Optimal Control of MDPs with Temporal Logic Constraints

Mária Svoreňová, Ivana Černá and Calin Belta

Abstract—In this paper, we focus on formal synthesis of control policies for finite Markov decision processes with non-negative real-valued costs. We develop an algorithm to automatically generate a policy that guarantees the satisfaction of a correctness specification expressed as a formula of Linear Temporal Logic, while at the same time minimizing the expected average cost between two consecutive satisfactions of a desired property. The existing solutions to this problem are sub-optimal. By leveraging ideas from automata-based model checking and game theory, we provide an optimal solution. We demonstrate the approach on an illustrative example.

I. INTRODUCTION

Markov Decision Processes (MDP) are probabilistic models widely used in various areas, such as economics, biology, and engineering. In robotics, they have been successfully used to model the motion of systems with actuation and sensing uncertainty, such as ground robots [17], unmanned aircraft [21], and surgical steering needles [1]. MDPs are central to control theory [4], probabilistic model checking and synthesis in formal methods [3], [9], and game theory [13].

MDP control is a well studied area (see e.g., [4]). The goal is usually to optimize the expected value of a cost over a finite time (e.g., stochastic shortest path problem) or an average expected cost in infinite time (e.g., average cost per stage problem). Recently, there has been increasing interest in developing MDP control strategies from rich specifications given as formulas of probabilistic temporal logics, such as Probabilistic Computation Tree Logic (PCTL) and Probabilistic Linear Temporal Logic (PLTL) [12], [17]. It is important to note that both optimal control and temporal logic control problems for MDPs have their counterpart in game theory, we provide an optimal solution. We demonstrate the approach on an illustrative example.

II. PRELIMINARIES

A. MDP Control

Definition 1: A Markov decision process (MDP) is a tuple \( \mathcal{M} = (S, A, P, AP, L, g) \), where \( S \) is a non-empty finite set of states, \( A \) is a non-empty finite set of actions, \( P : S \times A \times S \rightarrow [0, 1] \) is a transition probability function.
such that for every state $s \in S$ and action $\alpha \in A$ it holds that $\sum_{s' \in S} P(s, \alpha, s') \in \{0, 1\}$. AP is a finite set of atomic propositions, $L : S \rightarrow 2^{AP}$ is a labeling function, and $g : S \times A \rightarrow \mathbb{R}_0^+$ is a cost function. An initialized Markov decision process is an MDP $M = (S, A, P, AP, L, g)$ with a distinctive initial state $s_{init} \in S$.

An action $\alpha \in A$ is called enabled in a state $s \in S$ if $\sum_{s' \in S} P(s, \alpha, s') = 1$. With a slight abuse of notation, $A(s)$ denotes the set of all actions enabled in a state $s$. We assume $A(s) \neq \emptyset$ for every $s \in S$.

A run of an MDP $M$ is an infinite sequence of states $\rho = s_0 s_1 \ldots \in S^\omega$ such that for every $i \geq 0$, there exists $\alpha_i \in A(s_i)$, $P(s_i, \alpha_i, s_{i+1}) > 0$. We use $\text{Run}_M(s)$ to denote the set of all runs of $M$ that start in a state $s \in S$. Let $\text{Run}_M = \bigcup_{s \in S} \text{Run}_M(s)$. A finite run $\sigma = s_0 \ldots s_n \in S^+$ of $M$ is a finite prefix of a run in $M$ and $\text{Run}_{\text{fin}}(\sigma)$ denotes the set of all finite runs of $M$ starting in a state $s \in S$. Let $\text{Run}_{\text{fin}} = \bigcup_{s \in S} \text{Run}_{\text{fin}}(s)$. The length $|\sigma| = n+1$ of a finite run $\sigma = s_0 \ldots s_n$ is also referred to as the number of stages of the run. The last state of $\sigma$ is denoted by $\text{last}(\sigma) = s_n$.

The word induced by a run $\rho = s_0 s_1 \ldots$ of $M$ is an infinite sequence $L(s_0) L(s_1) \ldots \in (2^{AP})^\omega$. Similarly, a finite run of $M$ induces a finite word from the set $(2^{AP})^+$.

**Definition 2:** Let $M = (S, A, P, AP, L, g)$ be an MDP. An end component (EC) of the MDP $M$ is an MDP $N = (S_N, A_N, P|_{N}, AP, L|_{N}, g|_{N})$ such that $\emptyset \neq S_N \subseteq S$, $\emptyset \neq A_N \subseteq A$. For every $s \in S_N$ and $\alpha \in A_N(s)$ it holds that $\{s' \in S \mid P(s, \alpha, s') > 0\} \subseteq S_N$. For every pair of states $s, s' \in S_N$, there exists a finite run $\sigma \in \text{Run}_{\text{fin}}(s)$ such that $\text{last}(\sigma) = s'$. We use $P|_{N}$ to denote the function $P$ restricted to the sets $S_N$ and $A_N$. Similarly, we use $L|_{N}$ and $g|_{N}$ with the obvious meaning. If the context is clear, we only use $P, L, g$ instead of $P|_{N}, L|_{N}, g|_{N}$.

The number of ECs of an MDP $M$ can be up to exponential in the number of states of $M$ and they can intersect. On the other hand, MECs are pairwise disjoint and every EC is contained in a single MEC. Hence, the number of MECs of $M$ is bounded by the number of states of $M$.

**Definition 3:** Let $M = (S, A, P, AP, L, g)$ be a MDP. A control strategy for $M$ is a function $C : \text{Run}_{\text{fin}}(s) \rightarrow A$ such that for every $\sigma \in \text{Run}_{\text{fin}}(s)$ it holds that $C(\sigma) \in A(\text{last}(\sigma))$.

A strategy $C$ for which $C(\sigma) = C(\sigma')$ for all finite runs $\sigma, \sigma' \in \text{Run}_{\text{fin}}(s)$ with $\text{last}(\sigma) = \text{last}(\sigma')$ is called memoryless. In that case, we consider $C$ to be a function $C : S \rightarrow A$. A strategy is called finite-memory if it is defined as a tuple $C = (M, \text{act}, \Delta, \text{start})$, where $M$ is a finite set of modes, $\Delta : M \times S \rightarrow M$ is a transition function, $\text{act} : M \times S \rightarrow A$ selects an action to be applied in $M$, and $\text{start} : S \rightarrow M$ selects the starting mode for every $s \in S$.

A run $\rho = s_0 s_1 \ldots \in \text{Run}_M(s)$ of an MDP $M$ is called a run under a strategy $C$ for $M$ if for every $i \geq 0$, it holds that $P(s_i, C(\rho(i)), s_{i+1}) > 0$. A finite run under $C$ is a finite prefix of a run under $C$. The set of all infinite and finite runs of $M$ under $C$ starting in a state $s \in S$ are denoted by $\text{Run}_M(s)$ and $\text{Run}_{\text{fin}}(s)$, respectively. Let $\text{Run}_M = \bigcup_{s \in S} \text{Run}_M(s)$ and $\text{Run}_{\text{fin}} = \bigcup_{s \in S} \text{Run}_{\text{fin}}(s)$.

Let $M$ be an MDP, $s$ a state of $M$, and $C$ a strategy for $M$. The following probability measure is used to argue about the possible outcomes of applying $C$ in $M$ starting from $s$. Let $\sigma \in \text{Run}_{\text{fin}}(s)$ be a finite run. A cylinder set $\text{Cyl}(\sigma)$ of $\sigma$ is the set of all runs of $M$ under $C$ that have $\sigma$ as a finite prefix. There exists a unique probability measure $Pr^M_C(\sigma)$ on the $\sigma$-algebra generated by the set of cylinder sets of all runs in $\text{Run}_{\text{fin}}(s)$. For $\sigma = s_0 \ldots s_n \in \text{Run}_{\text{fin}}(s)$, it holds

$$Pr^M_C(\text{Cyl}(\sigma)) = \prod_{i=0}^{n-1} P(s_i, C(\sigma(i)), s_{i+1})$$

and $Pr^M_C(\text{Cyl}(\sigma)) = 1$. Intuitively, given a subset $X \subseteq \text{Run}_{\text{fin}}(s)$, $Pr^M_C(X)$ is the probability that a run of $M$ under $C$ that starts in $s$ belongs to the set $X$.

The following properties hold for any MDP $M$ (see, e.g., [3]). For every EC $N$ of $M$, there exists a finite-memory strategy $C$ for $M$ such that $M$ under $C$ starting from any state of $N$ never visits a state outside $N$ and all states of $N$ are visited infinitely many times with probability 1. On the other hand, having any, finite-memory or not, strategy $C$, a state $s$ of $M$ and a run $\rho$ of $M$ under $C$ that starts in $s$, the set of states visited infinitely many times by $\rho$ forms an end component. Let $\text{EC} \subseteq \text{EC}(M)$ be the set of all ECs of $M$ that correspond, in the above sense, to at least one run under the strategy $C$ that starts in the state $s$. We say that the strategy $C$ leads $M$ from the state $s$ to the set $\text{EC}$.

**B. Linear Temporal Logic**

**Definition 4:** Linear Temporal Logic (LTL) formulae over a set AP of atomic propositions are formed according to the following grammar:

$$\phi ::= true \mid a \mid \neg \phi \mid \phi \land \phi \mid X \phi \mid \phi U \phi \mid G \phi \mid F \phi,$$

where $a \in AP$ is an atomic proposition, $\neg$ and $\land$ are standard Boolean connectives, and $X$ (next), $U$ (until), $G$ (always), and $F$ (eventually) are temporal operators.

Formulae of LTL are interpreted over the words from $(2^{AP})^\omega$, such as those induced by runs of an MDP $M$ (for details see, e.g., [3]). For example, a word $w \in (2^{AP})^\omega$ satisfies $G \phi$ if $\phi$ holds in $w$ always and eventually, respectively. If the word induced by a run $\rho \in \text{Run}_M$ satisfies a formula $\phi$, we say that the run $\rho$ satisfies $\phi$. With slight abuse of notation, we also use states or sets of states of the MDP as propositions in LTL formulae.

For every LTL formula $\phi$, the set of all runs of $M$ that satisfy $\phi$ is measurable in the probability measure $Pr^M_C$ for any $C$ and $s$ [3]. With slight abuse of notation, we use LTL formulae as arguments of $Pr^M_C$. If for a state $s \in S$ it holds that $Pr^M_C(\phi) = 1$, we say that the strategy $C$ almost-surely satisfies $\phi$ starting from $s$. If $M$ is an initialized MDP and $Pr^M_C(\phi) = 1$, we say that $C$ almost-surely satisfies $\phi$. 
The LTL control synthesis problem for an initialized MDP \( \mathcal{M} \) and an LTL formula \( \phi \) over \( \mathcal{A} \) aims to find a strategy for \( \mathcal{M} \) that almost-surely satisfies \( \phi \). This problem can be solved using principles from probabilistic model checking [3], [12]. The algorithm itself is based on the translation of \( \phi \) to a Rabin automaton and the analysis of an MDP that combines the Rabin automaton and the original MDP \( \mathcal{M} \).

**Definition 5:** A deterministic Rabin automaton (DRA) is a tuple \( \mathcal{A} = (Q, 2^{AP}, \delta, q_0, \text{Acc}) \), where \( Q \) is a non-empty finite set of states, \( 2^{AP} \) is an alphabet, \( \delta: Q \times 2^{AP} \rightarrow Q \) is a transition function, \( q_0 \in Q \) is an initial state, and \( \text{Acc} \subseteq 2^Q \times 2^{\delta} \) is an accepting condition.

A run of \( \mathcal{A} \) is a sequence \( q_0q_1 \ldots \in Q^\omega \) such that for every \( i \geq 0 \), there exists \( A_i \in 2^{AP} \), \( \delta(q_i, A_i) = q_{i+1} \). We say that the word \( A_0A_1 \ldots \in (2^{AP})^\omega \) induces the run \( q_0q_1 \ldots \). A run of \( \mathcal{A} \) is called accepting if there exists a pair \( (B, G) \in \text{Acc} \) such that the run visits every state from \( B \) only finitely many times and at least one state from \( G \) infinitely many times.

For every LTL formula \( \phi \) over \( \mathcal{A} \), there exists a DRA \( A_\phi \) such that all and only words from \( (2^{AP})^\omega \) satisfying \( \phi \) induce an accepting run of \( A_\phi \) [14]. For translation algorithms see e.g., [16], and their online implementations, e.g., [15].

**Definition 6:** Let \( \mathcal{M} = (S, A, P, AP, L, g) \) be an initialized MDP and \( \mathcal{A} = (Q, 2^{AP}, \delta, q_0, \text{Acc}) \) be a DRA. The product of \( \mathcal{M} \) and \( \mathcal{A} \) is the initialized MDP \( \mathcal{P} = (S_P, A_P, P_P, AP_P, L_P, g_P) \), where \( S_P = S \times Q \), \( P_P((s,q), \alpha, (s', q')) = P((s, \alpha, s'), q') \) if \( q' = \delta(q, L(s)) \) and 0 otherwise, \( A_P = Q \), \( L_P((s,q)) = q \), \( g_P((s,q), \alpha) = g(s, \alpha) \). The initial state of \( \mathcal{P} \) is \( s_{init}^{\mathcal{P}} = (s_{init}, q_0) \).

Using the projection on the first component, every (finite) run of \( \mathcal{P} \) projects to a (finite) run of \( \mathcal{M} \) and vice versa. For every (finite) run of \( \mathcal{M} \), there exists a (finite) run of \( \mathcal{P} \) that projects to it. Analogous correspondence exists between strategies for \( \mathcal{P} \) and \( \mathcal{M} \). It holds that the projection of a finite-memory strategy for \( \mathcal{P} \) is also finite-memory. More importantly, for the product \( \mathcal{P} \) of an MDP \( \mathcal{M} \) and a DRA \( A_\phi \) for an LTL formula \( \phi \), the probability of satisfying the accepting condition \( \text{Acc} \) of \( A_\phi \) under a strategy \( \mathcal{C} \) for \( \mathcal{P} \) starting from the initial state \( s_{init}^{\mathcal{P}} \), i.e.,

\[
\Pr_{s_{init}^{\mathcal{P}}}^{\mathcal{P}, \mathcal{C}} \left( \bigvee_{(B,G) \in \text{Acc}} (\text{FG}(\neg B) \land \text{GF} G) \right),
\]

is equal to the probability of satisfying the formula \( \phi \) in the MDP \( \mathcal{M} \) under the projected strategy \( C \) starting from the initial state \( s_{init} \).

**Definition 7:** Let \( \mathcal{P} = (S_P, A_P, P_P, AP_P, L_P, g_P) \) be the product of an MDP \( \mathcal{M} \) and a DRA \( \mathcal{A} \). An accepting end component (AEC) of \( \mathcal{P} \) is defined as an end component \( N = (S_N, A_N, P_P, AP_P, L_P, g_P) \) of \( \mathcal{P} \) for which there exists a pair \( (B, G) \) in the acceptance condition of \( \mathcal{A} \) such that \( L_P(S_N) \cap B = \emptyset \) and \( L_P(S_N) \cap G \neq \emptyset \). We say that \( N \) is accepting with respect to the pair \( (B, G) \). An AEC \( N' = (S_{N'}, A_{N'}, P_P, AP_P, L_P, g_P) \) is called maximal (MAEC) if there is no AEC \( N'' = (S_{N''}, A_{N''}, P_P, AP_P, L_P, g_P) \) such that \( N'' \neq N', S_N' \subseteq S_N, A_{N'}((s,q)) \subseteq A_{N''}((s,q)) \) for every \( (s,q) \in SP \) and \( N \) and \( N' \) are accepting with respect to the same pair. We use AEC(\( \mathcal{P} \)) and MAEC(\( \mathcal{P} \)) to denote the set of all accepting end components and maximal accepting end components of \( \mathcal{P} \), respectively.

Note that MAECs that are accepting with respect to the same pair are always disjoint. However, MAECs that are accepting with respect to different pairs can intersect.

From the discussion above it follows that a necessary condition for almost-sure satisfaction of the accepting condition \( \text{Acc} \) by a strategy \( \mathcal{C} \) for \( \mathcal{P} \) is that there exists a set \( \text{maec} \subseteq \text{MAEC}(\mathcal{P}) \) of MAECs such that \( \mathcal{C} \) leads the product from the initial state to maec.

### III. Problem Formulation

Consider an initialized MDP \( \mathcal{M} = (S, A, P, AP, L, g) \) and a specification given as an LTL formula \( \phi \) over AP of the form

\[
\phi = \varphi \land \text{GF } \pi_{sur},
\]

where \( \pi_{sur} \in \text{AP} \) is an atomic proposition and \( \varphi \) is an LTL formula over AP. Intuitively, a formula of such form states two partial goals — mission goal \( \varphi \) and surveillance goal \( \text{GF } \pi_{sur} \). To satisfy the whole formula the system must accomplish the mission and visit the surveillance states \( S_{sur} = \{ s \in S \mid \pi_{sur} \in L(s) \} \) infinitely many times.

The motivation for this form of specification comes from applications in robotics, where persistent surveillance tasks are often a part of the specification. Note that the form in Eq. (1) does not restrict the full LTL expressivity since every LTL formula \( \phi_1 \) can be translated into a formula \( \phi_2 \) of the form in Eq. (1) that is associated with the same set of runs of \( \mathcal{M} \). Explicitly, \( \phi_2 = \phi_1 \land \text{GF } \pi_{sur} \), where \( \pi_{sur} \) is such that \( \pi_{sur} \in L(s) \) for every state \( s \in S \).

In this work, we focus on a control synthesis problem, where the goal is to almost-surely satisfy a given LTL specification, while optimizing a long-term quantitative objective. The objective is to minimize the average expected cumulative cost between consecutive visits to surveillance states.

Formally, we say that every visit to a surveillance state completes a surveillance cycle. In particular, starting from the initial state, the first visit to \( S_{sur} \) completes the first surveillance cycle of a run. We use \( \sharp(\sigma) \) to denote the number of completed surveillance cycles in a finite run \( \sigma \) plus one.

For a strategy \( C \) for \( \mathcal{M} \), the cumulative cost in the first \( n \) stages of applying \( C \) to \( \mathcal{M} \) starting from a state \( s \in S \) is

\[
g_{M,C}(s, n) = \sum_{i=0}^{n} g(\sigma_{s,n}^M(i), C(\sigma_{s,n}^M(i))),
\]

where \( \sigma_{s,n}^M \) is the random variable whose values are finite runs of length \( n + 1 \) from the set \( \text{Run}_{s,n}^M \) (the probability of a finite run \( \sigma \) is \( \Pr_{s,n}^M(\text{Cyl}(\sigma)) \)). Note that \( g_{M,C}(s, n) \) is also a random variable. Finally, we define the average expected cumulative cost per surveillance cycle (ACPC) in the MDP \( \mathcal{M} \) under a strategy \( C \) as a function \( V_{M,C}: S \rightarrow \mathbb{R}^+_0 \) such that for a state \( s \in S \)

\[
V_{M,C}(s) = \lim_{n \to \infty} \frac{E \left( g_{M,C}(s, n) \right)}{\sharp(\sigma_{s,n}^M)},
\]
The problem we consider in this paper can be formally stated as follows.

**Problem 1:** Let \( M = (S, A, P, AP, L, g) \) be an initialized MDP and \( \phi \) be an LTL formula over AP of the form in Eq. (1). Find a strategy \( C \) for \( M \) such that \( C \) almost-surely satisfies \( \phi \) and, at the same time, \( C \) minimizes the ACPC value \( V_{M,C}(s_{init}) \) among all strategies almost-surely satisfying \( \phi \).

The above problem was recently investigated in [11]. However, the solution presented by the authors is guaranteed to find an optimal strategy only if every MAEC \( N \) of the product \( P \) of the MDP \( M \) and the DRA for the specification satisfies certain conditions (for details see [11]). In this paper, we present a solution to Problem 1 that always finds an optimal strategy if one exists. The algorithm is based on principles from probabilistic model checking [3] and game theory [5], whereas the authors in [11] mainly use results from dynamic programming [4].

In the special case when every state of \( M \) is a surveillance state, Problem 1 aims to find a strategy that minimizes the average expected cost per stage among all strategies almost-surely satisfying \( \phi \). The problem of minimizing the average expected cost per stage (ACPS) in an MDP, without considering any correctness specification, is a well studied problem in optimal control, see e.g., [4]. It holds that there always exists a stationary strategy that minimizes the ACPS value starting from the initial state. In our approach to Problem 1, we use techniques for solving the ACPS problem to find a strategy that minimizes the ACPC value.

**IV. Solution**

Let \( M = (S, A, P, AP, L, g) \) be an initialized MDP and \( \phi \) an LTL formula over AP of the form in Eq. (1). To solve Problem 1 for \( M \) and \( \phi \) we leverage ideas from game theory [5] and construct an optimal strategy for \( M \) as a combination of a strategy that ensures the almost-game theory [5] and construct an optimal strategy for \( M \) over \( \phi \) to find a strategy that minimizes the ACPC value starting from the initial state. In our approach to Problem 1, we use techniques for solving the ACPS problem to find a strategy that minimizes the ACPC value.

First, we find a set \( \text{maec}^* \) of MAECs of \( P \) and a strategy \( C_0 \) that leads \( P \) from the initial state to the set \( \text{maec}^* \). We require that \( C_0 \) and \( \text{maec}^* \) minimize the weighted average of the values \( V_N^V \) for \( N \in \text{maec}^* \). The strategy \( C_P \) applies \( C_0 \) from the initial state until \( P \) enters the set \( \text{maec}^* \).

Second, we solve the problem of how to control the product once a state of an MAEC \( N \in \text{maec}^* \) is visited. Intuitively, we combine two finite-memory strategies, \( C_{N}^V \) for the almost-sure satisfaction of the accepting condition \( \text{Acc} \) and \( C_{N}^V \) for maintaining the average expected cumulative cost per surveillance cycle. To satisfy both objectives, the strategy \( C_{P} \) is played in rounds. In each round, we first apply the strategy \( C_{N}^V \), then the strategy \( C_{N}^V \), for each (finite) number of steps.

**A. Finding an optimal set of MAECs**

Let \( \text{MAEC}(P) \) be the set of all MAECs of the product \( P \) that can be computed as follows. For every pair \((B, G) \in \text{Acc}\), we create a new MDP from \( P \) by removing all its states with label in \( B \) and the corresponding actions. For the new MDP, we use one of the algorithms in [10], [9], [7] to compute the set of all its MECs. Finally, for every MEC, we check whether it contains a state with label in \( G \).

In this section, the aim is to find a set \( \text{maec}^* \subseteq \text{MAEC}(P) \) and a strategy \( C_0 \) for \( P \) that satisfy conditions formally stated below. Since the strategy \( C_0 \) will only be used to enter the set \( \text{maec}^* \), it is constructed as a partial function.

**Definition 8:** A partial strategy \( \zeta \) for the MDP \( M \) is a partial function \( \zeta : S \rightarrow A \) or a subset \( \zeta \subseteq S \times A \). The set \( \text{Run}^{M}_{\text{init}} \) of runs of \( M \) under \( \zeta \) contains all infinite runs of \( M \) that follow \( \zeta \) and all those finite runs of \( M \) under \( \zeta \) for which \( \zeta ((\text{last}(\sigma))) \) is not defined. A finite run of \( M \) under \( \zeta \) is then a finite prefix of a run under \( \zeta \). The probability measure \( \text{Pr}_{M,\zeta}^{\text{init}} \) is defined in the same manner as in Sec. II-A. We also extend the semantics of LTL formulas to finite words. For example, a formula \( FG \phi \) is satisfied by a finite word in some non-empty suffix of the word \( \phi \) always holds.

The conditions on \( \text{maec}^* \) and \( C_0 \) are as follows. First, the partial strategy \( C_0 \) leads \( P \) to the set \( \text{maec}^* \), i.e.,

\[
\Pi_{\text{P,Set}^{\text{init}}}^{C_{0}}(\text{FG} (\bigcup_{N \in \text{maec}^*} S_{N})) = 1. \tag{2}
\]

Second, we require that \( \text{maec}^* \) and \( C_0 \) minimize the value

\[
\sum_{N \in \text{maec}^*} \text{Pr}_{\text{P,Set}^{\text{init}}}^{C_{0}}(\text{FG} S_{N}) \cdot V_{N}^{V}. \tag{3}
\]

The procedure to compute the optimal ACPC value \( V_{N}^{V} \) for an MAEC \( N \) of \( P \) is described in the next section. Assume we already computed this value for each MAEC of \( P \). The algorithm to find the set \( \text{maec}^* \) and partial strategy \( C_0 \) is based on an algorithm for stochastic shortest path (SSP) problem. The SSP problem is one of the basic optimization problems for MDPs. Given an initialized MDP and its state \( t \), the goal is to find a strategy under which the MDP almost-surely reaches the state \( t \), so called terminal state, while
minimizing the expected cumulative cost. If there exists at least one strategy almost-surely reaching the terminal state, then there exists a stationary optimal strategy. For details and algorithms see e.g., [4].

The partial strategy \( C_0 \) and the set \( \text{maec}^* \) are computed as follows. First, we create a new MDP \( \mathcal{P}' \) from \( \mathcal{P} \) by considering only those states of \( \mathcal{P} \) that can reach the set MAEC(\( \mathcal{P} \)) with probability 1 and their corresponding actions. The MDP \( \mathcal{P}' \) can be computed using backward reachability from the set MAEC(\( \mathcal{P} \)). If \( \mathcal{P}' \) does not contain the initial state \( \mathcal{ss}_\text{init} \), there exists no solution to Problem 1. Otherwise, we add a new state \( t \) and for every MAEC \( \mathcal{N} \in \text{MAEC}(\mathcal{P}') = \text{MAEC}(\mathcal{P}) \), we add a new action \( \alpha_0 \) to \( \mathcal{P}' \). From each state \( (s,q) \in S_N, \mathcal{N} \in \text{MAEC}(\mathcal{P}') \), we define a transition under \( \alpha_0 \) to \( t \) with probability 1 and set its cost to \( V_s \). All other costs in the MDP are set to 0. Finally, we solve the SSP problem for \( \mathcal{P}' \) and the state \( t \) as the terminal state. Let \( C_{\text{SSP}} \) be the resulting stationary optimal strategy for \( \mathcal{P}' \). For every \( (s,q) \in S_T \), we define \( C_0((s,q)) = C_{\text{SSP}}((s,q)) \) if the action \( C_{\text{SSP}}((s,q)) \) does not lead from \( (s,q) \) to \( t \), \( C_0((s,q)) \) is undefined otherwise. The set \( \text{maec}^* \) is the set of all MAECs \( \mathcal{N} \) for which there exists a state \( (s,q) \) such that \( C_{\text{SSP}}((s,q)) = \alpha_0 \).

**Proposition 1:** The set \( \text{maec}^* \) and the partial stationary strategy \( C_0 \) resulting from the above algorithm satisfy the conditions in Eq. (2) and Eq. (3).

**Proof:** Both conditions follow directly from the fact that the strategy \( C_{\text{SSP}} \) is an optimal solution to the SSP problem for \( \mathcal{P}' \) and \( t \).

---

**B. Optimizing ACP Value in an MAEC**

In this section, we compute the minimum ACP value \( V_N^* \) that can be attained in an MAEC \( \mathcal{N} \in \text{MAEC}(\mathcal{P}) \) and construct the corresponding strategy for \( \mathcal{N} \). Essentially, we reduce the problem of computing the minimum ACP value to the problem of computing the minimum ACPS value by reducing \( \mathcal{N} \) to an MDP such that every state of the reduced MDP is labeled with the surveillance proposition \( \pi_{\text{sur}} \).

Let \( \mathcal{N} = (S_N, A_N, P_P, AP_P, L_P, g_P) \) be an MAEC, and \( S_{\text{sur}} = (S_{\text{sur}}, A_{\text{sur}}, P_{\text{sur}}, AP_{\text{sur}}, L_P, g_{\text{sur}}) \) be an MDP with states \( (s,q) \in S_N \) and actions \( \pi_{\text{sur}} \in L_P((s,q)) \). Let \( S_{\text{sur}} \) denote the set of all such states in \( S_N \). We reduce \( \mathcal{N} \) to an MDP \( N_{\text{sur}} \):

\[
N_{\text{sur}} = (S_{\text{sur}}, A_{\text{sur}}, P_{\text{sur}}, AP_{\text{sur}}, L_P, g_{\text{sur}})
\]

using Alg. [1]. For the sake of readability, we use singleton states such as \( v \) instead of pairs such as \( (s,q) \) to denote the states of \( \mathcal{N} \). The MDP \( N_{\text{sur}} \) is constructed from \( \mathcal{N} \) by eliminating states from \( S_N \) until \( S_{\text{sur}} \) contains one by one in arbitrary order. The actions \( A_{\text{sur}} \) are partial stationary strategies for \( \mathcal{N} \) in which we remember all the states and actions we eliminated. Later we prove that the transition probability \( P_{\text{sur}}(v, \zeta, v') \) for states \( v, v' \in S_{\text{sur}} \) and an action \( \zeta \in A_{\text{sur}}(v) \) is the probability that in \( \mathcal{N} \) under the partial stationary strategy \( \zeta \), if we start from the state \( v \), the next state that will be visited from the set \( S_{\text{sur}} \) is the state \( v' \), i.e., the first surveillance cycle is completed by visiting \( v' \). The cost \( g_{\text{sur}}(v, \zeta) \) is the expected cumulative cost gained in \( \mathcal{N} \) using partial stationary strategy \( \zeta \) from \( v \) until we reach a state in \( S_{\text{sur}} \).

In Fig. [1] we demonstrate the reduction on an example using the notation introduced in Alg. [1]. On the left side, we see a part of an MAEC \( \mathcal{N} \) with five states and two actions. First, we build an MDP \( X = (S_X, A_X, P_X, AP_P, L_P, g_X) \) from \( \mathcal{N} \) by transforming every action of every state to a partial stationary strategy with a single pair given by the state and the action. The MDP \( X \) is used in the algorithm as an auxiliary MDP to store the current version of the reduced system. Assume we want to reduce the state \( v \). We consider all "incoming" and "outgoing" actions of \( v \) and combine them pairwise as follows. There is only one outgoing action from \( v \) in \( X \), namely \( \zeta \), and only one incoming action, namely action \( \zeta_{\text{old}} \) of state \( v_{\text{from}} \). Since \( \zeta \) and \( \zeta_{\text{old}} \) do not conflict as partial stationary strategies on any state of \( \mathcal{N} \), we merge them to create a new partial stationary strategy \( \zeta_{\text{new}} \) that is an action of \( v_{\text{from}} \). The transition probability \( P_X(v_{\text{from}}, \zeta_{\text{new}}, v_{\text{to}}) \) for a state \( v_{\text{to}} \) of \( X \) is computed as the sum of the transition probability \( P_X(v_{\text{from}}, \zeta_{\text{old}}, v_{\text{to}}) \) of transiting from \( v_{\text{from}} \) to \( v_{\text{to}} \) using the old action \( \zeta_{\text{old}} \) and the probability of entering \( v_{\text{to}} \) by first transiting from \( v_{\text{from}} \) to \( v \) using \( \zeta_{\text{old}} \) and from \( v \) eventually reaching \( v_{\text{to}} \) using \( \zeta \). The cost \( g_X(v_{\text{from}}, \zeta_{\text{new}}) \) is the expected cumulative cost gained starting from \( v_{\text{from}} \) by first applying action \( \zeta_{\text{old}} \) and if we transit to \( v \), applying \( \zeta \) until a state different from \( v \) is reached. Now that we considered every pair of an incoming and outgoing action of \( v \), the state \( v \) and its incoming and outgoing actions are reduced. The modified MDP \( X \) is depicted on the right side of Fig. [1].

**Proposition 2:** Let \( \mathcal{N} = (S_N, A_N, P_P, AP_P, L_P, g_P) \) be an MAEC and \( N_{\text{sur}} = (S_{\text{sur}}, A_{\text{sur}}, P_{\text{sur}}, AP_{\text{sur}}, L_P, g_{\text{sur}}) \) its reduction resulting from Alg. [1]. The minimum ACP value that can be attained in \( N_{\text{sur}} \) starting from any of its states is the same and we denote it \( V_{\text{sur}} \). There exists a stationary strategy \( C_{N_{\text{sur}}} \) for \( N_{\text{sur}} \) that attains this value regardless of the starting state in \( N_{\text{sur}} \). Both \( V_{\text{sur}} \) and \( C_{N_{\text{sur}}} \) can be computed as a solution to the ACPS problem for \( N_{\text{sur}} \). It holds that \( V_{\text{sur}} = V_{\text{sur}}^* \), and from \( C_{N_{\text{sur}}} \), one can construct a finite-memory strategy \( C_{\text{sur}} \) for \( N_{\text{sur}} \) that in every state \( s_{\text{sur}} \) in the terminal state \( \mathcal{N} \) attains the optimal ACP value \( V_{\text{sur}} \).

**Proof:** We prove the following correspondence between \( N \) and \( N_{\text{sur}} \). For every \( v, v' \in S_{\text{sur}} \) and \( \zeta \in A_{\text{sur}}(v) \), it holds that \( \zeta \) is a well-defined partial stationary strategy for \( \mathcal{N} \). The transition probability \( P_{\text{sur}}(v, \zeta, v') \) is the probability that in \( \mathcal{N} \) when applying \( \zeta \) starting from \( v \), the first surveillance cycle is completed by visiting \( v' \), i.e.,

\[
P_{\text{sur}}(v, \zeta, v') = Pr_{\text{sur}}(\zeta(X(\sim S_{\text{sur}} U \nu))).
\]

The cost \( g_{\text{sur}}(v, \zeta) \) is the expected cumulative cost gained in \( N \) when applying \( \zeta \) starting from \( v \) until the first surveillance cycle is completed. On the other hand, for every partial stationary strategy \( \zeta \) for \( \mathcal{N} \) such that

\[
Pr_{\text{sur}}(\zeta(F S_{\text{sur}})) = 1
\]

for some \( v \in S_{\text{sur}} \), there exists an action \( \zeta' \in A_{\text{sur}}(v) \) such that the action \( \zeta' \) corresponds to the partial stationary
strategy $\zeta$ in the above sense, i.e.,
\[
P_{\text{sur}}(v, \zeta', v') = \Pr_{v,\zeta'}^{N}(X(\neg S_{\text{sur}} \cup v'))
\]
for every $v' \in S_{\text{sur}}$, and the cost $g_{\text{sur}}(v, \zeta')$ is the expected cumulative cost gained in $N$ when we apply $\zeta'$ starting from $v$ until we reach a state in $S_{\text{sur}}$.

To prove the first part of the correspondence above, we prove the following invariant of Alg. 1. Let $X = (S_X, A_X, P_X, AP_p, LP_p, g_X)$ be the MDP from the algorithm after the initialization, before the first iteration of the while cycle. It is easy to see that all actions of $X$ are well-defined partial stationary strategies. For the transition probabilities, it holds that
\[
P_X(v_{from}, \zeta, v_{to}) = \Pr_{v,\zeta}^{N}(X(\neg S_X \cup v_{to}))
\]
for every $v_{from}, v_{to} \in S_X$ and $\zeta \in A_X(v_{from})$. The cost $g_X(v_{from}, \zeta)$ is the expected cumulative cost gained in $N$ starting from $v_{from}$ when applying $\zeta$ until we reach a state in $S_X$. We show that these conditions also hold after every iteration of the while cycle.

Let $X$ satisfy the conditions above and let $v \in S_X \setminus S_{\text{sur}}$. By removing the state $v$ from $S_X$, we obtain a new version of the MDP $X' = (S_X', A_X', P_{X'}, AP_p, LP_p, g_{X'})$. Note that $S_X' \cup \{v\} = S_X$. Let $v_{from} \in S_X'$ be a state of $X'$ and $\zeta_{\text{new}} \in A_{X'}(v_{from})$ be its action such that $\zeta_{\text{new}}$ has changed in the process of removing the state $v$. The action $\zeta_{\text{new}}$ is a well-defined partial stationary strategy because it must have been created as a union of an action $\zeta_{old}$ of $v_{from}$ and an action $\zeta$ of $v$, both from the previous version $X$, which do not conflict on any state from $S_X$.

Let $X'_{\zeta_{\text{old}}}$ denote the LTL formula $X(\neg S_X \cup v_{to})$. For a state $v_{to} \in S_X'$, we prove that
\[
P_{X'}(v_{from}, \zeta_{\text{new}}, v_{to}) = \Pr_{v_{from},\zeta_{\text{new}}}^{N}(X'_{\zeta_{\text{old}}} \cup \{v\})
\]
and then eventually reaching the state $v_{to}$ from $v$ using $\zeta$. This means
\[
\Pr_{v_{from},\zeta_{\text{new}}}^{N}(X'_{\zeta_{\text{old}}} \cup \{v\}) = \Pr_{v_{from}}^{N}(X'_{\zeta_{\text{old}}} \cup \{v\}) + \Pr_{v_{from}}^{N}(X'_{\zeta_{\text{old}}}) \cdot \Pr_{v}^{N}(\zeta) \cdot \Pr_{v}^{N}(v_{to})
\]
\[
= P_{X}(v_{from}, \zeta_{old}, v_{to}) + P_{X}(v_{from}, \zeta_{old}, v) \cdot \left( \sum_{i=0}^{\infty} P_{X}(v, \zeta, v) \cdot P_{X}(v, \zeta, v_{to}) \right)
\]
\[
= P_{X}(v_{from}, \zeta_{old}, v_{to}) + P_{X}(v_{from}, \zeta_{old}, v) \cdot \frac{P_{X}(v, \zeta, v_{to})}{1 - P_{X}(v, \zeta, v)}
\]
which is exactly as defined in Alg. 1.

Similarly, we prove that $g_{X'}(v_{from}, \zeta_{\text{new}})$ is the expected cumulative cost gained in $N$ starting from $v_{from}$ when applying $\zeta_{\text{new}}$ until we reach a state in $S_X'$. As $\zeta_{\text{new}} = \zeta_{old} \cup \zeta$, it is the expected cumulative cost of reaching a state in $S_X$ by using $\zeta_{old}$ plus, in the case we reach $v$, the expected cumulative cost of eventually reaching a state in $S_X'$, i.e., other than $v$, using $\zeta$. To be specific, we have
\[
g_{X}(v_{from}, \zeta_{old}) + g_{X}(v_{from}, \zeta_{old}, v) \cdot \left( \sum_{i=0}^{\infty} P_{X}(v, \zeta, v) \cdot (1 - P_{X}(v, \zeta, v)) \cdot (i + 1) \cdot g_{X}(v, \zeta) \right)
\]
\[
= g_{X}(v_{from}, \zeta_{old}) + g_{X}(v_{from}, \zeta_{old}, v) \cdot \frac{g_{X}(v, \zeta)}{1 - P_{X}(v, \zeta, v)},
\]
just as defined in Alg. 1. This completes the proof of the first part of the correspondence between $N$ and $N_{\text{sur}}$.

The second part of the correspondence between $N$ and $N_{\text{sur}}$ follows directly from the fact that, in the process of removing a state $v \in S_X \setminus S_{\text{sur}}$, we consider all combinations of actions of $v$ which eventually reach a state different from $v$, with all actions of all states $v_{from}$ having an action under which $v$ is reached with non-zero probability.

From the correspondence between $N$ and $N_{\text{sur}}$ it follows that in $N_{\text{sur}}$, there exists a finite run between every two states. Therefore, the minimum ACPC value that can be obtained in $N_{\text{sur}}$ from any of its states is the same and it is denoted by $V_{N_{\text{sur}}}$. Since every state of $N_{\text{sur}}$ is a surveillance
Algorithm 1 Reduction of an MAEC $\mathcal{N}$ to $\mathcal{N}_{\text{sur}}$

**Input:** $\mathcal{N} = (S_N, A_N, P_N, AP_N, L_P, g_P)$

**Output:** $\mathcal{N}_{\text{sur}} = (S_{\text{sur}}, A_{\text{sur}}, P_{\text{sur}}, AP_{\text{sur}}, L_P, g_P)$

1. Let $X = (S_X, A_X, P_X, AP_X, L_P, g_X)$ be an MDP where
   - $S_X := S_N$
   - for $v \in S_X$ :
     - $A_X(v) := \{\zeta \mid \zeta_a = \{(v, \alpha)\}, \alpha \in A_N(v)\}$
     - for $v, v' \in S_X, \zeta \in A_X$ :
       - $P_X(v, \zeta, v') := P_P(v, \zeta(v), v')$
     - for $v \in S_X, \zeta \in A_X$ :
       - $g_X(v, \zeta) := g_P(v, \zeta(v))$

2. while $S_X \setminus S_{\text{sur}} \neq \emptyset$ do
3.  let $v \in S_X \setminus S_{\text{sur}}$
4.  for all $\zeta \in A_X(v)$ do
5.   if $P_X(v, \zeta, v) < 1$ then
6.     for all $v_{\text{from}} \in S_X, \zeta_{\text{old}} \in A_X(v_{\text{from}})$ do
7.       if $P_X(v_{\text{from}}, \zeta_{\text{old}}, v) > 0$ and $\zeta_{\text{new}} \neq \zeta$ do not conflict for any state from $S_X$ then
8.       $\zeta_{\text{new}} := \zeta_{\text{old}} \cup \zeta$
9.       add $\zeta_{\text{new}}$ to $A_X(v_{\text{from}})$
10. for every $v_{\text{to}} \in S_X$ :
11.    $P_X(v_{\text{from}}, \zeta_{\text{new}}, v_{\text{to}}) := P_X(v_{\text{from}}, \zeta_{\text{old}}, v_{\text{to}}) + P_X(v_{\text{from}}, \zeta_{\text{old}}, v) P_X(v, \zeta, v_{\text{to}})$
12.  end if
13. end for
14. end if
15. remove $\zeta$ from $A_X(v)$
16. end for
17. remove $v$ from $S_X$
18. end while
19. return $X$

The strategy $C_N^V$ is a then finite-memory strategy

$$C_N^V = (M, \text{act}, \Delta, \text{start}),$$

where $M = S_{\text{sur}} \cup \{\text{init}\}$ is the set of modes, $\Delta: M \times S_N \rightarrow M$ is the transition function such that for every $m \in M, v \in S_N$

$$\Delta(m, v) = \begin{cases} 
    m & \text{if } v \notin S_{\text{sur}}, \\
    v & \text{otherwise.}
\end{cases}$$

The function act: $M \times S_N \rightarrow A_N$ that selects an action to be applied in $N$ is for $m \in M, v \in S_N$ defined as

$$\text{act}(m, v) = \begin{cases} 
    C_N^V(m, v) & \text{if } m \in S_{\text{sur}} \\
    \zeta_{\text{init}}(v) & \text{otherwise.}
\end{cases}$$

Finally, start: $S_N \rightarrow S_{\text{sur}}$ selecting the starting mode for $v \in S_N$ is defined as

$$\text{start}(v) = \begin{cases} 
    v & \text{if } v \in S_{\text{sur}}, \\
    m & \text{where } C_N^V(m, v) \text{ is defined,} \\
    \text{init} & \text{otherwise.}
\end{cases}$$

The strategy attains the ACPC value $V_N$ since it only simulates the strategy $C_N^V$ by unwrapping the corresponding partial strategies.

The following property of the strategy $C_N^V$ is crucial for the correctness of our approach to Problem 1.

**Proposition 3:** For every $(s, q) \in S_N$, it holds that

$$\lim_{n \rightarrow \infty} P_{(s, q)}^N C_N^V(\{\rho \mid \frac{g_P(\rho(t^n))}{l} \leq V_N^*\}) = 1,$$

where $g_P(\rho(t^n))$ denotes the cumulative cost gained in the first $n$ surveillance cycles of a run $\rho \in \text{Run}_{\text{sur}}^N((s, q))$. Hence, for every $\epsilon > 0$, there exists $j(\epsilon) \in \mathbb{N}$ such that if the strategy $C_N^V$ is applied from a state $(s, q) \in S_N$ for any $l \geq j(\epsilon)$ surveillance cycles, then the average expected cumulative cost per surveillance cycle in these $l$ surveillance cycles is at most $V_N^* + \epsilon$ with probability at least $1 - \epsilon$, i.e.,

$$P_{(s, q)}^N C_N^V(\{\rho \mid \frac{g_P(\rho(t^n))}{l} \leq V_N^* + \epsilon\}) \geq 1 - \epsilon.$$  

**Proof:** In [7] the authors prove that a strategy solving the ACPS problem for an MDP satisfies a property analogous to the one in the proposition. Especially, for the strategy $C_{\text{sur}}^V$ for the reduced MDP $N_{\text{sur}}$, it holds that for any state $(s, q) \in S_{\text{sur}}$

$$\lim_{n \rightarrow \infty} P_{(s, q)}^N C_{\text{sur}}^V(\{\rho \mid \frac{g_{\text{sur}}(\rho(t^n))}{n} \leq V_{\text{sur}}^*\}) = 1,$$

where $g_{\text{sur}}(\rho(t^n))$ denotes the cumulative cost gained in the first $n$ stages of a run $\rho \in \text{Run}_{\text{sur}}^N((s, q))$. The proposition then follows directly from the construction of the strategy $C_N^V$ from the strategy $C_N^V$.
C. Almost-sure acceptance in an MAEC

Here we design a strategy for an MAEC $N' \in \text{MAEC}(\mathcal{P})$ that guarantees almost-sure satisfaction of the acceptance condition $Acc$ of $A_0$. Let $(B, G)$ be a pair in $Acc$ such that $N'$ is accepting with respect to $(B, G)$, i.e., $L_{P}(S_N)' \cap B = \emptyset$ and $L_{P}(S_N)' \cap G \neq \emptyset$. There exists a stationary strategy $C_{N'}^\phi$ for $N'$ under which a state with label in $G$ is reached with probability 1 regardless of the starting state, i.e.,

$$P_{(s,q)}^{N,C_{N'}^\phi}(F \mathit{G}) = 1$$

for every $(s,q) \in S_N'$. The existence of such a strategy follows from the fact that $N'$ is an EC [3]. Moreover, we construct $C_{N'}^\phi$ to minimize the expected cumulative cost before reaching a state in $S_N' \cap S \times G$.

The strategy $C_{N'}^\phi$ is found as follows. Let $N'$ be an MDP that is created from $N$ by adding a new state $t$ and a new action $a_t$. From every state $(s,q) \in S_N' \cap S \times G$, we define a new transition under $a_t$ to $t$ with probability 1 and cost 0. Let $C_{SSP}$ be a stationary optimal strategy for the SSP problem for $N'$ and $t$ as the terminal state. Let $\alpha_G$ be a pair from the accepting condition of $N' \in \text{maec}$. The existence of such a strategy follows from the fact that $N'$ is an EC [3]. Moreover, we construct $C_{N'}^\phi$ to minimize the expected cumulative cost before reaching a state in $S_N' \cap S \times G$.

The strategy $C_{N'}^\phi$ is found as follows. Let $N'$ be an MDP that is created from $N$ by adding a new state $t$ and a new action $a_t$. From every state $(s,q) \in S_N' \cap S \times G$, we define a new transition under $a_t$ to $t$ with probability 1 and cost 0. Let $C_{SSP}$ be a stationary optimal strategy for the SSP problem for $N'$ and $t$ as the terminal state. For a state $(s,q) \in S_N'$, we define $C_{N'}^\phi((s,q)) = C_{SSP}((s,q))$ if the state $(s,q)$ does not have a label in $G$, otherwise $C_{N'}^\phi((s,q)) = \alpha$ for some $\alpha \in \Delta_N((s,q))$.

**Proposition 4:** The strategy $C_{N'}^\phi$ for $N'$ resulting from the above algorithm almost-surely reaches the set $S_N' \cap S \times G$ and minimizes the expected cumulative cost before reaching the set, regardless of the initial state.

**Proof:** It follows directly from the fact that $C_{SSP}$ optimally solves the SSP problem for the MDP $N'$ and $t$.

D. Optimal strategy for $P$

Finally, we are ready to construct the strategy $C_P$ for the product $P$ that projects to an optimal solution for $M$.

First, starting from the initial state $s_{P\text{init}}$, $C_P$ applies the strategy $C_0$ resulting from the algorithm described in Sec. IV until a state of an MAEC in the set $\text{maec}^*$ is reached. Let $N' \in \text{maec}^*$ denote the MAEC and let $(B, G) \in \text{Acc}$ be a pair from the accepting condition of $A_0$ such that $N'$ is accepting with respect to $(B, G)$.

Now, the strategy $C_P$ starts to play the rounds. Each round consists of two phases. First, play the strategy $C_{N'}^\phi$ from Sec. IV until a state with label in $G$ is reached. Let us denote $k_{i}$ the number of steps we play $C_{N'}^\phi$ in $i$-th round. The second phase applies the strategy $C_{N'}^\phi$ from Sec. IV until the number of completed surveillance cycles in the second phase of the current round is $l_{i}$. The number $l_{i}$ is any natural number for which

$$l_{i} \geq \max\{j(\frac{1}{i}), i \cdot k_{i} \cdot g_{P\text{max}}\},$$

where $j(\frac{1}{i})$ is from Prop. 3 and $g_{P\text{max}}$ is the maximum value of the costs $g_P$. After applying the strategy $C_{N'}^\phi$ for $l_{i}$ surveillance cycles, we proceed to the next round $i + 1$.

**Theorem 1:** The strategy $C_P$ almost-surely satisfies the accepting condition $Acc$ of $A_0$ and at the same time, $C_P$ minimizes the ACPC value $V_{P,C_P}(s_{P\text{init}})$ among all strategies for $P$ almost-surely satisfying $Acc$.

**Proof:** From Prop. 4 it follows that when applying the strategy $C_0$ from the initial state $s_{P\text{init}}$, the set $\text{maec}^*$ is reached with probability 1.

Assume that $P$ enters MAEC $N' \in \text{maec}^*$ that is accepting with respect to a pair $(B, G) \in \text{Acc}$. Let $i$ be the current round of $C_P$ and $\epsilon_i = \frac{1}{i}$. According to Prop. 4 a state with a label in $G$ is almost-surely reached. In addition, using Prop. 3 the average expected cumulative cost per surveillance cycle in the $i$-th round is at most

$$k_{i} \cdot g_{P\text{max}} + l_{i}(V_{N'}^* + \epsilon_{i}) =$$

$$= V_{N'}^* + \epsilon_{i} + \frac{k_{i} \cdot g_{P\text{max}}}{l_{i}}$$

$$\leq V_{N'}^* + \epsilon_{i} + \frac{1}{i}$$

$$(l_{i} \geq i \cdot k_{i} \cdot g_{P\text{max}})$$

with probability at least $1 - \frac{1}{i}$. Therefore, in the limit, in the MAEC $N'$, we both satisfy the LTL specification and reach the optimal ACPC value with probability 1. Together with the fact that $\text{maec}^* \cap C_0$ satisfy the condition in Eq. (3), we have that $C_P$ is an optimal strategy for $P$.

E. Complexity and discussion

The size of a Rabin automaton for an LTL formula $\phi$ is in the worst case doubly exponential in the size of the set AP. However, studies such as [16] show that in practice, for many LTL formulas, automata are much smaller and manageable.

Once the product $P$ is built, we compute the set MAEC($P$) by running $|\text{Acc}|$-times an algorithm for ME decomposition, which is polynomial in the size of $P$. The size of the set MAEC($P$) is in the worst case $|\text{Acc}| \cdot |S_P|$. For each MAEC $N'$, we compute its reduction $N_{\text{ar}}$ using Alg. 1 in time $O(|S_{N'}| \cdot |A_N|^{O(|S_{N'}|)})$. The optimal ACPC value $V_{N'}^*$ and an optimal finite-memory strategy $C_{N'}^\phi$ are then found in time polynomial in the size of the reduced MDP.

The algorithm for finding the strategy $C_0$ and the optimal set $\text{maec}^*$ are again polynomial in the size of $P$. Similarly, computing a stationary strategy $C_{N'}^\phi$ for an MAEC $N' \in \text{maec}^*$ is polynomial in the size of $N'$.

As was proved in Sec. IV-D, the presented solution to Problem 1 is correct and complete. However, the resulting optimal strategy $C_P$ for $P$, and hence the projected strategy $C$ for $M$ as well, is not a finite-memory strategy in general. The reason is that in the second phase of every round $i$, the strategy $C_{N'}^\phi$ is applied for $l_{i}$ surveillance cycles and $l_{i}$ is generally growing with $i$.

This, however, does not prevent the solution to be effectively used. The following simple rule can be applied to avoid performing all $l_{i} \geq \max\{j(\frac{1}{i}), i \cdot k_{i} \cdot g_{P\text{max}}, j(\frac{1}{i})\}$ surveillance cycles in every round $i$. When the computation is in the second phase of round $i$ and the product is in an MAEC $N' \in \text{maec}^*$, after completion of every surveillance cycle, we can check whether the average cumulative cost per surveillance cycle in round $i$ is at most $V_{N'}^* + \frac{1}{i}$. If yes, we can proceed to the next round $i + 1$, otherwise continue...
with the second phase of round $i$. As the simulation results in Sec. IV show, the use of this simple rule dramatically decreases the number of performed surveillance cycles in almost every round.

On the other hand, the complexity of the resulting strategy $C$ for $\mathcal{M}$ can be reduced from non-finite-memory to finite-memory in the following case. Assume that for every $N \in \text{maec}^*$, the optimal ACPC strategy $C_N^V$ leads to an EC that contains a state from $G$, where $N$ is accepting with respect to the pair $(B, G) \in \text{Acc}$. In this case, the optimal strategy $C_{\mathcal{P}}$ can be defined as a finite-memory strategy that first applies the strategy $C_0$ to reach a state of an MAEC $N \in \text{maec}^*$, and from that point on, only applies the strategy $C_N^V$.

V. CASE STUDY

We implemented the solution presented in Sec. IV in Java and applied it to a persistent surveillance robotics example [20]. In this section, we report on the simulation results.

Consider a mobile robot moving in a partitioned environment. The motion of the robot is modeled by the initialized MDP $\mathcal{M}$ shown in Fig. 2a. The set AP of atomic propositions contains two propositions base and job. As depicted in Fig. 2a, state 0 is the base location and state 8 is the job location. At the job location, the robot performs some work, and at the base, it reports on its job activity.

The robot’s mission is to visit both base and job location infinitely many times. In addition, at least one job must be performed after every visit of the base, before the base is visited again. The corresponding LTL formula is $\phi = \text{GF base} \land \text{GF job} \land G (\text{base} \Rightarrow X(\neg \text{base} \lor \text{job})))$.

While satisfying the formula, we want to minimize the expected average cost between two consecutive jobs, i.e., the surveillance proposition $\pi_{\text{sur}} = \text{job}$.

In the simulation, we use a Rabin automaton $A_\phi$ for the formula that has 5 states and the accepting condition contains 1 pair. The product $\mathcal{P}$ of the MDP $\mathcal{M}$ and $A_\phi$ has 50 states and one MAEC $N$ of 19 states. The optimal set of MAECs $\text{maec}^* = \{N\}$. The optimal ACPC value $V_N^\phi = 40.5$. In Fig. 2b, we list the projections of strategies $C_0, C_N^V, C_N^V$ for $\mathcal{P}$ to strategies $C_{\text{init}}, C_{p1}, C_{p2}$ for $\mathcal{M}$, respectively. The optimal strategy $C$ for $\mathcal{M}$ is then defined as follows. Starting from the initial state 0, apply strategy $C_{\text{init}}$ until a state is reached, where $C_{\text{init}}$ is no longer defined. Start round number 1. In $i$-th round, proceed as follows. In the first phase of the round, apply strategy $C_{p1}$ until the base is reached and then for one more step (the product $\mathcal{P}$ has to reach a state from the Rabin pair). Let $k_i$ denote the number of steps in the first phase of round $i$. In the second phase, use strategy $C_{p2}$ for $l_i = \max\{i \cdot k_i, 10, j(\frac{1}{4})\}$ surveillance cycles, i.e., until the number of jobs performed by the robot is $l_i$. We also use the rule described in Sec. IV-E to shorten the second phase, if possible.

Let us summarize the statistical results we obtained for 5 executions of the strategy $C$ for $\mathcal{M}$, each of 100 rounds. The number $k_i$ of steps in the first phase of a round $i > 1$ was always 5 because in such case, the first phase starts at the job location and the strategy $C_{p1}$ needs to be applied for exactly 4 steps to reach the base. Therefore, in every round $i > 1$, the number $l_i$ is at least 50 $i$, e.g., in round 100, $l_i \geq 5000$. However, using the rule described in Sec. IV-E the average number of jobs per round was 130 and the median was only 14. In particular, the number was not increasing with the round. On the contrary, it appears to be independent from the history of the execution. In addition, at most 2 rounds in each of the executions finished only at the point, when the number of jobs performed by the robot in the second phase reached $l_i$. The average ACPC value attained after 100 rounds was 40.56.

In contrast to our solution, the algorithm proposed in [11] does not find an optimal strategy for $\mathcal{M}$. Regardless of the initialization of the algorithm, it always results in a sub-optimal strategy, namely the strategy $C_{p1}$ from Fig. 2b that has ACPC value 50.5.

VI. CONCLUSION

In this paper, we focus on the problem of designing a control strategy for an MDP with surveillance task, and at the same time, to minimize the expected average cumulative cost between visits of surveillance states. This problem was previously
addressed in [11], where the authors propose a sub-optimal solution based on dynamic programming. In contrast to this work, we exploit recent results from theoretical computer science, namely game theory and probabilistic model checking, to provide a sound and complete solution to this control problem.

REFERENCES

[1] 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. Citeseer, 2007.

[2] K. Apt and E. Grädel. Lectures in Game Theory for Computer Scientists. Cambridge University Press, 2011.

[3] C. Baier and J. Katoen. Principles of model checking. The MIT Press, 2008.

[4] D. Bertsekas. Dynamic Programming and Optimal Control, vol. II. Athena Scientific Optimization and Computation Series. Athena Scientific, 2007.

[5] K. Chatterjee and L. Doyen. Energy and Mean-Payoff Parity Markov Decision Processes. In Mathematical Foundations of Computer Science 2011, volume 6907 of Lecture Notes in Computer Science, pages 206–218. Springer Berlin Heidelberg, 2011.

[6] K. Chatterjee and L. Doyen. Games and Markov Decision Processes with Mean-Payoff Parity and Energy Parity Objectives. In Mathematical and Engineering Methods in Computer Science, volume 7119 of Lecture Notes in Computer Science, pages 37–46. Springer Berlin Heidelberg, 2012.

[7] K. Chatterjee and M. Henzinger. Faster and Dynamic Algorithms for Maximal End-Component Decomposition and Related Graph Problems in Probabilistic Verification. In Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA’11, pages 1318–1336, 2011.

[8] Y. Chen, J. Tumova, and C. Belta. LTL robot motion control based on automata learning of environmental dynamics. In IEEE International Conference on Robotics and Automation, ICRA’12, pages 5177–5182, 2012.

[9] C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. Journal of the ACM, 42(4):857–907, July 1995.

[10] L. de Alfaro. Formal Verification of Probabilistic Systems. PhD thesis, Stanford University, 1997. Technical report STAN-CS-TR-98-1601.

[11] X. Ding, S. Smith, C. Belta, and D. Rus. MDP Optimal Control under Temporal Logic Constraints. In The 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), pages 532 –538, dec. 2011.

[12] X. Ding, S. L. Smith, C. Belta, and D. Rus. LTL Control in Uncertain Environments with Probabilistic Satisfaction Guarantees. In Proceedings of the 18th IFAC World Congress, volume 18, 2011.

[13] J. Filar and K. Vrieze. Competitive Markov Decision Processes. Springer, 1996.

[14] E. Grädel, W. Thomas, and T. Wilke. Automata, Logics, and Infinite Games: A Guide to Current Research, volume 2500 of Lecture Notes in Computer Science. Springer, 2002.

[15] J. Klein. lt2dstar – LTL to Deterministic Streett and Rabin Automata, 2007. http://www.ltl2dstar.de/.

[16] J. Klein and C. Baier. Experiments with deterministic ω-automata for formulas of linear temporal logic. Theoretical Computer Science, 363(2):182 – 195, 2006.

[17] M. Lahijanian, S. B. Andersson, and C. Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. IEEE Transactions on Robotics, 28:396–409, 2011.

[18] M. Svořeřová, J. Tůmáčová, J. Barnat, and I. Černá. Attraction-Based Receding Horizon Path Planning with Temporal Logic Constraints. In Proceedings of the 51th IEEE Conference on Decision and Control, CDC’12, pages 6749–6754, 2012.

[19] M. Svořeřová, I. Černá, and C. Belta. Optimal Receding Horizon Control for Finite Deterministic Systems with Temporal Logic Constraints. In The 2013 American Control Conference, ACC’13, 2013. To appear.

[20] M. Svořeřová, I. Černá, and C. Belta. Simulation of Optimal Control of MDPs with Temporal Logic Constraints, 2013. http://www.fi.muni.cz/~x175388/simulationCDC13.

[21] S. Temizer, M. J. Kochenderfer, L. P. Kaelbling, T. Lozano-Perez, and J. K. Kuchar. Collision Avoidance for Unmanned Aircraft using Markov Decision Processes. In AIAA Guidance, Navigation and Control Conference, 2010.