Fortuna: Model Checking Priced Probabilistic Timed Automata

Jasper Berendsen, David N. Jansen and Frits Vaandrager
Institute for Computing and Information Sciences
Radboud University Nijmegen
Nijmegen, the Netherlands
Email: {jasperb,dnjansen,fvaan}@cs.ru.nl

***Abstract*—We introduce FORTUNA, the first tool for model checking priced probabilistic timed automata (PPTAs). FORTUNA can handle the *combination* of real-time, probabilistic and cost features, which is required for addressing key design trade-offs that arise in many practical applications. For example the Zeroconf, Bluetooth, IEEE802.11 and Firewire protocols, protocols for sensor networks, and scheduling problems with failures. PPTAs are an extension of probabilistic timed automata (PTAs) with cost-rates and discrete cost increments on states. FORTUNA is able to compute the maximal probability by which a state can be reached under a certain cost-bound (and time-bound). Although this problem is undecidable in general, there exists a semi-algorithm that produces a non-decreasing sequence of probabilities converging to the maximum. This paper presents a number of crucial optimizations of that algorithm. We compare the performance of FORTUNA with existing approaches for PTAs. Surprisingly, although PPTAs are more general, our techniques exhibit superior performance.**

Supported by NWO/EW project 612.000.103 FRAAI, and the European Community's 7th Framework Programme No. 214755 (QUASIMODO).

***Keywords*-probabilistic timed automata; priced timed automata; probabilistic reachability; symbolic algorithms;**

I. INTRODUCTION

Model checking technology has initially been developed for finite state models. Both hardware and communication protocols may be modelled naturally in terms of finite state models, and in these areas model checking has been very successful [1]. In practice, however, finite state models are often not sufficiently rich. A characteristic of embedded and cyber-physical systems, is that they have to meet a multitude of *quantitative* constraints. These constraints involve the resources that a system may use (computation resources, power consumption, memory usage, communication bandwidth, costs, etc.), assumptions about the environment in which it operates (arrival rates, hybrid behaviour), and requirements on the services that the system has to provide (timing constraints, QoS, availability, fault tolerance, etc.). In order to handle quantitative constraints, model checking technology has been extended with features for specifying real-time, probabilistic behaviour, and costs. Efficient tools have been developed such as Uppaal [2], Uppaal Cora [3] and PRISM [4], that have been successfully applied to challenging problems in different areas [2]–[4]. Nevertheless, until now no model checking tool was able to handle the *combination* of real-time, probabilistic and cost features. For many practical applications, however, the key design trade-offs can only be addressed by models that incorporate all these features. We give three examples:

- Operation of the Zeroconf protocol [5] depends in an essential way on both timing and probabilities. In order to determine the optimal value for some parameters (like the number of retransmissions) one needs a cost function that combines timing delays and cost of failure [5], [6].
- Timing plays an essential role in communication protocols for sensor networks. In a network with battery-powered devices, the limited energy budget can be modelled using costs. Probabilities are needed to model node failure and message loss.
- In scheduling problems it may be useful to consider the probability that a resource (e.g., a production machine) breaks down or produces imperfect output.

This paper presents FORTUNA, the first model checking tool that is able to deal with the combination of probabilities, costs and timing. FORTUNA is based on the model of priced probabilistic timed automata (PPTAs) introduced in [7], [8]. PPTAs equip timed automata with prices and probabilities on discrete transitions. Cost-*rates* indicate the increase of cost per time unit, whereas prices on discrete transitions indicate instantaneous costs. PPTAs are the orthogonal extension of both probabilistic timed automata (PTAs) [9] and priced timed automata [10], [11], as PTAs extend timed automata with probabilities on discrete transitions and priced timed automata extend timed automata with prices. FORTUNA is able to compute the fundamental problem of cost-bounded maximal probabilistic reachability (CBMR) for PPTA. CBMR determines the maximal probability by which a state can be reached under a certain cost-bound (and time-bound.) Sections VI gives two examples that show the usefulness of CBMR in practice.

As PTAs are PPTAs with trivial cost parameters, we were able to compare the performance of FORTUNA with existing approaches for PTAs that compute maximal probabilistic reachability. The comparison is made on a number of existing PTA case studies to the best approaches available: the game-based verification of [12], the backwards reachability approach of [13], and the mcpta tool [14]. Surprisingly, although FORTUNA is more general, it shows the best performance.

FORTUNA adds a number of crucial optimizations to the algorithm described in [7], that performs symbolic backwards exploration. Like that work, FORTUNA only adds intersections of symbolic states to the state space, thereby reducing the number of stored states. To compute the probability, the explored symbolic state graph is transformed into a Markov decision process that is analysed with existing techniques. For PPTA FORTUNA may not terminate, since the problem is known to be undecidable in general [15]. But, for increasing exploration depth, the produced sequence of probabilities is non-decreasing and converges to the maximal probability [7].

The optimizations described in this article increase performance drastically. Three optimizations make modifications to the symbolic state graph that is explored, by generating abstractions that preserve probabilistic reachability. The proofs are done in a rigorous way by the use of (probabilistic) simulation relations. The last optimization employs Hasse diagram data structures to speed up comparisons between symbolic states.

Due to lack of space, some technical details and all proofs have been omitted, but they can be found in [16]. The FORTUNA tool and case studies discussed in this paper are available from http://www.cs.ru.nl/J.Berendsen/fortuna.

II. PRELIMINARIES

Markov Decision Processes: (MDPs) are widely used to formally model and analyse systems that exhibit both nondeterministic and probabilistic behaviour.

Definition 2.1: Let Act be a fixed set of actions. An *MDP* is a tuple $M = (S, s_{\mathsf{init}}, T)$, where S is a set of states, $s_{\mathsf{init}} \in S$ is the initial state, and $T \subseteq S \times \mathsf{Act} \times \mathrm{Dist}(S)$ is a probabilistic transition relation. Here $\mathrm{Dist}(S)$ is the set of subdistributions over S. We require that T is total in the sense that, for each state s, there exists an action a and a distribution μ such that $(s, a, \mu) \in T$.

Note that our definition of MDPs allows sub-distributions for the transition relation. Sub-distributions can be transformed to full distributions by adding a single trapping state to the MDP where all 'missing' probabilities lead to. For the rest of the paper we implicitly apply this transformation when needed.

A *probabilistic transition* $s \xrightarrow{a,\mu}$ is made by nondeterministically selecting an action a and a distribution $\mu \in \mathrm{Dist}(S)$ such that $(s, a, \mu) \in T$. A *transition* $s \xrightarrow{a,\mu} r$ is made by the probabilistic transition $s \xrightarrow{a,\mu}$ followed by choosing next state $r \in S$ with probability $\mu(r) > 0$. In case μ is a Dirac distribution we also write $s \xrightarrow{a} r$. An *infinite path* starting in state s_0 is an infinite sequence of transitions: $\omega = s_0 \xrightarrow{a_0,\mu_0} s_1 \xrightarrow{a_1,\mu_1} s_2 \xrightarrow{a_2,\mu_2} \cdots$ such that $(s_i, a_i, \mu_i) \in T$ for all i. Let ω^i denote the ith state in the path ω, i.e. $\omega^i = s_i$. A *finite path* is a finite prefix of an infinite path. We denote the last state in a finite path ω by $\mathrm{last}(\omega)$. The *length* of a path is the number of transitions that it contains. A path of length 0 consists only of the starting state. We assume a special action label $\tau \in \mathsf{Act}$ that denotes *internal* transitions. We assume internal transitions to be non-probabilistic, i.e. $s \xrightarrow{\tau,\mu}$ implies $\mu = \{r \mapsto 1\}$, for some r. When we have a finite path of τ-transitions $s_0 \xrightarrow{\tau} \cdots \xrightarrow{\tau} s_n$ we write $s_0 \Rightarrow s_n$.

An *automaton* can be seen as an MDP, where the probabilistic transition relation uses only Dirac distributions. In this case, for ease of notation, we may write the transition relation as $D \subseteq S \times \mathsf{Act} \times S$.

To reason about an MDP, we have to resolve its nondeterministic choices. To this end, we consider *deterministic policies* (also called strategies, schedulers, or adversaries.) A deterministic policy is a function mapping every finite path ω in an MDP to an action $a \in \mathsf{Act}$ and a distribution $\mu \in \mathrm{Dist}(S)$ such that $(\mathrm{last}(\omega), a, \mu) \in T$. By classical techniques, for some policy A, we can define a probability measure on a set of infinite paths leaving from a state. Moreover, for a set of states G, we can define the *reachability probability* $\mathsf{ProbReach}_A(G)$, which is the likelihood to reach a state in G in a finite number of transitions under A, starting from the initial state. Similarly, we can define $\mathsf{ProbReach}^{\leq n}_A(G)$, where only paths that visit G in at most n transitions contribute to the probability. Since we are interested in optimal behaviour, we consider *maximal reachability probability*, for arbitrary paths and bounded paths, respectively: $\mathsf{SupProbReach}_M(G) \stackrel{\mathsf{def}}{=} \sup_A \mathsf{ProbReach}_A(G)$ and $\mathsf{SupProbReach}^{\leq n}_M(G) \stackrel{\mathsf{def}}{=} \sup_A \mathsf{ProbReach}^{\leq n}_A(G)$, for some MDP M.

For finite MDPs, these probabilities and the optimal policies can be computed using several techniques including: value iteration, (modified) policy iteration and linear programming [17]. From that work we also know that randomized instead of deterministic policies will not improve on the maximal reachability probability.

III. PRICED PROBABILISTIC TIMED AUTOMATA

A *clock* is a real-valued variable that can be used to measure the elapse of time. All clocks run at the same pace. A *clock valuation* assigns a non-negative value to each clock in some set $\mathbb{X}$. Let $\mathbb{R}^{\mathbb{X}}_{\geq 0}$ denote the set of all possible clock valuations. For $d \in \mathbb{R}_{\geq 0}$, let $v+d$ denote the clock valuation that maps each $x \in \mathbb{X}$ to $v(x)+d$. For $R \subseteq \mathbb{X}$, let $v[R := 0]$ denote the clock valuation in which the clocks in R have been reset, i.e. $v[R := 0](x)$ equals 0 if $x \in R$ and $v(x)$

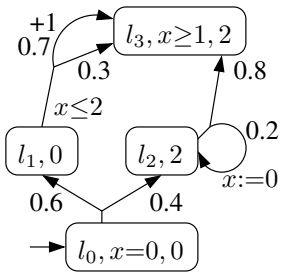


Figure 1. Example PPTA

otherwise. A *guard* is a conjunction of inequalities where the value of a single clock is compared to an integer. Formally, the set $\mathsf{Guards}(\mathbb{X})$ of guards g is defined by the grammar:

$$g ::= x \bowtie b \mid g \wedge g \mid \mathsf{true}, \quad \text{where } x \in \mathbb{X}, b \in \mathbb{N}, \bowtie \in \{<, \leq, \geq, >\}.$$

We write $v \models g$ when valuation v satisfies the constraints of guard g.

Definition 3.1: A *priced probabilistic timed automaton* (PPTA) is a tuple $\mathcal{A} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{inv}, \mathsf{edges}, \dot{\$})$, where:

- L is a finite set of locations;
- $l_{\mathsf{init}} \in L$ is the initial location;
- $\mathbb{X}$ is a finite set of clocks;
- $\mathsf{inv} : L \to \mathsf{Guards}(\mathbb{X})$ assigns an invariant to each location;
- $\mathsf{edges} \subseteq L \times \mathsf{Guards}(\mathbb{X}) \times \mathrm{Dist}(2^{\mathbb{X}} \times \mathbb{N} \times L)$ is a finite set of edges; and
- $\dot{\$} : L \to \mathbb{N}$ associates a cost-rate with each location.

For edge $(l, g, p) \in \mathsf{edges}$, l denotes the source location, g the guard, and p a distribution over *instantaneous effects*, which consist of a set of clocks to be reset, a cost increment, and a destination location. For the rest of the paper we fix a PPTA $\mathcal{A} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{inv}, \mathsf{edges}, \dot{\$})$.

Example 1: Figure 1 shows a PPTA with clock x. Locations are represented by rounded boxes, with branching arrows between them denoting the edges of the PPTA. The initial location is l_0. Invariants and cost-rates are written in the locations. Guards (e. g. $x{\leq}2$) are placed next to the source location; probabilities, resets and cost increments are at the branches (e. g. probability 0.3 and $x{:=}0$.) Together, the reset, cost increment, and target location, from an instantaneous effect. Probabilities 1, guards that always hold, and instantaneous cost increments of 0 are left out.

Intuitively, a PPTA behaves as follows. It always is in a state consisting of a location l, a clock valuation v, and the cost incurred until now c. A policy makes the nondeterministic choice between delaying or which edge to take. Only edges with guards satisfying the current valuation are available. Delaying will increase each clock by the quantity of delay and the accumulated cost by the quantity of delay times the cost-rate $\dot{\$}(l)$. When taking an edge, the instanteneous effect is chosen probabilistically. This determines which clocks are reset, a cost increment, and the next location.

Definition 3.2: Fix $\mathsf{Act} = \mathbb{R}_{\geq 0} \cup \{\tau\}$. The *semantics* of PPTA $\mathcal{A}$, denoted $[\![\mathcal{A}]\!]$, is given by the MDP $(S, s_{\mathsf{init}}, T)$, where $S = \{(l, v, c) \mid l \in L \wedge v \models \mathsf{inv}(l) \wedge c \in \mathbb{R}\}$, i.e. states consist of a location, a clock valuation, and the accumulated cost; $s_{\mathsf{init}} = (l_{\mathsf{init}}, \{x \mapsto 0 \mid x \in \mathbb{X}\}, 0)$; and $((l, v, c), d, \mu) \in T$ iff $(l, v, c) \in S$, $d \in \mathbb{R}_{\geq 0}$, $v + d \models \mathsf{inv}(l)$ and one of the following conditions holds:

- either $\mu(l, v{+}d, c + \dot{\$}(l)d) = 1$
- or there exists $(l, g, p) {\in} \mathsf{edges}$ such that $v{+}d {\models} g$ and for all $(l', v', c') \in S$: $\mu(l', v', c') = \sum_{R \subseteq \mathbb{X} \text{ s.t.} v' = (v+d)[R:=0]} p(R, c' - d{\cdot}\dot{\$}(l) - c, l')$

If the first condition holds then we call $((l, v, c), d, \mu)$ a *time transition,* otherwise we call it a *delayed discrete transition.*

Cost-bounded maximal reachability (CBMR) is the maximal probability by which a goal location in a PPTA can be reached, without the accumulated cost exceeding some bound. For PPTA $\mathcal{A}$ we fix $l_{\mathsf{goal}} \in L$ to be the goal state, and $c_{\mathsf{bound}} \in \mathbb{N}$ to be the cost-bound.

Definition 3.3: CBMR is the probability $\mathsf{SupProbReach}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}})$, where $\sigma_{\mathsf{goal}} = \{l_{\mathsf{goal}}\} \times \mathbb{R}^{\mathbb{X}}_{\geq 0} \times [0, c_{\mathsf{bound}}]$.

We now define predecessor operations on sets of states of $[\![\mathcal{A}]\!]$. The *discrete predecessor* of a set of states Z via edge e and instantaneous effect f, gives all states in $[\![\mathcal{A}]\!]$ that can reach some state in Z via edge e and instantaneous effect f. The *time predecessor* of a set of states Z, gives all states in $[\![\mathcal{A}]\!]$ that can reach some state in Z by letting time elapse.

Definition 3.4: Let $[\![\mathcal{A}]\!] = (S, s_{\mathsf{init}}, T)$. For any $Z \subseteq S$, $e = (l, g, p) \in \mathsf{edges}$. Let $f = (R, h, l')$ such that $p(f) > 0$. The discrete predecessor and time predecessor operations are:

$\mathsf{dpre}_{e,f}(Z) \stackrel{\mathsf{def}}{=} \{(l, v, c) {\in} S \mid v {\models} g \wedge (l', v[R{:=}0], c{+}h) {\in} Z\}$

and $\mathsf{tpre}(Z) \stackrel{\mathsf{def}}{=} \{(l, v, c) {\in} S \mid \exists d {\geq} 0 . (l, v{+}d, c{+}d{\cdot}\dot{\$}(l)) {\in} Z\}$.

IV. The Algorithm

Algorithm 1 is used for computing CBMR for paths of length up to $\mathsf{maxlength}$ (possibly ∞). It gets as inputs a CBMR problem and $\mathsf{maxlength}$.[1] Algorithm 1 returns a finite automaton $(\mathit{Visited}, \sigma_{\mathsf{init}}, D)$, which we will call *reachability graph*. The reachability graph captures symbolically all transitions by which a target state may be reached. Definition 4.1 defines an MDP M for the reachability graph. Now, from Theorem 4.2, we see that: 1) the maximal reachability probability in M is an upperbound on the CBMR solution. 2) if $\mathsf{maxlength} = \infty$ the upperbound is precise. 3) increasing $\mathsf{maxlength}$ leads to a higher or equal upperbound on CBMR with unbounded lenght.

Zones are sets of states of $[\![\mathcal{A}]\!]$ that share the same location. Since clocks and cost take real values, zones may

[1] As a minor condition, $\mathsf{inv}(l_{\mathsf{init}})$ should require all clocks to be zero.

Algorithm 1: The basic backwards reachability algorithm

1 **Input:** PPTA $\mathcal{A} = (L, l_{\mathsf{init}}, \mathbb{X}, \mathsf{inv}, \mathsf{edges}, \dot{\$}), l_{\mathsf{goal}}$, $c_{\mathsf{bound}}, \mathsf{maxlength}$, where $\mathsf{inv}(l_{\mathsf{init}}) = \bigwedge_{x \in \mathbb{X}}(x = 0)$
Output: $(\mathit{Visited}, \sigma_{\mathsf{init}}, D)$
2 $\sigma_{\mathsf{goal}} := l_{\mathsf{goal}} \times \mathsf{inv}(l_{\mathsf{goal}}) \times [0, c_{\mathsf{bound}}]$
3 $\sigma_{\mathsf{init}} := \{(l_{\mathsf{init}}, \{x \mapsto 0 \mid x \in \mathbb{X}\}, 0)\}$
4 **if** $\sigma_{\mathsf{init}} \subseteq \sigma_{\mathsf{goal}}$ **then** $\sigma_{\mathsf{init}} := \sigma_{\mathsf{goal}}$
5 $\mathit{Visited} := \{\sigma_{\mathsf{goal}}, \sigma_{\mathsf{init}}\}$
6 $D := \{(\sigma_{\mathsf{goal}}, \tau, \sigma_{\mathsf{goal}}), (\sigma_{\mathsf{init}}, \tau, \sigma_{\mathsf{init}})\}$
7 $\mathit{Waiting}_0 := \{\sigma_{\mathsf{goal}}\}$
8 **for** $i := 1$ **to** maxlength
9 **if** $\mathit{Waiting}_{i-1} = \emptyset$ **then break**
10 $\mathit{Waiting}_i := \emptyset$
11 **foreach** $\rho \in \mathit{Waiting}_{i-1}$
12 **foreach** $e \in$ edges and f is an inst. effect of e going to location of ρ
13 $\sigma := \mathsf{dpre}_{e,f}(\mathsf{tpre}(\rho))$
14 **if** $\sigma \neq \emptyset$
15 $D := D \cup \{(\sigma, e, f, \rho)\}$
16 **if** $\sigma \notin \mathit{Visited}$ **then** $\mathit{Waiting}_i := \mathit{Waiting}_i \cup \{\sigma\}$, $\mathit{Visited} := \mathit{Visited} \cup \{\sigma\}$
17 **foreach** $(\sigma', a', \rho') \in D$
18 **if** $\sigma \cap \sigma' \neq \emptyset$
19 $D := D \cup \{(\sigma \cap \sigma', e, f, \rho), (\sigma \cap \sigma', a', \rho')\}$
20 **if** $\sigma \cap \sigma' \notin \mathit{Visited}$
21 $\mathit{Waiting}_i := \mathit{Waiting}_i \cup \{\sigma \cap \sigma'\}$, $\mathit{Visited} := \mathit{Visited} \cup \{\sigma \cap \sigma'\}$
22 **return** $(\mathit{Visited}, \sigma_{\mathsf{init}}, D)$

contain infinitely many states. In [7] it is shown that the zones generated by our algorithm have a finite representation called multi-priced zones, which are a subclass of convex polyhedra.

Visited are the zones that are generated by Algorithm 1 in a backward fashion: starting from the zone of goal states (σ_{goal}), more zones are generated by repeatedly computing predecessors of explored zones. The predecessor of a zone is computed by first computing the time predecessor, and from that the discrete predecessor. The algorithm proceeds in a breadth-first way. In addition to predecessors, intersections of explored zones are added to *Visited*, which are zones themselves.

Throughout this work, edges of a reachability graph may be called *directions* to distinguish them from the edges in a PPTA. An important property of any direction (σ, e, f, ρ) generated by Algorithm 1 is that all states in σ can reach a state in ρ by a delayed discrete transition with 0 delay using edge e and instantaneous effect f, followed by a time transition. In the algorithm, *Visited* maintains the visited zones, and $\mathit{Waiting}_i$ contains the zones to be explored after iteration i. Backwards exploration starts from the zone of goal states σ_{goal}. The zone σ_{init} contains the initial state and becomes the initial state of the reachability graph. D maintains the directions of the reachability graph. Lines 13–15 are concerned about the predecessor of a zone and the resulting direction. Lines 17–21 are concerned with the intersections. Intersections are crucial for the following reason. Given zones ρ and ρ', let $\sigma = \mathsf{dpre}_{e,f_1}(\mathsf{tpre}(\rho))$ and $\sigma' = \mathsf{dpre}_{e,f_2}(\mathsf{tpre}(\rho'))$ for some $e \in$ edges and f_1, f_2 instantaneous effects of e. Now, $\sigma \cap \sigma'$ contains the states that can reach ρ as well as ρ': A state in ρ is reached by taking edge e, probabilistically choosing f_1, and delaying some time. A state in ρ' is reached by taking the same edge e, probabilistically choosing f_2, and delaying some (possibly different) time. Whenever $f_1 \neq f_2$ and the goal can be reached via both ρ and ρ', the probabilities of choosing either effect can be added. This explains why we first took the time predecessor and then the discrete predecessor: only after the probabilistic choice has been resolved, the policy gets the information whether f_1 or f_2 was chosen and can decide how long to delay to reach ρ or ρ', respectively. This reasoning can easily be generalized to intersections of more than two zones. At line 19, the intersection gets outgoing directions to the target zones of the two directions. These directions capture the possibility of states in the intersection to reach σ_{goal} via either way. Note that in the next iterations more directions may be added that start from the same intersection.

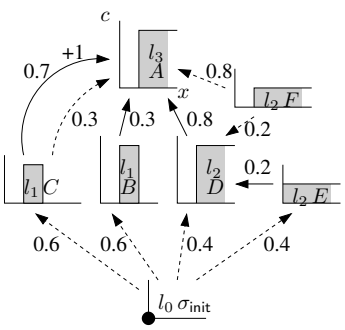


Figure 2. Reachability graph

Example 2: Figure 2 shows the reachability graph that Algorithm 1 would generate for the PPTA of Figure 1 if $l_{\mathsf{goal}} = l_3$ and $c_{\mathsf{bound}} = 3$. In the figure, zones are labeled $\sigma_{\mathsf{init}}, A, B, \ldots$, and are represented by planes in coordinate systems with x and c at the axes. Furthermore, zones have a location. Labels on directions are left out and probabilities are added in such a way that there is no ambiguity from which edge in the PPTA they were generated. Plain edges are the ones resulting from line 15. Thus, for example B is the predecessor of A. The predecessors of $B, \ldots, E$ are left out for brevity. Dashed edges are the ones resulting from line 19. Note that F is the intersection of D and E, while C is the intersection of C and B. Zone F has only an empty predecessor. The next definition is used to transform any reachability graph that is generated by Algorithm 1 into an

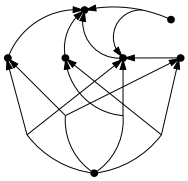


Figure 3. Without optimizations

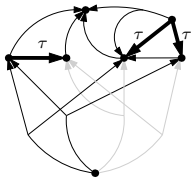


Figure 4. MDP when using Optimization 1

MDP. As such, it generates the symbolic semantics for the PPTA under the given CBMR problem.

Example 3: Figure 3 shows the MDP of Definition 4.1 for the reachability graph of Figure 2. The positions of the states correspond to the positions of the zones in the reachability graph. For brevity probabilities and labels have been left out.

Definition 4.1: Given reachability graph $Q = (S, s_{\mathsf{init}}, D)$, we define the Markov decision process $\mathsf{MDP}(Q) = (S, s_{\mathsf{init}}, T)$, where $(s, a, \mu) \in T$ if and only if either

- $a = \tau$ and $\mu = \{r \mapsto 1\}$ and there exists $(s, \tau, r) \in D$; or
- $a = e$ and there exists $D_{e,\mu} \subseteq D$ such that
  1) $\forall (s', e', f', r') \in D_{e,\mu}.s' = s \wedge e' = e$
  2) $\forall (s_1, e_1, f_1, r_1), (s_2, e_2, f_2, r_2) \in D_{e,\mu}.$ $r_1 \neq r_2 \implies f_1 \neq f_2$
  3) $D_{e,\mu}$ is maximal
  4) $\forall r \in S.\mu(r) = \sum_{(s,(l,g,p),f,r) \in D_{e,\mu}} p(f)$

Theorem 4.2: Assume Algorithm 1 is executed with input PPTA $\mathcal{A}$, location l_{goal}, $c_{\mathsf{bound}} \in \mathbb{N}$, and $\mathsf{maxlength} \in \mathbb{N} \cup \{\infty\}$. And on the output we define M by Definition 4.1. Now all of the following hold:

1) $\forall n \leq \mathsf{maxlength}.$ $\mathsf{SupProbReach}^{\leq n}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}}) \leq \mathsf{SupProbReach}_M(\{\sigma_{\mathsf{goal}}\})$
2) If $\mathsf{maxlength} = \infty$ we have that: $\mathsf{SupProbReach}_{[\![\mathcal{A}]\!]}(\sigma_{\mathsf{goal}}) = \mathsf{SupProbReach}_M(\{\sigma_{\mathsf{goal}}\})$
3) Let $\mathsf{maxlength} = m \neq \infty$, and construct M' in the same way as M except that $\mathsf{maxlength}$ is set to $m + 1$. Then, the probability does not decrease: $\mathsf{SupProbReach}_M(\{\sigma_{\mathsf{goal}}\}) \leq \mathsf{SupProbReach}_{M'}(\{\sigma_{\mathsf{goal}}\})$

Definition 4.3: Given a set of zones Σ, for any $\rho, \sigma \in \Sigma$, we say ρ *covers* σ, written $\rho \supset\!\!- \sigma$, iff $\rho \supset \sigma$ and there exists no $\alpha \in \Sigma$ such that $\rho \supset \alpha \supset \sigma$.

Optimization 1 extends Algorithm 1. To understand its basic idea, regard two zones ρ and ρ' with $\rho \supset \rho'$. All states in a zone have the same probabilistic transitions available, as long as these contribute to reaching the goal with maximal probability. If execution of the MDP enters ρ', this corresponds to an execution of the semantics that enters some state $r \in \rho'$. But, execution of the MDP might as well enter ρ, since $r \in \rho$. Therefore, we may add a τ-direction from ρ' to ρ. A τ-direction has probability 1 and does not correspond with any transition in the semantics. Fragment B adds the minimal number of τ-directions to obtain a path of τ-directions from any subzone to a superzone. The optimization comes from Fragment A that removes directions from any zone σ to ρ, when ρ' is reachable from σ with the same label. A policy can simply go to ρ by first going to ρ' and then taking τ-directions.

Optimization 1:
Fragment A: Inserted after lines 15 and 19 of Alg. 1.

15bis,19bis **foreach** $(\sigma, e, f, \rho), (\sigma', e', f', \rho') \in D$, with $(\sigma', e', f') = (\sigma, e, f)$
15ter,19ter **if** $\rho' \subset \rho$ **then** $D := D \backslash \{(\sigma, e, f, \rho)\}$

Fragment B: Replaces line 22 of Algorithm 1.

$22'$ **foreach** $\rho, \rho' \in$ *Visited*
$23'$ **if** $\rho \supset\!\!- \rho'$ **then** $D := D \cup \{(\rho', \tau, \rho)\}$
$24'$ **return** $(\mathit{Visited}, \sigma_{\mathsf{init}}, D)$

The benefit of Optimization 1 comes from the way Definition 4.1 works: given an edge, sets $D_{e,\mu}$ are constructed by taking all possible (maximal) combinations of instantaneous effects of that edge. Given some edge e with n instantaneous effects, a zone with m_i outgoing directions that use the i-th instantaneous effect would have $m_1 \cdot m_2 \cdot \ldots \cdot m_n$ possibilities for $D_{e,\mu}$, which gives a blow-up exponential in n.

Example 4: Recall Example 2. With Optimization 1 the reachability graph will differ from the one in Figure 2: the direction from σ_{init} to B will not be present, and there will be τ-directions (C, τ, B), (F, τ, D) and (F, τ, E). The generated MDP is depicted in Figure 4. The thin edges are edges no longer present due to the optimization, whereas the thick edges are added.

Theorem 4.4: Assume that we change Algorithm 1 with Optimization 1, then Theorem 4.2 still holds.

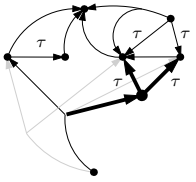
Intersections are only useful if they capture probabilistic branching. Optimization 2 is straightforward, and is also made in [13] to suppress intersections that have only outgoing transitions with the same probability resolution.

Optimization 2: Replaces line 18 of Algorithm 1.

$18'$ **if** $\sigma \cap \sigma' \neq \emptyset$ **and** $(\sigma' = \sigma_{\mathsf{init}}$ **or** $\exists f' \neq f.a' = (e, f'))$

Theorem 4.5: Both for Algorithm 1 changed with Optimizations 1 and 2, and Algorithm 1 changed with only Optimization 2, theorem 4.2 still holds.

We now define a transformation from a reachability graph Q to a reachability graph $\bar{Q}$. The transformation reduces the number of probabilistic transitions in $\mathsf{MDP}(\bar{Q})$ w.r.t.



R2.8cm

Figure 5. MDP with all optimizations

$\mathsf{MDP}(Q)$, but does not affect the maxim reachability probability. The zones of Q are maintained by the transformation, but $\bar{Q}$ has extra *intermediate* zones. Two or more directions leaving a zone in Q that have the same label are replaced in $\bar{Q}$ by a direction with the same label going to a fresh intermediate zone. From the fresh intermediate zone there are τ-directions going to the goal zones of the original directions in Q. The benefit is the same as for Optimization 1: we reduce the number of outgoing directions from a state that have the same label.

Example 5: Recall Example 4. Optimization 3 creates an intermediate state with τ transitions to zone D and zone E. Figure 5 shows the generated MDP.

In $\bar{Q}$ all directions leaving a zone have a different label, except for τ-directions, that all use the label τ. Because of these properties of directions in $\bar{Q}$, from Definition 4.1 we can see that for each zone and each direction leaving that zone there is only one possibility to construct a probabilistic transition. Each direction has a corresponding τ-direction from the intermediate zone and there is one direction to the intermediate zone, so in total at most one extra direction is needed per instantaneous effect per zone.

Optimization 3: Given reachability graph $(S, s_{\mathsf{init}}, D)$. We define the reachability graph $(S \cup I, s_{\mathsf{init}}, \bar{D})$, where $I \subseteq S \times \mathsf{Act}$ and for any $(s, a, r) \in D$

- if $a \neq \tau$ and there exists $(s, a, r') \in D$ such that $r' \neq r$, then $(s, a, (s, a)) \in \bar{D}$ and $((s, a), \tau, r) \in \bar{D}$.
- otherwise: $(s, a, r) \in \bar{D}$.

Theorem 4.6: Given reachability graph Q. Let $\bar{Q}$ be obtained from Q by applying the transformation of Optimization 3, then for any G:

$$\mathsf{SupProbReach}_{\mathsf{MDP}(\bar{Q})}(G) = \mathsf{SupProbReach}_{\mathsf{MDP}(Q)}(G)$$

It is not hard to see that also without Optimization 1, Optimization 3 will still make sure each zone has no two outgoing directions with the same label, except for τ-directions. However, Optimization 1 is useful for two reasons: 1) It works during the construction of the reachability graph, i.e. directions from a zone are removed as soon as it gets a new outgoing direction. 2) It needs no extra 'intermediate' states.

V. Implementation Issues

A straightforward enhancement FORTUNA employs is to consider only locations that are reachable. To this end a standard symbolic *forward* exploration is performed, not taking probabilities into account. Forward exploration cannot be used to calculate the probability directly. It can only give an upperbound [9]. Since there are no guards on cost, lowering the cost in a state by say C will not influence the locations reachable, although they are reachable with C less cost. The exploration amounts to minimum cost reachability in a priced timed automaton which is a decidable problem [10]. We use convex polyhedra instead of the multi-priced zones of [7] to represent zones, because there is a advanced library for them available: the Parma Polyhedra Library [18]. The library offers operations on convex polyhedra, such as intersection, inclusion checking, and time predecessor. Another advantage is that convex polyhedra will allow for an extension to more general classes of automata, such as the probabilistic linear hybrid automata of [19]. A disadvantage could be reduced performance. Some operations of the Parma Polyhedra Library take considerable computation time, most notably the inclusion and intersection operations on polyhedra. To reduce the number of intersections, we maintain a Hasse diagram structure for zones that share the same location. The Hasse diagrams use the $\supset\!-$ relation to order their zones. Now, as a new zone enters the explored state space on line 17, it will be inserted in the Hasse diagram that corresponds to its location. This means more inclusion checks to determine the position of the new element in the Hasse diagram, but the benefit comes at line 18 of Algorithm 1: If one zone includes the other, the smaller is the intersection of the two, and inclusion can now be quickly decided based on the Hasse diagram. Additionally, we reuse the diagram for the τ-directions of Optimization 1: they are no longer stored.

VI. Case Studies and Model Checking Results

We present two simple case studies that illustrate the practical usefulness of PPTAs and CBMR. In addition, we present experimental results for some PTA case studies, taken from [12], which do not include costs. Even though these case studies only exploit part of the functionality of FORTUNA, they allow us to compare the performance of FORTUNA to other tools. We also demonstrate the usefulness of our optimizations by comparing non-optimized versions of our algorithm to optimized ones. FORTUNA uses a CCS style parallel composition, whereas the approaches that we compare with have a CSP style composition. Due to this, the models used by the different tools are not entirely isomorphic. We tried to stay as close as possible to the original case studies. We did not change the number of locations in the PTAs, but only added intermediate locations on some transitions when needed. Two of the case studies are concerned with computing minimal probabilistic reachability. In contrast to the other two methods, FORTUNA does not directly compute the minimum, but computes maximal probabilistic reachability instead. For these particular models the minimum can be computed using maximal probabilistic

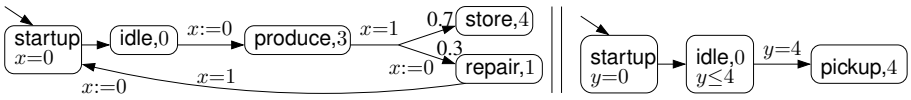


Figure 6. A Production Plant

reachability using the technique described in [13] that works on slightly augmented models. Assume we want to compute minimal probabilistic reachability for the goal location l_{goal} within some deadline, where l_{goal} is reachable from the initial state with probability 1 for any policy. We augment the model with a new location l_{ex} that denotes that the deadline is exceeded. From all locations other than l_{goal} we add edges to l_{ex} that are enabled as soon as the deadline is exceeded. We can now compute the maximal probability of reaching l_{ex}. One minus this value gives the wanted minimal probability. All experiments are carried out on a 2GHz PC with 2GB RAM, which is similar to the hardware used in [12]. We do not compare on memory use as these statistics from the other approaches are unavailable to us. However, for the instance of the case study that is the largest in both states and directions (firewire 100k), FORTUNA needs only 70MB.

A production plant: This case study has been inspired by a case study on a lacquer production plant [20]. Although small, it easily scales up to more elaborate plant models. The PPTA on the left of Figure 6 models a production plant. Initially the system is in the startup location, but goes immediately to the idle location. At some point in time, a scheduler may decide to produce the lacquer. Production takes 1 day and costs 3 credits. With probability 0.7 production succeeds and the lacquer is stored, which costs 4 credits per day. With probability 0.3 production fails; the machine needs to be cleaned, after which production can start again. The PPTA on the right of Figure 6 models the customer. After 4 days the customer will try to pick up the product. The CBMR is the maximal probability to reach the locations store and pickup with cost at most 9. Note that leaving a customer waiting also costs 4 credits per day. FORTUNA calculates a maximal probability of 0.91. In order to realize this, the plant scheduler should wait for 1.5 days before starting production.

CSMA with energy constraints: IEEE 802.3 CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) is a protocol to avoid data collision on a single channel. The authors of [13] model CSMA with PTAs and are able to compute the maximal probability that both senders have successfully sent their data within a deadline. In certain settings power consumption is important. Sending data consumes power, and typically when a node is listening to receive data, it consumes more power than in other modes. We build a PPTA from the PTA in [13], due to lack of space we refer to their figures and further explanation. We added the following cost to their model: an instantaneous cost of 50 upon a *send*, cost-rate 1 in the *wait* and *done* locations, and cost-rate 3 in the *transmit* location. All other locations use cost-rate 0. With CBMR we are now able to compute the maximal probability by which both nodes are done, but total power consumption is not more than c_{bound}.

Experiments on optimizations: We have presented Optimizations 1–3 and the optimization that uses Hasse diagrams. We conjecture Optimizations 2 and 3 are always useful, as early experiments indicated, and do not experiment with turning them off. The benefit of the other optimizations becomes apparent from Table I. The first column shows the situation when using only Optimizations 2 and 3. The second column puts Optimization 1 into play. Finally, the last column also adds the use of Hasse diagrams. The implementation gives no guarantees on the order in which zones are explored. The explored zones determine which directions are present. Therefore when trying to add an intersection, Optimization 2 may or may not suppress this. As a result, the number of zones, directions, and τ-directions may vary between different runs of the algorithm. Each experiment is repeated 10 times. We use the format $a{\pm}b$ to express the calculated mean a and the calculated standard deviation b, where b has the same significance as the least significant digit of a. For each optimization level the number of generated states is approximately the same. We have displayed the number of directions for each optimization level, and the number of τ-directions when using Hasse diagrams. The second column shows a strong reduction in the number of directions, for all case studies except "csma". The effect is mainly a reduction in memory usage. Also the benefit of using Hasse diagrams is clear. There are some great reductions in the number of directions, as well as in the run-time.

Comparison to other Approaches: Table II compares the performance of FORTUNA to the *game-based verification* approach of [12], and the *backwards reachability* approach of [13]. The statistics are take from [12]. The probabilities computed by FORTUNA, as shown in the last column, vary slightly from those results on the larger instances as a result of rounding errors. The *backwards reachability* approach is included in the comparison because it is closest to our approach in its workings. Other approaches for maximal probabilistic reachability on PTA use quite different techniques. Game-based verification [12] uses an abstraction–refinement scheme to iteratively generate tighter lower and upper bounds on the solution. The digital clocks

Table I
PERFORMANCE STATISTICS OF THE OPTIMIZATIONS

Case study (parameters) [min /max]		Optimizations 2 and 3		Optimizations 1–3		Optimizations 1–3 and Hasse diagrams		
		Dirs.	Time (s)	Dirs.	Time (s)	Dirs.	τ-dirs.	Time (s)
csma_cost	8k	2357±29	3.041±35	1538±20	2.873±28	947±0	956±0	**1.831±26**
(c_bound)	9k	5283±140	12.98±11	2780±27	12.026±63	1697±1	1795±0	**4.388±32**
[max]	10k	12589±250	67.56±35	5277±199	62.684±21	2754±0	2922±0	**11.08±16**
	11k	28464±1004	296.1±103	9921±543	271.6±11	4223±0	4662±0	**26.59±18**
csma	2 4	374±0	0.267±9	360±0	**0.265±7**	290±0	172±0	0.270±7
(max_backoff	2 8	1002±0	0.752±12	948±0	**0.726±5**	742±0	476±0	0.735±8
collisions)	4 4	1999±0	2.406±27	1941±0	**2.366±18**	1593±0	812±0	2.418±34
[max]	4 8	4617±0	7.901±62	4247±0	**7.223±37**	3273±0	2092±0	7.241±38
csma_abst	1k 1	781±1	0.467±14	574±3	0.459±22	362±0	309±0	**0.323±19**
(deadline	2k 1	1591±5	0.863±15	1036±7	0.811±17	602±0	552±0	**0.627±24**
bcmax)	3k 1	10790±5	7.053±82	8375±125	6.879±59	1499±0	1527±0	**3.445±49**
[min]	1k 2	5236±3	4.401±95	2263±6	3.99±13	1298±1	1061±0	**1.885±54**
	2k 2	13576±27	17.03±11	5326±52	13.92±18	2955±0	2737±0	**7.75±18**
	3k 2	51856±123	102.8±10	25492±425	83.68±87	5298±0	5354±0	**46.14±44**
firewire_abst	5k	290±0	0.034±5	220±2	0.029±6	102±0	52±0	**0.024±5**
(deadline)	10k	1727±0	0.216±5	1284±0	0.192±6	276±0	169±0	**0.081±6**
[min]	20k	18694±2	4.610±20	14864±2	4.135±16	946±0	629±0	**0.587±8**
	100k	$>$ 1 hour		$>$ 1 hour		20884±0	14516±0	**221.8±11**
nrp_malicious	5	567±15	0.204±12	312±3	0.192±10	244±3	168±5	**0.141±9**
(deadline)	10	2300±25	1.629±29	997±5	1.480±24	654±12	478±5	**0.711±14**
[max]	20	7404±37	13.07±10	2569±14	12.043±45	1436±27	1107±4	**3.135±73**

approach of [8] translates a PTA in a finite state machine where all clock values are discretized. It is proven that discretization does not change reachability probabilities. We do not compare with the digital clocks approach since the same authors have shown game-based verification to be much faster [12]. We have also looked at the approach of [14] that automatically generates a digital clocks model from a powerful modeling language that subsumes PTAs. It is shown that in most cases the automatic generation and verification of a digital clocks model is faster than the verification of a manually made digital clocks model as in [8]. Although an elaborate comparison is preferred, FORTUNA seems much better in terms of time and space for model checking. The model "csma_abst" is also used in [14], but with deadline 1800. Verification takes 230 seconds and 305 MB, on a faster machine. Uppaal-Pro is another tool for checking maximal reachability on PTA, available from [21]. Since, at the time of writing, development of Uppaal-Pro has not stabilized yet, we have not included it in our comparison. As FORTUNA uses the backward reachability approach with new optimizations, it improves on the latter. For all instances FORTUNA is faster than game-based verification, often several orders of magnitude. Why FORTUNA out-performs game-based verification is hard to say, as both approaches are very different in nature. However, we see the following possible reasons: 1) Like backward reachability, FORTUNA does not calculate the difference between two zones, but only intersections. As a result, the number of states is much smaller, as can be seen in the table. 2) FORTUNA does forward exploration of the reachable state space. 3) FORTUNA uses the efficient Parma Polyhedra Library [18] to do operations on zones. 4) FORTUNA has been implemented in C++, but we do not know the implementation language for the other tools.

VII. CONCLUSION

We have presented FORTUNA, the first tool for model checking PPTA. FORTUNA is able to compute CBMR. It uses novel optimizations that drastically improve the backward reachability algorithm. Although FORTUNA is more general, it outperforms existing tools for PTAs by several orders of magnitude on a number of case studies in computing maximal probabilistic reachability. Obvious directions for future research are extending the PPTA model or extending the properties that can be analyzed. Linear hybrid systems with discrete probabilistic [19] branching strictly include PPTAs. Since our implementation uses general polyhedra, an extension in this direction could be easy. Like in the case studies, for special cases minimal probabilistic reachability can be computed with maximal probabilistic reachability on a transformed model. However, in general one would need to implement new techniques like the ones in [13]. Other properties include expected cost reachability [8] or probability-bounded minimal cost reachability. In the latter, compared to CBMR, the probability is bounded instead of the cost, and the cost is optimized instead of the probability. Finally it is useful to have a logic, like PTCTL [9], in which properties can be expressed, and simple properties can be combined to more elaborate ones. Users of FORTUNA enter models as hard-wired C++ code, using calls to an interface. Although this interface is quite clear, a user-interface is preferable. To increase useablity, the actual policy and the traces it generates should be given as feedback to the user. Another interesting feature would be to output the probability at each depth of exploration. This sequence of probabilities is non-decreasing. The algorithm may be stopped when the outcome is large enough compared to some objective. In this iterative approach at each iteration one can benefit from the probabilities calculated in the previous iteration.

Table II
PERFORMANCE STATISTICS AND COMPARISONS

Case study (parameters) [min /max]		FORTUNA		Game-based verification [12]		Backwards reachability [13]		Min/Max reachability probability
		States	Time (s)	States	Time (s)	States	Time (s)	
csma	2 4	224±0	**0.270±7**	6,476	3.9	243	20.7	0.143555
(max_backoff	2 8	572±0	**0.735±8**	18,196	8.9	575	77.8	0.00525932
collisions)	4 4	1082±0	**2.418±34**	34,826	20.5	303	1443.7	0.0769043
[max]	4 8	2315±0	**7.241±38**	239,298	431.4	> 1 hour		1.65363e-5
csma_abst	1k	254±0	**0.323±19**	6,392	1.9	366	68.2	0.0
(deadline)	2k	437±0	**0.627±24**	24,173	20.7	722	367.8	0.869791
[min]	3k	1178±0	**3.445±49**	79,608	448.0	1,736	1436.3	0.999820099
firewire_abst	5k	64±0	**0.024±5**	205	0.25	63	2.45	0.78125
(deadline)	10k	181±0	**0.081±6**	1,023	1.76	180	3.8	0.9747314
[min]	20k	641±0	**0.587±8**	9,059	26.1	640	26.4	0.999629555
nrp_malicious	5	123±2	**0.141±9**	1,663	1.5	75	2.9	0.100072
(deadline)	10	293±2	**0.711±14**	8,080	11.1	408	117.3	0.105447
[max]	20	632±2	**3.135±73**	49,622	218.1	1,108	1606.5	0.105658

REFERENCES

[1] R. Kurshan, "Verification technology transfer," in *25 Years of Model Checking*, ser. LNCS, O. Grumberg and H. Veith, Eds., vol. 5000. Springer, 2008, pp. 46–64.

[2] G. Behrmann, A. David, and K. Larsen, "A tutorial on uppaal." in *SFM*, ser. LNCS, M. Bernardo and F. Corradini, Eds., vol. 3185. Berlin: Springer, 2004, pp. 200–236.

[3] G. Behrmann, K. Larsen, and J. Rasmussen, "Optimal scheduling using priced timed automata," *SIGMETRICS Performance Evaluation Review*, vol. 32, no. 4, pp. 34–40, 2005.

[4] M. Kwiatkowska, G. Norman, and D. Parker, "Prism: Probabilistic model checking for performance and reliability analysis," *ACM SIGMETRICS Performance Evaluation Review*, vol. 36, no. 4, pp. 40–45, 2009.

[5] J. Berendsen, B. Gebremichael, F. Vaandrager, and M. Zhang, "Formal specification and analysis of zeroconf using Uppaal," *ACM Trans. Embedded Comput. Syst.*, 2010, to appear. [Online]. Available: http://www.mbsd.cs.ru.nl/publications/papers/fvaan/zeroconf/full.html

[6] H. Bohnenkamp, P. van der Stok, H. Hermanns, and F. Vaandrager, "Cost-optimization of the ipv4 zeroconf protocol," in *DSN*. IEEE Computer Society, 2003, pp. 531–540.

[7] J. Berendsen, D. Jansen, and J.-P. Katoen, "Probably on time and within budget: On reachability in priced probabilistic timed automata," in *QEST*. IEEE Computer Society, 2006, pp. 311–322.

[8] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston, "Performance analysis of probabilistic timed automata using digital clocks," *Formal Methods in System Design*, vol. 29, no. 1, pp. 33–78, 2006.

[9] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, "Automatic verification of real-time systems with discrete probability distributions," *Theoretical Computer Science*, vol. 282, no. 1, pp. 101–150, 2002.

[10] G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager, "Minimum-cost reachability for priced timed automata," in *Proceedings* 4^{th} *International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, ser. LNCS, M. Di Benedetto and A. Sangiovanni-Vincentelli, Eds., vol. 2034. Springer-Verlag, Mar. 2001, pp. 147–161.

[11] R. Alur, S. La Torre, and G. Pappas, "Optimal paths in weighted timed automata," in *Proceedings* 4^{th} *International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, ser. LNCS, M. Di Benedetto and A. Sangiovanni-Vincentelli, Eds., vol. 2034. Springer-Verlag, Mar. 2001.

[12] M. Kwiatkowska, G. Norman, and D. Parker, "Stochastic games for verification of probabilistic timed automata," in *FORMATS*, ser. LNCS, J. Ouaknine and F. Vaandrager, Eds., vol. 5813. Springer, 2009, pp. 212–227.

[13] M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang, "Symbolic model checking for probabilistic timed automata," *Information and Computation*, vol. 205, no. 7, pp. 1027–1077, 2007.

[14] A. Hartmanns and H. Hermanns, "A modest approach to checking probabilistic timed automata," in *QEST*. IEEE Computer Society Press, 2009, pp. 187–196.

[15] J. Berendsen, T. Chen, and D. Jansen, "Undecidability of cost-bounded reachability in priced probabilistic timed automata," in *TAMC*, ser. LNCS, J. Chen and S. Cooper, Eds., vol. 5532. Springer, 2009, pp. 128–137.

[16] J. Berendsen, D. Jansen, and F. Vaandrager, "Fortuna: Model checking priced probabilistic timed automata," Institute for Computing and Information Sciences, Radboud University Nijmegen, Report, 2009. [Online]. Available: http://www.cs.ru.nl/J.Berendsen/fortuna/

[17] M. Puterman, *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley, 2005.

[18] R. Bagnara, P. M. Hill, and E. Zaffanella, "The Parma Polyhedra Library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems," *Science of Computer Programming*, vol. 72, no. 1–2, pp. 3–21, 2008. [Online]. Available: http://www.cs.unipr.it/ppl/Documentation/BagnaraHZ08SCP.pdf

[19] J. Sproston, "Decidable model checking of probabilistic hybrid automata," in *Formal Techniques in Real-Time and Fault-tolerant Systems (FTRTFT)*, ser. LNCS, M. Joseph, Ed., vol. 1926. Berlin: Springer, 2000, pp. 31–45.

[20] A. Mader, H. Bohnenkamp, Y. Usenko, D. Jansen, J. Hurink, and H. Hermanns, "Synthesis and stochastic assessment of cost-optimal schedules," Centre for Telematics and Information Technology, University of Twente, Enschede, Tech. Rep. TR-CTIT-06-14, 2006, http://eprints.eemcs.utwente.nl/2694/.

[21] A. Haugstad, "Uppaal pro," http://www.cs.aau.dk/~arild/uppaal-probabilistic/.

[22] M. Di Benedetto and A. Sangiovanni-Vincentelli, Eds., *Proceedings 4^{th} International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, ser. LNCS, vol. 2034. Springer-Verlag, Mar. 2001.