Counterexamples in Probabilistic Model Checking

Tingting Han[1,2] and Joost-Pieter Katoen[1,2]

[1] Software Modelling and Verification, RWTH Aachen, Germany
[2] Formal Methods and Tools, University of Twente, The Netherlands

Abstract. This paper considers algorithms and complexity results for the generation of counterexamples in model checking of probabilistic until-formulae in discrete-time Markov chains (DTMCs). It is shown that finding the strongest evidence (i.e, the most probable path) that violates a (bounded) until-formula can be found in polynomial time using single-source (hop-constrained) shortest path algorithms. We also show that computing the smallest counterexample that is mostly deviating from the required probability bound can be found in a pseudo-polynomial time complexity by adopting a certain class of algorithms for the (hop-constrained) k shortest paths problem.

1 Introduction

A major strength of model checking is the possibility to generate counterexamples in case a property is violated. The shape of a counterexample depends on the temporal logic used. For LTL and the universal fragment of CTL, a single path through the system model suffices to indicate the refutation of the property. For existentially quantified path-formulae in logics such as CTL, either witnesses are provided (to indicate why the property holds), or more advanced structures such as trees of paths [10] or annotated paths [28] are provided as counterexample. Counterexamples are of utmost importance in model checking: they provide diagnostic feedback (also in cases where only a fragment of the entire model can be searched), they constitute the key to successful abstraction-refinement techniques [9], and are at the core of obtaining (optimal) schedules in e.g. timed model checking [7]. As a result, advanced counterexample generation and analysis techniques have intensively been investigated, see e.g., [23, 6, 12].

In probabilistic model checking, however, counterexample generation is almost not developed [2, 3]. Probabilistic model checking is a technique to verify system models in which transitions are equipped with randomness. Popular models are discrete- and continuous-time Markov chains (DTMCs and CTMCs), and variants thereof which exhibit nondeterminism. Efficient model-checking algorithms have been developed for these models, and have been applied to case studies from various application areas. The crux of probabilistic model checking is to combine techniques from numerical mathematics with standard reachability analysis. In this way, properties such as "the probability to reach a set of goal states is at most 0.6" can be automatically checked (up to a certain precision).

In the probabilistic setting, typically there is no single trace (but rather a set of them) that indicates why a given property is refuted. In case of a property refutation, current probabilistic model checkers produce a log file that shows the computed probability for all states. This information is too detailed to aid as a useful support in finding the cause(s). This paper considers algorithms and complexity results for the generation of counterexamples in model checking of (a safe fragment of) probabilistic CTL [20] on DTMCs. We concentrate on properties of the form $\mathcal{P}_{\leqslant p}(\Phi \mathcal{U}^{\leqslant h} \Psi)$. In case s refutes this formula, the probability of all paths in s satisfying $\Phi \mathcal{U}^{\leqslant h} \Psi$ exceeds p. We first consider the generation of *strongest evidences* for violation, i.e., paths satisfying $\Phi \mathcal{U}^{\leqslant h} \Psi$ that have the largest probability mass. Strongest evidences "contribute" mostly to the property refutation. For unbounded until (i.e., $h{=}\infty$), determining strongest evidences is equivalent to a standard shortest path (SP) problem; in case h is bounded, we obtain a special case of the (resource) constrained shortest path (CSP) problem [1] that can be solved in $\mathcal{O}(hm)$ where m is the number of transitions in the DTMC.

As most probable paths may have a very small probability mass, their information may be limited. As a next step, therefore, we consider the problem of determining most probable subtrees. Whereas in traditional model checking one is interested in the shortest counterexample, we consider trees of *smallest size* that exceed the probability bound *maximally*. The problem of generating such smallest, most indicative counterexamples can be casted as a k shortest paths problem. For unbounded-until formulae (i.e., $h{=}\infty$), it is shown that the generation of such smallest counterexamples can be found in pseudo-polynomial time by adopting k shortest path algorithms [14, 17, 26] that can compute k on the fly. For bounded until-formulae, we propose a variant of the recursive enumeration algorithm of Jiménez and Marzal [22]. The time complexity of this adapted algorithm is $\mathcal{O}(hm{+}hk\log(\frac{m}{n}))$, where n is the number of states in the DTMC.

2 Preliminaries

This section introduces DTMCs and the logic PCTL-safety.

2.1 DTMCs

Definition 1 (DTMCs). A (labelled) *discrete-time Markov chain* (DTMC) is a tuple $\mathcal{D} = (S, \mathbf{P}, L)$ where:

- S is a finite set of states;
- $\mathbf{P} : S \times S \to [0, 1]$ is a probability matrix satisfying $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ for all $s \in S$;
- $L : S \to 2^{AP}$ is a labelling function which assigns to each state $s \in S$ the set $L(s)$ of atomic propositions that are valid in s.

A state s in $\mathcal{D}$ is called absorbing if $\mathbf{P}(s, s) = 1$. W.l.o.g. we assume a DTMC to have a unique initial state.

Definition 2 (Paths). Let $\mathcal{D} = (S, \mathbf{P}, L)$ be a DTMC.

– An *infinite path* σ in $\mathcal{D}$ is an infinite sequence $s_0 \cdot s_1 \cdot s_2 \cdot ...$ of states such that $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geqslant 0$.
– A *finite path* in $\mathcal{D}$ is a finite prefix of an infinite path.

For state s and path σ, $\sigma \cdot s$ denotes the path obtained by extending σ by s. Let $|\sigma|$ denote the length of the path σ, i.e., $|s_0 \cdot s_1 \cdot ... \cdot s_n| = n$, $|s_0| = 0$ and $|\sigma| = \infty$ for infinite σ. For $0 \leqslant i \leqslant |\sigma|$, $\sigma[i] = s_i$ denotes the i-th state in σ. $\mathit{Path}(s)$ denotes the set of infinite paths that start in state s, formally, $\mathit{Path}(s) = \{\sigma \mid \sigma[0] = s\}$. Let $\mathit{Path}_{\mathit{fin}}(s)$ denotes the set of finite paths that start in state s.

A DTMC $\mathcal{D}$ enriched with an initial state s_0 induces a probability space. The underlying σ-algebra from the basic cylinders is induced by the finite paths starting in s_0. The probability measure $\Pr^{\mathcal{D}}_{s_0}$ (briefly Pr) induced by $(\mathcal{D}, s_0)$ is the unique measure on this σ-algebra where:

$$\Pr\{\underbrace{\sigma \in \mathit{Path}(s_0) \mid s_0 \cdot s_1 \cdot ... \cdot s_n \text{ is a prefix of } \sigma}_{\text{basic cylinder of the finite path } s_0 \cdot s_1 \cdot ... \cdot s_n}\} \;=\; \prod_{0 \leqslant i < n} \mathbf{P}(s_i, s_{i+1}).$$

Example 3. Fig. 1 illustrates a simple DTMC consisting of 10 states. s is the initial state, $AP = \{a, b\}$ and L is given through the subsets of AP labelling the states as $L(s) = L(s_i) = \{a\}$, for $1 \leqslant i \leqslant 4$; $L(t_1) = L(t_2) = L(t_3) = \{b\}$ and $L(u_1) = L(u_2) = \emptyset$. The DTMC contains no absorbing states. $\sigma_1 = s \cdot u_1 \cdot u_2 \cdot s_1 \cdot t_1 \cdot s_3$ is a finite path in this DTMC with $\Pr\{\sigma_1\} = 0.1 \times 0.4 \times 0.8 \times 0.1 \times 1 = 0.0032$ and $|\sigma_1| = 5$, $\sigma_1[3] = s_1$. $\sigma_2 = s \cdot s_3 \cdot (s_4 \cdot t_3)^{\omega}$ is an infinite path, $|\sigma_2| = \infty$.

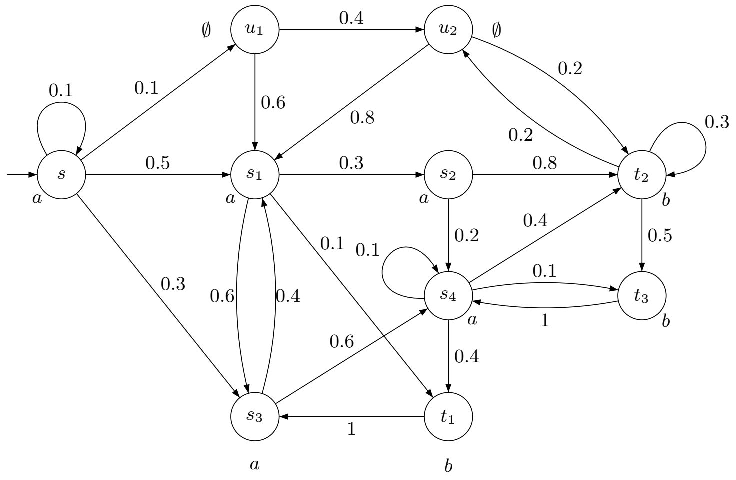


Fig. 1. An example DTMC

2.2 PCTL-safety

In this paper, counterexamples are explored when properties specified in the safety fragment of PCTL [5] are violated. The syntax and semantics of PCTL-safety are given and its expressiveness is illustrated by some examples.

Syntax. Let $p \in [0,1]$ and let AP denote a fixed, finite set of atomic propositions ranged over by $a, b, c, \ldots$. The syntax of PCTL-safety state formulae (in positive normal form, i.e., negations can only occur adjacent to atomic propositions.) is defined as follows:

$$\Phi ::= \mathrm{tt} \mid \mathrm{ff} \mid a \mid \neg a \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \mathcal{P}_{\leqslant p}(\phi)$$

where ϕ is path-formula defined by

$$\phi ::= \Phi \mathcal{U}^{\leqslant h} \Phi,$$

where $h \in \mathbb{N} \cup \{\infty\}$. We call the operator $\mathcal{U}^{\leqslant h}$ *unbounded until* if $h = \infty$ and abbreviate it as $\mathcal{U}$; and call it *bounded until* otherwise. For the sake of simplicity, we do not consider the next-operator. Note that the main distinction with PCTL [20] is that the probability bounds are upper bounds, and formulae are required to be in positive normal form.

As for CTL, the temporal operator $\Diamond^{\leqslant h}$ (eventually) can be derived as

$$\mathcal{P}_{\leqslant p}(\Diamond^{\leqslant h}\Phi) = \mathcal{P}_{\leqslant p}(\mathrm{tt}\ \mathcal{U}^{\leqslant h}\ \Phi)$$

The dual form is defined as $\mathcal{P}_{>p}(\Box^{\leqslant h}\Phi) = \mathcal{P}_{\leqslant 1-p}(\Diamond^{\leqslant h}\neg\Phi)$. Note that the negation can be "pushed" inside Φ until finally it is adjacent to an atomic proposition (see [5])[1]. For the more general case, $\mathcal{P}_{>p}(\Phi\mathcal{W}^{\leqslant h}\Psi) = \mathcal{P}_{\leqslant 1-p}(\neg\Psi\mathcal{U}^{\leqslant h}\neg(\Phi \vee \Psi))$. Thus in the PCTL-safety fragment, four forms $\mathcal{P}_{\leqslant p}(\Phi\mathcal{U}^{\leqslant h}\Psi)$, $\mathcal{P}_{>p}(\Phi\mathcal{W}^{\leqslant h}\Psi)$, $\mathcal{P}_{\leqslant p}(\Diamond^{\leqslant h}\Phi)$ and $\mathcal{P}_{>p}(\Box^{\leqslant h}\Phi)$ are allowed.

Semantics. Let DTMC $\mathcal{D} = (S, \mathbf{P}, L)$. The semantics of PCTL-safety is defined by a satisfaction relation, denoted $\models$, which is characterized as the least relation over the states in S (infinite paths in $\mathcal{D}$, respectively) and the state formulae (path formulae). The semantics of the propositional fragment is identical to that for CTL. The meaning of the probabilistic operator is formalized as follows [20]. The semantics of PCTL-safety state formulae is defined for path formula ϕ as:

$s \models \mathrm{tt}$	iff	*true*	$s \models \Phi \vee \Psi$	iff	$s \models \Phi$ or $s \models \Psi$
$s \models \mathrm{ff}$	iff	*false*	$s \models \Phi \wedge \Psi$	iff	$s \models \Phi$ and $s \models \Psi$
$s \models a$	iff	$a \in L(s)$	$s \models \mathcal{P}_{\leqslant p}(\phi)$	iff	$Prob(s, \phi) \leqslant p$
$s \models \neg a$	iff	$a \notin L(s)$			

[1] During the transformation, arbitrary PCTL-formulae can occur, but the result is again a PCTL-safety formula.

Let $\mathit{Path}(s, \phi)$ denote the set of infinite paths that start in state s and satisfy ϕ. Formally, $\mathit{Path}(s, \phi) = \{\sigma \in \mathit{Path}(s) \mid \sigma \models \phi\}$. Here, $\mathit{Prob}(s, \phi) = \Pr\{\sigma \in \mathit{Path}(s, \phi)\}$ denotes the probability of $\mathit{Path}(s, \phi)$. Let σ be an infinite path in $\mathcal{D}$. The semantics of PCTL-safety path formulae is defined as:

$$\sigma \models \Phi \mathcal{U}^{\leqslant h} \Psi \quad \text{iff} \quad \exists i \leqslant h \text{ such that } \sigma[i] \models \Psi \text{ and } \forall j : 0 \leqslant j < i.(\sigma[j] \models \Phi).$$

For finite path σ, $\models$ is defined in a similar way by changing the range of i to $i \leqslant \max\{h, |\sigma|\}$. Let $\mathit{Path}_{\mathit{fin}}(s, \phi)$ denote the set of finite paths starting in s that fulfill ϕ.

Example 4. We give some examples to illustrate how the PCTL-safety formulae are utilized to specify system properties.

- In Fig. 1, $\mathcal{P}_{\leqslant 0.27}(a\mathcal{U}b)$ asserts that the probability of reaching a b-state via an a-path is at most 0.27.
- Let *error* be an atomic proposition that characterizes any state in which a system error has occurred. Then $\mathcal{P}_{\leqslant 0.001}(\Diamond^{\leqslant 50} \mathit{error})$ asserts that the probability for a system error to occur within 50 steps is at most 0.001. Dually, $\mathcal{P}_{>0.999}(\Box^{\leqslant 50} \neg \mathit{error})$ states that the probability for not having a system error (running successfully) within 50 steps exceeds 0.999.
- Let *red* and *green* be two atomic propositions. $\mathcal{P}_{>0.8}(\mathit{green}\ \mathcal{W}\ \mathit{red})$ asserts that the probability of either being green forever, or reaching a red state via a green path, is greater than 0.8. Stated differently, with probability at most 0.2, a state is reached that is neither red nor green via a path that does not contain a red state, which can be specified by the dual formula $\mathcal{P}_{\leqslant 0.2}((\neg \mathit{red})\mathcal{U}(\neg \mathit{green} \wedge \neg \mathit{red}))$.

3 Counterexamples in a probabilistic setting

Let us first consider what a counterexample actually is. To that end, consider the formula $\mathcal{P}_{\leqslant p}(\phi)$, where ϕ is a path-formula. If state s refutes $\mathcal{P}_{\leqslant p}(\phi)$:

$$\begin{array}{ll} & s \not\models \mathcal{P}_{\leqslant p}(\phi) \\ \text{iff} & \text{not } (\mathit{Prob}(s, \phi) \leqslant p) \\ \text{iff} & \mathit{Prob}(s, \phi) > p \\ \text{iff} & \Pr\{\sigma \mid \sigma \in \mathit{Path}(s, \phi)\} > p \end{array}$$

So, $\mathcal{P}_{\leqslant p}(\phi)$ is refuted by state s whenever the total probability mass of all ϕ-paths that start in s exceeds p. This indicates that a counterexample for $\mathcal{P}_{\leqslant p}(\phi)$ is in general a *set* of paths satisfying ϕ. As ϕ is an until-formula and can be witnessed by finite state sequences, finite paths do suffice in counterexamples. As a counterexample should exceed p, a maximally probable ϕ-path is a strong evidence for the violation of $\mathcal{P}_{\leqslant p}(\phi)$. For counterexamples that are as small as possible, i.e., that contain the smallest possible set of paths indicating the refutation, such maximally probable paths are essential.

Definition 5 (Strongest evidence). A *strongest evidence* for violating $\mathcal{P}_{\leqslant p}(\phi)$ in state s is a finite path $\sigma \in \mathit{Path}_{\mathit{fin}}(s, \phi)$ such that $\Pr\{\sigma\} \geqslant \Pr\{\sigma'\}$ for any $\sigma' \in \mathit{Path}_{\mathit{fin}}(s, \phi)$.

Dually, a strongest evidence for violating $\mathcal{P}_{\leqslant p}(\phi)$ is a strongest witness for $\mathcal{P}_{>p}(\phi)$ [2]. Note that a strongest evidence does not need to be a counterexample as its probability mass may be (far) below p. A counterexample is defined as follows:

Definition 6 (Counterexample). A *counterexample* for $\mathcal{P}_{\leqslant p}(\phi)$ in state s is a set C of paths such that $C \subseteq \mathit{Path}_{\mathit{fin}}(s, \phi)$ and $\Pr(C) > p$.

Note that counterexamples are always finite as we consider non-strict upper-bounds in the probability operator[3]. Let $CX_p(s, \phi)$ be the set of counterexamples for $\mathcal{P}_{\leqslant p}(\phi)$ in state s. For $C \in CX_p(s, \phi)$ and C's superset C': $C \subseteq C' \subseteq \mathit{Path}_{\mathit{fin}}(s, \phi)$, it follows that $C' \in CX_p(s, \phi)$, since $\Pr(C') \geqslant \Pr(C) > p$. A counterexample is called *minimal* if it is minimal w.r.t. $\subseteq$. Note that a counterexample for state s is a set of finite paths that all start in s, and thus can be considered as a finite tree rooted at s.

As in conventional model checking, we are not interested in generating arbitrary counterexamples, but those that are easy to comprehend, and provide a clear evidence of the refutation of the formula. So, akin to shortest counterexamples we define the notion of a smallest, most indicative counterexample. Such smallest counterexamples should contain as few paths as possible—allowing easier analysis of the cause of refutation—but whose probability is clearly exceeding p.

Definition 7 (Smallest counterexample). $C \in CX_p(s, \phi)$ is a *smallest (most indicative) counterexample* if:

1. $|C| \leqslant |C'|$, for any $C' \in CX_p(s, \phi)$ and
2. $\Pr(C) \geqslant \Pr(C'')$, for any $C'' \in CX_p(s, \phi)$ and $|C| = |C''|$.

Intuitively, a smallest counterexample is mostly deviating from the required probability bound given that it has the smallest number of paths. Any smallest counterexample is minimal, but not necessarily the reverse. Note that the strongest evidence, minimal counterexample or smallest counterexample may not be unique, as paths may have equal probability. As a result, not every strongest evidence is contained in a minimal (or smallest) counterexample. However, any smallest counterexample contains at least one strongest evidence.

Example 8. In the DTMC in Fig. 1, we have $s \not\models \mathcal{P}_{\leqslant 0.27}(a\mathcal{U}b)$. Let $\sigma_1 = s{\cdot}s_1{\cdot}s_2{\cdot}t_2$, $\sigma_2 = s{\cdot}s_1{\cdot}s_3{\cdot}s_4{\cdot}t_1$, $\sigma_3 = s{\cdot}s_1{\cdot}s_3{\cdot}s_4{\cdot}t_2$, $\sigma_4 = s{\cdot}s_3{\cdot}s_4{\cdot}t_1$, $\sigma_5 = s{\cdot}s_3{\cdot}s_4{\cdot}t_2$,

[2] $\mathcal{P}_{>p}(\phi)$ is a PCTL-formula, not necessarily a safe one.

[3] For strict upper bounds in the probability operator, i.e., $\mathcal{P}_{<p}(\phi)$, a counterexample C may contain infinite paths, since $\Pr(C) = p$ is a counterexample. The limit of the sum of path probabilities obeying a geometric distribution may equal p, but the limit requires infinite paths.

$\sigma_6 = s{\cdot}s_3{\cdot}s_4{\cdot}t_3$, where:

$\Pr\{\sigma_1\} = 0.12,\ \Pr\{\sigma_2\} = \Pr\{\sigma_3\} = \Pr\{\sigma_4\} = \Pr\{\sigma_5\} = 0.072,\ \Pr\{\sigma_6\} = 0.018.$

Path σ_1 is a strongest evidence, as it is the maximally probable path from s to $\{t_1, t_2, t_3\}$, i.e., b-states. The set $C_1 = \{\sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6\}$ with $\Pr(C_1) = 0.306$ is a counterexample, but neither a minimal counterexample, nor a smallest counterexample, as $C_2 = \{\sigma_2, \sigma_3, \sigma_4, \sigma_5\} \subset C_1$ with $\Pr(C_2) = 0.288$ is also a counterexample. C_2 is a minimal counterexample, since the probability of any proper subset of C_2 is less than 0.27. But C_2 is not a smallest counterexample, as $C_3 = \{\sigma_1, \sigma_2, \sigma_3, \sigma_4\}$ with $\Pr(C_3) = 0.336$ is a counterexample too and $|C_2| = |C_3|$ but $\Pr(C_3) > \Pr(C_2)$. In fact, any set containing the strongest evidence and any three paths in C_2 is a smallest counterexample.

In the remainder of the paper, we first consider the computation of strongest evidences. Formally, we consider the strongest evidence problem (SE), that for a given state s with $s \not\models \mathcal{P}_{\leqslant p}(\phi)$, determines the strongest evidence for this violation. Subsequently, we consider the corresponding smallest counterexample problem (SC). For both cases, we distinguish between until-formulae for which $h{=}\infty$ (unbounded until) and $h \in \mathbb{N}$ (bounded until) as distinctive algorithms are used for these cases.

4 From DTMC to a weighted digraph

Prior to finding strongest evidences or shortest counterexamples, we first modify the DTMC and turn it into a weighted directed graph. Let $\phi = \Phi\mathcal{U}^{\leqslant h}\Psi$, $h = \infty or h \in \mathbb{N}$ and $Sat(\Phi) = \{s \in S \mid s \models \Phi\}$ for any state-formula Φ. Due to the bottom-up traversal of the model-checking algorithm over the formula ϕ, we may assume that $Sat(\Phi)$ and $Sat(\Psi)$ are known.

Step 1: Adapting the DTMC. First, we make all states in the DTMC $\mathcal{D} = (S, \mathbf{P}, L)$ that neither satisfy Φ nor Ψ absorbing. Then we add an extra state t so that all the Ψ-states are equipped with a transition to t with probability 1 (all other outgoing transitions of Ψ-states are omitted). State t can thus only be reached via a Ψ-state. The thus obtained DTMC $\mathcal{D}' = (S', \mathbf{P}', L')$ has state space $S \cup \{t\}$ for $t \notin S$. The transition probabilities in $\mathcal{D}'$ are defined as follows:

$$\begin{cases} \mathbf{P}'(s', s') = 1 \text{ and } \mathbf{P}'(s', s'') = 0 \text{ for } s'' \neq s' & \text{if } s' \notin Sat(\Phi) \cup Sat(\Psi) \text{ or } s' = t \\ \mathbf{P}'(s', t) = 1 \text{ and } \mathbf{P}'(s', s'') = 0 \text{ for } s'' \neq t & \text{if } s' \in Sat(\Psi) \\ \mathbf{P}'(s', s'') = \mathbf{P}(s', s'') \text{ for } s'' \in S \text{ and } \mathbf{P}'(s', t) = 0 & \text{otherwise} \end{cases}$$

$L'(s') = L(s')$ for $s' \in S$ and $L'(t) = \{at_t\}$, where $at_t \notin L(s')$ for any $s' \in S$, i.e., at_t uniquely identifies being at state t. Remark that all the $(\neg\Phi \wedge \neg\Psi)$-states could be collapsed into a single state, but this is not further explored here. The time complexity of this transformation is $\mathcal{O}(n)$ where $n = |S|$.

It is evident that the validity of $\Phi\mathcal{U}^{\leqslant h}\Psi$ is not affected by this amendment of the DTMC. All paths in $\mathcal{D}'$ of length at most $h + 1$ that end in t satisfy

$\Phi\mathcal{U}^{\leqslant h}\Psi$ in $\mathcal{D}$. More precisely, any finite path satisfying $(\Phi \vee \Psi)\mathcal{U}^{\leqslant h+1} at_t$ in $\mathcal{D}'$ has a finite path in $\mathcal{D}$ satisfying $\Phi\mathcal{U}^{\leqslant h}\Psi$. The following lemma guarantees that the later results in $\mathcal{D}'$ also hold in $\mathcal{D}$.

Lemma 9. *Let $\sigma' = \sigma{\cdot}t$ be a path in $Path_{fin}(s)$ in $\mathcal{D}'$. Then:*

1. $\sigma' \models (\Phi \vee \Psi)\mathcal{U}^{\leqslant h+1} at_t$ *in DTMC* $\mathcal{D}'$;
2. $\sigma \models \Phi\mathcal{U}^{\leqslant h}\Psi$ *in DTMC* $\mathcal{D}$;
3. $\Pr\{\sigma\} = \Pr\{\sigma'\}$.

For the rest of the technical report, we suppose that all the DTMCs are the results of Step 1, and the logic formula considered is $(\Phi \vee \Psi)\mathcal{U}^{\leqslant h+1} at_t$.

Step 2: Conversion into a weighted digraph. As a second preprocessing step, the DTMC obtained in the first phase is transformed into a weighted digraph. Recall that a weighted digraph is a tuple $\mathcal{G} = (V, E, w)$ where V is a finite set of vertices, $E : V \times V$ is a set of edges, and $w : E \to \mathbb{R}_{\geqslant 0} \cup \{\infty\}$ is a function assigning non-negative weights to edges.

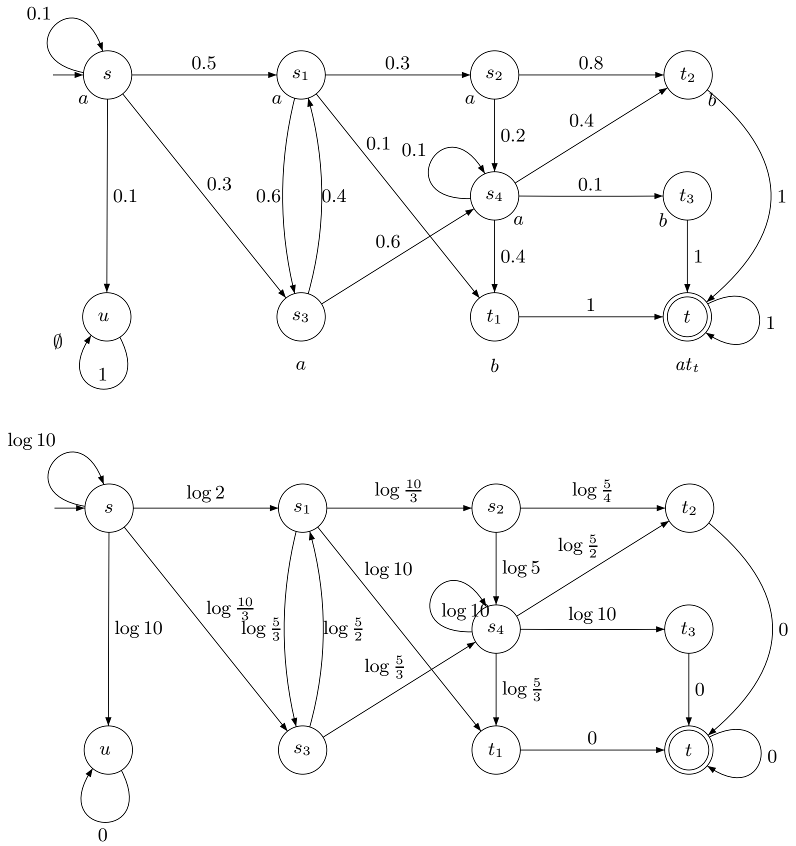
Definition 10 (Weighted digraph of a DTMC). For DTMC $\mathcal{D} = (S, \mathbf{P}, L)$, the weighted digraph $\mathcal{G}_D = (V, E, w)$ where:

- $V = S$ and $(v, v') \in E$ iff $\mathbf{P}(v, v') > 0$, and
- $w(v, v') = \begin{cases} \log(\mathbf{P}(v, v')^{-1}) & \text{if } \mathbf{P}(v, v') > 0, \\ \infty & \text{otherwise.} \end{cases}$

Note that in any DTMC, $\mathbf{P}(s, s') \in [0, 1]$, thus $\mathbf{P}(s, s')^{-1} \in [1, +\infty)$, and consequently, $\log(\mathbf{P}(s, s')^{-1}) \in [0, +\infty)$. Thus, we indeed obtain a non-negatively weighted digraph. Note that this transformation can be done in $\mathcal{O}(m)$, where $m = |\mathbf{P}|$.

Example 11. The transformation from DTMC in Fig. 1 to a weighted digraph is illustrated in Fig. 2. For the path formula $\phi = a\mathcal{U}b$, in Fig. 2(a) all the b-states (i.e., t_1, t_2, t_3) are made absorbing and take a transition with probability 1 into the new state t (indicated by a double circle). All the $(\neg a \wedge \neg b)$-states (i.e., u_1, u_2) are made absorbing and then (to simplify the figure) collapsed into one state u. In Fig. 2(b), the resulting weighted digraph is depicted where all the states remain the same, however the edge weights are obtained by taking the logarithm of the reciprocal of the corresponding transition probabilities.

A path σ from s to t in $\mathcal{G}$ is a sequence $\sigma = v_0{\cdot}v_1{\cdot}\ldots{\cdot}v_j \in V^+$, where $v_0 = s, v_j = t$ and $(v_i, v_{i+1}) \in E$, for $0 \leqslant i < |\sigma|$. As for paths in DTMCs, $|\sigma|$ denotes the length of σ. The *distance* of finite path $\sigma = v_0{\cdot}v_1{\cdot}\ldots{\cdot}v_j$ in graph $\mathcal{G}$ is $d(\sigma) = \sum_{i=0}^{j-1} w(v_i, v_{i+1})$. Due to the fact that multiplication of probabilities in $\mathcal{D}$ corresponds to addition of weights in $\mathcal{G}_D$, and that weights are based on taking the logarithm of the reciprocal of the transition probabilities in $\mathcal{D}$, distances in $\mathcal{G}$ and path-probabilities in DTMC $\mathcal{D}$ are related as follows.



(a) result of Step 1

(b) result of Step 2

Fig. 2. Transformation from DTMC to weighted digraph

Lemma 12. *Let σ and σ' be finite paths in DTMC $\mathcal{D}$ and its graph $\mathcal{G}_D$. Then:*

$$\Pr\{\sigma'\} \geqslant \Pr\{\sigma\} \quad \textit{iff} \quad d(\sigma') \leqslant d(\sigma).$$

Proof. Consider the following finite path σ in $\mathcal{D}$:

$$\sigma = s_0 \xrightarrow{\mathbf{P}(s_0,s_1)} s_1 \xrightarrow{\mathbf{P}(s_1,s_2)} s_2 \cdots s_{n-1} \xrightarrow{\mathbf{P}(s_{n-1},s_n)} s_n$$

By definition, its probability is:

$$\Pr\{\sigma\} = \mathbf{P}(s_0, s_1) \cdot \mathbf{P}(s_1, s_2) \cdots \mathbf{P}(s_{n-1}, s_n) = \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})$$

Consider an alternative path σ':

$$\sigma' = s'_0 \xrightarrow{\mathbf{P}(s_0,s'_1)} s'_1 \xrightarrow{\mathbf{P}(s'_1,s'_2)} s'_2 \cdots s'_{m-1} \xrightarrow{\mathbf{P}(s'_{m-1},s'_m)} s'_m$$

$$\Pr\{\sigma'\} = \mathbf{P}(s'_0, s'_1) \cdot \mathbf{P}(s'_1, s'_2) \cdots \mathbf{P}(s'_{m-1}, s'_m) = \prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1})$$

If $\Pr\{\sigma'\} \geqslant \Pr\{\sigma\}$, we have:

$$\begin{array}{lc} & \prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1}) \geqslant \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1}) \\ \Longleftrightarrow & \dfrac{1}{\prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1})} \leqslant \dfrac{1}{\prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})} \\ \Longleftrightarrow & \log(\dfrac{1}{\prod_{i=0}^{m-1} \mathbf{P}(s'_i, s'_{i+1})}) \leqslant \log(\dfrac{1}{\prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})}) \\ \Longleftrightarrow & \sum_{i=0}^{m-1} \log(\dfrac{1}{\mathbf{P}(s'_i, s'_{i+1})}) \leqslant \sum_{i=0}^{n-1} \log(\dfrac{1}{\mathbf{P}(s_i, s_{i+1})}) \\ \Longleftrightarrow & \sum_{i=0}^{m-1} w(s'_i, s'_{i+1}) \leqslant \sum_{i=0}^{n-1} w(s_i, s_{i+1}) \\ \Longleftrightarrow & d(\sigma^*) \leqslant d(\sigma) \end{array}$$

□

Note that Lemma 12 also holds for infinite paths. The following lemma specifies the correspondence between paths in DTMC D and its weighted digraph.

Lemma 13. *For any path σ from s to t in DTMC $\mathcal{D}$ and $k > 0$:*

1. *σ is a k-th most probable path in $\mathcal{D}$ iff σ is a k-th shortest path in $\mathcal{G}_D$;*
2. *σ is a k-th most probable path of at most h hops in $\mathcal{D}$ iff σ is a k-th shortest path of at most h hops in $\mathcal{G}_D$;*

The correspondence between path probabilities in the DTMC and distances in its weighted digraph constitutes the basis for the remaining algorithms.

5 Finding strongest evidences

This section considers algorithms for determining strongest evidences, i.e., maximally probable paths.

5.1 Unbounded until

Based on the results of Lemma 13.1 where $k = 1$, we consider the shortest path problem.

Definition 14 (SP problem). Given a weighted digraph $\mathcal{G} = (V, E, w)$ and $s, t \in V$, the *shortest path* (SP) problem is to determine a path σ from s to t such that $d(\sigma) \leqslant d(\sigma')$ for any path σ' from s to t in $\mathcal{G}$.

From Lemma 13.1 together with the transformation of a DTMC into a weighted digraph, it follows that there is a polynomial reduction from the SE problem for unbounded until to the SP problem. As the SP problem is known to be in PTIME, it follows:

Theorem 15. *The SE problem for unbounded until is in PTIME.*

Various efficient algorithms [13, 8, 15, 11] exist for the SP problem, e.g., when using Dijkstra's algorithm, the SE problem for unbounded until can be solved in time $\mathcal{O}(m + n \log n)$ when using appropriate data structures such as Fibonacci heaps.

5.2 Bounded until

Lemma 13.2 when $k = 1$ applies when considering maximally probable paths of a certain maximal hop count. This suggests to consider the hop-constrained shortest path problem.

Definition 16 (HSP problem). Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and $h \in \mathbb{N}$, the *hop-constrained SP* (HSP) problem is to determine a path σ in $\mathcal{G}$ from s to t with $|\sigma| \leqslant h$ such that $d(\sigma) \leqslant d(\sigma')$ for any path σ' from s to t with $|\sigma'| \leqslant h$.

The HSP problem is a special case of the constrained shortest path (CSP) problem [27, 1], where the only constraint is the hop count.

Definition 17 (CSP problem). Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and resource constraints λ^i, for $1 \leqslant i \leqslant c$. Edge $e \in E$ uses $r^i(e) \geqslant 0$ units of resource i. The *(resource) constrained shortest path* problem (CSP) is to determine a shortest path σ in $\mathcal{G}$ from s to t such that $\sum_{e \in \sigma} r^i(e) \leqslant \lambda^i$ for $1 \leqslant i \leqslant c$.

The CSP problem is NP-complete, even for a single resource constraint [1]. However, if each edge uses a constant unit of that resource (such as the hop count), the CSP problem can be solved in polynomial time, cf. [18], problem ND30. Thus:

Theorem 18. *The SE problem for bounded until is in PTIME.*

For $h \geqslant n-1$, it is possible to use Dijkstra's SP algorithm (as for unbounded until), as a shortest path does not contain cycles. If $h < n-1$, however, Dijkstra's algorithm does not guarantee to obtain a shortest path of at most h hops. We, therefore, adopt the Bellman-Ford (BF) algorithm [8, 15, 11] which fits well to our problem as it proceeds by increasing hop count. It can be readily modified to generate a shortest path within a given hop count. In the sequel of the paper, this algorithm is generalised for computing shortest counterexamples. The BF-algorithm is based on a set of recursive equations; we extend these with the hop count h. For $v \in V$, let $\pi(v,h)$ denote the shortest path from s to v of at most h hops (if it exists). Then:

$$\pi(v,h) = \begin{cases} s & \text{if } v = s \text{ and } h \geqslant 0; & (1a) \\ \perp & \text{if } v \neq s \text{ and } h = 0; & (1b) \\ \arg\min_u \{d(\pi(u,h{-}1) \cdot v) \mid (u,v) \in E\} & \text{if } v \neq s \text{ and } h > 0. & (1c) \end{cases}$$

where $\perp$ denotes undefined. The last clause states that $\pi(v,h)$ consists of the shortest path to v's predecessor u, i.e., $\pi(u,h{-}1)$, extended with edge (u,v). Note that $\min_u \{d(\pi(u,h{-}1) \cdot v) \mid (u,v) \in E\}$ is the distance of the shortest path; by means of arg, the path is obtained. It follows (cf. [24]) that equation (1) characterizes the shortest path from s to v in at most h hops, and can be solved in time $\mathcal{O}(hm)$. As $h < n{-}1$, this is indeed in PTIME. Recall that for $h \geqslant n{-}1$, Dijkstra's algorithm has a favorable time complexity.

Remark 19. Note that the self-loop of vertex t is neglected when computing the hop-constrained shortest path using BF algorithm. This is because the logic operator is the bounded until $\mathcal{U}^{\leqslant h}$ instead of point interval until $\mathcal{U}^{=h}$, so that once it reaches t within the hop bound, the path formula holds and the self-loop does not change the path probability.

Example 20. To illustrate the BF algorithm, we compute the shortest path in at most 4 hops from s to t in our example in Fig.2(b), i.e., $\pi(t,4)$. In order to compute $\pi(t,4)$, three predecessors of t are considered so that $\pi(t_1,3)$, $\pi(t_2,3)$, $\pi(t_3,3)$ are invoked. Again, to compute $\pi(t_1,3)$, two predecessors of t_1 are considered so that $\pi(s_1,2), \pi(s_4,2)$ are invoked. In sequel, $\pi(s,1)$ and $\pi(s_3,1)$ are invoked for $\pi(s_1,2)$, where $\pi(s,1)$ is s, defined by equation (1a). $\pi(s_3,1)$ is derived by invoking its two predecessors $\pi(s,0)$, which is s (by (1a)) and $\pi(s_1,0)$ which is $\perp$ (by (1b)). The computation is given in Fig. 3. Note that $\pi(s_4,2), \pi(s_3,1), \pi(s,0), \pi(s_1,0), \pi(s_3,0)$ are indicated more than once (with more than one incoming edge), but are (like in dynamic programming) computed only once.

Remark 21. Alternatively, the *Viterbi algorithm* [16, 29] for probabilistic automata can be applied to our problem. The Viterbi algorithm determines the most probable path that generates a given trace. Let $\mathcal{D}'$ be a DTMC that is obtained after the first step described in Section 4, and suppose that $L'(s)$ contains the set of atomic propositions that are valid in s and all subformulae of the formula under consideration. (Note that these labels are known due to the recursive descent nature of the PCTL model checking algorithm.) Let $tr(\sigma)$ denote the projection of a path $\sigma = s_0{\cdot}s_1{\cdot}\ldots{\cdot}s_h$ on its trace γ, i.e., $\gamma = tr(\sigma) = L'(s_0)L'(s_1)\cdots L'(s_h)$. $\sigma{\downarrow_i}$ denotes the prefix of path σ truncated at length i (thus ending in s_i), formally, $\sigma{\downarrow_i} = \sigma[0]{\cdot}\sigma[1]{\cdot}\ldots{\cdot}\sigma[i]$. Thus, $tr(\sigma{\downarrow_i}) = L'(s_0)L'(s_1)\cdots L'(s_i)$. $\gamma{\downarrow_i}$ denotes the prefix of trace γ with length i. Let $\rho(\gamma,i,v)$ denote the probability of the most probable path $\sigma{\downarrow_i}$ whose trace equals

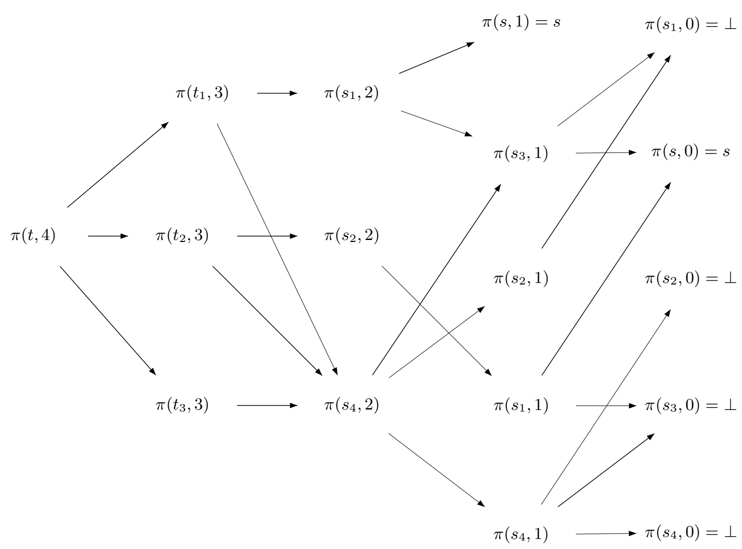


Fig. 3. An example run of the Bellman-Ford algorithm

$\gamma{\downarrow_i}$ and reaches state v. $\rho(\gamma, i, v)$ can be formally defined as follows:

$$\rho(\gamma, i, v) = \max_{tr(\sigma\downarrow_i)=\gamma_i} \prod_{j=0}^{i-1} \mathbf{P}(s_j, s_{j+1}) \cdot \mathbf{1}_v(s_i),$$

where $\mathbf{1}_v(s_i)$ is the characteristic function of v, i.e., $\mathbf{1}_v(s_i)$ returns 1, if $s_i = v$; 0, else.

The Viterbi algorithm provides an algorithmic solution to compute $\rho(\gamma, i, v)$:

$$\rho(\gamma, i, v) = \begin{cases} 1 & \text{if } s = v \text{ and } i = 0; \\ 0 & \text{if } s \neq v \text{ and } i = 0; \\ \max_{u \in S} \rho(\gamma, i-1, u) \cdot \mathbf{P}(u, v) & \text{otherwise.} \end{cases}$$

By computing $\rho(\Phi^h\Psi, h, s_h)$, the Viterbi algorithm determines the most probable h-hop path $\sigma = s_0{\cdot}s_1{\cdot}...{\cdot}s_h$ that generates the trace $\gamma = L'(s_0)L'(s_1)...L'(s_h)$ $= \Phi^h\Psi$ with length $(h{+}1)$. For our SE problem for bounded until, the trace of the most probable hop-constrained path from s to t is among $\{\Psi at_t, \Phi\Psi at_t, ..., \Phi^h\Psi at_t\}$. The self-loop at vertex t with probability 1 can make sure that all these paths have length $h{+}1$ but not change their probabilities, e.g., the path with trace Ψat_t can be extended so that the trace becomes $\Psi at_t{}^{h+1}$. We obtain the most probable path for $\Phi\mathcal{U}^{\leqslant h}\Psi$ by computing $\rho((\Phi{\vee}\Psi{\vee}at_t)^{h+1}at_t, h{+}1, t)$ using the Viterbi algorithm. The time complexity of the Viterbi algorithm is $\mathcal{O}(hm)$, as for the BF algorithm.

6 Finding smallest counterexamples

Recall that a smallest (most indicative) counterexample is a counterexample of minimal cardinality, whose probability deviates maximally from the required probability bound. In this section, we investigate algorithms for computing such smallest counterexamples. First observe that any smallest counterexample that contains, say k paths, contains the k most probable paths. This follows from the fact that any non-k most probable path can be exchanged with a more probable path, without changing the size of the counterexample, but by increasing its probability.

6.1 Unbounded until

Lemma 13.1 is applicable here. This suggests to consider the k shortest paths problem.

Definition 22 (KSP problem). Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$, and $k \in \mathbb{N}$, the k *shortest paths* (KSP) problem is to find k distinct shortest paths between s and t in G, if such paths exist.

Theorem 23. *The SC problem for unbounded until is a KSP problem.*

Proof. We prove that a smallest counterexample of size k, contains k most probable paths. It is proven by contradiction. Let C be a smallest counterexample for ϕ with $|C| = k$, and assume C does not contain the k most probable paths satisfying ϕ. Then there is a path $\sigma \notin C$ satisfying ϕ such that $\Pr\{\sigma\} > \Pr\{\sigma'\}$ for some $\sigma' \in C$. Let $C' = C \setminus \{\sigma'\} \cup \{\sigma\}$. Then C' is a counterexample for ϕ, $|C| = |C'|$ and $\Pr(C) > \Pr(C')$. This contradicts C being a smallest counterexample. □

The question remains how to obtain k. Various algorithms for the KSP problem require k to be known a priori. This is inapplicable in our setting, as the number of paths in a shortest counterexample is implicitly provided by the probability bound in the PCTL-formula. We therefore consider algorithms that allow to determine k on the fly, i.e., that can halt at any k and resume if necessary. A good candidate is Eppstein's algorithm [14]. Although this algorithm has the best known asymptotic time complexity, viz. $\mathcal{O}(m+n \log n+k)$, in practice the recursive enumeration algorithm (REA) by Jiménez and Marzal [22] prevails. This algorithm has a time complexity in $\mathcal{O}(m+kn \log \frac{m}{n})$ and is based on a generalisation of the recursive equations for the BF-algorithm, and is readily adaptable to the case for bounded h, as we demonstrate below. Note that the time complexity of KSP algorithms depends on k, and as k may be exponential, their complexity is *pseudo*-polynomial.

6.2 Bounded until

Similar to the bounded until case for strongest evidences, we now consider the KSP problem where the length of paths is constrained, as Lemma 13.2 is applicable here.

Definition 24 (HKSP problem). Given a weighted digraph $\mathcal{G} = (V, E, w)$, $s, t \in V$ and $h, k \in \mathbb{N}$, the *hop-constrained* KSP (HKSP) problem is to determine k shortest paths each of length at most h between s and t.

Similar to Theorem 23 we obtain:

Theorem 25. *The SC problem for bounded until is a HKSP problem.*

To our knowledge, algorithms for the HKSP problem do not exist. In order to solve the HKSP problem, we propose a new algorithm that is strongly based on Jiménez and Marzal's REA algorithm [22]. The advantage of adapting this algorithm is that k can be determined on the fly, an essential characteristic for our setting. The algorithm is a conservative extension of the REA algorithm.

For $v \in V$, let $\pi^k(v,h)$ denote the k-th shortest path from s to v of length at most h (if it exists). As before, we use $\perp$ to denote the non-existence of a path. We establish the following equations:

$$\pi^k(v,h) = \begin{cases} s & \text{if } k=1, v=s \text{ and } h \geqslant 0 \quad (2a) \\ \perp & \text{if } (k>1, v=s, h=0) \text{ or } (v \neq s, h=0) \quad (2b) \\ \arg\min_\sigma\{d(\sigma) \mid \sigma \in Q^k(v,h)\} & \text{otherwise} \quad (2c) \end{cases}$$

where $Q^k(v,h)$ is a set of candidate paths among which $\pi^k(v,h)$ is chosen. The candidate sets are defined by:

$$Q^k(v,h) = \begin{cases} \{\pi^1(u,h{-}1)\cdot v \mid (u,v) \in E\} \\ \qquad \text{if } k=1, v \neq s \text{ or } k=2, v=s \\ (Q^{k-1}(v,h) - \{\pi^{k'}(u,h{-}1)\cdot v\}) \cup \{\pi^{k'+1}(u,h{-}1)\cdot v\} \\ \qquad \text{if } k>1 \text{ and } u, k' \text{ are the node and index,} \\ \qquad \text{such that } \pi^{k-1}(v,h) = \pi^{k'}(u,h{-}1)\cdot v \end{cases} \tag{3}$$

Assume that the path $\pi^{k'+1}(u,h{-}1)\cdot v{=}\perp$ if it does not exist, which happens when $Q^{k'+1}(u,h{-}1){=}\emptyset$. Note that $\perp\cdot v{=}\perp$ for any $v \in V$. $Q^k(v,h){=}\emptyset$ if it only contains $\perp$.

If $k{=}1$, the shortest path to v's predecessor u is extended with the edge to v. In the latter clause, $\pi^{k'}(u,h{-}1)$ denotes the selected $(k{-}1)$-st shortest path from s to u, where u is the direct predecessor of v. Paths in $Q^k(v,h)$ for $k>1$ are thus either candidate paths for $k{-}1$ where the selected path is eliminated (first summand) or the $(k'{+}1)$-st shortest path from s to u extended with edge (u,v) (second summand). Note that for the source state s, there is no need to define $Q^k(s,h)$ as $\pi^k(s,h)$ is defined by equations $(2a)$ and $(2b)$, which act as termination conditions.

Proposition 26. *The equations* (2*a*)-(2*c*) *and* (3) *characterize the hop-constrained* k *shortest paths from* s *to* v *in at most* h *hops.*

Proof. Let $\mathcal{R}^k(v,h)$ denote the set of the k shortest paths from s to v in at most h hops. Each path in $\mathcal{R}^k(v,h)$ reaches v from some vertex $u \in \mathit{Pred}(v)=\{w \in V \mid (w,v) \in E\}$. In order to compute $\pi^k(v,h)$, we should consider for every $u \in \mathit{Pred}(v)$, *all paths* from s to u that do not yield a path in $\mathcal{R}^{k-1}(v,h)$. However, since $k_1{<}k_2$ implies that $d(\pi^{k_1}(u,h{-}1)){+}w(u,v) \leqslant d(\pi^{k_2}(u,h{-}1)){+}w(u,v)$, only *the shortest* of these paths needs to be taken into account when computing $\pi^k(v,h)$. Thus we can associate to (v,h) a set of candidate paths $Q^k(v,h)$ among which $\pi^k(v,h)$ can be chosen, that contains at most one path for each predecessor $u \in \mathit{Pred}(v)$. This set Q^k is recursively defined by equation (3). $\square$

The adapted REA. The adapted REA for computing the k shortest paths from s to t which each consist of at most h hops is sketched as follows. The algorithm is based on the recursive equations given just above.

i Compute $\pi^1(t,h)$ by the BF algorithm and set $k := 1$.

ii Repeat until $\pi^k(t,h)$ does not exist or $\displaystyle\sum_{i=1}^{k} \Pr\{\pi^i(t,h)\} > p$:

  (a) Set $k := k{+}1$ and compute $\pi^k(t,h)$ by invoking $\mathit{NextPath}(v,h,k)$.

For $k{>}1$, and once $\pi^1(v,h),\ldots,\pi^{k-1}(v,h)$ are available, $\mathit{NextPath}(t,h,k)$ computes $\pi^k(v,h)$ as follows:

1. If $h{\leqslant}0$, goto step 4.
2. If $k{=}2$, then set $Q[v,h] := \{\pi^1(u,h{-}1){\cdot}v \mid (u,v) \in E \text{ and } \pi^1(v,h) \neq \pi^1(u,h{-}1){\cdot}v\}$.
3. Let u and k' be the node and index such that $\pi^{k-1}(v,h) = \pi^{k'}(u,h{-}1){\cdot}v$.
   (a) If $\pi^{k'+1}(u,h{-}1)$ has not yet been computed, invoke $\mathit{NextPath}(u,h{-}1,k'{+}1)$.
   (b) If $\pi^{k'+1}(u,h{-}1)$ exists, then insert $\pi^{k'+1}(u,h{-}1){\cdot}v$ in $Q[v,h]$.
4. If $Q[v,h] \neq \emptyset$, then select and delete a path with minimum weight from $Q[v,h]$ and assign it to $\pi^k(v,h)$, else $\pi^k(v,h)$ does not exist.

In the main program, first the shortest path from s to t is determined using, e.g., the BF-algorithm. The intermediate results are recorded, e.g., all the paths in Fig. 3. Then, the k shortest paths are determined iteratively using the subroutine *NextPath*. The computation terminates when the k-th shortest path does not exist, or the total probability mass of the k shortest paths so far exceeds the bound p. Recall that p is the lower bound of the PCTL formula to be checked. Note that $Q[v,h]$ in the algorithm corresponds to $Q^k(v,h)$, where k is the parameter of the program. In steps 2 through 3, the set $Q^k(v,h)$ is determined from $Q^{k-1}(v,h)$ according to equation (3). In the final step, $\pi^k(v,h)$ is selected from $Q^k(v,h)$ according to equation (2*c*).

Example 27. We illustrate how the algorithm progresses by computing 3 shortest paths in Fig. 2 (b) with hop count $h = 4$, i.e., $\pi^1(t,4)$, $\pi^2(t,4)$ and $\pi^3(t,4)$.

By BF algorithm, we compute $\pi^1(t,4) = s{\cdot}s_1{\cdot}s_2{\cdot}t_2{\cdot}t$, several shortest paths to different destinations with different hop constraints are derived as a by-product as we showed in Example 20, where $\pi(v,h)$ in the previous example is $\pi^1(v,h)$ here. We summarize the paths below:

$h=3$	$h=2$	$h=1$	$h=0$
		$\pi^1(s,1)=s{\cdot}s$	$\pi^1(s,0)=s$
$\pi^1(t_1,3)=s{\cdot}s_1{\cdot}t_1$	$\pi^1(s_1,2)=s{\cdot}s_1$	$\pi^1(s_1,1)=s{\cdot}s_1$	$\pi^1(s_1,0)=\perp$
$\pi^1(t_2,3)=s{\cdot}s_1{\cdot}s_2{\cdot}t_2$	$\pi^1(s_2,2)=s{\cdot}s_1{\cdot}s_2$	$\pi^1(s_2,1)=\perp$	$\pi^1(s_2,0)=\perp$
$\pi^1(t_3,3)=s{\cdot}s_3{\cdot}s_4{\cdot}t_3$	$\pi^1(s_4,2)=s{\cdot}s_3{\cdot}s_4$	$\pi^1(s_3,1)=s{\cdot}s_3$	$\pi^1(s_3,0)=\perp$
		$\pi^1(s_4,1)=\perp$	$\pi^1(s_4,0)=\perp$

The algorithm for $\pi^2(t,4)$ is running in Fig. 4.

$k=2$, invoke $\mathit{NextPath}(t,4,2)$:
2. set $Q[t,4]=\{\pi^1(t_1,3){\cdot}t,\ \pi^1(t_3,3){\cdot}t\}$. /*$t_1,t_2,t_3$ are predecessors of t*/
3. $u=t_2$, $k'=1$, $h=3$. /* $\pi^1(t,4)=\pi^1(t_2,3){\cdot}t=s{\cdot}s_1{\cdot}s_2{\cdot}t_2{\cdot}t$*/
3(a). compute $\pi^2(t_2,3)$ by invoking $\mathit{NextPath}(t_2,3,2)$:
    2. $(k=2)$ set $Q[t_2,3]=\{\pi^1(s_4,2){\cdot}t_2\}$.
    3. $u=s_2$, $k'=1$, $h=2$. /* $\pi^1(t_2,3)=\pi^1(s_2,2){\cdot}t_2=s{\cdot}s_1{\cdot}s_2{\cdot}t_2$ */
    3(a). compute $\pi^2(s_2,2)$ by invoking $\mathit{NextPath}(s_2,2,2)$:
        2. $(k=2)$ set $Q[s_2,2]=\emptyset$.
        3. $u=s_1$, $k'=1$, $h=1$. /* $\pi^1(s_2,2)=\pi^1(s_1,1){\cdot}s_2=s{\cdot}s_1{\cdot}s_2$ */
        3(a). compute $\pi^2(s_1,1)$ by invoking $\mathit{NextPath}(s_1,1,2)$:
            2 $(k=2)$ set $Q[s_1,1]=\emptyset$.
            3. $u=s$, $k'=1$, $h=0$. /* $\pi^1(s_1,1)=\pi^1(s,0){\cdot}s_1=s{\cdot}s_1$ */
            3(a). compute $\pi^2(s,0)$ by invoking $\mathit{NextPath}(s,0,2)$:
                1. Goto 4.
                4. $\pi^2(s,0)=\perp$.
            3(b). $Q[s_1,1]=\emptyset$.
            4. $\pi^2(s_1,1)=\perp$.
        3(b). $Q[s_2,2]=\emptyset$.
        4. $\pi^2(s_2,2)=\perp$.
    3(b). $Q[t_3,3]=\{\pi^1(s_4,2){\cdot}t_2\}$.
    4. $\pi^2(t_2,3)=s{\cdot}s_3{\cdot}s_4{\cdot}t_2$.
3(b)*. $Q[t,4]=\{\pi^1(t_1,3){\cdot}t,\ \pi^1(t_3,3){\cdot}t,\ \pi^2(t_3,3){\cdot}t\}$. /*$d(\pi^1(t_1,3){\cdot}t)=d(\pi^2(t_3,3){\cdot}t)$*/
4. $\pi^2(t,4)=\pi^1(t_1,3){\cdot}t=s{\cdot}s_3{\cdot}s_4{\cdot}t_1{\cdot}t$. /* $\pi^1(t_1,3){\cdot}t$ is the shortest in $Q[t,4]$ */

Fig. 4. An example of the adapted REA

Note that in order to compute $\pi^2(t,4)$, the vertices along $\pi^1(t,4)=s{\cdot}s_1{\cdot}s_2{\cdot}t_2{\cdot}t$ are computed successively, i.e., $\pi^2(t_2,3)$, $\pi^2(s_2,2)$, $\pi^2(s_1,1)$ and $\pi^2(s,0)$, by recursively invoking *NextPath*. The hop count of each invocation is decreased by 1, so that *NextPath* is invoked 5 times when $h=4$ and $k=2$. Similarly, to compute $\pi^3(t,4)$, the vertices along $\pi^2(t,4)=s{\cdot}s_3{\cdot}s_4{\cdot}t_1{\cdot}t$ ($\pi^2(t_1,3)$, $\pi^2(s_4,2)$, $\pi^2(s_3,1)$ and $\pi^2(s,0)$) are needed. Actually, in this particular example, $\pi^3(t,4)$ can be easily derived, since in Step $3(b)^*$, $d(\pi^1(t_1,3){\cdot}t)=d(\pi^2(t_3,3){\cdot}t)$, then $\pi^3(t,4)$ definitely equals $\pi^2(t_3,3){\cdot}t$.

Time complexity. Before we analyze the algorithm time complexity, we first prove that the recursive calls to *NextPath* to compute $\pi^k(t,h)$ visit, in the worst case, all the vertices in $\pi^{k-1}(t,h)$, which is at most h.

Lemma 28. *For $k{>}1$ and for all $v \in V$, the computation of $\pi^k(v,h)$ by means of NextPath(v,h,k) may recursively generate calls to NextPath$(u,h{-}1,j)$ only for vertices u in $\pi^{k-1}(v,h)$.*

Proof. Suppose $\pi^{k-1}(v,h) = u_1{\cdot}u_2{\cdot}...{\cdot}u_p$, where $u_1{=}s$ and $u_p{=}t$. For every $i = 1,...,p$, let k_i be the index such that $\pi^{k_i}(u_i) = u_1{\cdot}u_2{\cdot}...{\cdot}u_i$. Since $\pi^{k-1}(v,h) = \pi^{k_{p-1}}(u_{p-1},h{-}1){\cdot}v$, *NextPath*$(v,h,k)$ may require a recursive call to *NextPath*$(u_{p-1}, h{-}1, k_{p-1}{+}1)$ in case $\pi^{k_{p-1}+1}(u_{p-1},h{-}1)$ has not been already computed; since $\pi^{k_{p-1}}(u_{p-1},h{-}1) = \pi^{k_{p-2}}(u_{p-2},h{-}2){\cdot}u_{p-1}$, *NextPath*$(u_{p-1},h{-}1,k_{p-1}+1)$ may require a recursive call to *NextPath*$(u_{p-2},h{-}2,k_{p-2}{+}1)$; and so on. In the worst case, the recursive calls extend through the nodes $u_p, u_{p-1}, ..., u_1$. If the recursion reaches $\pi^1(s,h')(0{\leqslant}h'{\leqslant}h)$ or $\pi^{k'}(v,0)(k'{>}0, v{\neq}s)$ or $\pi^{k''}(s,0)(k'' > 1)$ so that the termination conditions in equation (2a) and (2b) or Algorithm Step 1 hold, then no more recursive calls are performed. ☐

To determine the computational complexity of the algorithm, we assume the candidate sets to be implemented by heaps (as in [22]). The k shortest paths to a vertex v can be stored in a linked list, where each path $\pi^k(v,h) = \pi^{k'}(u){\cdot}v$ is compactly represented by its length and a back pointer to $\pi^{k'}(u)$. Using these data structures, we obtain:

Proposition 29. *The time complexity of the adapted REA is $\mathcal{O}(hm{+}hk\log(\frac{m}{n}))$.*

Proof. The computation of the first step takes $\mathcal{O}(hm)$ using the BF-algorithm. Due to Lemma 28, the number of recursive invocations to *NextPath* is bounded by h, the maximum length of $\pi^{k-1}(t,h)$. At any given time, the set $Q^k(v,h)$ contains at most $|\mathit{Pred}(v)|$ paths where $\mathit{Pred}(v) = \{u \mid (u,v) \in E\}$, i.e., one path for each predecessor vertex of v. By using heaps to store the candidate sets, a minimal element can be determined and deleted (cf. Step4) in $\mathcal{O}(\log|\mathit{Pred}(v)|)$ time. Insertion of a path (as in Steps 2 and 3(b)) takes the same time complexity. Since $\sum_{v\in V}|\mathit{Pred}(v)| = m$, $\sum_{v\in V}\log|\mathit{Pred}(v)|$ is maximized when all vertices have an equal number of predecessors, i.e., $|\mathit{Pred}(v)| = \frac{m}{n}$. Hence, it takes $\mathcal{O}(h\log(\frac{m}{n}))$ to compute $\pi^k(v,h)$. We have k such paths to compute, yielding $\mathcal{O}(hm + hk\log(\frac{m}{n}))$.

Note that the time complexity is pseudo-polynomial due to the dependence on k which may be exponential in n. As in our setting, k is not known in advance, this can not be reduced to a polynomial time complexity.

7 Conclusion

Summary of results. We have investigated the computation of strongest evidences (maximally probable paths) and smallest counterexamples for PCTL model checking of DTMCs. Relationships to various kinds of shortest path problems have been established. Summarizing we have obtained:

counterexample problem	shortest path problem	time complexity
SE (until)	SP	$\mathcal{O}(m + n \log n)$
SE (bounded until)	HSP	$\mathcal{O}(hm)$
SC (until)	KSP	$\mathcal{O}(m + n \log n + k)$
SC (bounded until)	HKSP	$\mathcal{O}(hm + hk \log(\frac{m}{n}))$

where n and m are the number of states and transitions, h is the hop bound, and k is the number of shortest paths.

For DTMCs with rewards, we can establish along the same lines as in this paper that determining strongest evidences for violating reward- and hop-bounded until-formulae boils down to solving a non-trivial instance of the CSP problem. As this problem is NP complete, efficient algorithms for finding counterexamples for PRCTL [4] will be hard to obtain.

Further research. Topics for further research are: experimental research of the proposed algorithms in probabilistic model checking, considering loopless paths (see e.g., [21, 25, 19]), and extension towards continuous-time models.

Related work. With the notable exception of [2, 3], counterexample generation for probabilistic model checking has not been addressed before. Aljazzar *et al.* [2] consider the generation of a most probable path for timed reachability in CTMCs. They map this onto a bounded-until problem on DTMCs, and use heuristics (Z^*) for determining the most probable path. Unbounded until is not considered, and neither a correctness proof nor complexity results are provided. Recently, [3] generalises this heuristic-based approach for CTMCs to obtain failure subgraphs, i.e., counterexamples. To our knowledge, smallest counterexamples have not been considered yet.

Acknowledgement. This research has been performed as part of the QUPES project that is financed by the Netherlands Organization for Scientific Research (NWO). David N. Jansen is kindly acknowledged for remarks on an earlier version of this paper.

References

1. R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network Flows: Theory, Algorithms and Applications*, Prentice Hall, Inc., 1993.
2. H. Aljazzar, H. Hermanns and S. Leue. Counterexamples for timed probabilistic reachability. FORMATS 2005, LNCS 3829: 177-195, 2005.
3. H. Aljazzar and S. Leue. Extended directed search for probabilistic timed reachability. FORMATS 2006. (to appear)
4. S. Andova, H. Hermanns and J.-P. Katoen. Discrete-time rewards model-checked. FORMATS 2003, LNCS 2791: 88-104, 2003.

5. C. Baier, J.-P. Katoen, H. Hermanns and V. Wolf. Comparative branching-time semantics for Markov chains. *Inf. Comput.* 200(2): 149-214 (2005).
6. T. Ball, M. Naik and S. K. Rajamani. From symptom to cause: localizing errors in counterexample traces. POPL: 97-105, 2003.
7. G. Behrmann, K. G. Larsen and J. I. Rasmussen. Optimal scheduling using priced timed automata. *ACM SIGMETRICS Perf. Ev. Review* 32(4): 34-40 (2005).
8. R. Bellman. On a routing problem. *Quarterly of Appl. Math.*, 16(1): 87-90 (1958).
9. E.M. Clarke, O. Grumberg, S. Jha, Y. Lu and H. Veith: Counterexample-guided abstraction refinement. CAV, LNCS 1855: 154-169, 2000.
10. E.M. Clarke, S. Jha, Y. Lu and H. Veith. Tree-like counterexamples in model checking. LICS: 19-29 (2002).
11. T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein. *Introduction to Algorithms*, 2001. Section 24.1: The Bellman-Ford algorithm, pp.588-592.
12. L. de Alfaro, T.A. Henzinger and F. Mang. Detecting errors before reaching them. CAV, LNCS 2725: 186-201, 2000.
13. E.W. Dijkstra. A note on two problems in connection with graphs. *Num. Math.*, 1:395-412 (1959).
14. D. Eppstein. Finding the k shortest paths. *SIAM J. Comput.* 28(2): 652-673 (1998).
15. L.R. Ford jr. and D.R. Fulkerson. *Flows in Networks*, Princeton Univ. Press, 1962.
16. G.D. Forney. The Viterbi algorithm. *Proc. of the IEEE* 61(3): 268-278 (1973).
17. B. L. Fox. k-th shortest paths and applications to the probabilistic networks. In *ORSA/TIMS National Mtg*, volume 23, page B263. *Bull. Operations Research Soc. of America*, 1975.
18. M.R. Garey and D.S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
19. E. Hadjiconstantinou and N. Christofides. An efficient implementation of an algorithm for finding K shortest simple paths. *Networks* 34(2): 88-101 (1999).
20. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.* 6(5): 512-535 (1994).
21. J. Hershberger, M. Maxel and S. Suri. Finding the k shortest simple paths: A new algorithm and its implementation. ALENEX 2003, in *Proc. of the Fifth Workshop on Algorithm Engineering and Experiments*, Baltimore, USA, pp. 26-36, 2003.
22. V.M. Jiménez and A. Marzal. Computing the K shortest paths: A new algorithm and an experimental comparison. WAE 1999, LNCS 1668: 15-29, 1999.
23. H. Jin, K. Ravi and F. Somenzi. Fate and free will in error traces. *STTT* 6(2): 102-116 (2004).
24. E.L. Lawler. *Combinatorial Optimization: Networks and Matroids.* Holt, Reinhart, and Winston, 1976.
25. E.Q.V. Martins and M.M.B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *4OR* 1(2): 121-133 (2003).
26. E.Q.V. Martins, M.M.B. Pascoal and J.L.E. Dos Santos. Deviation algorithms for ranking shortest paths. *Int. J. Found. Comput. Sci.* 10(3): 247-262 (1999).
27. K. Mehlhorn and M. Ziegelmann. Resource constrained shortest paths. ESA 2000, LNCS 1879: 326-337, 2000.
28. S. Shoham and O. Grumberg. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. CAV, LNCS 2725: 275-287, 2003.
29. E. Vidal, F. Thollard, C. de la Higuera, F. Casacuberta and R.C. Carrasco. Probabilistic finite-state machines-Part I. *IEEE Trans. Pattern Anal. Mach. Intell.* 27(7): 1013-1025 (2005).