Nested Value Iteration for Partially Satisfiable Co-Safe LTL Specifications
(Extended Abstract)

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

Overview We describe our recent work (Lacerda, Parker, and Hawes 2015) on cost-optimal policy generation for co-safe linear temporal logic (LTL) specifications that are not satisfiable with probability one in a Markov decision process (MDP) model. We provide an overview of the approach to pose the problem as the optimisation of three standard objectives in a trimmed product MDP. Furthermore, we introduce a new approach for optimising the three objectives, in decreasing order of priority, based on “nested” value iteration, where one value table is kept for each objective.

Our overall goal is to generate policies that maximise the probability of success and minimise the undiscounted expected cumulative cost to achieve a task specified in the co-safe fragment of LTL (i.e., a task that can be completed in a finite horizon). 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. We formalise this notion as a progression reward defined over the LTL automaton.

Given an MDP and a co-safe LTL specification, we show that the problems of (i) maximising the probability of satisfying a co-safe LTL formula; (ii) maximising the progression reward (i.e., fulfilling as much of the specification 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. The main steps of the construction of this MDP are depicted in Fig. 1.

The problems above are conflicting. In (Lacerda, Parker, and Hawes 2015), we implement objective prioritisation, using multi-objective model checking techniques. However, this approach is quite inefficient: in our experiments, calculating the optimal policy for a trimmed product MDP with ~20,000 states and ~100,000 transitions took ~9 minutes on a standard laptop computer. Here, we introduce a new, more efficient approach, based on a nested version of value iteration.

A Metric for Task Progression A co-safe LTL formula \( \varphi \) over atomic propositions \( AP \) is defined by:

\[ \varphi := \text{true} \mid p \mid \neg p \mid \varphi \land \varphi \mid X \varphi \mid F \varphi \mid \varphi U \varphi, \text{ where } p \in AP. \]

\[ p := d \varphi(q,q'), \varphi \]

\[ \varphi := \text{true} \mid p \mid \neg p \mid \varphi \land \varphi \mid X \varphi \mid F \varphi \mid \varphi U \varphi, \text{ where } p \in AP. \]

\[ p := d \varphi(q,q'), \varphi \]

\[ p := d \varphi(q,q'), \varphi \]

\[ p := d \varphi(q,q'), \varphi \]

\[ p := d \varphi(q,q'), \varphi \]

\[ p := d \varphi(q,q'), \varphi \]

\[ p := d \varphi(q,q'), \varphi \]

\[ p := d \varphi(q,q'), \varphi \]
Construction of the Trimmed Product Let \( M = (S, \pi, A, \delta_M, AP, Lab) \) be an MDP, where: \( S \) is a finite set of states; \( \pi \in S \) is the initial state; \( A \) is a finite set of actions; \( \delta_M : S \times A \times S \rightarrow \{0, 1\} \) is a probabilistic transition function, where \( \sum_{s' \in S} \delta_M(s, a, s') \in \{0, 1\} \) for all \( s, a \in S \), \( a \in A \); \( AP \) is a set of atomic propositions; and \( Lab : S \rightarrow 2^{AP} \) is a labelling function, such that \( p \in Lab(s) \) iff \( p \) is true in \( s \in S \), and \( c : S \times A \rightarrow 0,\infty \), be a cost structure over \( M \). Given a DFA \( A_\phi = (Q, \pi, Q_F, 2^{AP}, \delta_A) \), one can build the cross-product \( M_\phi = M \otimes A_\phi = (S \times Q, \pi, Q_F, 2^{AP}, \delta_M, \delta_A) \), and extend the cost structure \( c \) to \( M_\phi \) as \( c_\phi((s, q), a) = c(s, a) \). \( M_\phi \) behaves like the original MDP \( M \), but is augmented with information about the satisfaction of \( \phi \). Once a path of \( M_\phi \) reaches an accepting state (i.e., a state of the form \((s, q_F)\)), it satisfies \( \phi \). The construction of the product MDP is well known and is such that \( M_\phi \) preserves the probabilities of paths from \( M \) (see, e.g., (Baier and Katoen 2008)).

However, given that the probability of reaching an accepting state in \( M_\phi \) is not one, the calculation of the minimal expected cost in this structure does not converge. In order to tackle this issue, we start by encoding \( \phi \) as a reward structure \( r_\phi : (S \times Q) \times A \rightarrow [0, \infty) \):

\[
r_\phi((s, q), a) = \sum_{(s', q') \in S \times Q} \delta_{M_\phi}((s, q), a, (s', q')) p_\phi(q, q')
\]

The main insight for building the trimmed product MDP \( M_\phi^{r_\phi} \) is noticing that the progression metric is built in such a way that any infinite run of \( M_\phi \) will eventually reach a state where no more progression reward can be gathered. This is due to the system reaching an accepting state, or a state where no more can be done towards the formula satisfaction. To tackle this issue, we start by encoding \( \phi \) as a reward structure \( r_\phi \) for all states which (i) have zero probability of gathering more progression reward; and (ii) are not an immediate successor of a state with non-zero probability of gathering more progression reward. All transitions to these states are also removed from the structure. By doing this, we guarantee convergence of the calculation of the cumulative cost, because we removed all end components in the model with a positive cost. The states that satisfy condition (i) above, but not condition (ii), are terminal states in \( M_\phi^{r_\phi} \), i.e., no action can be executed once they are reached.

Thus, our three objectives are reduced to standard problems in \( M_\phi^{r_\phi} \): (i) maximising the probability of satisfying \( \phi \) is reduced to maximising the probability of reaching an accepting state; (ii) maximising progression is reduced to the infinite-horizon maximisation of the cumulative value of \( r_\phi \); and (iii) minimising the cost to reach a state where no more progression is accumulated can be reduced to the infinite-horizon minimisation of the cumulative value of \( c_\phi \).

Algorithm 1

Input: Trimmed product \( M_\phi^{r_\phi} \), cost \( c_\phi \), progression reward \( r_\phi \).
Output: Optimal policy \( \pi : S \times Q \rightarrow A \).

1: for all \( s \in S \times Q \) do
2: \( V_\phi(s) \leftarrow \begin{cases} 1 & \text{if } s = (s, q_F), \text{ where } q_F \in Q_F \\ 0 & \text{otherwise} \end{cases} \)
3: \( V_\phi(s) \leftarrow 0 \)
4: \( V_\phi(s) \leftarrow \begin{cases} \infty & \text{if } s \text{ is a terminal state} \\ 0 & \text{otherwise} \end{cases} \)
5: end for
6: while \( V_\phi \) or \( V_\phi \) or \( V_\phi \) have not converged do
7: for all \( s \in S \times Q \) which are not terminal do
8: for all \( a \in A(s) \) do
9: \( v_\phi \leftarrow \sum_{s' \in S \times Q} \delta_{M_\phi}((s, q), a, (s', q')) V_\phi(s') \)
10: \( v_\phi \leftarrow r_\phi((s, q), a) + \sum_{s' \in S \times Q} \delta_{M_\phi}((s, q), a, (s', q')) V_\phi(s') \)
11: \( v_\phi \leftarrow c_\phi((s, q), a) \)
12: if \( v_\phi > V_\phi \) then
13: \( V_\phi(s) \leftarrow v_\phi \)
14: \( V_\phi(s) \leftarrow v_\phi \)
15: \( V_\phi(s) \leftarrow v_\phi \)
16: \( \pi(s) \leftarrow a \)
17: else if \( v_\phi = V_\phi \) then
18: \( V_\phi(s) \leftarrow v_\phi \)
19: \( V_\phi(s) \leftarrow v_\phi \)
20: \( \pi(s) \leftarrow a \)
21: else if \( v_\phi = V_\phi \) then
22: \( V_\phi(s) \leftarrow v_\phi \)
23: \( \pi(s) \leftarrow a \)
24: end if
25: end for
26: end for
27: end while

However, they just take into account single state reachability, while we use co-safe LTL goals, which are more general, and introduce the notion of task progression.

This approach greatly improves the efficiency of policy generation for our problem, when compared with our previously approach based on constrained multi-objective queries. It solves our illustrative problem in ~10 seconds, rather than the ~9 minutes it took previously.

Acknowledgement This work has received funding from EU FP7 under grant agreement No 600623, STRANDS.

References