled Markov Decision Process). *A labeled Markov decision process (MDP) is a tuple $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$, where $S = \{1, \dots, n\}$ is a finite set of states; U is a finite set of controls (actions) (with slight abuse of notations we also define the function $U(i)$, where $i \in S$*

and $U(i) \subseteq U$ to represent the available controls at state i); $P : S \times U \times S \rightarrow [0, 1]$ 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; $\mathcal{L} : S \rightarrow 2^\Pi$ is a labeling function and $g : S \times U \rightarrow \mathbb{R}^+$ is such that $g(i, u)$ is the expected (non-negative) cost when control $u \in U(i)$ is taken at state i .

We define a control function $\mu : S \rightarrow U$ such that $\mu(i) \in U(i)$ for all $i \in S$. A infinite sequence of control functions $M = \{\mu_0, \mu_1, \dots\}$ 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 . Given an initial state s_0 , an infinite sequence $r_{\mathcal{M}}^M = s_0 s_1 \dots$ 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*. If $\mu_k = \mu$ for all k , then we call it a *stationary* policy and we denote it 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 $\text{Paths}_{\mathcal{M}}^M$ and $\text{FPaths}_{\mathcal{M}}^M$ 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 $\text{Paths}_{\mathcal{M}}^M$. For a path $r_{\mathcal{M}}^M = s_0 s_1 \dots s_m s_{m+1} \dots \in \text{Paths}_{\mathcal{M}}^M$, the *prefix* of length m of $r_{\mathcal{M}}^M$ is the finite subsequence $s_0 s_1 \dots s_m$. Let $\text{Paths}_{\mathcal{M}}^M(s_0 s_1 \dots s_m)$ denote the set of all paths in $\text{Paths}_{\mathcal{M}}^M$ with the prefix $s_0 s_1 \dots s_m$. (Note that $s_0 s_1 \dots s_m$ is a finite path in $\text{FPaths}_{\mathcal{M}}^M$.) Then, the probability measure $\Pr_{\mathcal{M}}^M$ on the smallest σ -algebra over $\text{Paths}_{\mathcal{M}}^M$ containing $\text{Paths}_{\mathcal{M}}^M(s_0 s_1 \dots s_m)$ for all $s_0 s_1 \dots s_m \in \text{FPaths}_{\mathcal{M}}^M$ is the unique measure satisfying:

$$\Pr_{\mathcal{M}}^M\{\text{Paths}_{\mathcal{M}}^M(s_0 s_1 \dots s_m)\} = \prod_{0 \leq k < n} P(s_k, \mu_k(s_k), s_{k+1}).$$

Finally, we can define the probability that an MDP \mathcal{M} under a policy M satisfies an LTL formula ϕ . A path $r_{\mathcal{M}}^M = s_0 s_1 \dots$ deterministically generates a word $o = o_0 o_1 \dots$, 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 $\{r_{\mathcal{M}}^M \in \text{Paths}_{\mathcal{M}}^M : \mathcal{L}(r_{\mathcal{M}}^M) \models \phi\}$ is measurable. We define:

$$\Pr_{\mathcal{M}}^M(\phi) := \Pr_{\mathcal{M}}^M\{r_{\mathcal{M}}^M \in \text{Paths}_{\mathcal{M}}^M : \mathcal{L}(r_{\mathcal{M}}^M) \models \phi\} \quad (1)$$

as the probability of satisfying ϕ for \mathcal{M} under 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 weighted MDP $\mathcal{M} = (S, U, P, s_0, \Pi, \mathcal{L}, g)$ and an LTL formula ϕ over Π . As proposed in [18], we assume that formula ϕ is of the form:

$$\phi = \square \diamond \pi \wedge \psi, \quad (2)$$

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. We assume that there exists at least one policy M of \mathcal{M} such that \mathcal{M} under M satisfies ϕ almost surely, i.e., $\Pr_{\mathcal{M}}^M(\phi) = 1$ (in this case we simply say M satisfies ϕ almost surely).

We let \mathbb{M} be the set of all policies and \mathbb{M}_ϕ be the set of all policies satisfying ϕ almost surely. Note that if there exists a control policy satisfying ϕ almost surely, then there typically exist many (possibly infinite number of) such policies.

We would like to obtain the optimal policy such that ϕ is almost surely satisfied, and the expected cost in between visiting a state satisfying π is minimized. To formalize this, 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 $r_{\mathcal{M}}^M = s_0 s_1 \dots$, 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 in N stages 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, we have:

Problem III.1. Find a policy $M = \{\mu_0, \mu_1, \dots\}$, $M \in \mathbb{M}_\phi$ that minimizes

$$J(s_0) = \limsup_{N \rightarrow \infty} E \left\{ \frac{\sum_{k=0}^N g(s_k, \mu_k(s_k))}{C(r_{\mathcal{M}}^M, N)} \right\}, \quad (3)$$

where $E\{\cdot\}$ denotes the expected value.

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

$$J^s(s_0) = \limsup_{N \rightarrow \infty} E \left\{ \frac{\sum_{k=0}^N g(s_k, \mu_k(s_k))}{N} \right\}, \quad (4)$$

over \mathbb{M} , with the noted difference that the right-hand-side (RHS) of (4) 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.

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. For example, in a data gathering mission [18], 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 upload. In such cases, the ACPS cost function does not translate to a meaningful performance criterion. In fact, a policy minimizing (4) may even produce an infinite cost in (3). Nevertheless, we will make the connection between the ACPS and the ACPC problems in Sec. IV.

Remark III.2 (Optimization Criterion). *The optimization criterion in Prob. III.1 is only meaningful for specifications where π is satisfied infinitely often. Otherwise, the limit from Eq. (3) is infinite (since g is a positive-valued function) and Prob. III.1 has no solution. This is the reason for choosing ϕ in the form $\Box \diamond \pi \wedge \psi$ and for only searching among policies that almost surely satisfy ϕ .*

IV. SOLVING THE AVERAGE COST PER CYCLE PROBLEM

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 (4). The reader interested in more details is referred to [22], [23] and references therein.

Definition IV.1 (Weak Accessibility Condition). *An MDP \mathcal{M} is said to satisfy the Weak Accessibility (WA) condition if there exist $S_r \subseteq S$, 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.*

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*. For a stationary policy, it 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 is unichain, \mathcal{M} is called unichain.

Recall that the set of states of \mathcal{M} is denoted by $\{1, \dots, n\}$. 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 vector g_μ where $g_\mu(i) = g(i, \mu(i))$. For each stationary policy μ , we can obtain a so-called gain-bias pair (J_μ^s, h_μ^s) , where

$$J_\mu^s = P_\mu^* g_\mu, \quad h_\mu^s = H_\mu^s g_\mu \quad (5)$$

with

$$P_\mu^* = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{k=0}^{N-1} P_\mu^k, \quad H_\mu = (I - P_\mu + P_\mu^*)^{-1} - P_\mu^*. \quad (6)$$

The vector $J_\mu^s = [J_\mu^s(1), \dots, J_\mu^s(n)]^T$ is such that $J_\mu^s(i)$ is the ACPS starting at initial state i under policy μ . Note that the limit in (6) exists for any stochastic matrix P_μ , and P_μ^* is stochastic. Therefore, the lim sup in (4) can be replaced by the limit for a stationary policy. Moreover, (J_μ^s, h_μ^s) satisfies

$$J_\mu^s = P_\mu J_\mu^s, \quad J_\mu^s + h_\mu^s = g_\mu + P_\mu h_\mu^s. \quad (7)$$

By noting that

$$h_\mu^s + v_\mu^s = P_\mu v_\mu^s, \quad (8)$$

for some vector v_μ^s , we see that $(J_\mu^s, h_\mu^s, v_\mu^s)$ is the solution of $3n$ linear equations with $3n$ unknowns.

It has been shown that there exists a stationary optimal policy μ^* minimizing (4) over all policies, where its gain-bias

pair (J^s, h^s) satisfies the Bellman's equations for average cost per stage problems:

$$J^s(i) = \min_{u \in \bar{U}(i)} \sum_{j=1}^n P(i, u, j) J^s(j) \quad (9)$$

and

$$J^s(i) + h^s(i) = \min_{u \in \bar{U}(i)} \left[g(i, u) + \sum_{j=1}^n P(i, u, j) h^s(j) \right], \quad (10)$$

for all $i = 1, \dots, n$, where \bar{U}_i is the set of controls attaining the minimum in (9). Furthermore, if \mathcal{M} is single-chain, the optimal average cost does not depend on the initial state, i.e., $J_{\mu^*}^s(i) = \lambda$ for all $i \in S$. In this case, (9) is trivially satisfied and \bar{U}_i in (10) can be replaced by $U(i)$. Hence, μ^* with gain-bias pair $(\lambda \mathbf{1}, h)$ is optimal over all policies if for all stationary policies μ we have:

$$\lambda \mathbf{1} + h \leq g_{\mu} + P_{\mu} h, \quad (11)$$

where $\mathbf{1} \in \mathbb{R}^n$ is a vector of all 1s and \leq is component-wise.

B. Optimality conditions for ACPC problems

Now we derive equations similar to (9) and (10) for ACPC problems, without considering the satisfaction constraint, i.e., we do not limit the set of policies to \mathbb{M}_{ϕ} 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 (3).*

Note that, for reasons that will become clear in Sec. V, we assume in Prob. IV.2 that the MDP is communicating. However, it is possible to generalize the results in this section to an MDP that is not communicating.

We limit our attention to stationary policies. We will show that, similar to the majority of problems in dynamic programming, there exist optimal stationary policies, 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 [22]).

Definition IV.3 (Proper Policies). *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 denote $J_{\mu} = [J_{\mu}(1), \dots, J_{\mu}(n)]^T$ where $J_{\mu}(i)$ is the ACPC in (3) 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 $J_{\mu}(i) = \infty$. We will first consider only proper policies.

Without loss of generality, we assume that $S_{\pi} = \{1, \dots, m\}$ (i.e., states $m+1, \dots, n$ are not in S_{π}). Given a proper policy μ , we obtain its transition matrix P_{μ} as described in Sec. IV-A. Our goal is to express J_{μ} in terms of P_{μ} , similar to (5) 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 $\tilde{P}(i, \mu, j)$. We can obtain

this probability for all $i \in S$ and $j \in S_{\pi}$ by the following proposition:

Proposition IV.4. $\tilde{P}(i, \mu, j)$ satisfies

$$\tilde{P}(i, \mu, j) = \sum_{k=m+1}^n P(i, \mu(i), k) \tilde{P}(k, \mu(k), j) + P(i, \mu(i), j). \quad (12)$$

Proof. From i , the next state can either be in S_{π} or not. The first term in the RHS of (12) 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 \tilde{P}_{μ} such that

$$\tilde{P}_{\mu}(i, j) = \begin{cases} \tilde{P}(i, \mu, j) & \text{if } j \in S_{\pi} \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

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

Using (12), we can express \tilde{P}_{μ} 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:

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

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

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

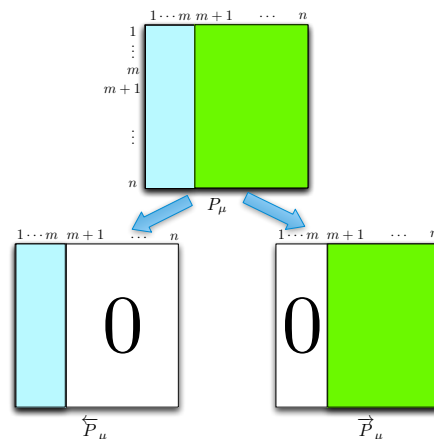


Fig. 1. The constructions of \overleftarrow{P}_{μ} and \overrightarrow{P}_{μ} from P_{μ} .

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 $j \in S_{\pi}$, matrix \overrightarrow{P}_{μ} is transient, i.e.,

$\lim_{k \rightarrow \infty} \vec{P}_\mu^k = 0$. From linear algebra (see, e.g., Ch. 9.4 of [24]), since \vec{P}_μ is transient and sub-stochastic, $I - \vec{P}_\mu$ is non-singular. ■

We can then write (12) as the following matrix equation:

$$\tilde{P}_\mu = \vec{P}_\mu \tilde{P}_\mu + \overleftarrow{P}_\mu. \quad (16)$$

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

$$\tilde{P}_\mu = (I - \vec{P}_\mu)^{-1} \overleftarrow{P}_\mu. \quad (17)$$

Note that (16) and (17) do not depend on the ordering of the states of \mathcal{M} , i.e., S_π does not need to be equal to $\{1, \dots, m\}$.

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

$$\tilde{g}(i, \mu) = \sum_{k=m+1}^n P(i, \mu(i), k) \tilde{g}(k, \mu) + g(i, \mu(i)). \quad (18)$$

Proof. The first term of RHS of (18) 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}_μ such that $\tilde{g}_\mu(i) = \tilde{g}(i, \mu)$, and note that (18) can be written as:

$$\begin{aligned} \tilde{g}_\mu &= \vec{P}_\mu \tilde{g}_\mu + g_\mu \\ \tilde{g}_\mu &= (I - \vec{P}_\mu)^{-1} g_\mu, \end{aligned} \quad (19)$$

where g_μ is defined in Sec. IV-A.

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

$$J_\mu = \limsup_{C \rightarrow \infty} \frac{1}{C} \sum_{k=0}^{C-1} \tilde{P}_\mu^k \tilde{g}_\mu, \quad (20)$$

where C represents the cycle count. Since \tilde{P}_μ is a stochastic matrix, the lim sup in (20) can be replaced by the limit, and we have

$$J_\mu = \lim_{C \rightarrow \infty} \frac{1}{C} \sum_{k=0}^{C-1} \tilde{P}_\mu^k \tilde{g}_\mu = \tilde{P}_\mu^* \tilde{g}_\mu, \quad (21)$$

where P^* for a stochastic matrix P is defined in (6).

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

$$J_\mu = J_{\tilde{\mu}}^s. \quad (22)$$

Moreover, we define $h_\mu := h_{\tilde{\mu}}^s$. 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 policies that can be mapped to a proper policy as $\mathbb{M}_{\tilde{\mu}}$. We see that a proper policy minimizing the ACPC maps to a policy over $\mathbb{M}_{\tilde{\mu}}$ minimizing the ACPS.

The by-product of the above analysis is that, if μ is proper, then $J_\mu(i)$ is finite for all i , since \tilde{P}_μ^* 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: $S_{\rightarrow\pi}$ is the set of states in S that cannot reach S_π and $S_{\leftarrow\pi}$ as the set of states that can reach S_π with positive probability. Since μ is improper and $g(i, u)$ is positive-valued, $S_{\rightarrow\pi}$ is not empty and $J_\mu(i) = \infty$ for all $i \in S_{\rightarrow\pi}$. Moreover, states in $S_{\rightarrow\pi}$ cannot visit $S_{\rightarrow\pi}$ by definition. Since \mathcal{M} is communicating, there exists some actions at some states in $S_{\rightarrow\pi}$ such that, if applied, all states in $S_{\rightarrow\pi}$ 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 $i \in S_{\rightarrow\pi}$. Therefore $J_{\mu'}(i) < J_\mu(i)$ for all $i \in S_{\rightarrow\pi}$. ■

Using the connection to the ACPS problem, we have:

Proposition IV.8. 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 (12), $\tilde{P}_\mu(i, j)$ 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 a single-chain MDP 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 (9) and (10), and in particular the case when the optimal cost is the same for all initial

states (11), policy $\tilde{\mu}^*$ with the ACPS gain-bias pair $(\lambda \mathbf{1}, h)$ satisfying for all $\tilde{\mu} \in \mathbb{M}_{\tilde{\mu}}$:

$$\lambda \mathbf{1} + h \leq g_{\tilde{\mu}} + P_{\tilde{\mu}} h \quad (23)$$

is optimal. Equivalently, μ^* that maps to $\tilde{\mu}^*$ is optimal over all proper policies. The following proposition shows that this policy is optimal over all policies in \mathbb{M} , stationary or not.

Proposition IV.9. *The proper policy μ^* that maps to $\tilde{\mu}^*$ satisfying (23) is optimal over \mathbb{M} .*

Proof. Consider a $M = \{\mu_1, \mu_2, \dots\}$ and assume it to be optimal. We first consider that M is stationary for each cycle, and the policy is μ_k for the k -th cycle. Among this type of policies, 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 $\{\tilde{\mu}_1, \tilde{\mu}_2, \dots\}$. Since the optimal policy of the ACPS is $\tilde{\mu}^*$ (stationary). Then we can conclude that if M is stationary in between cycles, then optimal policy for each cycle is μ^* and thus $M = \mu^*$.

Now we assume that M is not stationary for each cycle. 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 [22]. 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 $M = \mu^*$. The proof is complete. ■

Unfortunately, it is not clear how to find the optimal policy from (23) except 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 (9) and (10), so that the optimizations 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 $(J_{\tilde{\mu}}^s, h_{\tilde{\mu}}^s)$, in terms of μ . From (7), we have

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

By manipulating the above equations, we can now show that J_μ and h_μ can be expressed in terms of μ (analogous to (7)) instead of $\tilde{\mu}$ via the following proposition:

Proposition IV.10. *We have*

$$J_\mu = P_\mu J_\mu, \quad J_\mu + h_\mu = g_\mu + P_\mu h_\mu + \vec{P}_\mu J_\mu. \quad (24)$$

Moreover, we have

$$(I - \vec{P}_\mu) h_\mu + v_\mu = P_\mu v_\mu, \quad (25)$$

for some vector v_μ .

Proof. We start from (7):

$$J_\mu = P_{\tilde{\mu}} J_\mu, \quad J_\mu + h_\mu = g_{\tilde{\mu}} + P_{\tilde{\mu}} h_\mu. \quad (26)$$

For the first equation in (26), using (17), we have

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

For the second equation in (26), using (17) and (19), we have

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

Thus, (26) can be expressed in terms of μ as:

$$J_\mu = P_\mu J_\mu, \quad J_\mu + h_\mu = g_\mu + P_\mu h_\mu + \vec{P}_\mu J_\mu.$$

To compute J_μ and h_μ , we need an extra equation similar to (8). Using (8), we have

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

which completes the proof. ■

From Prop. IV.10, similar to the ACPS problem, (J_μ, h_μ, v_μ) can be solved together by 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.11. *The policy μ^* with gain-bias pair $(\lambda \mathbf{1}, h)$ satisfying*

$$\begin{aligned} \lambda + h(i) &= \min_{u \in U(i)} \\ \left[g(i, u) + \sum_{j=1}^n P(i, u, j) h(j) + \lambda \sum_{j=m+1}^n P(i, u, j) \right], \quad (27) \end{aligned}$$

for all $i = 1, \dots, n$, is the optimal policy minimizing (3) over all policies in \mathbb{M} .

Proof. The optimality condition (27) can be written as:

$$\lambda \mathbf{1} + h \leq g_\mu + P_\mu h + \vec{P}_\mu \lambda \mathbf{1}, \quad (28)$$

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 $a + c \leq b + c$.

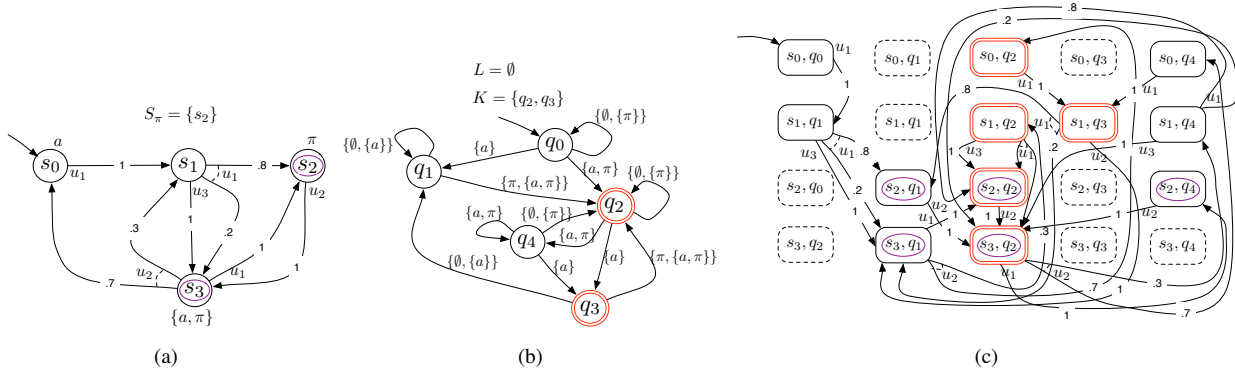


Fig. 2. The construction of the product MDP between a labeled MDP and a DRA. In this example, the set of atomic proposition is $\{a, \pi\}$. (a): A labeled MDP where the label on top of a state denotes the atomic propositions assigned to the state. The number on top of an arrow pointing from a state 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}_ϕ corresponding to the LTL formula $\phi = \square \diamond \pi \wedge \square \diamond a$. 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, 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 states of $K_{\mathcal{P}}$ are marked by double-strokes and states of $S_{\mathcal{P}\pi}$ are marked by ovals. The states with dashed borders are unreachable, and they are removed from $S_{\mathcal{P}}$.

From (28) we have

$$\begin{aligned}
\lambda \mathbf{1} + h &\leq g_\mu + P_\mu h + \vec{P}_\mu \lambda \mathbf{1} \\
\lambda \mathbf{1} - \vec{P}_\mu \lambda \mathbf{1} + h &\leq g_\mu + P_\mu h \\
\lambda \mathbf{1} - \vec{P}_\mu \lambda \mathbf{1} + h &\leq g_\mu + (\overleftarrow{P}_\mu + \vec{P}_\mu) h \\
\lambda \mathbf{1} - \vec{P}_\mu \lambda \mathbf{1} + h - \vec{P}_\mu h &\leq g_\mu + \overleftarrow{P}_\mu h \\
(I - \vec{P}_\mu)(\lambda \mathbf{1} + h) &\leq g_\mu + \overleftarrow{P}_\mu h \quad (29)
\end{aligned}$$

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

$$(I - \vec{P}_\mu)^{-1} = I + \vec{P}_\mu + \vec{P}_\mu^2 + \dots$$

Therefore, since all entries of \vec{P}_μ are non-negative, all entries of $(I - \vec{P}_\mu)^{-1}$ are non-negative. Thus, continuing from (29), we have

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

for all proper policies μ and all $\tilde{\mu} \in \mathbb{M}_{\tilde{\mu}}$. Hence, $\tilde{\mu}^*$ satisfies (23) and μ^* is optimal over all proper policies. Using Prop. IV.9, the proof is complete. ■

We will present an algorithm that uses Prop. IV.11 to find the optimal policy in the next section. Note that, unlike (23), (27) can be checked for any policy μ by finding the minimum for all states $i = 1, \dots, n$, which is significantly easier than searching over all proper policies.

V. SYNTHESIZING THE OPTIMAL POLICY UNDER LTL CONSTRAINTS

In this section we outline an approach for Prob. III.1. We aim for a computational framework, which in combination with results of [15] 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 as $\mathcal{R}_\phi = (Q, 2^{\Pi}, \delta, q_0, F)$ with $F = \{(L(1), K(1)), \dots, (L(M), K(M))\}$ where $L(i), K(i) \subseteq Q$ for all $i = 1, \dots, M$. We now obtain an MDP as the product of a labeled MDP \mathcal{M} and a DRA \mathcal{R}_ϕ , 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 labeled 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 obtained from a tuple $\mathcal{P} = (S_{\mathcal{P}}, U, P_{\mathcal{P}}, s_{\mathcal{P}0}, F_{\mathcal{P}}, S_{\mathcal{P}\pi}, g_{\mathcal{P}})$, where*

- (i) $S_{\mathcal{P}} = S \times Q$ is a set of states;
- (ii) U 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:

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

- (iv) $s_{\mathcal{P}0} = (s_0, q_0)$ is the initial state;
- (v) $F_{\mathcal{P}} = \{(L_{\mathcal{P}}(1), K_{\mathcal{P}}(1)), \dots, (L_{\mathcal{P}}(M), K_{\mathcal{P}}(M))\}$ where $L_{\mathcal{P}}(i) = S \times L(i)$, $K_{\mathcal{P}}(i) = S \times K(i)$, for $i = 1, \dots, M$;
- (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. After removing those states (via a simple graph search), \mathcal{P} is a valid MDP and is the desired product MDP. With a slight abuse of notation we still denote the product MDP as \mathcal{P} and always assume that unreachable states are removed. An example of a product MDP between a labeled MDP and a DRA corresponding to the LTL formula $\phi = \square \diamond \pi \wedge \square \diamond a$ is shown in Fig. 2.

There is an one-to-one correspondence between a path $s_0 s_1, \dots$ on \mathcal{M} and a path $(s_0, q_0)(s_1, q_1) \dots$ 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 $(s_0, q_0)(s_1, q_1) \dots$, the corresponding path $s_0 s_1 \dots$ 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}). *Given policy $M_{\mathcal{P}} = \{\mu_0^{\mathcal{P}}, \mu_1^{\mathcal{P}}, \dots\}$ on \mathcal{P} , where $\mu_k^{\mathcal{P}}((s, q)) \in U_{\mathcal{P}}((s, q))$, it induces policy $M = \{\mu_0, \mu_1, \dots\}$ 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 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 (3) 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 $(S_{\mathcal{C}}, U_{\mathcal{C}}, P_{\mathcal{C}}, K_{\mathcal{C}}, S_{\mathcal{C}\pi}, g_{\mathcal{C}})$ 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 $j \in S_{\mathcal{C}}$, in which case $P_{\mathcal{C}}(i, u, j) = P(i, u, j)$. An accepting maximal end component (AMEC) is the largest such end component such that $K_{\mathcal{C}} \neq \emptyset$ and $S_{\mathcal{C}} \cap L_{\mathcal{P}} = \emptyset$.*

Note that, an AMEC always contains at least one state in $K_{\mathcal{P}}$ and no state in $L_{\mathcal{P}}$. Moreover, it is ‘‘absorbing’’ in the sense that the state does not leave an AMEC once entered. 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 $F_{\mathcal{P}}$, and the states of this AMEC are shown in Fig. 3.

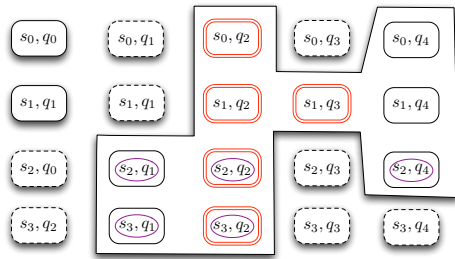


Fig. 3. The states of the only AMEC corresponding to the product MDP in Fig. 2.

A procedure to obtain all AMECs of an MDP was provided in [13]. From probabilistic model checking, a policy $M = M_{\mathcal{P}}|_{\mathcal{M}}$ almost surely satisfies ϕ (i.e., $M \in \mathbb{M}_{\phi}$) if and only if, under policy $M_{\mathcal{P}}$, there exists AMEC \mathcal{C} such that the probability of reaching $S_{\mathcal{C}}$ from initial state (s_0, q_0) is 1 (in this case, we call \mathcal{C} a *reachable* AMEC). In [15], such an optimal policy was found by dynamic programming or

solving a linear program. For states inside \mathcal{C} , since \mathcal{C} itself is a communicating MDP, a policy (not unique) can be easily constructed such that a state in $K_{\mathcal{C}}$ is infinitely often visited, satisfying the LTL specification.

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 policies obtained in [15]–[17] essentially disregard the behavior inside an AMEC, because, from the verification point of view, the behavior inside an AMEC is for the most part irrelevant, as long as a state in $K_{\mathcal{P}}$ 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 help obtaining a solution. Our approach consists of the following steps:

- (i) Convert formula ϕ to a DRA \mathcal{R}_{ϕ} and obtain the product MDP \mathcal{P} between \mathcal{M} and \mathcal{R}_{ϕ} ;
- (ii) Obtain the set of reachable AMECs, denoted as \mathcal{A} ;
- (iii) For each $\mathcal{C} \in \mathcal{A}$: Find a stationary policy $\mu_{\rightarrow \mathcal{C}}^*(i)$, defined for $i \in S \setminus S_{\mathcal{C}}$, that reaches $S_{\mathcal{C}}$ with probability 1 ($\mu_{\rightarrow \mathcal{C}}^*$ is guaranteed to exist and obtained as in [15]); Find a stationary policy $\mu_{\circlearrowleft \mathcal{C}}^*(i)$, defined for $i \in S_{\mathcal{C}}$ minimizing (3) for MDP \mathcal{C} and set $S_{\mathcal{C}\pi}$ while satisfying the LTL constraint; Define $\mu_{\mathcal{C}}^*$ to be:

$$\mu_{\mathcal{C}}^* = \begin{cases} \mu_{\rightarrow \mathcal{C}}^*(i) & \text{if } i \notin S_{\mathcal{C}} \\ \mu_{\circlearrowleft \mathcal{C}}^*(i) & \text{if } i \in S_{\mathcal{C}} \end{cases}, \quad (30)$$

and denote the ACPC of $\mu_{\circlearrowleft \mathcal{C}}^*$ as $\lambda_{\mathcal{C}}$;

- (iv) We find the solution to Prob. III.1 by:

$$J^*(s_0) = \min_{\mathcal{C} \in \mathcal{A}} \lambda_{\mathcal{C}}, \quad (31)$$

and the optimal policy is $\mu_{\mathcal{C}^*}^*|_{\mathcal{M}}$, where \mathcal{C}^* is the AMEC attaining the minimum in (31).

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

Proposition V.4. *For each $\mathcal{C} \in \mathcal{A}$, let $\mu_{\mathcal{C}}^*$ to be constructed as in (30), where $\mu_{\circlearrowleft \mathcal{C}}^*$ is a stationary policy satisfying two optimality conditions: (i) its ACPC gain-bias pair is $(\lambda_{\mathcal{C}} \mathbf{1}, h)$, where*

$$\lambda_{\mathcal{C}} + h(i) = \min_{u \in U_{\mathcal{C}}(i)} \left[g_{\mathcal{C}}(i, u) + \sum_{j \in S_{\mathcal{C}}} P(i, u, j) h(j) + \lambda_{\mathcal{C}} \sum_{j \notin S_{\mathcal{C}\pi}} P(i, u, j) \right], \quad (32)$$

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

Proof. Given $\mathcal{C} \in \mathcal{A}$, 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 1, (ii) $S \setminus S_C$ is not visited thereafter, and (iii) K_C is visited infinitely often. We see that, by the definition of AMECs, a policy almost surely satisfying ϕ belongs to $\mathbb{M}_C|_{\mathcal{M}}$ for a $C \in \mathcal{A}$. Thus, $\mathbb{M}_\phi = \cup_{C \in \mathcal{A}} \mathbb{M}_C|_{\mathcal{M}}$.

Since $\mu_C^*(i) = \mu_{\rightarrow C}^*(i)$ if $i \notin S_C$, the state reaches S_C with probability 1 and in a finite number of stages. We denote the probability that $j \in S_C$ is the first state visited in S_C when C is reached from initial state $s_{\mathcal{P}0}$ as $\tilde{P}_C(j, \mu_{\rightarrow C}^*, s_{\mathcal{P}0})$. Since the ACPC for the finite path from the initial state to a state $j \in S_C$ is 0 as the cycle index goes to ∞ , the ACPC from initial state $s_{\mathcal{P}0}$ under policy μ_C^* is

$$J(s_0) = \sum_{j \in S_C} \tilde{P}_C(j, \mu_{\rightarrow C}^*, s_{\mathcal{P}0}) J_{\mu_C^*}(j). \quad (33)$$

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

$$\begin{aligned} J(s_0) &= \sum_{j \in S_C} \tilde{P}_C(j, \mu_{\rightarrow C}^*, (s_0, q_0)) \lambda_C \\ &= \lambda_C. \end{aligned} \quad (34)$$

Applying Prop. IV.11, we see that $\mu_{\circ C}^*$ satisfies the optimality condition for MDP C with respect to set $S_{C\pi}$. Since there exists a state of K_C in each recurrent class of $\mu_{\circ C}^*$, a state in K_C is visited infinitely often and it satisfies the LTL constraint. Therefore, μ_C^* as constructed in (30) is optimal over \mathbb{M}_C and $\mu_C^*|_{\mathcal{M}}$ is optimal over $\mathbb{M}_C|_{\mathcal{M}}$ (due to equivalence of expected costs between $M_{\mathcal{P}}$ and $M_{\mathcal{P}}|_{\mathcal{M}}$). Since $\mathbb{M}_\phi = \cup_{C \in \mathcal{A}} \mathbb{M}_C|_{\mathcal{M}}$, we have that $J^*(s_0) = \min_{C \in \mathcal{A}} \lambda_C$ and the policy corresponding to C^* attaining this minimum is the optimal policy. ■

We can relax the optimality conditions for $\mu_{\circ C}^*$ in Prop. V.4 and require that there exist a state $i \in K_C$ in one recurrent class of $\mu_{\circ C}^*$. 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 [22, p. 203]). We can then use (30) to obtain the optimal policy μ_C^* for 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.4, for a given $C \in \mathcal{A}$. Note that Alg. 1 is similar in nature to policy iteration algorithms for ACPS problems.

Proposition V.5. *Given C , Alg. 1 terminates in a finite number of iterations. If it returns policy $\mu_{\circ C}$ with “optimal”, then $\mu_{\circ C}$ satisfies the optimality conditions in Prop. V.4. If C is unichain (i.e., each stationary policy of C contains one recurrent class), then Alg. 1 is guaranteed to return the optimal policy $\mu_{\circ C}^*$.*

Proof. If C is unichain, then since it is also communicating, $\mu_{\circ C}^*$ contains a single recurrent class (and no transient state). In this case, since K_C is not empty, states in K_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

Algorithm 1 : Policy iteration algorithm for ACPC

Input: $C = (S_C, U_C, P_C, K_C, S_{C\pi}, g_C)$

Output: Policy $\mu_{\circ C}$

- 1: Initialize μ^0 to a proper policy containing K_C in its recurrent classes (such a policy can always be constructed since C is communicating)
- 2: **repeat**
- 3: Given μ^k , compute J_{μ^k} and h_{μ^k} with (24) and (25)
- 4: Compute for all $i \in S_C$:

$$\bar{U}(i) = \arg \min_{u \in U_C(i)} \sum_{j \in S_C} P(i, u, j) J_{\mu^k}(j) \quad (35)$$

- 5: **if** $\mu^k(i) \in \bar{U}(i)$ for all $i \in S_C$ **then**
- 6: Compute, for all $i \in S_C$:

$$\begin{aligned} \bar{M}(i) = \arg \min_{u \in \bar{U}(i)} & \left[g_C(i, u) + \sum_{j \in S_C} P(i, u, j) h_{\mu^k}(j) \right. \\ & \left. + \sum_{j \notin S_{C\pi}} P(i, u, j) J_{\mu^k}(j) \right] \end{aligned} \quad (36)$$

- 7: Find μ^{k+1} such that $\mu^{k+1}(i) \in \bar{M}(i)$ for all $i \in S_C$, and contains a state of K_C in its recurrent classes. If one does not exist, **Return:** μ^k with “not optimal”
 - 8: **else**
 - 9: Find μ^{k+1} such that $\mu^{k+1}(i) \in \bar{U}(i)$ for all $i \in S_C$, and contains a state of K_C in its recurrent classes. If one does not exist, **Return:** μ^k with “not optimal”
 - 10: **end if**
 - 11: Set $k \leftarrow k + 1$
 - 12: **until** μ^k with gain-bias pair satisfying (32) and **Return:** μ^k with “optimal”
-

not assuming C to be unichain) is similar to the proof of convergence for the policy iteration algorithm for the ACPS problem (see [22, 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 (32) instead of the optimality conditions for the ACPS problem ((9) and (10)). ■

If we obtain the optimal policy for each $C \in \mathcal{A}$, then we use (31) to obtain the optimal solution for Prob. III.1. If for some 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{A} and use (31) 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.

Remark V.6 (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 [20] 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 $|S| \times |Q|$. 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 (35) and (36) for all i is also dominated by $|S_{\mathcal{C}}|^2$, for each iteration, the computational complexity is $O(|S_{\mathcal{C}}|^3)$.

VI. CASE STUDY

The algorithmic framework developed in this paper is implemented in MATLAB, and here we provide an example as a case study. Consider the MDP \mathcal{M} shown in Fig. 4, which can be viewed as the dynamics of a robot navigating in an environment with the set of atomic propositions $\{\text{pickup}, \text{dropoff}\}$. In practice, this MDP can be obtained via an abstraction process (see [1]) from the environment, where its probabilities of transitions can be obtained from experimental data or accurate simulations.

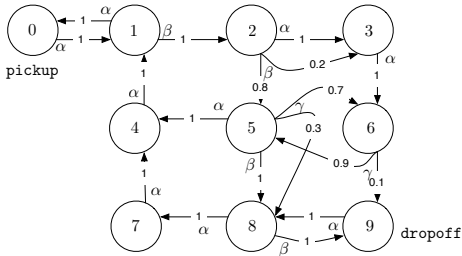


Fig. 4. MDP capturing a robot navigating in an environment. $\{\alpha, \beta, \gamma\}$ is the set of controls at states. The cost of applying α, β, γ at a state where the control is available is 5, 10, 1, respectively. (e.g., $g(i, \alpha) = 5$ if $\alpha \in U(i)$)

The goal of the robot is to continuously perform a pickup-delivery task. The robot is required to pick up items at the state marked by pickup (see Fig. 4), and drop them off at the state marked by dropoff. It is then required to go back to pickup and this process is repeated. This task can be written as the following LTL formula:

$$\phi = \square \diamond \text{pickup} \wedge \square (\text{pickup} \Rightarrow \bigcirc (\neg \text{pickup} \mathcal{U} \text{dropoff})).$$

The first part of ϕ , $\square \diamond \text{pickup}$, enforces that the robot repeatedly pick up items. The remaining part of ϕ ensures that new items cannot be picked up until the current items are dropped off. We denote pickup as the optimizing proposition, and the goal is to find a policy that satisfies ϕ with probability 1 and minimizes the expected cost in between visiting the pickup state (i.e., we aim to minimize the expected cost in between picking up items).

We generated the DRA \mathcal{R}_ϕ using the ltl2dstar tool [21] with 13 states and 1 pair $(L, K) \in F$. The product MDP \mathcal{P} after removing unreachable states contains 31 states (note that \mathcal{P} has 130 states without removing

unreachable states). There is one AMEC \mathcal{C} corresponding to the only pair in $F_{\mathcal{P}}$ and it contains 20 states. We tested Alg. 1 with a number of different initial policies and Alg. 1 produced the optimal policy within 2 or 3 policy updates in each case (note that \mathcal{C} is not unichain). For one initial policy, the ACPC was initially 330 at each state of \mathcal{C} , and it was reduced to 62.4 at each state when the optimal policy was found. The optimal policy is as follows:

State	0	1	2	3	4	5	6	7	8	9
After pickup	α	β	α	α	α	γ	γ	α	β	α
After dropoff	α	α	α	α	α	α	γ	α	α	α

The first row of the above table shows the policy after pick-up but before drop-off and the second row shows the policy after drop-off and before another pick-up.

VII. CONCLUSIONS

We have developed a method to automatically generate a control policy for a dynamical system modelled as a Markov Decision Process (MDP), in order to satisfy specifications given as Linear Temporal Logic formulas. The control policy satisfies the given specification almost surely, if such a policy exists. In addition, the policy optimizes the average 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 currently pursuing several future directions. First, we aim to solve the problem completely and find an algorithm that guarantees to always return the optimal policy. Second, we are interested to apply the optimization criterion of average cost per cycle to more complex models such as Partially Observable MDPs (POMDPs) and semi-MDPs.

REFERENCES

- [1] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta, ‘‘Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees,’’ in *Proc ICRA*, Anchorage, AK, 2010, pp. 3227 – 3232.
- [2] 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 *Proc AIAA GN&C*, Toronto, Canada, Aug. 2010.
- [3] R. Alterovitz, T. Siméon, and K. Goldberg, ‘‘The stochastic motion roadmap: A sampling framework for planning with Markov motion uncertainty,’’ in *RSS*, Atlanta, GA, Jun. 2007.
- [4] H. Kress-Gazit, G. Fainekos, and G. J. Pappas, ‘‘Where’s Waldo? Sensor-based temporal logic motion planning,’’ in *Proc ICRA*, Rome, Italy, 2007, pp. 3116–3121.
- [5] S. Karaman and E. Frazzoli, ‘‘Sampling-based motion planning with deterministic μ -calculus specifications,’’ in *Proc CDC*, Shanghai, China, 2009, pp. 2222–2229.
- [6] S. G. Loizou and K. J. Kyriakopoulos, ‘‘Automatic synthesis of multiagent motion tasks based on LTL specifications,’’ in *Proc CDC*, Paradise Island, Bahamas, 2004, pp. 153–158.
- [7] T. Wongpiromsarn, U. Topcu, and R. M. Murray, ‘‘Receding horizon temporal logic planning for dynamical systems,’’ in *Proc CDC*, Shanghai, China, 2009, pp. 5997–6004.
- [8] E. M. Clarke, D. Peled, and O. Grumberg, *Model checking*. MIT Press, 1999.
- [9] 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.

- [10] M. Kloetzer and C. Belta, "A fully automated framework for control of linear systems from temporal logic specifications," *IEEE Trans Automatic Ctrl*, vol. 53, no. 1, pp. 287–297, 2008.
- [11] M.Kloetzer and C. Belta, "Dealing with non-determinism in symbolic control," in *Hybrid Systems: Computation and Control*, ser. Lect. Notes Comp. Science, M. Egerstedt and B. Mishra, Eds. Springer Verlag, 2008, pp. 287–300.
- [12] L. De Alfaro, "Formal verification of probabilistic systems," Ph.D. dissertation, Stanford University, 1997.
- [13] C. Baier, J.-P. Katoen, and K. G. Larsen, *Principles of Model Checking*. MIT Press, 2008.
- [14] 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.
- [15] X. C. 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, Aug. 2011, to appear.
- [16] C. Courcoubetis and M. Yannakakis, "Markov decision processes and regular events," *IEEE Trans Automatic Ctrl*, vol. 43, no. 10, pp. 1399–1418, 1998.
- [17] C. Baier, M. Größer, M. Leucker, B. Bollig, and F. Ciesinski, "Controller synthesis for probabilistic systems," in *Proceedings of IFIP TCS'2004*. Kluwer, 2004.
- [18] S. L. Smith, J. Tůmová, C. Belta, and D. Rus, "Optimal path planning under temporal constraints," in *Proc IROS*, Taipei, Taiwan, Oct. 2010, pp. 3288–3293.
- [19] E. Gradel, W. Thomas, and T. Wilke, *Automata, logics, and infinite games: A guide to current research*, ser. Lect. Notes Comp. Science. Springer Verlag, 2002, vol. 2500.
- [20] 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.
- [21] J. Klein, "ltl2dstar - LTL to deterministic Streett and Rabin automata," <http://www.ltl2dstar.de/>, 2007, viewed September 2010.
- [22] D. Bertsekas, *Dynamic programming and optimal control, vol. II*. Athena Scientific, 2007.
- [23] M. L. Puterman, *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley and Sons, 1994.
- [24] L. Hogben, *Handbook of linear algebra*. CRC Press, 2007.