Optimal Policy Generation for Partially Satisfiable Co-Safe LTL Specifications

Bruno Lacerda, David Parker and Nick Hawes
School of Computer Science, University of Birmingham
Birmingham, United Kingdom
{b.lacerda, d.a.parker, n.a.hawes}@cs.bham.ac.uk

Abstract

We present a method to calculate cost-optimal policies for task specifications in co-safe linear temporal logic over a Markov decision process model of a stochastic system. Our key contribution is to address scenarios in which the task may not be achievable with probability one. We formalise a task progression metric and, using multi-objective probabilistic model checking, generate policies that are formally guaranteed to, in decreasing order of priority: maximise the probability of finishing the task; maximise progress towards completion, if this is not possible; and minimise the expected time or cost required. We illustrate and evaluate our approach in a robot task planning scenario, where the task is to visit a set of rooms that may be inaccessible during execution.

1 Introduction

Markov decision processes (MDPs) are a widely used formalism to model sequential decision-making problems for scenarios where there is inherent uncertainty about a system's evolution. In general, planning for MDPs is performed by defining a reward structure over the model, and using dynamic programming techniques such as value or policy iteration [Puterman, 1994] to generate a policy that maximises the cumulative discounted reward over an infinite horizon. This approach has the drawback of requiring the designer to map the planning goal into a reward structure over the MDP, which can be quite cumbersome and error-prone.

Thus, in recent years, there has been increasing interest in the use of higher-level formalisms to specify the planning problems, in order to allow the designer to tackle intricate goal behaviours more naturally. In particular, linear temporal logic (LTL) [Pnueli, 1981] has been proposed as a suitable specification language, due to it being a powerful and intuitive formalism to unambiguously specify a variety of tasks. Furthermore, algorithms and tools exist to generate provably correct policies from an MDP model of the system and an LTL task specification. These approaches build upon techniques from the field of formal verification, in particular, *probabilistic model checking*, which provides tools to reason not just about the probability of satisfying an LTL specification, but also other quantitative metrics using cost (or reward) functions. These might represent, for example, execution time or energy consumption.

In this paper, we are interested in generating policies that minimise the *undiscounted expected cumulative cost*[1] to achieve a task specified in the *co-safe* fragment of LTL (i.e., a task that can be completed in a finite horizon). Our novel contribution is to generate cost-optimal policies for tasks that cannot be achieved with probability one in the model (i.e., there is some probability that part of the LTL specification cannot be achieved). Defining cost-optimal policies for such tasks presents an additional challenge as, in general, the undiscounted cumulative cost calculation will not converge. Furthermore, we tackle the question of what to do when the task becomes unsatisfiable during execution. In many cases, even if the probability of satisfying the overall task is zero, it is still possible to fulfil part of it. An illustrative example is a robot that needs to navigate to every office in a building to perform security checks. During execution some doors might be closed, making offices inaccessible. This will make the overall task unsatisfiable, yet we still want the robot to check as many offices as it can.

To allow this partial task satisfaction, we define a *task progression metric* for co-safe LTL formulas, which can be encoded as a reward function on the MDP. Using this, we show that the problems of (i) maximising the probability of satisfying a co-safe LTL formula; (ii) maximising the task progression reward (i.e., fulfilling as much of the formula as possible); and (iii) minimising a cost function while performing (i) and (ii) can be solved independently by standard techniques in a *trimmed product MDP*. As there is a trade-off between these problems, we apply *multi-objective probabilistic model checking* techniques [Forejt *et al.*, 2011b] to find the best policy for the three objectives, in decreasing order of priority. We also describe how to analyse the resulting policy to calculate conditional expectations such as expected cost to success or expected cost to failure. This analysis provides extra information about the policy that can be used to, for example,

[1] Our goal is to provide *meaningful* formal guarantees on the system's performance, e.g., expected time for completion. For such properties, following the more typical approach of optimising the cumulative sum using a discount factor strips the optimisation result of a well-defined meaning.

monitor execution, or inform a higher-level task scheduler, e.g., [Mudrova and Hawes, 2015]. Finally, we evaluate the approach on the illustrative example described above.

2 Related Work

Finding cost-optimal policies for MDPs using LTL specifications has been tackled in several ways. [Svoreňová *et al.*, 2013; Ding *et al.*, 2014] study the problem of maximising the probability of satisfying a LTL specification, while minimising the long-term average cost to pass between two states that satisfy an "optimising" atomic proposition, which must be visited infinitely often. In [Lacerda *et al.*, 2014], cost-optimal policies for co-safe LTL are generated, and a mechanism for the dynamic addition of tasks during execution is presented. In these cases, the cost minimisation is done *only over states where the probability of satisfying the specification is one*. Conversely, [Ulusoy *et al.*, 2012; Cizelj and Belta, 2014; Wolff *et al.*, 2013; Lahijanian *et al.*, 2012] deal with maximising the probability of satisfying temporal logic specifications *but cost-optimality is not taken into account*.

Other work on policy generation for MDPs with temporal logic specifications have focused on using the more traditional approach of finding policies that maximise *discounted cumulative rewards over an infinite horizon*, while constraining such policies to the class that satisfies a set of "until" specifications [Teichteil-Königsbuch, 2012a; Sprauel *et al.*, 2014]. In these works, the temporal logic constraints are disconnected from the reward structure. This means that, in general, no optimal policies exist for such a problem. However, the authors prove that one can always build *randomised policies* that achieve a notion of ϵ-optimality. In our work, the main focus is the LTL specification, and the generation of cost-optimal policies that maximise its probability of satisfaction. This makes our class of optimal policies simpler: given that we prioritise maximisation of the probability of satisfaction over minimisation of the expected cost, the class of deterministic finite-history policies suffices to solve our problem.

There has also been work on generation of cost-optimal policies for specifications where the probability of success is not one. [Teichteil-Königsbuch, 2012b; Kolobov *et al.*, 2012] present approaches to generate cost-optimal policies to reach a target state, in the presence of *unavoidable dead ends*, i.e., states which cannot be avoided with probability one from the initial state, and for which the probability of reaching the target is zero. Our work extends these approaches by focusing on co-safe LTL, instead of simpler *single-state reachability*. This fact requires the construction of a *product MDP* from the original model and the co-safe LTL specification, and also introduces the notion of *partial satisfiability*, which is not present when the goal is reaching a single state. [Ding *et al.*, 2013] present an approach, based on *constrained* MDPs, to generate cost-optimal policies for co-safe LTL, where the probability of satisfying the specification is kept above a threshold. Contrary to our approach, this requires the threshold for probability of satisfaction to be provided by the designer beforehand, and does not include the notion of partial task specification.

In terms of dealing with the unsatisfiability of temporal logic specifications during execution, [Tumova *et al.*, 2013; Castro *et al.*, 2013] deal with the generation of controllers that only violate sets of safety rules for the shortest possible amount of time as possible. [Maly *et al.*, 2013] also defines a metric for task progression for co-safe LTL, and generates plans that satisfy as much of the specification as possible. Contrary to our work however, the focus of these works is on controlling a hybrid model of the system, thus *they have no notion of probabilistic outcomes*. Our contribution goes beyond these previous works by *both* dealing with the partial satisfaction of LTL and doing this in a cost-optimal manner.

3 Preliminaries

3.1 Markov Decision Processes

We model systems using *Markov decision processes* (MDPs) with atomic propositions labelling states. An MDP is a tuple $\mathcal{M} = \langle S, \overline{s}, A, \delta_{\mathcal{M}}, AP, Lab \rangle$, where: S is a finite set of states; $\overline{s} \in S$ is the initial state; A is a finite set of actions; $\delta_{\mathcal{M}} : S \times A \times S \to [0,1]$ is a probabilistic transition function, where $\sum_{s' \in S} \delta_{\mathcal{M}}(s, a, s') \in \{0,1\}$ for all $s \in S$, $a \in A$; AP is a set of atomic propositions; and $Lab : S \to 2^{AP}$ is a labelling function, such that $p \in Lab(s)$ iff p is true in $s \in S$.

We define the set of *enabled* actions in $s \in S$ as $A_s = \{a \in A \mid \delta_{\mathcal{M}}(s, a, s') > 0$ for some $s' \in S\}$. An MDP model represents the possible evolutions of the state of a system: in each state s, any of the enabled actions $a \in A_s$ can be selected and the probability of evolving to a successor state s' is then $\delta_{\mathcal{M}}(s, a, s')$. An infinite *path* through an MDP is a sequence $\sigma = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots$ where $\delta_{\mathcal{M}}(s_i, a_i, s_{i+1}) > 0$ for all $i \in \mathbb{N}$. A finite path $\rho = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \ldots \xrightarrow{a_{n-1}} s_n$ is a prefix of an infinite path. We denote the sets of all finite and infinite paths of $\mathcal{M}$ starting from state s by $FPath_{\mathcal{M},s}$ and $IPath_{\mathcal{M},s}$.

The choice of action to take at each step of the execution of an MDP $\mathcal{M}$ is made by a *policy*, which can base its decision on the history of $\mathcal{M}$ up to the current state. Formally, a policy is a function $\pi : FPath_{\mathcal{M},\overline{s}} \to A$ such that, for any finite path σ ending in state s_n, $\pi(\sigma) \in A_{s_n}$. Important classes of policy include those that are *memoryless* (which only base their choice on the current state) and *finite-memory* (which need to track only a finite set of "modes"). The set of all policies of MDP $\mathcal{M}$ is denoted by $\Pi_{\mathcal{M}}$.

Under a particular strategy π for $\mathcal{M}$, all nondeterminism is resolved and the behaviour of $\mathcal{M}$ is fully probabilistic. Formally, we can represent this using an (infinite) *induced discrete-time Markov chain*, whose states are finite paths of $\mathcal{M}$. This leads us, using a standard construction [Kemeny *et al.*, 1976], to the definition of a probability measure $Pr^{\pi}_{\mathcal{M},s}$ over the set of infinite paths $IPath_{\mathcal{M},s}$, which allows us to determine the probability of certain events holding under policy π. For an event $b : IPath_{\mathcal{M},s} \to \{0,1\}$, we write $Pr^{\pi}_{\mathcal{M},s}(b)$ for the probability of b holding under π. Given a (measurable) function $f : IPath_{\mathcal{M},s} \to \mathbb{R}$, we can define the *expected value* $E^{\pi}_{\mathcal{M},s}(f)$ of f with respect to the probability measure $Pr^{\pi}_{\mathcal{M},s}$. We can then consider the maximum probabilities or expected values over all policies: $Pr^{\max}_{\mathcal{M},s}(b) = \sup_{\pi} Pr^{\pi}_{\mathcal{M},s}(b)$ and $E^{\max}_{\mathcal{M},s}(f) = \sup_{\pi} E^{\pi}_{\mathcal{M},s}(f)$, respectively. Minimum values $Pr^{\min}_{\mathcal{M},s}(b)$ or $E^{\min}_{\mathcal{M},s}(f)$ are defined analogously.

The *probabilistic reachability problem* is the problem of calculating $Pr^{\max}_{\mathcal{M},s}(reach_{S'})$, along with the corresponding optimal policy. Given $S' \subseteq S$, we define the event of reaching a state in S' as $reach_{S'} : IPath_{\mathcal{M},s} \rightarrow \{0,1\}$ where $reach_{S'}(s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ...) = 1$ iff $s_i \in S'$ for some i. The *expected cumulative reward problem* is the problem of calculating $E^{\max}_{\mathcal{M},s}(cumul_r)$, along with the corresponding optimal policy. Let $r : S \times A \rightarrow \mathbb{R}_{\geq 0}$ be a *reward structure* over $\mathcal{M}$. We define the cumulative reward $cumul_r : IPath_{\mathcal{M},s} \rightarrow \mathbb{R}_{\geq 0}$ as $cumul_r(s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} ...) = \sum_{i=0}^{\infty} r(s_i, a_i)$. $E^{\min}_{\mathcal{M},s}(cumul_r)$ is defined analogously, and in this case we write c instead of r, and call it a *cost structure*.

We can solve the probabilistic reachability and expected cumulative reward problems by using standard MDP algorithms such as value or policy iteration [Puterman, 1994] (which for these problems yield memoryless policies). Note that for cumulative rewards, the expected value might not converge. Thus, we first perform a graph analysis to identify states for which the reward is infinite [Forejt *et al.*, 2011a].

In this paper, we also consider *multi-objective* properties, which optimise two or more distinct objectives, such as the probability of an event or the cumulative value of a cost or reward structure. To mix objective values, it is common to consider their *Pareto set*. However, as we need to prioritise certain values (to favour task completion over cost incurred, for example) our approach uses *constrained* queries, such as:

$$E^{\max}_{\mathcal{M},s}(cumul_r \triangleleft reach_{S'} \geq p)$$

which maximises expected cumulated reward over all policies for which the probability of reaching S' is at least p:

$$\sup\{E^{\pi}_{\mathcal{M},s}(cumul_r) \,|\, \pi \in \Pi_{\mathcal{M}} \text{ s. t. } Pr^{\pi}_{\mathcal{M},s}(reach_{S'}) \geq p\}.$$

3.2 Syntactically Co-Safe Linear Temporal Logic

Linear temporal logic (LTL) is an extension of propositional logic which allows us reasoning about infinite sequences of states. It was developed as a means for formal reasoning about concurrent systems [Pnueli, 1981], and provides a convenient and powerful way to formally specify a variety of qualitative properties. In this work, we use the syntactically co-safe class of LTL formulas. Co-safe LTL formulas φ over propositions AP are defined using the following grammar:

$$\varphi ::= true \mid p \mid \neg p \mid \varphi \wedge \varphi \mid \mathtt{X}\,\varphi \mid \varphi\,\mathtt{U}\,\varphi, \text{ where } p \in AP.$$

The X operator is read "next", meaning that the formula it precedes will be true in the next state. The U operator is read "until", meaning that its second argument will eventually become true in some state, and the first argument will be continuously true until this point. See, e.g., [Pnueli, 1981] for the formal semantics of the logic.

Given an infinite path σ, we write $\sigma \models \varphi$ to denote that σ satisfies formula φ. Furthermore, we write $Pr^{\max}_{\mathcal{M},s}(\varphi)$ to denote the maximum probability of satisfying φ from s in $\mathcal{M}$. It is known that this problem can be reduced to a reachability problem in a *product* MDP [Vardi, 1985].

Even though their semantics is defined over infinite sequences, co-safe LTL formulas always have a finite *good prefix* [Kupferman and Vardi, 2001]. Given an LTL formula φ and an infinite sequence of sets of atomic propositions $w = w_0 w_1 ... \in (2^{AP})^{\omega}$ such that $w \models \varphi$, we say w has a good prefix if there exists $n \in \mathbb{N}$ for which the truncated finite sequence $w|_n = w_0 w_1 ... w_n$ is such that the concatenation $w|_n \cdot w' \models \varphi$ for any infinite sequence $w' \in (2^{AP})^{\omega}$.

Furthermore, for any co-safe LTL formula φ written over AP, we can build a deterministic finite automaton (DFA) $\mathcal{A}_{\varphi} = \langle Q, \overline{q}, Q_F, 2^{AP}, \delta_{\mathcal{A}_{\varphi}} \rangle$, where: Q is a finite set of states; $\overline{q} \in Q$ is the initial state; $Q_F \subseteq Q$ is the set of accepting (i.e., final) states; 2^{AP} is the alphabet; and $\delta_{\mathcal{A}_{\varphi}} : Q \times 2^{AP} \rightarrow Q$ is a transition function. $\mathcal{A}_{\varphi}$ accepts exactly the good prefixes for φ [Kupferman and Vardi, 2001]. We extend $\delta_{\mathcal{A}_{\varphi}}$ to a function over finite sequences $\delta^{+}_{\mathcal{A}_{\varphi}} : Q \times (2^{AP})^{*} \rightarrow Q$ in the usual manner, by applying it sequentially to the finite sequence. Given that a good prefix satisfies φ regardless of how it is "completed', an accepting state $q_F \in Q_F$ is such that $\delta_{\mathcal{A}_{\varphi}}(q_F, \alpha) \in Q_F$ for all $\alpha \in 2^{AP}$.

4 Policy Generation for Partially Satisfiable Task Specifications

Given an MDP $\mathcal{M}$, a cost structure $c : S \times A \rightarrow \mathbb{R}_{\geq 0}$, and a co-safe LTL specification φ such that $Pr^{\max}_{\mathcal{M},\overline{s}}(\varphi) > 0$, our goal is to find a policy that: (1) maximises the probability of satisfying φ; (2) when φ becomes unsatisfiable (i.e., when we reach a state s such that $Pr^{\max}_{\mathcal{M},s}(\varphi) = 0$), gets as close as possible to satisfying φ; and (3) has a minimal expected cost to achieve (1) and (2). Before we formally define this problem, we define what we mean by "as close as possible".

4.1 Task Progression Metric

We propose a notion of *task progression*, defined from a distance metric $d_{\varphi} : Q \rightarrow \mathbb{Q}_{\geq 0}$ that maps each state of $\mathcal{A}_{\varphi}$ to a value representing how close we are to reaching an accepting state. In the following, we write $q \rightarrow^{*} q'$ if there is a sequence of transitions in $\mathcal{A}_{\varphi}$ that leads it from state q to state q'.

Let $Suc_q \subseteq Q$ be the set of successors of state q, and $|\delta_{q,q'}| \in \{0, ..., 2^{|AP|}\}$ be the number of transitions from q to q'. We define the following distance metric:

$$d_{\varphi}(q) = \begin{cases} 0 & \text{if } q \in Q_F \\ \min\limits_{q' \in Suc_q} \left\{ d_{\varphi}(q') + \dfrac{1}{|\delta_{q,q'}|} \right\} & \text{if } q \notin Q_F \text{ and} \\ & \exists q_F \in Q_F \text{ s.t.} \\ & q \rightarrow^{*} q_F \\ |Q| & \text{otherwise} \end{cases}$$

This metric represents the number of states that need to be visited, starting in q, to reach a state in Q_F, balanced by the number of transitions between these states. This balancing is done because we assume that, if there are more transitions between two DFA states, then it should be easier for the system to evolve to a state that makes the DFA move between them. Furthermore, if a state in Q_F is reachable from q, then $d_{\varphi}(q) \leq |Q| - 1$. So, if there is no path from q to Q_F, we set $d_{\varphi}(q)$ to the maximum value $|Q|$. The values d_{φ} can be calculated recursively by the following fixed-point calculation:

$$d^{0}_{\varphi}(q) = \begin{cases} 0 & \text{if } q \in Q_F \\ |Q| & \text{if } q \notin Q_F \end{cases}$$

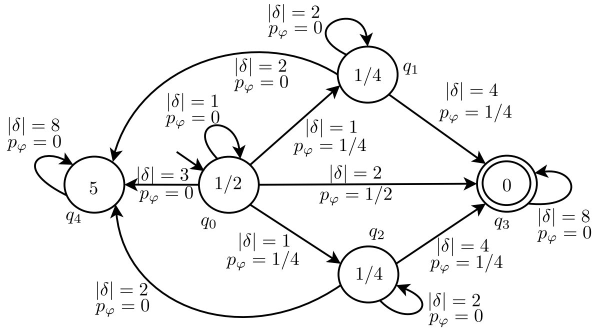

Figure 1: A DFA $\mathcal{A}_\varphi$ for $\varphi = ((\neg a) \,\mathtt{U}\, b) \wedge ((\neg a) \,\mathtt{U}\, c)$, labelled with distance d_φ, progression p_φ and transition counts $|\delta|$.

$$d_\varphi^{k+1}(q) = \min\left\{ d_\varphi^k(q), \min_{q' \in Suc_q} \left\{ d_\varphi^k(q') + \frac{1}{|\delta_{q,q'}|} \right\} \right\}$$

Using this distance metric, we can also define the *progression* $p_\varphi : Q \times Q \to \mathbb{Q}_{\geq 0}$ between two states of $\mathcal{A}_\varphi$ as a measure of how much the distance to an accepting state is reduced by moving between q and q':

$$p_\varphi(q, q') = \begin{cases} \max\{0, d_\varphi(q) - d_\varphi(q')\} & \text{if } q' \in Suc_q \text{ and} \\ & q' \not\to^* q \\ 0 & \text{otherwise} \end{cases}$$

We require $q' \not\to^* q$ in the first condition to guarantee that there are no cycles in the DFA with non-zero values for p_φ. This is needed in order to guarantee convergence of infinite sums of values of p_φ, as will be explained below. We can check existence of paths between q' and q by computing the strongly connected components (SCCs) of $\mathcal{A}_\varphi$. Finally, given that we are only interested in getting closer to accepting states, we only take positive progression values into account.

Example. Consider the DFA in Fig. 1. Inside each state q_i, we show the value $d_\varphi(q_i)$ and we label each edge with the number of transitions corresponding to that edge $|\delta|$, and the value of p_φ for the states connected by that edge. Note that moving from any state to the accepting state q_3 has a positive progression, as once we get to the accepting state we cannot leave it, due to the co-safe assumption. Also, $d_\varphi(q_4) = 5$ because there is no path from q_4 to the accepting state.

4.2 Problem Statement

We are now in a position to formalise the problem stated at the beginning of this section, which is sub-divided into three objectives. First, let us consider each problem in isolation. Let $\mathcal{M}$ be an MDP and φ a co-safe LTL formula.

Problem 1 *Calculate $Pr^{\max}_{\mathcal{M},\overline{s}}(\varphi)$, and find the corresponding policy:*

$$\pi_1 = \arg\max_\pi Pr^\pi_{\mathcal{M},\overline{s}}(\varphi)$$

Problem 2 *Let $\sigma = s_0 \overset{a_0}{\to} s_1 \overset{a_1}{\to} \ldots \in IPath_{\mathcal{M},\overline{s}}$, and $q_i^\sigma = \delta^+_{\mathcal{A}_\varphi}(\overline{q}, Lab(s_0)Lab(s_1)\ldots Lab(s_i))$. We define the total progression of σ as $prog(\sigma) = \sum_{i=0}^\infty p_\varphi(q_i^\sigma, q_{i+1}^\sigma)$. Calculate $E^{\max}_{\mathcal{M},\overline{s}}(prog)$, and find the corresponding policy:*

$$\pi_2 = \arg\max_\pi E^\pi_{\mathcal{M},\overline{s}}(prog)$$

Note that, by construction, the value of $prog(\sigma)$ always converges: any infinite sequence of visited states of a DFA always has an (infinite) suffix formed of only SCCs. Thus, we are guaranteed that for all $\sigma \in IPath_{\mathcal{M},s}$, $\exists k \in \mathbb{N}$ such that $p_\varphi(q_i^\sigma, q_{i+1}^\sigma) = 0$, for all $i \geq k$. We denote the minimum such k as $k^\sigma_{p_\varphi}$. Also, if $\sigma \models \varphi$, then $Lab(s_0)\ldots Lab(s_{k^\sigma_{p_\varphi}})$ is a good prefix for φ.

Problem 3 *Let $\sigma = s_0 \overset{a_0}{\to} s_1 \overset{a_1}{\to} \ldots \in IPath_{\mathcal{M},\overline{s}}$. We define the cumulated cost until progression p_φ is 0 as $cumul_c^{k^\sigma_{p_\varphi}}(\sigma) = \sum_{i=0}^{k^\sigma_{p_\varphi}} c(s_i, a_i)$. Calculate $E^{\min}_{\mathcal{M},\overline{s}}(cumul_c^{k^\sigma_{p_\varphi}})$, and find the corresponding policy:*

$$\pi_3 = \arg\min_\pi E^\pi_{\mathcal{M},\overline{s}}(cumul_c^{k^\sigma_{p_\varphi}})$$

These three problems are pairwise conflicting. For example, when the cost of reaching a state from where φ is not satisfiable is lower than the cost of reaching an accepting state, the solution for Problem 3 will be a policy that tries to make φ not satisfiable as soon as possible. Also, maximising progression might lead to policies that get close to an accepting state with high probability but have probability zero of reaching it, instead of policies that reach an accepting state with some probability. While the solution for each independent problem does not meet our overall goal, we can employ *multi-objective* probabilistic model checking techniques to look for a solution for the three problems in decreasing order of priority.

4.3 Generation of the Trimmed Product

Before we define the multi-objective based solution, we show that each of the three problems stated above can be reduced to simpler problems in a *trimmed product* MDP. We start by building the product MDP $\mathcal{M}_\varphi = \mathcal{M} \otimes \mathcal{A}_\varphi = \langle S \times Q, \overline{s_\varphi}, A, \delta_{\mathcal{M}_\varphi}, AP, Lab_\varphi \rangle$, and extending the cost structure c to $\mathcal{M}_\varphi$ as $c_\varphi((s,q),a) = c(s,a)$. $\mathcal{M}_\varphi$ behaves like the original MDP $\mathcal{M}$, but is augmented with information about the satisfaction of φ. Once a path of $\mathcal{M}_\varphi$ reaches an *accepting state* (i.e., a state of the form (s, q_F)), it is a good prefix for φ, and we are sure that φ is satisfied. The construction of the product MDP is well known and is such that $\mathcal{M}_\varphi$ preserves the probabilities of paths from $\mathcal{M}$ (see, e.g., [Baier and Katoen, 2008]), thus we can reduce Problem 1 to a reachability problem in $\mathcal{M}_\varphi$. Furthermore, Problem 2 can be reduced to an expected cumulative reward problem in $\mathcal{M}_\varphi$, by encoding p_φ as a reward structure $r_\varphi : (S \times Q) \times A \to \mathbb{Q}_{\geq 0}$:

$$r_\varphi((s,q),a) = \sum_{(s',q') \in S \times Q} \delta_{\mathcal{M}_\varphi}((s,q),a,(s',q')) p_\varphi(q,q')$$

However, Problem 3 cannot be reduced to an expected cumulative cost problem in $\mathcal{M}_\varphi$, because the value $k^\sigma_{p_\varphi}$ is not constant, being dependent on the path σ. Thus, we need to remove states from $\mathcal{M}_\varphi$ for which it is not possible to accumulate more p_φ. To do that, we start by defining the set of states for which we can gather more reward immediately as:

$$S^0_{r_\varphi} = \{(s,q) \in S \times Q \mid \exists a \in A \text{ s. t. } r_\varphi((s,q),a) > 0\}$$

Furthermore, we define the set of states for which it is still possible to gather more reward as:

$$S_{r_\varphi} = \{(s,q) \in S \times Q \mid Pr^{\max}_{\mathcal{M}_\varphi,(s,q)}(reach_{S^0_{r_\varphi}}) > 0\}$$

If $(s,q) \notin S_{r_\varphi}$ then, by construction of r_φ, it is guaranteed that we cannot progress further towards the goal from (s,q). Thus, such states are not needed to solve Problems 1 – 3.

We can calculate S_{r_φ} efficiently by using $S^0_{r_\varphi}$ as above, and finding the fixed-point of the following calculation:

$$\begin{aligned} S^{k+1}_{r_\varphi} = S^k_{r_\varphi} \cup \{s_\varphi \in S \times Q \mid \exists s'_\varphi \in S^k_{r_\varphi}, a \in A \text{ s.t.} \\ \delta_{\mathcal{M}_\varphi}(s_\varphi, a, s'_\varphi) > 0\} \end{aligned}$$

Along with S_{r_φ}, we also need to keep states which are successors of a state in S_{r_φ}, but are not in S_{r_φ}. These are the states for which we are certain that no more can be done towards satisfying φ, thus they are the states where we should stop accumulating costs. Thus, let:

$$\begin{aligned} Suc_{S_{r_\varphi}} = \{s_\varphi \in S \times Q \mid \exists s'_\varphi \in S_{r_\varphi}, a \in A \text{ s. t.} \\ \delta_{\mathcal{M}_\varphi}(s'_\varphi, a, s_\varphi) > 0\} \end{aligned}$$

We can now define the trimmed product MDP $\mathcal{M}^{r_\varphi}_\varphi = \langle S_{r_\varphi} \cup Suc_{S_{r_\varphi}}, \overline{s_\varphi}, A, \delta_{\mathcal{M}^{r_\varphi}_\varphi}, AP, Lab_\varphi \rangle$ as the restriction of $\mathcal{M}_\varphi$ to the set of states S_{r_φ} plus their possible successor states. More precisely, we project Lab_φ, c_φ, and r_φ to $S_{r_\varphi} \cup Suc_{S_{r_\varphi}}$, and only keep transitions from states in S_{r_φ}, i.e., $\delta_{\mathcal{M}^{r_\varphi}_\varphi}$ is defined as:

$$\delta_{\mathcal{M}^{r_\varphi}_\varphi}(s_\varphi, a, s'_\varphi) = \begin{cases} \delta_{\mathcal{M}_\varphi}(s_\varphi, a, s'_\varphi) & \text{if } s_\varphi \in S_{r_\varphi} \\ 0 & \text{if } s_\varphi \in Suc_{S_{r_\varphi}} \end{cases}$$

We just remove states for which the probability of accumulating progression reward is zero in the construction of $\mathcal{M}^{r_\varphi}_\varphi$, and we remove all the transitions (along with the corresponding costs) from all such states. This means that the solutions for Problems 1 and 2 for $\mathcal{M}^{r_\varphi}_\varphi$ are the same as the solutions for $\mathcal{M}_\varphi$. Furthermore, by removing those states, we guarantee convergence of the calculation of the cumulative cost, by removing *all* end components in the model with a positive cost. Also, once we reach a state where we cannot progress more towards the goal, we achieved the corresponding value for the cumulative cost, thus we can replace the path-dependent sum upper-bound $k^\sigma_{p_\varphi}$ on the cumulative cost minimisation by an infinite sum. Thus, letting $S_{\varphi,F} = \{(s,q) \in S_{r_\varphi} \cup Suc_{S_{r_\varphi}} \mid q \in Q_F\}$, we have:

$$Pr^{\max}_{\mathcal{M},\overline{s}}(\varphi) = Pr^{\max}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(reach_{S_{\varphi,F}})$$

$$E^{\max}_{\mathcal{M},\overline{s}}(prog) = E^{\max}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{r_\varphi})$$

$$E^{\min}_{\mathcal{M},\overline{s}}(cumul_c^{k^\sigma_{p_\varphi}}) = E^{\min}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi})$$

So, Problem 1 can be reduced to solving a reachability problem, and Problems 2 and 3 can be reduced to expected cumulative reward/cost problems, where convergence is guaranteed. These problems can be solved on the same underlying model $\mathcal{M}^{r_\varphi}_\varphi$ using standard techniques, such as value iteration. This allows us to solve a multi-objective problem with three functions to be optimised, as will be discussed next.

4.4 Multi-objective Model Checking

Finally, we use multi-objective probabilistic model checking to generate a policy that takes into account Problems 1, 2 and 3, in decreasing order of priority. As explained above, each of these can be computed using the trimmed product model $\mathcal{M}^{r_\varphi}_\varphi$. More precisely, we proceed as follows. We first compute the maximum probability p of completing the task:

$$p = Pr^{\max}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(reach_{S_{\varphi,F}})$$

Then, we find the maximum expected total progression that can be achieved whilst maintaining a task satisfaction probability of p, denoting this value by r:

$$r = E^{\max}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{r_\varphi} \triangleleft \ reach_{S_{\varphi,F}} \geq p)$$

Finally, we determine the policy which minimises the total expected cost, whilst ensuring that the task satisfaction probability and total progression remain at their optimal values of p and r, respectively:

$$c^* = E^{\min}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi} \triangleleft \ reach_{S_{\varphi,F}} \geq p \wedge cumul_{r_\varphi} \geq r)$$

By calculating c^*, we also get a corresponding policy π^* that optimises the three objectives in decreasing order of priority. Note that, even though in general policies for multi-objective specifications can be *randomised*, we are guaranteed by construction that there exists a deterministic policy that solves the overall problem, because the values p and r we use to constrain the multi-objective queries are the extreme values obtained from the previous query.

4.5 Calculation of Conditional Expectations

We can now generate a cost-optimal policy π^* for our problem, however the value we obtain for the expected cumulative cost is the expectation to reach a state for which it is not possible to gather more cumulative progression reward. This value can have a very large variance because the expected costs to reach each of these states can be very different. Thus, we are interested in "splitting" the expected cumulative cost value, into expected cost *to success*, and expected cost *to failure*. These conditional expectations are more informative and can be used for execution monitoring, or for a higher level task scheduler. We present a procedure to calculate the conditional expectation $E^{\pi^*}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi} \mid reach_{S_{\varphi,F}})$ efficiently. We start by noting that, given that π^* is a memoryless policy for $\mathcal{M}^{r_\varphi}_\varphi$, applying π^* to $\mathcal{M}^{r_\varphi}_\varphi$ induces a (finite) Markov chain $C = \langle S_{r_\varphi} \cup Suc_{S_{r_\varphi}}, \overline{s_\varphi}, \delta_C \rangle$, where the transition for each state $s_\varphi \in S_{r_\varphi} \cup Suc_{S_{r_\varphi}}$ is given by $\delta_C(s_\varphi, s'_\varphi) = \delta_{\mathcal{M}^{r_\varphi}_\varphi}(s_\varphi, \pi^*(s_\varphi), s'_\varphi)$. Furthermore, during the calculation of π^*, we calculated $Pr_{C,s_\varphi}(reach_{S_{\varphi,F}}) = Pr^{\pi^*}_{\mathcal{M}^{r_\varphi}_\varphi,s_\varphi}(reach_{S_{\varphi,F}})$ for all states of C. Let $S_{\not\models}$ be the set of all the states where such probability is zero. To calculate the expected cumulative cost *to success*, i.e., only for runs of C that reach $S_{\varphi,F}$, we trim C, and build $C_{\models} = \langle S_{\models}, \overline{s_\varphi}, \delta_{C_\models} \rangle$, where $S_\models = (S_{r_\varphi} \cup Suc_{S_{r_\varphi}}) \setminus S_{\not\models}$, and for all $s, s' \in S_\models$:

$$\delta_{C_\models}(s, s') = \delta_C(s, s') \frac{Pr_{C,s'}(reach_{S_{\varphi,F}})}{Pr_{C,s}(reach_{S_{\varphi,F}})}$$

So, to build $C_{\models}$ we simply remove all states for which the probability of satisfying the specification is zero, and normalise the transition function accordingly. Then:

$$E^{\pi^*}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi} \mid reach_{S_{\varphi,F}}) = E_{C_{\models},\overline{s_\varphi}}(cumul_{c_\varphi})$$

This value can be calculated efficiently, as it is a simple expectation over C. We can also calculate the expected cost to failure $E^{\pi^*}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi} \mid \neg reach_{S_{\varphi,F}})$, since we also know $Pr^{\pi^*}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(reach_{S_{\varphi,F}})$ and $E^{\pi^*}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi})$.

This method can be generalised over the trimmed product, in order to calculate $E^{\min}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi} \mid reach_{S_{\varphi,F}})$, and the corresponding policy. However, such an approach does not provide guarantees on probability of task satisfaction, hence it might have a large probability of failure (in spite of providing an optimal expectation for the satisfying runs). See [Baier *et al.*, 2014] for a proof of correctness of the construction of $C_{\models}$, and for a more general discussion on the calculation of conditional expectations for Markovian models.

5 Implementation and Evaluation

We implemented our approach in the widely used PRISM model checker [Kwiatkowska *et al.*, 2011], which already has support for solving MDPs against properties in LTL and, in particular, for multi-objective policy generation [Forejt *et al.*, 2011b]. The tool also includes multiple efficient solution engines, based on symbolic data structures (e.g., binary decision diagrams) designed to improve scalability.

Consider the topological map of an office environment in Fig. 2. We assume that v_0 should always be avoided (e.g. to prevent the robot blocking a fire exit), and that our task is to visit nodes v_1, v_6 and v_{18} (i.e., visit three rooms). This specification can be written in co-safe LTL as $((\neg v_0)\, \mathtt{U}\, v_1) \wedge ((\neg v_0)\, \mathtt{U}\, v_6) \wedge ((\neg v_0)\, \mathtt{U}\, v_{18})$.

The policy obtained for this specification simply tries to visit v_1, v_6 and v_{18} in the order which minimises expected cost (i.e., time), which is the numerical order in this case. However, since there is a chance the robot will end up in v_0 when navigating directly between v_3 and v_4, the policy takes a "detour" through v_{10}. This is because our approach prioritises robustness – in the form of maximisation of the probability of success – over optimality of expected time.

If a closed door is encountered during execution, the policy simply tries to go to the next unvisited closest node. This allows it to satisfy as much of the specification as possible. Based on the values in Fig. 2 the probability of the policy satisfying the specification is 0.729, with an expected time of 11.8 minutes. The values for the expected time of success and failure are 13.8 and 6.42, respectively. The lower time for failure is explained by the fact that the more of the specification the robot fails to satisfy (due to closed doors), the less it has to move. The MDP model used for the scenario on Fig. 2 has 10,206 states and 49,572 transitions. This number is due to the state space of the MDP having one additional dimension per door, each with value "unknown", "open", or "closed". The trimmed product has 21,222 states, and 100,926 transitions, and was generated in 0.354 seconds.

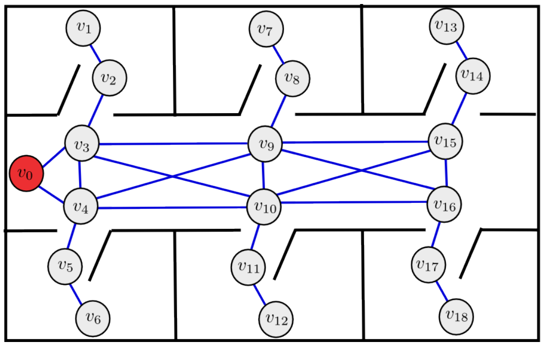

Figure 2: An office environment with 6 rooms, and the nodes used for patrolling. The initial node is v_3. Blue (bidirectional) edges represent possible navigation actions between nodes. The MDP model also includes the state of each door, and before navigating through a door the robot must check if it is open (we assume probability 0.9 of being open for each door). All navigation actions are successful with probability 1, except from v_3 to v_4, which might finish in v_0, with probability 0.2. A cost structure, representing the expected time taken to move between each pair of nodes, is defined by a coarse distance between the nodes.

Calculating $p = Pr^{\max}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(reach_{S_{\varphi,F}})$ took approximately 1 second, $r = E^{\max}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{r_\varphi} \triangleleft reach_{S_{\varphi,F}} \geq p)$ 305 seconds, and $E^{\min}_{\mathcal{M}^{r_\varphi}_\varphi,\overline{s_\varphi}}(cumul_{c_\varphi} \triangleleft reach_{S_{\varphi,F}} \geq p, cumul_{r_\varphi} \geq r)$ 227 seconds, yielding a total of approximately 9 minutes to generate the policy (on an Intel® Core™i7 @ 1.70GHz x 4 processor, 8GB of RAM). While the specification we tested is computationally challenging, featuring the combinatorial explosion of choosing the best order to visit a set of locations, the time taken to generate the policy is higher than desirable, given that the model is relatively small. This motivates the need to find more efficient approaches for objective prioritisation, and better modelling approaches for this type of environment, and will be the subject of future work. However, with this example we showcase the ability to generate optimal policies for our problem by using multi-objective model-checking techniques. Moreover, the policies can be calculated offline, and executing them requires negligible computational effort. Since these policies map *all* reachable states of the trimmed product to an action, this provides a robust plan that can handle every probabilistic environmental outcome without the need for replanning.

6 Conclusions

This paper presented a methodology for the generation of policies that maximise the probability of satisfying a co-safe LTL specification, whilst trying to satisfy it as much as possible when it becomes unsatisfiable, and minimising the expected cumulative cost while doing so. This approach is one of the first cost-optimal policy generation approach for LTL specifications over MDP models able to optimise an undiscounted expected cost over states where the probability of satisfying the specification is not one. Furthermore, the fact that we base our approach on probabilistic model checking techniques allows us to provide meaningful formal guarantees, which can be used to analyse the overall quality of service of the system. In particular, we show how to provide guarantees on the probability of task satisfaction, along with expected cumulative costs for success and failure. We also illustrated the approach on an example that motivates the need to investigate more efficient solutions to our objective prioritisation problem and to model certain environments as MDPs. This will be subject of future work, e.g., modelling of office-like environments, as we are interested in the deployment of service robots using these techniques, and investigating how the approaches in [Teichteil-Königsbuch, 2012b; Kolobov *et al.*, 2012] can be adapted to increase computational efficiency of our solution, as the problem solved there is very similar to our reduced problem in the product MDP.

Acknowledgements

The research leading to these results has received funding from the EU Seventh Framework Programme (FP7/2007-2013) under grant agreement No 600623, STRANDS.

References

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

[Baier *et al.*, 2014] C. Baier, J. Klein, S. Klüppelholz, and S. Märcker. Computing conditional probabilities in Markovian models efficiently. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2014.

[Castro *et al.*, 2013] L. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus. Incremental sampling-based algorithm for minimum-violation motion planning. In *Proc. of 52nd IEEE Conf. on Decision and Control (CDC)*, 2013.

[Cizelj and Belta, 2014] I. Cizelj and C. Belta. Control of noisy differential-drive vehicles from time-bounded temporal logic specifications. *Int. Journal of Robotics Research*, 33(8), 2014.

[Ding *et al.*, 2013] X. Ding, A. Pinto, and A. Surana. Strategic planning under uncertainties via constrained Markov decision processes. In *Proc. of the 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.

[Ding *et al.*, 2014] X. Ding, S. Smith, C. Belta, and D. Rus. Optimal control of Markov decision processes with linear temporal logic constraints. *IEEE Trans. on Automatic Control*, 59(5), 2014.

[Forejt *et al.*, 2011a] V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In *Formal Methods for Eternal Networked Software Systems (SFM)*, volume 6659 of *LNCS*. Springer, 2011.

[Forejt *et al.*, 2011b] V. Forejt, M. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Quantitative multi-objective verification for probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011.

[Kemeny *et al.*, 1976] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov chains*. Springer-Verlag, 2nd edition, 1976.

[Kolobov *et al.*, 2012] A. Kolobov, Mausam, and D. Weld. A theory of goal-oriented MDPs with dead ends. In *Proc. of 28th Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2012.

[Kupferman and Vardi, 2001] O. Kupferman and M. Vardi. Model checking of safety properties. *Formal Methods in System Design*, 19(3), 2001.

[Kwiatkowska *et al.*, 2011] M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *Computer Aided Verification (CAV)*, volume 6806 of *LNCS*. Springer, 2011.

[Lacerda *et al.*, 2014] B. Lacerda, D. Parker, and N. Hawes. Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications. In *Proc. of 2014 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

[Lahijanian *et al.*, 2012] M. Lahijanian, S. Andersson, and C. Belta. Temporal logic motion planning and control with probabilistic satisfaction guarantees. *IEEE Trans. on Robotics*, 28(2), 2012.

[Maly *et al.*, 2013] M. Maly, M. Lahijanian, L. Kavraki, H. Kress-Gazit, and M. Vardi. Iterative temporal motion planning for hybrid systems in partially unknown environments. In *Proc. of 16th Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, 2013.

[Mudrova and Hawes, 2015] L. Mudrova and N. Hawes. Task scheduling for mobile robots using interval algebra. In *Proc. of 2015 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2015.

[Pnueli, 1981] A. Pnueli. The temporal semantics of concurrent programs. *Theoretical Computer Science*, 13, 1981.

[Puterman, 1994] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[Sprauel *et al.*, 2014] J. Sprauel, F. Teichteil-Königsbuch, and A. Kolobov. Saturated path-constrained MDP: Planning under uncertainty and deterministic model-checking constraints. In *Proc. of 28th AAAI Conf. on Artificial Intelligence (AAAI)*, 2014.

[Svoreňová *et al.*, 2013] M. Svoreňová, I. Černá, and C. Belta. Optimal control of MDPs with temporal logic constraints. In *Proc. of 52nd IEEE Conf. on Decision and Control (CDC)*, 2013.

[Teichteil-Königsbuch, 2012a] F. Teichteil-Königsbuch. Path-constrained Markov decision processes: bridging the gap between probabilistic model-checking and decision-theoretic planning. In *Proc. of 2012 European Conf. on Artificial Intelligence (ECAI)*, 2012.

[Teichteil-Königsbuch, 2012b] F. Teichteil-Königsbuch. Stochastic safest and shortest path problems. In *Proc. of 26th AAAI Conf. on Artificial Intelligence (AAAI)*, 2012.

[Tumova *et al.*, 2013] J. Tumova, G. Hall, S. Karaman, E. Frazzoli, and D. Rus. Least-violating control strategy synthesis with safety rules. In *Proc. of 16th Int. Conf. on Hybrid Systems: Computation and Control (HSCC)*, 2013.

[Ulusoy *et al.*, 2012] A. Ulusoy, T. Wongpiromsarn, and C. Belta. Incremental control synthesis in probabilistic environments with temporal logic constraints. In *Proc. of 51st IEEE Conf. on Decision and Control (CDC)*, 2012.

[Vardi, 1985] M. Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Proc. of 26th IEEE Annual Symp. on Foundations of Comp. Sci. (FOCS)*, 1985.

[Wolff *et al.*, 2013] E. Wolff, U. Topcu, and R. Murray. Efficient reactive controller synthesis for a fragment of linear temporal logic. In *Proc. of 2013 IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.