Abstract

This paper presents an algorithm for cost-bounded probabilistic reachability in timed automata extended with prices (on edges and locations) and discrete probabilistic branching. The algorithm determines whether the probability to reach a (set of) goal location(s) within a given price bound (and time bound) can exceed a threshold \( p \in [0, 1] \). We prove that the algorithm is partially correct and show an example for which termination cannot be guaranteed.

1. Introduction

Timed automata are a prominent model for analyzing real-time systems. Efficient algorithms for timed reachability—can a certain goal (set of) location(s) be reached within a given time-bound?—are at the heart of successful model checkers for timed automata such as Uppaal [4]. As the state space of timed automata is infinite, these algorithms are based on finite symbolic representations such as regions and zones. Verification algorithms for timed automata have been applied to several application areas. Recently, the use of real-time model checkers for scheduling synthesis has become en vogue [9, 12] and elsewhere. The basic idea here is to model all resources as well as all individual tasks together with their (hard) deadlines as timed automata. The question whether there exists a schedule that meets all requirements (such as order of tasks, timing aspects and deadlines) can be formulated as timed reachability question and be tackled with model checkers such as Uppaal.

Scheduling synthesis has been the major motivation to enrich timed automata with prices [3, 5, 17]. Such prices can be interpreted as bonus, gain, or duallly, as cost. Price rates attached to locations indicate the increase of price per time unit, whereas prices attached to edges indicate instantaneous costs. (This is similar to state and impulse rewards, respectively, in Markov reward models [21].) The problem of minimal cost reachability on priced timed automata (also called weighted timed automata) has been shown to be decidable [3, 5]. The symbolic algorithms are based on priced extensions of the symbolic data structures used for timed automata, such as regions and zones. When interpreting prices as resource costs, these timed automata can be used to obtain minimal cost schedules. In combination with the use of heuristics, scheduling synthesis with (priced) timed automata can often handle larger problem instances than with standard approaches using, e.g., mixed-integer linear programming [19].

An important restriction, however, of these approaches is that resources are typically considered to be fully reliable. That is to say, resources are assumed to never break down and (e.g., in case of production machines) never produce imperfect output. In order to handle such situations, in this paper we investigate priced probabilistic timed automata (\( \mathbb{P}^2 \)TA, for short), which are a probabilistic extension of priced timed automata. This model is an orthogonal extension of priced as well as probabilistic timed automata [13]. When prices are omitted, probabilistic timed automata are obtained, whereas the deletion of probabilities yields priced timed automata. We define this model, provide its semantics and its symbolic semantics using the novel concept of multi-priced zones.

The main contribution of the paper is an algorithm for the problem we call cost-bounded probabilistic reachability. This backwards algorithm can be used to determine whether a (set of) goal location(s) can with probability greater than \( p \) be reached with cost at most \( c \).

Although cost-bounded reachability [3, 5] and maximal probabilistic reachability [13] are decidable, the problem...
of cost-bounded probabilistic reachability is not trivial as
the symbolic state space is not guaranteed to be finite. To
our knowledge, the algorithm presented in this paper is the
first semi-decidable algorithm for cost-bounded probabilis-
tic reachability. In case the answer to the problem is affirm-
ative, the algorithm will terminate. Furthermore, in cases
of a finite symbolic state space it will also terminate.

\(P^2\text{TA}\) are a subclass of probabilistic linear hybrid auto-
toma (PLHA), introduced by Sproston [20]. Cost-
bounded probabilistic reachability can be expressed in the
logic used, but this (decidable) model checking algorithm
only treats a subclass of PLHA and \(P^2\text{TA}\) with finitely many
symbolic states, namely those with a finite bisimulation
quotient. The most related work to ours is the recent work
by Mutsuda et al. [18], that considers the maximal proba-
ability to reach a set of target states via a single path. Instead,
our algorithm considers the total probability to reach the
target via any path.

Organization of the paper. Section 2 defines \(P^2\text{TA}\)
and their concrete semantics. Section 3 formalizes the
cost-bounded probabilistic reachability problem. Section 4
presents the symbolic semantics of \(P^2\text{TA}\) and operators on
the symbolic data structures. Section 5 presents the algo-

\[\text{2. Priced Probabilistic Timed Automata}\]

\[\text{2.1. Clocks and zones}\]

A clock is a real-valued variable that can be used to mea-
sure the elapse of time. In the rest of the paper, \(\mathbb{X}\) will
be a fixed, finite set of clocks. All clocks in \(\mathbb{X}\) run at the
same pace. A clock valuation assigns a non-negative value
to each clock. Let \(\mathbb{R}^\mathbb{X}_{\geq 0}\) denote the set of all possible clock
valuations. A clock valuation \(v\in \mathbb{R}^\mathbb{X}_{\geq 0}\) is thus a mapping
\(\mathbb{X}\rightarrow \mathbb{R}_{\geq 0}\). For \(d\in \mathbb{R}_{\geq 0}\), let \(v+d\) denote the clock valu-
a tion that maps each \(x\in \mathbb{X}\) to \(v(x) + d\). We use \(O\) for the
clock valuation that assigns the value zero to all clocks. For
\(r\subseteq \mathbb{X}\), let \(v[r := 0]\) denote the reset of the clocks in \(r\), i.e.
\[
v[r := 0](x) = \begin{cases} 
0 & \text{if } x \in r \\
v(x) & \text{otherwise}
\end{cases}
\]

A zone is a conjunction of inequalities where the value of
a single clock or the difference between two clocks is com-
pared to an integer. Formally, for the set \(\mathbb{X}\) of clocks the set
\(\text{Zones}(\mathbb{X})\) of zones \(Z\) is defined by the grammar:
\[
Z ::= x \bowtie b \mid x - y \bowtie b \mid Z \land Z \mid \text{true}
\]
\(\bowtie\) where \(x,y \in \mathbb{X}, b \in \mathbb{Z}, \bowtie \in \{\leq,\geq\}\)

\[(\text{In})\text{qualities such as } (x = b)\text{ and } (2 \leq x - y \leq 3)\]
are abbreviations for a conjunction of multiple inequalities.

For simplicity our definition of zones does not contain strict
inequalities. We conjecture that strict inequalities can eas-
ily be added by introducing extra book keeping variables
for strictness as in [10]. The semantics of zone \(Z\), denoted
\([Z]\), is the set of all clock valuations satisfying \(Z\). Note
that several zones may have the same semantics. We will
only consider the canonical zones of [13]. Figure 1 gives an
example zone.

\[\text{2.2. Model definition}\]

Before defining the priced probabilistic extension
of timed automata, let us recall the notion of a
(sub)distribution. A discrete probability subdistribution
over a finite set \(Q\) is a function \(\mu : Q \rightarrow [0,1]\) such that
\(\sum_{q \in Q} \mu(q) \leq 1\). \(\mu\) is called a distribution if the
inequality can be replaced by an equality. For (possibly uncountable)
set \(Q\), let \(\text{Dist}(Q)\) and \(\text{SubDist}(Q)\) be the set of distribu-
tions and subdistributions, respectively, over finite subsets
of \(Q\). The point distribution \(\{q \rightarrow 1\}\) is the discrete proba-
bility distribution such that \(\{q \rightarrow 1\}(q) = 1\) and all other
elements have probability zero.

\(P^2\text{TA}\) equip timed automata with prices (on edges and lo-
cations) [5] and discrete probabilistic branching [13]. Prices
on edges indicate the cost to evolve from one location to
another, while price rates attached to locations indicate the
cost to reside in a location per time unit. Edges are proba-
bilistic in the sense that a combination of resets and direct
successor location is selected in a random manner.

Definition 1 A priced probabilistic timed automaton
\(P^2\text{TA}\) is a tuple \(W = (L, l_{init}, \mathbb{X}, \text{inv}, pE, \$)\), where:

- \(L\) – finite set of locations;
- \(l_{init} \in L\) – the initial location;
- \(\mathbb{X}\) – finite set of clocks;
- \(\text{inv} : L \rightarrow \text{Zones}(\mathbb{X})\) – function assigning an invari-
ant to each location;
- \(pE \subseteq L \times \text{Zones}(\mathbb{X}) \times \mathbb{N} \times \text{Dist}(2^\mathbb{X} \times L)\) – proba-
bilistic edge relation such that \(\{l \rightarrow 0\text{inv}(l')\}\) for every \(l'\) and \(r \subseteq \mathbb{X}\) such that \(p(r,l') > 0\) and
\((l,g,h,p) \in pE\);
- \(\$ : L \rightarrow \mathbb{N}\) – function assigning a price rate to each
location.

For probabilistic edge \((l,g,h,p) \in pE\), \(l\) denotes the source
location, \( g \) the guard (which is a zone), \( h \) its price, and \( p \) a distribution on pairs of a set of clocks to be reset and a destination location. The set \( E_W \) of edges of a P²TA is defined as follows: \((l, g, h, p, r, l') \in E_W \) if \((l, g, h, p) \in pE \) and \( p(r, l') > 0 \). The restriction on probabilistic edges is introduced to prevent performing a transition (in the semantics) to a location for which the invariant does not hold. Zone \([r := 0]Z\) contains all clock valuations \( v \) such that \( v[r := 0] \in \{l, l'\} \).

**Example 1** Figure 2 provides an example P²TA with a single clock: \( X = \{x\} \). The circles represent the locations of the P²TA. The initial location is marked by the dangling incoming arrow. The invariants and price rates are written inside the locations. Invariants are omitted when true. The probabilistic edges are denoted by branching arrows between locations or a single arrow in case of a single branch. The guard and price are indicated next to the source location; the probabilities and reset sets at the branches. Trivial probabilities (i.e. equal to one) are omitted. The same applies to guards equal to true, and zero prices. The Greek letters are only part of the figure (not of the model), and are used to mark the probabilistic edges.

The intuitive semantics of P²TA is as follows. Each P²TA is mapped onto a (typically infinite-state) transition system. States in these transition systems consist of a location, a clock valuation, and the accumulated cost. Only states that satisfy the invariants are considered. Execution starts in the initial location (e.g. location \( l_0 \) in Figure 2), with all clocks and the accumulated cost equal to zero. As long as the invariant is satisfied, time may pass in a location. As a result, all clocks increment by the same value. The cost of delaying is determined by the price rate of the location: residing \( d \) time units in location \( l \) incurs the cost \( \hat{s}(l) \cdot d \). To accommodate for the probabilistic branching, Markov decision processes are used as semantic model (and not plain transition systems). A probabilistic edge that emanates from location \( l \) may be taken when the state of the system is in \( l \), the guard is satisfied, and all possible resulting states, as discussed below, satisfy their invariants. Taking the edge implies an increment of the accumulated cost by the price of the edge. On taking a probabilistic edge, the destination location and the reset set are chosen probabilistically according to the distribution of the edge. The reset set determines which clocks are reset. No time elapses when taking a probabilistic edge.

### 2.3. Probabilistic Systems

The semantics of P²TA is defined in terms of infinite state discrete-time Markov decision processes [21] and is defined along the lines of [15].

**Definition 2** [16]. A probabilistic system (PS) is a tuple \((S, \text{Steps})\), where \( S \) is a possibly infinite set of states, and \( \text{Steps} \subseteq S \times \text{Dist}(S) \) is a probabilistic transition relation such that for each \( s \in S \), there exists a distribution \( \mu \) such that \((s, \mu) \in \text{Steps}\).

A sub-probabilistic system (SPS) is a probabilistic system, with the only difference that \( \text{Steps} \subseteq S \times \text{SubDist}(S) \).

The restriction of having a distribution for each state is imposed to ensure that adversaries (defined below) always exist. For state \( s \) in SPS \((S, \text{Steps})\) with \((s, \mu) \in \text{Steps}\), the remaining probability \( 1 - \sum_{s'} \mu(s') \) may be interpreted as a deadlock probability. Note that any SPS can easily be transformed into a PS by adding a trapping state \( s_{\text{trap}} \) that is equipped with a self-loop with probability 1, such that all the ‘missing’ probabilities of the subdistributions \( \mu \) lead to \( s_{\text{trap}} \). All definitions on a PS in the rest of this paper are therefore easily lifted to a SPS.

Intuitively speaking, a (sub-)probabilistic system describes the following behaviour. Whenever the system is in some state, \( s \in S \) say, a distribution \( \mu \) with \((s, \mu) \in \text{Steps}\) is selected nondeterministically. Subsequently, the next state is selected probabilistically according to \( \mu \), i.e. the next state \( s' \) is selected with probability \( \mu(s') \). Thus, a transition involves resolving both a nondeterministic and a probabilistic choice.

An infinite path in PS \((S, \text{Steps})\) is an infinite alternating sequence: \( \omega = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \ldots \) such that \((s_i, \mu_i) \in \text{Steps} \) and \( \mu_i(s_{i+1}) > 0 \) for all \( i \). Let \( \omega(i) \) denote the \( i \)-th state in the path \( \omega \), i.e. \( \omega(i) = s_i \). A finite path is just a finite prefix of an infinite path. Let last(\( \omega \)) denote the last state in the finite path \( \omega \). The length of a path is the number of transitions involved. By definition, the minimal path of length zero only contains a single state.

To define a probability space corresponding to a PS, we have to resolve the nondeterministic choices. To this end, we consider deterministic adversaries, similar to schedulers or policies in a Markov decision process. An adversary \( A \) is a function mapping every finite path \( \omega \) in probabilistic system \((S, \text{Steps})\) to a distribution \( \mu \in \text{Dist}(S) \) such
that \((\text{last}(\omega), \mu) \in \text{Steps}\). Hereby all nondeterminism is resolved, therefore a probabilistic system together with an adversary yields a (discrete-time) Markov chain. For any adversary \(A\), let \(\text{Path}^A_{\text{full}}(s)\) and \(\text{Path}^A_{\text{fin}}(s)\) respectively denote the infinite paths and finite paths that start in \(s\) induced by \(A\). \(\text{Path}^A_{\text{full}}\) and \(\text{Path}^A_{\text{fin}}\) denote the entire set of infinite paths and finite paths induced by \(A\). \(\text{Prob}^A\) denotes the probability measure on \(\text{Path}^A_{\text{full}}(s)\), defined using classical techniques [15]. Let \(\text{Adv}_M\) denote all deterministic adversaries on probabilistic system \(M\).

The reach probability is the likelihood to reach a certain set of target states in a finite number of transitions under some adversary. For PS \((S, \text{Steps})\) with state \(s \in S\), target states \(S^T \subseteq S\), and adversary \(A\), the reach probability on infinite paths is defined as [15]:

\[
\text{ProbReach}^A(s, S^T) = \sup_{\omega \in \text{Path}^A_{\text{full}}(s)} \{ \exists i \in \mathbb{N} \cdot \omega(i) \in S^T \}
\]

The reach probability depends on the nondeterministic choices made by the adversary. A nondeterministic choice models branches in system execution that are not to be resolved probabilistically or for which the probability distribution is not known. Therefore the maximal reach probability is of interest, i.e., the maximal attainable value if all choices are optimal. The maximal reach probability for probabilistic system \(M = (S, \text{Steps})\) with \(s \in S\) and \(S^T \subseteq S\) is defined as:

\[
\text{Max Prob Reach}_M(s, S^T) = \sup_{A \in \text{Adv}_M} \text{ProbReach}^A(s, S^T)
\]

For finite state probabilistic systems, maximal and minimal reach probability are computable in polynomial time [8].

The following definition gives the reach probability using paths of finite length.

**Definition 3** Let \(M = (S, \text{Steps})\) be a PS and \(S^T \subseteq S\). For adversary \(A \in \text{Adv}_M\) and finite path \(\omega \in \text{Path}^A_{\text{fin}}\) let:

\[
P^A_0(\omega \overset{M}{\sim} S^T) = \begin{cases} 1 & \text{if last}(\omega) \in S^T \\ 0 & \text{otherwise} \end{cases}
\]

and for any \(n \in \mathbb{N}\), with \(\mu = A(\omega)\):

\[
P^A_{n+1}(\omega \overset{M}{\sim} S^T) =
\begin{cases} \\
1 & \text{if last}(\omega) \in S^T \\
\sum_{s \in S} \mu(s) \cdot P^A_n(\omega \overset{M}{\mu} \overset{M}{\sim} s \overset{M}{\sim} S^T) & \text{otherwise.}\end{cases}
\]

For arbitrary state \(s\) define:

\[
P^\text{max}_n(s \overset{M}{\sim} S^T) = \sup_{A \in \text{Adv}_M} P^A_n(s \overset{M}{\sim} S^T).
\]

For \(S\) finite, the computation of \(P^\text{max}_n(s \overset{M}{\sim} S^T)\) can be regarded as a linear optimization problem.

### 2.4. Timed Probabilistic Systems

**Definition 4** [16] A timed probabilistic system (TPS) is a tuple \((S, \text{Steps})\), where \(\text{Steps}\) has two extra labels in addition to the ones in Definition 2:

\[
\text{Steps} \subseteq S \times \mathbb{R}_{\geq 0} \times \{\text{time}, \text{disc}, \text{full}\} \times \text{Dist}(S).
\]

For \((s, d, i, \mu) \in \text{Steps}\), the real \(d\) denotes the duration that the system remains in state \(s\), \(i\) and \(\mu\) are used to mark discrete and time transitions respectively, with the following rules:

- **time determinism**: if \(i = \text{time}\), then \(\mu\) is a point distribution.
- **discrete transitions**: if \(i = \text{disc}\) then \(d = 0\),
- **Wang’s axiom**: \(s \overset{\text{time}}{\rightarrow} t\) if for all \(0 \leq d' \leq d\) there exists \(s'\) such that \(s \overset{\text{time}}{\rightarrow} s'\) and \(s' \overset{\text{time}}{\rightarrow} t\).

A full transition (marked as full) is a timed transition followed by a discrete transition. Formally: \(s \overset{\text{full}}{\rightarrow} u\) if and only if \(s \overset{\text{time}}{\rightarrow} t\) for some state \(t\) and \(\overset{\text{disc}}{\rightarrow} u\) for some state \(t\).

Labels on transitions are sometimes omitted when they are clear from the context. A path \(\omega\) in a TPS has the form: \(\omega = s_0 \overset{d_0, \mu_0}{\rightarrow} s_1 \overset{d_1, \mu_1}{\rightarrow} s_2 \overset{d_2, \mu_2}{\rightarrow} \cdots\) with \((s_i, d_i, i, \mu_i) \in \text{Steps}_i\) and \(\mu_i(s_{i+1}) > 0\) for all \(i \in \mathbb{N}\). The probability space corresponding to a TPS is defined analogously to the probability space of a PS by means of adversaries. A (deterministic) adversary of TPS \((S, \text{Steps})\) is a function mapping every finite path \(\omega\) to an element \((d, i, \mu)\), i.e., a duration, transition type and distribution, such that \((\text{last}(\omega), (d, i, \mu)) \in \text{Steps}\). Definition 3 is easily adapted to a TPS by adding durations to the transitions.

### 2.5. Semantics

**Definition 5** (P²TA Semantics) The semantics of P²TA \(W = (L, l_{\text{init}}, X, \text{inv}, pE, \$)\) is the tuple \([W] = (S, \text{Steps})\), where

\[
S = \{ (l, v, c) \mid l \in L \land v \in \text{inv}(l) \land c \in \mathbb{R}_{\geq 0} \}
\]

\[
\text{Steps} \subseteq S \times \mathbb{R}_{\geq 0} \times \{\text{time}, \text{disc}, \text{full}\} \times \text{Dist}(S).
\]

A probabilistic transition \((l, v, c), d, i, \mu) \in \text{Steps}\) if and only if one of the following conditions holds:

- **time**: \(i = \text{time}\), \(d \geq 0\), \(v + d \in \text{inv}(l)\), and \(\mu(l, v + d, c + \$) = 1\) (point distribution);
- **disc**: \(i = \text{disc}\), \(d = 0\), and there exists \((l, g, h, p) \in pE\) such that \(v \in [g]\) and for any \((l', v', c + h) \in S, \mu(l', v', c + h) = \sum_{r \in X} \mathbb{P}(r, l', v')\).

\[
\text{full} = \text{and the constituting time and discrete transition fulfill the two conditions above.}
\]

**Lemma 1** For each P²TA \(W\), \([W]\) is a TPS.
3. Cost-bounded probabilistic reachability

The cost-bounded probabilistic reachability problem is the question: “Is it possibly to reach a (set of) goal location(s) with probability greater than λ with cost at most κ?” The next definition formalizes this.

Definition 6 Given P^2TA W = (L, l_{init}, X, inv, pE, $\$), target locations $L^T \subseteq L$, a probability $\lambda \in [0, 1]$, cost bound $\kappa \in \mathbb{R}_{\geq 0}$, and initial state $s_{init} = (l_{init}, 0, 0)$. The cost-bounded probabilistic reachability problem $(W, L^T, \lambda, \kappa)$ is the question: “Does there exist an adversary $A \in Adv[W]$ such that $ProbReach^A(s_{init}, S^T) > \lambda$, with $S^T = \{(l, v, c) | l \in L^T \land v \in \mathbb{R}^X_{\geq 0} \land c \leq \kappa\}?"$

To solve $(W, L^T, \lambda, \kappa)$, the maximal reach probability (Section 2.3) needs to be computed on $[W]$, where we are interested in the reach probability on states that have a location in $L^T$ and cost at most $\kappa$. Formally, if $MaxProbReach_{[W]}(s_{init}, S^T) > \lambda$, with $s_{init}$ and $S^T$ as above, the answer to the cost-bounded reachability problem is “yes” and “no” otherwise.

Very related is the case where the cost must be at least some value. Without further proof we conjecture that all results in this paper can be altered to treat this case. Another variant is where the probability must be less than some value. We will not treat this variant, note however that it is more difficult, as we must exclude adversaries that have Zeno behavior, see e.g. [15].

4. Symbolic states

Symbolic states are used to represent a possibly infinite set of concrete states of the system. Typically in timed automata symbolic states consist of a location and a zone. Because cost is part of the system state, in P^2TA we cannot use zones. In [17] priced zones are introduced, which are zones augmented with a linear cost function. This cost function assigns a cost to each clock valuation of the zone. The problem is that priced zones (even if we allow a rational price bound), are not closed under intersection; for more details see [6]. Instead we will introduce multi-priced zones, which are zones augmented with a conjunction of linear inequalities that define an upper bound on the cost that can be associated with each valuation in the zone. Figure 3 shows an example. The gray plane is the underlying zone. Clearly multiple linear upper bounds on cost are allowed.

Definition 7 A Multi-Priced Zone (MP-zone) is a tuple $M = (Z, \Phi)$ where $Z$ is a zone and $\Phi$ is a formula defined by the grammar:

$\Phi ::= a \leq b_1 x_1 + \cdots + b_n x_n + b_0 | \Phi \land \Phi | true$

where $z$ is the cost variable, $x_1, \ldots, x_n$ are all the clocks in $Z$, $a, b_0, \ldots, b_n \in \mathbb{Z}$, and $a > 0$.

We define a valuation to be a pair consisting of a clock valuation and accumulated cost. The semantics for MP-zones $M$ are denoted $[M]$ and coincide with the set of all valuations $(v, c) \in \mathbb{R}^X_{\leq 0} \times \mathbb{R}_{\geq 0}$ that satisfy formula $(Z \land \Phi)$.

From the definition of MP-zones we see that the inequalities of $\Phi$ are not equivalent to any inequality of $Z$, because $a > 0$ implies that cost always is a nontrivial variable in the formula. MP-zones are a conjunction of linear inequalities, and therefore are a class of convex polyhedra. A zone $Z$ can easily be converted to the MP-zone $(Z, true)$, that does not place any constraints on the cost. The conjunction between two MP-zones $(Z_1, \Phi_1) \land (Z_2, \Phi_2)$ is simply the MP-zone $(Z_1 \land Z_2, \Phi_1 \land \Phi_2)$.

MP-zones will be used in our backward algorithm for P^2TA. Like classical work on backward methods [11], we define backward operators. On an arbitrary MP-zone $M$ three operators can be applied to find the predecessor valuations for the valuations in the semantics of $M$. MP-zones are closed under these operators, as stated in Lemma 2. The operators are computable by using quantifier elimination, or probably more efficient, as operations on convex polyhedra [2, 10]. We will not describe implementation details.

$$[M^{aq}] \equiv \{(v, c) | \exists d \geq 0. (v+d, c+qd) \in [M]\}$$

$$[r := 0]M \equiv \{(v, c) | (v|r := 0), c \in [M]\}$$

$$[\text{reset}]M \equiv \{(v, c) | (v, c+h) \in [M]\}$$

$[M^{aq}]$ are the valuations that can reach a valuation in $[M]$ by delaying an arbitrary amount of time under price rate $q$. $[r := 0]M$ are the valuations that can reach a valuation in $[M]$ by resetting clocks in set $r$. $[\text{reset}]M$ are the valuations that can reach a valuation in $[M]$ with price $h$.

Figure 4 gives examples for all operators for an MP-zone $M$ with one clock.
Lemma 2. For MP-zone \( M, q \in \mathbb{Z}, r \subseteq X, \) and \( h \in \mathbb{N} \):
\[ M^{1q}, [r := 0] M \text{ and } [+h] M \text{ are MP-zones} \]

Proof. Let \( M = (Z, \Phi) \). We can construct MP-zone \([+h] M\) in the following way: for all formulas \( ax \leq b_1x_1 + \cdots + b_nx_n + b_0 \) in \( \Phi \), replace \( b_0 \) with \( b_0 - h \).

Following the proof of Lemma 1 in [11], \([r := 0] M \) are all valuations that satisfy \( (Z \land \Phi)[r := 0] \), i.e., the formula obtained by replacing all occurrences of \( x \) in \( (Z \land \Phi) \) with 0. Clearly \( (Z \land \Phi)[r := 0] = ((Z[r := 0]) \land (\Phi[r := 0])) \). By theory on timed automata \( Z[r := 0] \) is a zone. \( \Phi[r := 0] \) satisfies the requirements for cost constraints in MP-zones, thus \([r := 0] M \) is a MP-zone.

Again let \( M = (Z, \Phi) \). Let \( \llbracket \Phi \rrbracket \) denote all valuations that satisfy \( \Phi \). For all valuations \((v, c) \) of \([r := 0] M \):
\[ (v, c) \in \llbracket M^{1q} \rrbracket \iff \exists d \geq 0. v+d \in \llbracket Z \rrbracket \land (v+d, c+qd) \in \llbracket \Phi \rrbracket \]

To eliminate the variable \( d \), let’s look at how to change the constraints in \( M \) to get constraints that describe \( M^{1q} \). The zone constraints in \( Z \) have to be replaced by the constraints in zone \( Z^{1q} \). Operator \( ^{1q} \) denotes the unpriced version of \( ^q \) on zones; details can be found in [11]. The cost constraints \( ax \leq b_1x_1 + \cdots + b_nx_n + b_0 \) remain unchanged, because \((v + d, c + qd) \) satisfies such constraints only if \((v, c) \) does. Other cost constraints have to be replaced by linear inequalities on the clocks and cost variable, which can be expressed as cost constraints again. Therefore, we can find a MP-zone that corresponds to \( M^{1q} \).

A symbolic state is a tuple \( \sigma = (l, M) \), with location \( l \), MP-zone \( M \). Its semantics is \( \llbracket \sigma \rrbracket = \{l\} \times \llbracket M \rrbracket \), which is a set of states. If \( \Psi \) is a set of symbolic states, then we let its pointwise extension be \( \llbracket \Psi \rrbracket = \cup (\llbracket \sigma \rrbracket | \sigma \in \Psi) \).

The semantics of the intersection of two symbolic states is:
\[ \llbracket (l_1, Z_1) \land (l_2, Z_2) \rrbracket = \llbracket (l_1, Z_1) \rrbracket \cap \llbracket (l_2, Z_2) \rrbracket \]

The next definition gives the time predecessor and discrete predecessor operators on symbolic states. Using Lemma 2 we get that symbolic states are closed under time predecessor and discrete predecessor.

Definition 8 (Predecessor of symbolic states) Let \( W = (L, l_{init}, X, inv, pE, \$) \), \( l \in L \) with \( q = \#(l) \), \( M = (Z, \Phi) \) be a multi-priced, zoned, and \( e := (l, g, h, p, r, l') \in E_W \). Then:
\[ tpre(l, M) = (l, (M^{1q}) \land (inv(l), true)) \]
\[ dpre_e(l', M) = (l, [r := 0]([+h] M) \land (g \land inv(l), true)) \]

5. Algorithm

Algorithm CBPRalg is used to answer the cost-bounded probabilistic reachability problem. As we will see it does not terminate for all instances of the problem. It takes a problem instance \( (W, L^T, \lambda, \kappa) \), and if it terminates, it outputs the answer true or false.

Algorithm 1: CBPRalg
\begin{enumerate}
\item Input: cost-bounded probabilistic reachability problem \( (W, L^T, \lambda, \kappa) \), where \( P^{\cdot TA} W = (L, l_{init}, X, inv, pE, \$) \)
\item Output: boolean
\item short-hand \( s_{init} = (l_{init}, 0, 0) \)
\item short-hand \( \Psi = \{(l, inv(l), z \leq \kappa | l \in L^T)\} \)
\item if \( \exists \sigma \in \Psi, s_{init} \in [\sigma] \) then \( R_0 := 1 \) else \( R_0 := 0 \)
\item foreach \((l, g, h, p) \in pE // Initialize all edge sets \)
\[ E_{(l,g,h,p)} := \emptyset \]
\item \( \text{Waiting} := \emptyset \text{ // Waiting is the set of symbolic states to be explored in iteration} \)
\item short-hand \( Visited = \Psi \cup \{\tau | \exists (l, g, h, p) \in pE. (\tau, \cdot, \cdot) \in E_{(l,g,h,p)}\} \)
\item for \( n := 1 \) to \( \infty \)
\item if \( R_{n-1} > \lambda \) then return \text{true}
\item if \( \text{Waiting}_{n} = \emptyset \) then return \text{false}
\item \( \text{Waiting}_{n+1} := \emptyset \)
\item foreach \( \tau \in \text{Waiting}_n \)
\item \( e = (l, g, h, p, r, l') \in E_W \), with \( l' = \text{loc}(\tau) \)
\item \( \text{dpres}(l', \text{tpres}(\tau)) \)
\item \( s_{\text{visited}} := s_{\text{visited}} \cup \text{Visited} \)
\item \( \text{Visited} := \emptyset \)
\item \( \text{Visited} := \emptyset \)
\item \( \text{Visited} := \emptyset \)
\item \( \text{Visited} := \emptyset \)
\item \( s_{\text{visited}} := s_{\text{visited}} \cup \text{Visited} \)
\item \( Q_n := (\text{Visited}, \text{Steps}) \), where \( (\sigma, \pi) \in \text{Steps} \) if and only if \( \sigma \in \Psi \) and \( \pi = \{\sigma \mapsto 1\} \) or there exists \( E_{\sigma} \subseteq E_{(l, g, h, p)} \) for some \((l, g, h, p) \in pE\) such that:
\begin{itemize}
\item \( \forall (\sigma', \cdot, \cdot) \in E_{\sigma}, \sigma' = \sigma \)
\item \( \forall (l, r, l', \tau), (\cdot, \cdot, \cdot) \in E_{\pi}, \tau \Rightarrow (l', r, \cdot) \neq (\cdot, \cdot, \cdot) \)
\item \( E_{\pi} \) is maximal
\item \( \forall \tau \in \text{Visited}. \pi(\tau) = \{p(r, l') | (\cdot, l', \cdot) \in E_{\pi}\} \)
\end{itemize}
\item \( R_n := \max_{s_{\text{visited}} \in \text{Visited}(\pi)} \text{MaxProbReach}_Q_n(s_{\text{visited}}, \pi) \)
\end{enumerate}

Execution of CBPRalg establishes a sequence \([R_n]_{n=1}^{\infty}\) of values for each iteration. \([R_n]_{n=1}^{\infty}\) is nondecreasing and converges to the maximal reach probability for the states of interest. When the values get higher than \( \lambda \), CBPRalg will return true.

CBPRalg is based on earlier backward symbolic algorithms for probabilistic reachability in PTA [14, 15]. Similarly to that work, to preserve the probabilistic branching, one must consider the intersections of symbolic states that have edges from the same distribution. The intuition is that by computing intersections, we get representations for states that have paths via multiple branches of a single probabilistic edge leading to the target, thereby enlarging the maximal reach probability. The difference in our algorithm is that we do a breadth-first backward exploration of the symbolic state space, and compute the probability exploration of each iteration.
Each iteration has two stages. In the first stage, first termination conditions are checked. In case of no termination, the symbolic state space is explored one step further in depth. A graph, referred to as the symbolic graph, is constructed based on the symbolic state space explored so far. In the second stage, a SPS is constructed from the symbolic graph. On the SPS the maximal reach probability is computed.

The symbolic state space starts as \( \Psi \), defined by the equality on line 3. \( \| \Psi \| \) are all target states, for which the maximal reach probability is of interest. Line 4 gives the verdict of \( CBP R_{laf} \) for zero iterations.

### 5.1. Stage 1: Symbolic Graph generation

Line 10 compares the last value of the generated sequence \( R_\lambda \) \( \lambda \) may and conclude a positive verdict. When no more symbolic states have to be explored, line 11 returns false, because the maximal reach probability will not get any higher.

The set \( Visited \) contains the explored symbolic states so far; it is a short-hand notation, because it can be completely derived from the sets \( E_{(l,g,h,p)} \) (line 8 of \( CBP R_{laf} \)). Together, the sets \( E_{(l,g,h,p)} \) are the edges of the symbolic graph. An edge means that for every state in the semantics of the source symbolic state there is a possibility of taking a probabilistic transition, followed by a time transition, such that some state in the target symbolic state is reached. Formally \( (\sigma, r, l', \tau) \in E_{(l,g,h,p)} \), if \( \sigma \subseteq dpree_{(l,g,h,p,r,l')} (tpre(\tau)) \).

On line 19 an edge in the symbolic graph is added between the symbolic state and its predecessor. Lines 20-24 add intersections between the predecessor and previously explored symbolic states. These intersections are only generated between symbolic states that can reach \( \Psi \) using an edge in the symbolic graph corresponding to the same probabilistic edge \( (l, g, h, p) \) of the \( P^2 TA \), but another element from the distribution (condition \( (r, l') \neq (r, l') \) on line 21). Only these intersections are of interest, as they can enlarge the maximal reach probability. On line 24 the intersection symbolic state gets the two outgoing edges from the intersecting symbolic states.

An edge represents a probabilistic transition followed by a time transition for the following reason: we have to compute intersections of symbolic states immediately before the probabilistic choices. After the probabilistic choice, an adversary will choose the amount of time to delay. If we would have chosen otherwise, a transition from an intersection could represent different delays depending on the outcome of the probabilistic choice, while the probabilistic choice is the last part of such a transition.

**Example 2** Given the cost-bounded probabilistic reachability problem \( \{ W, \{ I \}, \geq, 0.99, 3 \} \), with \( W \) the \( P^2 TA \) of Figure 2

![Figure 5. Symbolic graph for \( P^2 TA \) of Figure 2](image)

**Figure 2.** \( CBP R_{laf} \) will terminate when \( n = 4 \) on line 10 as we will show later. Figure 5 shows the symbolic graph that is generated at the moment \( n \) becomes 4 on line 9. Next to each edge the set \( (E_{\alpha}, E_{\beta}, E_{\gamma}, E_{\delta} \) or \( E_{\tau} \) to which it belongs is denoted. The reset set is mentioned when nonempty. All elements of an edge in the symbolic graph are clear from the figure. \( Visited \) starts as \( \{ A \} \). In the first iteration \( B, C \) and \( A \) are added. Slightly abusing notation: \( B = dpree_{(\gamma, 3, \{ r \}, t_3)} (tpre(A)), C = dpree_{(b, 3, \{ r \}, t_2)} (tpre(A)), A = dpree_{(r, 3, \{ r \}, t_3)} (tpre(A)) \). In the next iteration \( E, D, F \), and \( E \land F \) are added to \( Visited \). Intersection \( E \land F = H \) is added because \( E \) and \( F \) have an outgoing edge both belonging to \( E_{\alpha} \), and condition \( (r, l') \neq (r, l') \) holds, as the destination location is different. \( H \) gets edges to \( B \) and \( C \). In the last iteration \( G, E \land G, E \land F \land G \) are added to \( Visited \). \( E \land G \) is not different from \( H \) that was already explored, but the edge \( E \to D \) is added. Similarly \( H \land G \) is added, which is also not different from \( H \); note that \( F \land G \) is not added because \( (r, l') = (r, l') \) in this case.

In general the intersection between e.g., three symbolic states is computed by first intersecting two symbolic states and then intersecting the result with the third one. More generally the intersection of a set of symbolic states is computed by applying pairwise intersection until the intersection of all symbolic states of the original set is reached. The order in which symbolic states are intersected does not make a difference as can be seen from lines 23 and 24.

### 5.2. Stage 2: SPS construction

On line 25, the sub-probabilistic system \( Q_n \) is constructed from the sets \( E_{(l,g,h,p)} \). The state space of \( Q_n \) is \( Visited \). For each symbolic state in \( Visited \), probabilistic edges are constructed, by taking together as many edges from one set \( E_{(l,g,h,p)} \) as possible. This set is denoted \( E_n \).
The first three bullets after line 25 are the conditions on edges in $E_x$. Condition 1 and 3 are self-explaning. Condition 2 ensures that the edges in $E_x$ correspond to different edges in the $P^2$TA. The fourth bullet describes how to construct the probabilistic edge with distribution $\pi$ in $Q_n$. Finally, $R_n$ is computed on line 26 for the constructed SPS, using the technique of [8].

Example 3 We continue Example 2. Figure 6 shows the SPS constructed for the symbolic graph of Figure 5. Of course the symbolic graph includes all symbolic graphs of previous iterations, as it is only extended in each iteration. For this reason the SPS of Figure 6 contains sub-SPSes of all previous iterations. Nodes $E$, $F$, $G$ and $H$ can all serve as starting node, as the initial state $(s_{init})$ is in the time predecessor of all corresponding symbolic states. For iterations $n = 0 \ldots 3$, we can compute the values of $R_n$, which are $R_0 = 0.0$, $R_1 = 0.0$, $R_2 = 0.97$, $R_3 = 1.0$. Clearly $CBPRalg$ terminates when $n = 4$. If we would alter the problem by taking probability threshold 0.9 it terminates when $n = 3$. Take our original problem, but assume that $R_3 = 0.98$. Then $CBPRalg$ terminates on line 11 when $n = 4$, because for this $P^2$TA the number of symbolic states is finite.

5.3. Non-terminating example of $CBPRalg$

Take the cost-bounded probabilistic reachability problem $(W, \{I_2\}, \geq, \lambda, 2)$, with $W$ the $P^2$TA of Figure 7 and $\lambda = 0.374$. $CBPRalg$ will terminate on line 11 when $n = 5$. Figure 7 also shows the constructed symbolic graph when $n = 5$. It becomes clear that $CBPRalg$ would generate an infinite number of symbolic states. When $n = 6$ symbolic states with $x \leq \frac{1}{3}$ would be generated, when $n = 7$ with $x \leq \frac{1}{4}$ and so on. Figure 8 shows the SPS generated from the symbolic graph, where the plain probabilistic edges are exactly the ones used by the maximizing adversaries. As explained before the SPS also contains the SPS for iterations 0 \ldots 4. The dashed arrows are some other probabilistic edges. $s_{init}$ is part of the time predecessors of $C$, $D$, $E$, \ldots, $I$, thus all of these symbolic states may serve as starting point in computation of the maximal reach probability on the SPS. For $n \geq 2$ we have $R_n > 0$. The values together with the symbolic states used as starting point are: $(R_2 = .3, C)$, $(R_3 = .36, E)$, $(R_4 = .372, G)$, $(R_5 = .374, I)$.

By observing the $P^2$TA we can see that from the initial state $s_{init}$ the optimal adversary will choose the probabilistic edge $\alpha$ arbitrarily often without spending any time in location $l_0$. When in state $(l_2, 0, 0)$ the adversary chooses to spend one time unit and after that it chooses to go to $l_2$. Such an adversary will yield the cost-bounded maximal reach probability for cost-bound 2, which is $\frac{3}{8} = 0.375$.

$CBPRalg$ generates a non-decreasing sequence $[R_n]_{n=0,\ldots,\infty}$, which approaches 0.375 but will never reach it. If we choose $\lambda \geq 0.375$, $CBPRalg$ should return $false$, but instead it does not terminate, because for an arbitrary iteration it is unknown if some future iteration will yield a higher value for $R_n$ letting $R_n > \lambda$ hold.

5.4. Partial correctness of $CBPRalg$

The following is our main theorem. It resembles Proposition 14 in [15].
Corollary 1 Assume given the cost-bounded probabilistic reachability problem \((W, L, \lambda, \kappa)\), with \(W = (L, \ell_{init}, X, inv, pE, S)\). Apply algorithm \(CBPRalg\), where \(Q_n = (\Sigma_n, Steps_n)\) is the SPS generated in iteration \(n\) and \(\Psi\) as in \(CBPRalg\). For all \(n \in \mathbb{N}, \sigma \in \Sigma_n\) the following holds:

\[
\text{MaxProbReach}_{Q_n+1}((\sigma, \Psi)) \geq \text{MaxProbReach}_{Q_n}((\sigma, \Psi))
\]

We can prove that the sequence of probabilities generated in \(CBPRalg\) converges to the actual maximal reach probability, i.e. \(\lim_{n \to \infty} R_n = \text{MaxProbReach}_{\Psi}(s_{init}, [\Psi])\).

6. Related work

In [5], and independently in [3], decidability of minimal cost reachability on linear priced timed automata (LPTA) is proven. LPTA are a subclass of \(P^2TA\). All edges in a LPTA are deterministic. A LPTA can be written as a \(P^2TA\) that has probabilistic edges that only use point distributions. If we want to calculate cost-bounded reachability on \(P^2TA\) (disregarding probabilities), we can construct a LPTA by replacing each probabilistic edge by a set of edges, where every edge has one possibility of reset and target location from the distribution on the probabilistic edge.

In [13], maximal probabilistic reachability on probabilistic timed automata (PTA) is shown to be decidable. PTA are a \(P^2TA\) without prices. The notion of zenoness does not interfere with cost, therefore zenoness can be checked on the PTA obtained from a \(P^2TA\) by removing all prices, using the method of [16].

In [14], a semi-decidable algorithm is presented to compute the maximal reach probability for symbolic probabilistic systems. PTA, but also \(P^2TA\), are an instance of such symbolic probabilistic systems. The algorithm of [15] is derived from this algorithm for the special case of PTA. It is a decidable algorithm and is more efficient than the approach of [13], because it uses zones in its abstractions, which are much coarser than the regions used by [13]. Moreover, it is shown that minimal probabilistic reachability is decidable for PTA.

Our algorithm is based on that of [15]. Although \(P^2TA\) can be handled by the algorithm of [14], our algorithm will terminate in more cases. This comes at the expense of more complexity in the form of numerical computation in every iteration. The problem with [14] is that in the first step a symbolic state space is generated, and secondly the maximal reach probability is computed using the symbolic state space. However, in certain cases, for example that of Figure 7, the symbolic state space is infinite and the first step will not terminate. Therefore, in our algorithm the symbolic state space is explored in a breadth-first way and a lower bound on the maximal reach probability is computed.

![Figure 8. SPS for symbolic graph in Figure 7](image-url)
in each iteration. Based on this lower bound our algorithm may terminate.

Our non-terminating example is based on that in [7]. Although in [7] they do not use probabilities, their model requires also intersections between symbolic states to be made. An important remark that carries over to this work, is that from the infinite symbolic state space, we cannot conclude that the problem under investigation is undecidable.

7. Conclusion

We introduced the model Priced Probabilistic Timed Automata (P²TA), automata with prices, probabilities and (real) time. We investigated cost-bounded probabilistic reachability properties on P²TA. These properties state that a location is reachable with cost \( \leq \kappa \) and with maximal probability \( \geq \lambda \). Although cost-bounded reachability [3, 5] and maximal probabilistic reachability [13] (both on locations in P²TA) are known to be decidable, the problem of cost-bounded probabilistic reachability is not trivial; it may happen that the generated symbolic state space is infinite.

We constructed a backward algorithm that computes cost-bounded probabilistic reachability. It is the first (semi-)decidable algorithm that combines prices and probabilities for this class of models. In case the answer to the problem is affirmative, the algorithm will terminate. Furthermore, in cases of a finite symbolic state space it will also terminate. Based on this, we can conclude that cost-bounded maximal probabilistic reachability is at least semi-decidable.

The model can be applied, for example, to solve scheduling problems where the use of resources incurs cost and resources may fail with some probability. Is it still possible to ensure a high production quality within the given budget? Another application lies in interpreting cost as energy consumption in a battery-powered system with unreliable parts, for example a sensor network. Is a high reliability without depleting batteries achievable?

Directions for further research, and/or improvements are numerous, see [6] for some ideas.

References


