Model-Checking One-Clock Priced Timed Automata

Patricia Bouyer[1⋆], Kim G. Larsen[2⋆⋆], and Nicolas Markey[1⋆]

[1] LSV, CNRS & ENS de Cachan, France
{bouyer,markey}@lsv.ens-cachan.fr
[2] Aalborg University, Denmark
kgl@cs.aau.dk

Abstract. We consider the model of priced (a.k.a. weighted) timed automata, an extension of timed automata with cost information on both locations and transitions. We prove that model-checking this class of models against the logic WCTL, CTL with cost-constrained modalities, is PSPACE-complete under the "single-clock" assumption. In contrast, it has been recently proved that the model-checking problem is undecidable for this model as soon as the system has three clocks. We also prove that the model-checking of WCTL* becomes undecidable, even under this "single-clock" assumption.

1 Introduction

An interesting direction of real-time model-checking that has recently received substantial attention is the extension and re-targeting of timed automata technology towards optimal scheduling and controller synthesis [1, 18, 7]. In particular, as part of this effort, the notion of priced (or weighted) timed automata [4, 3] has been promoted as a useful extension of the classical model of timed automata allowing continuous consumption of resources (e.g. energy) to be modelled and analyzed.

A number of optimization problems have been shown decidable for priced timed automata including minimum-cost reachability [4, 3], optimal (minimum and maximum cost) reachability in multi-priced settings [17] and cost-optimal infinite schedules [6, 7].

Unfortunately, the addition of cost comes with a price: certain problems become undecidable for priced timed automata. In fact, in [11] it has recently been shown that the problem of determining cost-optimal winning strategies for priced timed games is not computable. Also, by the same authors, it has been shown that the model-checking problem for priced timed automata w.r.t. WCTL —CTL with cost-constrained modalities— is undecidable [10]. In [5] it has been shown that these negative results hold even for priced timed (game) automata with no more than three clocks.

⋆ Partly supported by ACI "Sécurité & Informatique" CORTOS.
⋆⋆ Partly supported by an invited professorship from ENS Cachan.

However, when restricting to the setting of priced timed game automata with a single clock, the most recent work in [9] shows that the optimal cost of winning and (almost-) optimal strategies are computable problems. In this paper we focus on model-checking problems for priced timed automata with a single clock. In particular we show that the model-checking problem with respect to WCTL is PSPACE-complete under the "single clock" assumption. This is rather surprising as model-checking TCTL (the only cost variable is the time elapsed) under the same assumption is also PSPACE-complete [15]. We also prove that the model-checking of WCTL* becomes undecidable, even under this "single clock" assumption.

The paper is organized as follows: In Section 2, we present the model of priced timed automata, the logic WCTL and develop an example. In Section 3, we state the main result of the paper. In Section 4, we study the granularity which is required for model-checking the logic WCTL. In Section 5, we first propose an EXPTIME algorithm for model-checking one-clock priced timed automata against WCTL formulas, then refine it to get a PSPACE algorithm, and finally give an example. In Section 6, we prove that model-checking one-clock priced timed automata against WCTL* formulas is undecidable.

2 Preliminaries

2.1 Priced Timed Automata

Let $\mathcal{X}$ be a set of clock variables. The set of clock constraints (or guards) over $\mathcal{X}$ is defined by the grammar "$g ::= x \sim c \mid g \wedge g$" where $x \in \mathcal{X}$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$. The set of all clock constraints is denoted $\mathcal{B}(\mathcal{X})$. When a valuation $v : \mathcal{X} \to \mathbb{R}_+$ satisfies a clock constraint g is defined in a natural way (v satisfies $x \sim c$ whenever $v(x) \sim c$), and we then write $v \models g$. We denote by v_0 the valuation that assigns zero to all clock variables, by $v+t$ ($t \in \mathbb{R}_+$) the valuation that assigns $v(x)+t$ to all $x \in \mathcal{X}$, and for $R \subseteq \mathcal{X}$ we write $v[R \to 0]$ to denote the valuation that assigns zero to all variables in R and agrees with v for all $\mathcal{X} \setminus R$.

Definition 1. *A* priced timed automaton *(PTA for short) is a tuple* $\mathcal{A} = (Q, q_0, \mathcal{X}, T, \eta, (P_i)_{1 \leq i \leq p})$ *where* Q *is a finite set of* locations*,* $q_0 \in Q$ *is the* initial location*,* $\mathcal{X}$ *is a set of* clocks*,* $T \subseteq Q \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times Q$ *is the set of* transitions*,* $\eta \colon Q \to \mathcal{B}(\mathcal{X})$ *defines the* invariants *of each location, and* $P_i \colon Q \cup T \to \mathbb{N}$ *is a* cost (or price) function*.*

The semantics of a PTA $\mathcal{A}$ is given as a labeled timed transition system $\mathcal{T} = (S, s_0, \to)$ where $S \subseteq Q \times \mathbb{R}_+^{\mathcal{X}}$ is the set of states, $s_0 = (q_0, v_0)$[3] is the initial state, and the transition relation $\to \subseteq S \times \mathbb{R}_+^p \times S$ is defined as:

1. *(discrete transition)* $(q, v) \xrightarrow{c} (q', v')$ if there exists $(q, g, R, q') \in E$ s.t. $v \models g$, $v' = [R \leftarrow 0]v$, $v' \models \eta(q')$, and $c_i = P_i(q, g, R, q')$ for every $1 \leq i \leq p$;

[3] v_0 assigns zero to each clock.

2. *(delay transition)* $(q,v) \xrightarrow{c} (q, v+t)$ if $\forall 0 \le t' \le t$, $v+t' \models \eta(q)$, and $c_i = t \cdot P_i(q)$ for every $1 \le i \le p$.

A *run* of a PTA is a path in the underlying transition system. Given a run $\varrho = s_0 \xrightarrow{c^0} s_1 \xrightarrow{c^1} \cdots \xrightarrow{c^{n-1}} s_n$, its ith-cost is $P_i(\varrho) = \sum_{j=0}^{n-1} c_i^j$. A *position* along a run ϱ is an occurrence of a state (q,v) along ϱ. Let π be such a position, then $\varrho[\pi]$ denotes the corresponding state, whereas $\varrho_{\le\pi}$ denotes the finite prefix of ϱ ending at position π.

Remark 1. In the model of priced timed automata, the cost variables are *observers*: the values of these variables don't constrain the behaviour of the system (the behaviours of a priced timed automaton are those of the underlying timed automaton), but can be used as evaluation functions. For instance, problems such as "optimal reachability" [4, 3], "optimal infinite schedules" [6] or "optimal reachability timed games" [2, 8, 11, 5] have recently been investigated. The problem we consider in this paper is closely related to these kinds of problems: we will use WCTL as a language for evaluating the performances of a system.

2.2 The Logic WCTL

Let AP be a set of atomic propositions. The logic WCTL[4] [10] extends CTL with cost constraints. Its syntax is given by the following grammar:

$$\text{WCTL} \ni \phi \ ::= \ \texttt{true} \mid a \mid \neg\phi \mid \phi \vee \phi \mid \mathsf{E}\,\phi \mathsf{U}_{P\sim c}\phi \mid \mathsf{A}\,\phi \mathsf{U}_{P\sim c}\phi$$

where $a \in \mathsf{AP}$, P is a cost function, c ranges over $\mathbb{N}$, and $\sim \in \{<, \le, =, \ge, >\}$.

We interpret formulas of WCTL over labeled PTA, *i.e.* PTA having a labeling function ℓ which associates with every location q a subset of AP.

Definition 2. *Let $\mathcal{A}$ be a labeled PTA. The satisfaction relation of WCTL is defined over configurations (q,v) of $\mathcal{A}$ as follows:*

$$\begin{array}{rcl}
(q,v) \models \texttt{true} & & \\
(q,v) \models p & \Leftrightarrow & a \in \ell(q) \\
(q,v) \models \neg\phi & \Leftrightarrow & (q,v) \not\models \phi \\
(q,v) \models \phi_1 \vee \phi_2 & \Leftrightarrow & (q,v) \models \phi_1 \textit{ or } (q,v) \models \phi_2 \\
(q,v) \models \mathsf{E}\,\phi_1 \mathsf{U}_{P\sim c}\phi_2 & \Leftrightarrow & \textit{there is an infinite run } \varrho \textit{ in } \mathcal{A} \\
 & & \textit{from } (q,v) \textit{ s.t. } \varrho \models \phi_1 \mathsf{U}_{P\sim c}\phi_2 \\
(q,v) \models \mathsf{A}\,\phi_1 \mathsf{U}_{P\sim c}\phi_2 & \Leftrightarrow & \textit{any infinite run } \varrho \textit{ in } \mathcal{A} \textit{ from } (q,v) \\
 & & \textit{satisfies } \varrho \models \phi_1 \mathsf{U}_{P\sim c}\phi_2 \\
\varrho \models \phi_1 \mathsf{U}_{P\sim c}\phi_2 & \Leftrightarrow & \textit{there exists } \pi > 0 \textit{ position along } \varrho \textit{ s.t.} \\
 & & \varrho[\pi] \models \phi_2, \textit{ for all position } \pi' > 0 \\
 & & \textit{before } \pi \textit{ on } \varrho,\ \varrho[\pi'] \models \phi_1, \\
 & & \textit{and } P(\varrho_{\le\pi}) \sim c
\end{array}$$

[4] WCTL stands for "Weighted CTL", following [10] terminology. It would have been more natural to call it "Priced CTL" (PCTL) in our setting, but this would have been confusing with "Probabilistic CTL" [13].

If $\mathcal{A}$ is not clear from the context, we may write $(q, v), \mathcal{A} \models \phi$ instead of simply $(q, v) \models \phi$.

As usual, we will use shortcuts as $\mathsf{E}\,\mathsf{F}_{P\sim c}\phi \equiv \mathsf{E}\,\mathtt{true}\,\mathsf{U}_{P\sim c}\phi$, or $\mathsf{A}\,\mathsf{G}_{P\sim c}\phi \equiv \neg\mathsf{E}\,\mathsf{F}_{P\sim c}\neg\phi$. Moreover, if the cost function P is unique or clear from the context, we may write $\phi\mathsf{U}_{\sim c}\psi$ instead of $\phi\mathsf{U}_{P\sim c}\psi$.

We write WCTL* for the extension of WCTL similar to the extension CTL* of CTL [12]: temporal modality $\mathsf{U}_{\sim c}$ can then be nested independently of path quantifiers.

2.3 Example

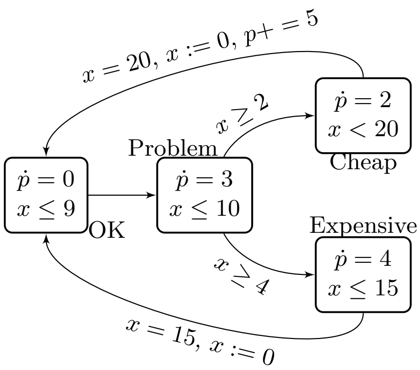

Fig. 1. Repair problem as a PTA

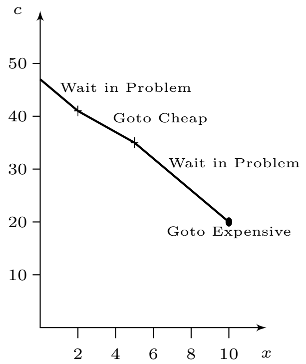

Fig. 2. Minimum cost of repair and associated strategy in location $\mathsf{Problem}$

The $\mathsf{1PTA}$ of Fig. 1 models a never-ending process of repairing problems, which are bound to occur repeatedly with a certain frequency. The repair of a problem has a certain cost, captured in the model by the cost variable c. As soon as a problem occurs (modeled by the $\mathsf{Problem}$ location) the value of c grows with rate 3, until actual repair is taking place in one of the locations Cheap (rate 2) or $\mathsf{Expensive}$ (rate 4). At most 20 time units after the occurrence of a problem it will have been repaired one way or another. In this setting we are interested in properties concerning the cost of repairs as stated by the following WCTL formulas (all satisfied by the model):

$$\begin{array}{l}\mathsf{A}\,\mathsf{G}\big(\mathsf{Problem} \implies \mathsf{E}\,\mathsf{F}_{c\leq 47}\mathsf{OK}\big)\\ \mathsf{A}\,\mathsf{G}\big(\mathsf{Problem} \implies \mathsf{A}\,\mathsf{F}_{c\leq 56}\mathsf{OK}\big)\\ \mathsf{A}\,\mathsf{G}\big(\neg\mathsf{E}\,(\mathsf{OK}\ \mathsf{U}_{t\geq 8}(\mathsf{Problem} \wedge \neg\mathsf{E}\,\mathsf{F}_{c<30}\mathsf{OK}))\big)\end{array}$$

where t holds for the time elapsed (special cost variable with rate 1).

Here the first property claims that whenever a problem occurs it may be repaired (*i.e.* reach the location OK) within a total cost of 47. In fact Fig. 2 gives the minimum cost of repair —as well as an optimal strategy— for any state of the form $(\mathsf{Problem}, x)$ with $x \in [0, 10]$. Correspondingly, the minimum cost of reaching OK from states of the form (Cheap, x) (resp. (Expensive, x)) is given by the expression $45 - 2x$ (resp. $60 - 4x$). The second property states that no matter which method is used for the repair, it will cost no more than 56. Finally, the third property claims that whenever the system has been OK for at least 8 time units before a problem occurs, then there must be a way of solving the problem with a total cost less than 30. In fact, as indicated in Fig. 2, any state (Problem, x) with $x \geq \frac{20}{3}$ satisfies the WCTL property $\mathsf{E}\,\mathsf{F}_{c\leq 30}\mathsf{OK}$.

3 Main Result

We focus on one-clock priced timed automata (1PTA for short), *i.e.* priced timed automata where $|\mathcal{X}| = 1$. The main result of this paper is the following theorem:

Theorem 3. *Model-checking WCTL on 1PTA is* PSPACE*-complete.*

The PSPACE lower bound is a consequence of the PSPACE-hardness of the model-checking of TCTL, the restriction of WCTL to time constraints, over 1PTA [15].

The PSPACE upper bound is rather involved, and will be done in two steps: *i*) first we will exhibit a set of regions which will be correct for model-checking WCTL formulas, see Section 4; *ii*) then we will use this result to propose a PSPACE algorithm for model-checking WCTL, see Section 5.

Finally, it is worth reminding here that the model-checking of WCTL over priced timed automata with three clocks is undecidable [5].

4 Sufficient Granularity for Model-Checking WCTL

The proof of Theorem 3 is rather involved and partly relies on the following proposition, which exhibits a set of *regions* on which truth of WCTL formulas is uniform.

Proposition 4. *Let Φ be a WCTL formula and let $\mathcal{A}$ be a 1PTA. Then there exist finitely many constants $0 = a_0 < a_1 < \ldots < a_n < a_{n+1} = +\infty$ s.t. for every location q of $\mathcal{A}$, for every $0 \leq i \leq n$, the truth of Φ is uniform over $\{(q, x) \mid a_i < x < a_{i+1}\}$. Moreover,*

- *$\{a_0, ..., a_n\}$ contains all the constants appearing in clock constraints of $\mathcal{A}$;*
- *the constants are integral multiples of $1/C^{\hbar(\Phi)}$ where $\hbar(\Phi)$ is the* constrained temporal height *of Φ, i.e. the maximal number of nested constrained modalities in Φ, and C is the lcm of all positive costs labeling a location of $\mathcal{A}$;*
- *a_n equals the largest constant M appearing in the guards of $\mathcal{A}$;*
- *$n \leq M \cdot C^{\hbar(\Phi)} + 1$.*

As a corollary, we recover the partial decidability result of [10], stating that the model-checking of 1PTA with a *stopwatch cost*[5] against WCTL formulas is decidable using classical one-dimensional regions of timed automata (*i.e.* with granularity 1).

[5] *I.e.* cost with rates in $\{0, 1\}$.

Proof. The proof of this proposition is by structural induction on Φ. We focus on the case when $\Phi = \mathsf{E}\,\phi \mathsf{U}_{P\sim c}\psi$ (we will simply write $\Phi = \mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$): the cases of atomic propositions, boolean combinations are straightforward, unconstrained modalities require no refinement of the granularity (a basic CTL algorithm handles this case), and the other modalities will be reduced to this main case.

Assume that the result has been proved for WCTL subformulas ϕ and ψ, and that we have merged all constants for ϕ and ψ: we thus have constants $0 = a_0 < a_1 < \ldots < a_n < a_{n+1} = +\infty$ such that for every location q of $\mathcal{A}$, for every $0 \leq i \leq n$, the truth of ϕ and that of ψ are both uniform over $\{(q,x) \mid a_i < x < a_{i+1}\}$. The granularity of these constants is $1/C^{\max(\hbar(\phi),\hbar(\psi))} = 1/C^{\hbar(\Phi)-1}$. We will exhibit extra constants such that the above proposition then also holds for formula $\Phi = \mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$. For the sake of simplicity, we will call *regions* all elementary intervals (a_i, a_{i+1}) and singletons $\{a_i\}$. We also assume that $\mathcal{A}$ has no discrete costs (*i.e.* $P(T) = \{0\}$). The general case would be handled in a similar way, and will be developed in the long version of this paper.

In order to compute the set of states satisfying $\mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$, we compute for every state (q,x) all costs of paths from (q,x) to some region (q',r), along which ϕ continuously holds, and such that a ψ-state can be reached immediately from (q',r). We then check whether we can achieve a cost satisfying "$\sim c$". We thus explain how we compute the set of possible costs between a state (q,x) and a region (q',r) in $\mathcal{A}$.

For each index i, we restrict the automaton $\mathcal{A}$ to transitions whose guards contain the interval (a_i, a_{i+1}), and that do not reset the clock. We denote by $\mathcal{A}_i$ this restricted automaton. Let q and q' be two locations of $\mathcal{A}_i$. As stated by the following lemma, the set of costs of paths between (q,a_i) and (q',a_{i+1}) is an interval that can be easily computed:

Lemma 5. *Let $S_i(q,q')$ be the set of locations that are reachable from (q,a_i) and co-reachable from (q',a_{i+1}) in $\mathcal{A}_i$ (assuming $a_{i+1} \neq +\infty$), and assume it is non-empty. Let $c^{i,q,q'}_{\min}$ and $c^{i,q,q'}_{\max}$ be the minimum and maximum costs among the costs of locations in $S_i(q,q')$. Then the set of all possible costs of paths going from (q,a_i) to (q',a_{i+1}) in $\mathcal{A}_i$ is an interval $\langle (a_{i+1}-a_i)\cdot c^{i,q,q'}_{\min} ; (a_{i+1}-a_i)\cdot c^{i,q,q'}_{\max} \rangle$. The interval is left-closed iff there exist two locations r and s (with possibly $r = s$) in $S_i(q,q')$ with cost $c^{i,q,q'}_{\min}$ such that*[6] *$(q,a_i) \rightsquigarrow^*_{\mathcal{A}_i} (r,a_i)$, $(r,a_i) \rightsquigarrow^*_{\mathcal{A}_i} (s,a_{i+1})$, and $(s,a_{i+1}) \rightsquigarrow^*_{\mathcal{A}_i} (q',a_{i+1})$. The interval is right-closed iff there exists two locations r and s in $S_i(q,q')$ with cost $c^{i,q,q'}_{\max}$ such that $(q,a_i) \rightsquigarrow^*_{\mathcal{A}_i} (r,a_i)$, $(r,a_i) \rightsquigarrow^*_{\mathcal{A}_i} (s,a_{i+1})$, and $(s,a_{i+1}) \rightsquigarrow^*_{\mathcal{A}_i} (q',a_{i+1})$.*

The conditions on left/right-closures characterize the fact that it is possible to instantaneously reach/leave a location with minimal/maximal cost, or if a small positive delay has to be waited (due to a strict guard).

Proof. Obviously the costs of all paths in $\mathcal{A}_i$ belong to the interval $(a_{i+1}-a_i) \cdot [c^{i,q,q'}_{\min}, c^{i,q,q'}_{\max}]$. We will now prove that the set of costs is an interval containing $(a_{i+1}-a_i) \cdot (c^{i,q,q'}_{\min} ; c^{i,q,q'}_{\max})$.

[6] The notation $\alpha \rightsquigarrow^*_{\mathcal{A}_i} \alpha'$ means that there is a path in $\mathcal{A}_i$ from α to α'.

Let $\tau_{\min}$ (resp. $\tau_{\max}$) be a sequence of transitions in $\mathcal{A}_i$ leading from (q, a_i) to (q', a_{i+1}) and going through a location with minimal (resp. maximal) cost. Easily enough, the possible costs of the paths following $\tau_{\min}$ (resp. $\tau_{\max}$) form an interval whose left (resp. right) bound is $c_{\min}^{i,q,q'} \cdot (a_{i+1} - a_i)$ (resp. $c_{\max}^{i,q,q'} \cdot (a_{i+1} - a_i)$).

Now, if c and c' are the respective costs of q and q', then $\frac{1}{2} \cdot (c + c') \cdot (a_{i+1} - a_i)$ is in both intervals. Indeed, the path following $\tau_{\min}$ (resp. $\tau_{\max}$) which delays $\frac{1}{2} \cdot (a_{i+1} - a_i)$ time units in q, then directly goes to q' and waits there for the remaining $\frac{1}{2} \cdot (a_{i+1} - a_i)$ time units achieves the above-mentioned cost. This implies that the set of all possible costs is an interval.

The bound $c_{\min}^{i,q,q'} \cdot (a_{i+1} - a_i)$ is reached iff there is a path from (q, a_i) to (q', a_{i+1}) which delays only in locations with cost $c_{\min}^{i,q,q'}$. This is precisely the condition expressed in the lemma. The same holds for the upper bound $c_{\max}^{i,q,q'} \cdot (a_{i+1} - a_i)$. $\square$

Similar results clearly hold for other kinds of regions:

- between a state (q, a_i) and a region $(q', (a_i, a_{i+1}))$ with $a_{i+1} \neq +\infty$, the set of possible costs is an interval $\langle 0; c_{\max}^{i,q,q'} \cdot (a_{i+1} - a_i))$, where 0 can be reached iff it is possible to go from (q, a_i) to some state (q'', a_i) with $P(q'') = 0$.
- between a state (q, x), with $x \in (a_1, a_{i+1})$, and (q', a_{i+1}), the set of costs is $(a_{i+1} - x) \cdot \langle c_{\min}^{i,q,q'}; c_{\max}^{i,q,q'} \rangle$, with similar conditions as above for the bounds of the interval.
- between a state (q, x), with $x \in (a_1, a_{i+1})$, and region $(q', (a_i, a_{i+1}))$ (assuming $a_{i+1} \neq +\infty$), the set of possible costs is $[0, c_{\max}^{i,q,q'} \cdot (a_{i+1} - x))$;
- between a state (q, a_n) and a region $(q', (a_n, a_{n+1}))$ (with $a_{n+1} = +\infty$), the set of possible costs is either $[0, 0]$, if no positive cost rate is reachable and co-reachable, or $\langle 0, +\infty)$ otherwise. If the latter case, 0 can be achieved iff it is possible to reach a state (q'', a_n) with $P(q'') = 0$;
- between a state (q, x), with $x \in (a_n, a_{n+1})$ and $a_{n+1} = +\infty$, and a region $(q', (a_n, a_{n+1}))$, the set of costs is either $[0, 0]$ or $[0, +\infty)$, with the same conditions as previously.

We use these computations and build a graph G labeled by intervals which will store all possible costs between symbolic states (*i.e.* pairs (q, r), where q is a location and r a region) in $\mathcal{A}$. Vertices of G are pairs $(q, \{a_i\})$ and $(q, (a_i, a_{i+1}))$, and tuples $(q, x, \{a_i\})$ and $(q, x, (a_i, a_{i+1}))$, where q is a location of $\mathcal{A}$. Their roles are as follows: vertices of the form (q, x, r) are used to initiate a computation, they represent a state (q, x) with $x \in r$. States $(q, \{a_i\})$ are "regular" steps in the computation, while states $(q, (a_i, a_{i+1}))$ are used either for finishing a computation, or just before resetting the clock (there will be no edge from $(q, (a_i, a_{i+1}))$ to any $(q', \{a_{i+1}\})$).

Edges of G are defined as follows:

- $(q, \{a_i\}) \rightarrow (q', \{a_{i+1}\})$ if there is a path from (q, a_i) to (q', a_{i+1}). This edge is then labeled with an interval $\langle (a_{i+1} - a_i) \cdot c_{\min}^{i,q,q'}; (a_{i+1} - a_i) \cdot c_{\max}^{i,q,q'} \rangle$, the nature of the interval (left-closed and/or right-closed) depending on the criteria exposed in Lemma 5.

– $(q, \{a_i\}) \to (q', \{a_i\})$ if there is an instantaneous path from (q, a_i) to (q', a_i) in $\mathcal{A}$, the edge is then labeled with the interval $[0, 0]$ (remember that we assumed there are no discrete costs on transitions of $\mathcal{A}$).
– $(q, \{a_i\}) \to (q', \{a_0\})$ if there is a transition in $\mathcal{A}$ enabled when the value of the clock is a_i and resetting the clock. It is labeled with $[0, 0]$.
– $(q, (a_i, a_{i+1})) \to (q', \{a_0\})$ if there is a transition in $\mathcal{A}$ enabled when the value of the clock is in (a_i, a_{i+1}) and resetting the clock. It is labeled with $[0, 0]$.
– $(q, \{a_i\}) \to (q', (a_i, a_{i+1}))$ if there is a path from (q, a_i) to some (q', α) with $a_i < \alpha < a_{i+1}$. This edge is labeled with the interval $\langle 0; (a_{i+1} - a_i) \cdot c^{i,q,q'}_{\max} \rangle$.
– $(q, x, \{a_i\}) \to (q, \{a_i\})$ labeled with $[0, 0]$.
– $(q, x, (a_i, a_{i+1})) \to (q', \{a_{i+1}\})$ if there is a path from some (q, α) with $a_i < \alpha < a_{i+1}$ to (q', a_{i+1}). This edge is labeled with $(a_{i+1} - x) \cdot \langle c^{i,q,q'}_{\min}; c^{i,q,q'}_{\max} \rangle$.
– $(q, x, (a_i, a_{i+1})) \to (q', (a_i, a_{i+1}))$ labeled with $[0, (a_{i+1} - x) \cdot c^{i,q,q'}_{\max} \rangle$.

Figure 3 represents one part of this graph. Note that each path π of this graph is naturally associated with an interval $\iota(\pi)$ (possibly depending on variable x if we start from a node $(q, x, (a_i, a_{i+1})))$ by summing up all intervals labeling transitions of π.

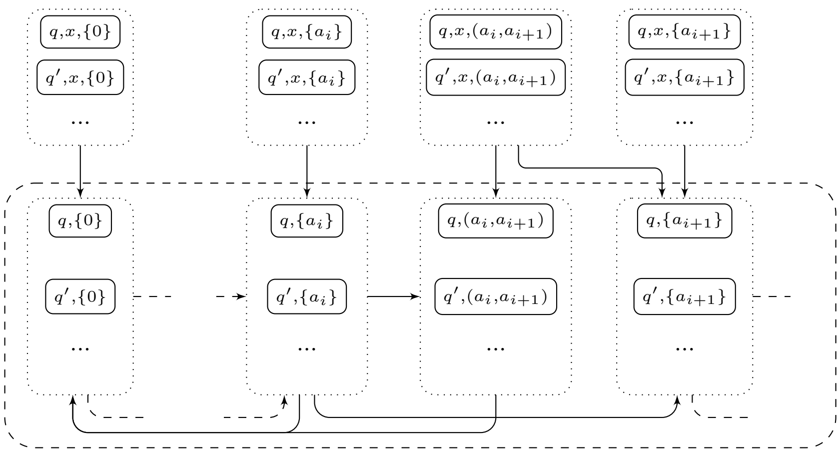

Fig. 3. (Schematic) representation of the graph G (intervals omitted)

The correctness of graph G w.r.t. costs is stated by the following lemma, which is a direct consequence of the previous investigations.

Lemma 6. *Let q and q' be two locations of $\mathcal{A}$. Let r and r' be two regions, and let $\alpha \in r$. Let $d \in \mathbb{R}^+$. There exists a path π in G from a state (q, x, r) to (q', r') with cost $d \in \iota(\pi)(\alpha)$ if, and only if, there is a path in $\mathcal{A}$ with total cost d, and going from (q, α) to some (q', β) with $\beta \in r'$.*

Corollary 7. *Fix two regions r and r'. Then the set of possible costs of paths in G from (q, x, r) to (q', r') is of the form*

$$\bigcup_{m\in\mathbb{N}} \langle \alpha_m - \beta_m \cdot x; \alpha'_m - \beta'_m \cdot x\rangle$$

(possibly with β_m and/or $\beta'_m = 0$, and/or $\alpha'_m = +\infty$). Moreover,

- *all constants α_m and α'_m are either integral multiples of $1/C^{\max(\hbar(\phi),\hbar(\psi))}$ or $+\infty$, and constants β_m and β'_m are either costs of the automaton or 0;*
- *if $r = (a_n, +\infty)$, then $\beta_m = \beta'_m = 0$ for all m.*

Proof (Sketch). The set of possible costs can be computed by guessing the Parikh image of a possible path. Then the set of possible costs along that path has the form given in the statement. And as the set of possible Parikh images is countable, we obtain the (possibly infinite) union of intervals of the corollary. □

Lemma 8. *For every location q, the set of clock values x such that (q, x) satisfies $\mathsf{E}\,\phi\mathsf{U}_{\sim c}\psi$ is a finite union of intervals. Moreover,*

- *the bounds of those intervals are integral multiples of $1/C^{\hbar(\Phi)}$;*
- *the largest finite bound of those intervals is at most the maximal constant appearing in the guards of the automaton.*

Proof (Sketch). It is possible to prove that the (possibly infinite) union of intervals of the previous corollary can be reduced, for checking formula $\mathsf{E}\,\phi\mathsf{U}_{\sim c}\psi$, to a finite union of such intervals.

Then, new constants α we need to consider for checking $\mathsf{E}\,\phi\mathsf{U}_{\sim c}\psi$ are such that $\alpha_m - \beta_m \cdot \alpha = c$, *i.e.* $\alpha = (\alpha_m - c)/\beta_m$. Thus α is an integral multiple of $1/C^{\hbar(\Phi)}$. □

This concludes the induction step for formula $\mathsf{E}\,\phi\mathsf{U}_{\sim c}\psi$ when the automaton has no discrete cost. Extending this result to other modalities and to automata with discrete cost is a rather technical matter that gives no new insights on the model-checking problem; we thus postpone the proofs of these two extensions to the full version of this paper. ■

Remark 2. The exponential number of constants a_i's is unavoidable in general. Indeed, consider the 1PTA $\mathcal{A}$ displayed on Fig.4. Using a WCTL formula, we will require that the cost is exactly 4 between a and b. That way, if clock x equals $x_0.x_1x_2x_3\dots x_n\dots$ (this is the binary representation of a real in the interval $(0, 2)$) when leaving a, then it will be equal to $x_1.x_2x_3\dots x_n\dots$ in b. We consider the WCTL formula $\phi(X) = \mathsf{E}\Big((a \vee b)\mathsf{U}_{=0}(\neg a \wedge \mathsf{E}\,(\neg b\mathsf{U}_{=4}(b \wedge X)))\Big)$, where X is a formula we will specify. Then formula $\phi(\mathsf{E}\,\mathsf{F}_{=0}c)$ states that we can go from a to b with cost 4, and that $x = 0$ when arriving in b (since we can fire the transition leading to c). From the remark above, this can only be true if $x = 0$ or $x = 1$ in a. Now, consider formula $\phi(\mathsf{E}\,\mathsf{F}_{=0}c \vee \phi(\mathsf{E}\,\mathsf{F}_{=0}c))$. If

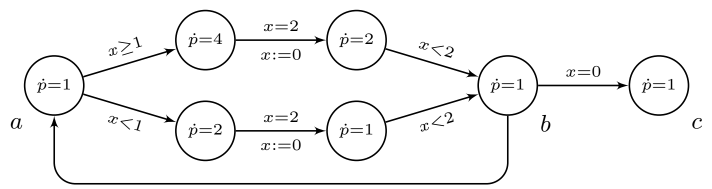

Fig. 4. The 1PTA $\mathcal{A}$

it holds in state a, then state c can be reached after exactly one or two rounds in the automaton, *i.e.*, if the value of x is in $\{0, 1/2, 1, 3/2\}$. Clearly enough, nesting ϕ n times characterizes values of the clocks of the form $p/2^{n-1}$ where p is an integer strictly less than 2^n.

5 Algorithms and Complexity

In this section, we provide two algorithms for model-checking WCTL on 1PTA. The first algorithm runs in EXPTIME, whereas the second one runs in PSPACE, thus matching the PSPACE lower bound. However, it is easier to first explain the first algorithm, and then reuse part of it in the second algorithm. Finally, we will pursue the example of Subsection 2.3 for illustrating our PSPACE algorithm.

5.1 An EXPTIME Algorithm

The correctness of the algorithm we propose for model-checking 1PTA against WCTL properties relies on the properties we have proved in the previous section: if $\mathcal{A}$ is an automaton with maximal constant M, writing C for the l.c.m. of all costs labeling a location, and if Φ is a WCTL formula of size n, then the satisfaction of Φ is uniform on the regions $(m/C^n; (m+1)/C^n)$ with $m < M \cdot C^n$, and also on $(M; +\infty)$. The idea is thus to test the satisfaction of Φ for each state of the form $(q, k/2C^n)$ for $0 \leq k \leq (M \cdot 2C^n) + 1$ (*i.e.* at the bounds and in the middle of each region).

To check the truth of $\Phi = \mathsf{E}\,\phi \mathsf{U}_{P\sim c}\psi$ in state (q, x) with $x = k/2C^n$, we will use the graph G that we have defined in Section 4. From the state (q, x, r) of G, where r is the region containing $k/2C^n$, we check if $\mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$ (say) holds by non-deterministically discovering a witness. This requires the following lemma:

Lemma 9. *Let s be the smallest positive cost in $\mathcal{A}$, and C be the lcm of all positive costs of $\mathcal{A}$. Let q be a location of $\mathcal{A}$, and $x \in \mathbb{R}^+$. Let $\Phi = \mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$ be a WCTL formula of size n. Then $(q, x) \models \Phi$ iff there exists a trajectory in $\mathcal{A}$, from (q, x) and satisfying $\phi \mathsf{U}_{\sim c}\psi$, and whose projection in G visits at most $N = \lfloor c \cdot C^n/s \rfloor + 2$ times each state of G.*

Proof (Sketch). Let τ be a trajectory in $\mathcal{A}$, starting from (q, x) and satisfying $\phi \mathsf{U}_{\sim c}\psi$. To that trajectory corresponds a trajectory ρ in G, starting in (q, x, r). Consider a cycle in that trajectory ρ: either it has a global cost interval $[0, 0]$, in which case it can be removed and still yields a witnessing trajectory; or it has a global cost interval of the form $\langle a, b\rangle$ with $b > 0$. In that case, letting s be the smallest positive cost of the automaton, we know that $b \geq s/C^n$. Now, if some state of G is visited (strictly) more than $N = \lfloor c \cdot C^n/s\rfloor + 2$ times along ρ, we build a trajectory ρ' from ρ by removing extraneous cycles, in such a way that each state of G is visited at most N times along ρ (and that ρ starts and ends in the same states). Since we assumed that ρ does not contain cycles with cost interval $[0;0]$, we know that the upper bound of the accumulated cost along ρ' is above c. Also, the lower bound of the accumulated costs along ρ' is less than that of ρ. Since ρ "contains" a trajectory witnessing $\phi \mathsf{U}_{\sim c}\psi$, the cost interval of ρ contains a value satisfying $\sim c$, thus so does the cost interval of ρ'. In other words, ρ' still contains a trajectory witnessing $\phi \mathsf{U}_{\sim c}\psi$. □

We now describe our algorithm: assuming we have computed, for each state q of $\mathcal{A}$, the intervals of values of x where ϕ (resp. ψ) holds, we non-deterministically guess the successive states of a trajectory in G. At each step, we also have to guess the intermediary states that are visited (between $(q, \{a_i\})$ and $(q', \{a_{i+1}\})$), and check that they satisfy ϕ when x is in (a_i, a_{i+1}). This verification can be achieved in PSPACE. Moreover, at each step of this algorithm for checking that $(q, x) \models \mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$, we only need to store a polynomial amount of information: the current position in G, the number of steps so far, and the interval of costs accumulated so far. At each point, the algorithm may non-deterministically decide to go to a ψ-state, and will check that the cost constraint is satisfied. In that case, it returns `yes`. Otherwise, when the number of steps reaches $|G| \cdot (\lfloor c \cdot C^n/s\rfloor + 2)$ (which is exponential), the procedure stops and returns `no`.

Thus, our procedure for checking that $(q, x) \models \mathsf{E}\,\phi \mathsf{U}_{\sim c}\psi$ is in PSPACE. Still, since we store all the intervals for each location of the automaton and each subformula, the whole algorithm requires an exponential amount of space, but it runs in exponential time.

The other existential modalities are handled by reducing to the case of $\mathsf{E}\,\mathsf{U}_{\sim c}$, as explained in Section 4. We assume that no universal modality appears in the formula by replacing them with negated existential ones.

5.2 A PSPACE Algorithm

The PSPACE algorithm will reuse some parts of the previous algorithm, but it will improve on space performance by storing only the minimal information required, preferring to spend time on reconstructing model-checking information rather than to spend space on storing it. Our method is thus similar in spirit to the space-efficient, on-the-fly algorithm for TCTL presented in [14].

We will then need, while guessing a witness for $\mathsf{E}\,\phi \mathsf{U}_{P\sim c}\psi$, to check that all intermediary states satisfy formula ϕ. As ϕ might be itself a WCTL formula with several nested modalities, we will fork a new computation of our algorithm on formula ϕ from each intermediary state. The maximal number of threads running simultaneaously is at most the depth of the parsing tree of formula Φ. When a thread is preempted we only need to store a polynomial amount of information in order to be able to resume it. Indeed, it is sufficient to store for each preempted thread a triple (α, K, I) where α is a node a graph G, K is the value of a counter bounded by $|G| \cdot (\lfloor c \cdot C^n/s \rfloor + 2)$ counting the number of steps of the path we are guessing (we know that a witness can be bounded by this constant), and I is an interval corresponding to the accumulated cost along the path being guessed.

The algorithm thus runs as follows: we start by labeling the root of the tree by $\alpha = (q, x, r)$, $K = 0$ and $I = [0; 0]$. Then we guess a path in G starting from (q, x, r), and when a new state (q', r') is added, we increment the value of K, update the value of the interval, as described in the previous section. Then, either we choose to verify that the state satisfies ϕ, or the constraint $P \sim c$ can be satisfied by the new interval and we verify in addition that the new state satisfies ψ. Moreover, we need to prove that all intermediary states (see the EXPTIME algorithm) also satisfy ϕ (it is of course sufficient to check intermediary with clock values of the form $h/2C^n$). All these verifications of ϕ or ψ are done by starting a new thread in the computation, and a new guess of path can start for a subformula of the original one... when all these computations are finished, we can continue guessing the original path for formula Φ, and so on.

The number of nested guesses can be bounded by the depth of the parsing tree of Φ, because when a new thread starts, it starts from a node which is a child of the previous node. Thus, the memory which is needed in this algorithm is the parsing tree of formula Φ with each node labeled by a tuple which can be stored in polynomial space, which leads to a globally PSPACE algorithm.

Example 1. We illustrate our PSPACE algorithm on our initial example, with formula $\Phi = \neg \mathsf{E}\,(\mathsf{OK}\ \mathsf{U}_{t\leq 8}(\mathsf{Problem} \wedge \neg \mathsf{E}\,\mathsf{F}_{c<30}\mathsf{OK}))$. We write $g = 1/C^2$ for the resulting granularity as defined in Prop. 4, and consider a starting state, e.g. $(\mathsf{OK}, x = mg)$.

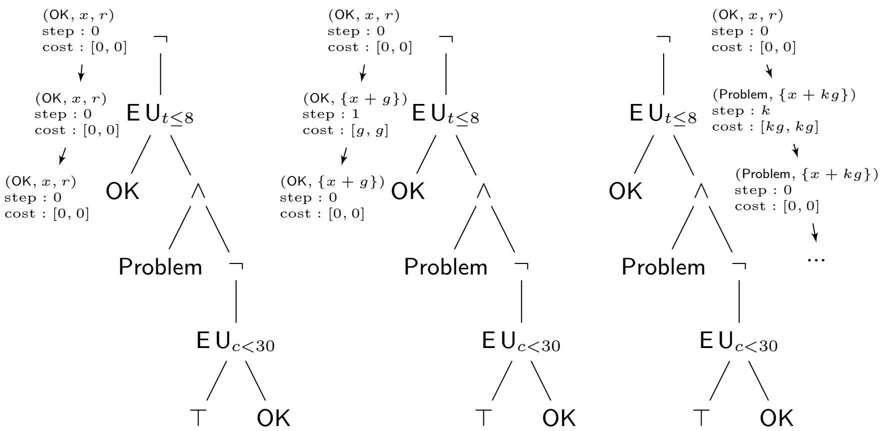

Fig. 5. Execution of our PSPACE algorithm on the initial example.

Fig. 5 show three steps of our algorithm. The first step represents the first iteration, where subformula OK is satisfied at the beginning of the trajectory. At step 2, the execution goes to $(\mathsf{OK}, x+g)$: we check that the left-hand-side formula still holds in $(\mathsf{OK}, x+g)$ (as depicted), but also in intermediary states. The third figure corresponds to k steps later, when the algorithm decides to go to the right-hand-part of $\mathsf{E}\,\mathsf{U}_{t\leq 8}$. In that case, of course, it is checked that $kg \leq 8$, and then goes on verifying the second until subformula.

6 Undecidability of WCTL* Model-Checking

The logic WCTL* is an extension of WCTL that allows nesting of modalities without existential or universal quantifications. We prove that it is undecidable on 1PTAs. To our knowledge, the complexity of TCTL* model-checking has not been studied on one-clock timed automata. However, it is in EXPSPACE on durational Kripke structures, a discrete-time extension of Kripke structures [16].

Theorem 10. *Model-checking WCTL* over 1PTA is undecidable.*

Proof (Sketch). We encode the halting problem for a two-counter machine $\mathcal{M}$ as a model-checking problem for WCTL* over 1PTA. The counters c_1 and c_2 are encoded by clock x being equal to $1/(2^{c_1} \cdot 3^{c_2})$.

We first explain how we encode an instruction incrementing counter c_1, say "$\mathtt{q_j}$: c_1:=c_1+1; goto $\mathtt{q_k}$". Such an instruction is encoded by the automaton displayed on Fig. 6 (where costs are written in locations). We will require that the price between the date at which we enter (or equivalently exit) $\mathtt{q_j}$ and the date at which we enter $\mathtt{q_k}$ is exactly 1. This is enforced by checking the following path formula (with nested until modalities) when entering q_j:

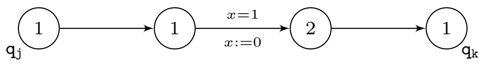

Fig. 6. Incrementing a counter

$$\varphi_{\text{incr1}} = \mathtt{q_j}\,\mathsf{U}_{=0}(\neg\mathtt{q_j} \wedge (\neg\mathtt{q_k}\mathsf{U}_{=1}\mathtt{q_k}))$$

This ensures that clock x has been divided by 2, *i.e.*, that counter c_1 has been incremented. Decrementation can be handled in a similar way by setting the cost of the second (resp. third) location to 2 (resp. 1) and enforcing global cost along that module to be 2. Those operations easily adapt to counter c_2.

Testing if counter c_1 equals 0 reduces to checking that the value of clock x is of the form $1/3^{c_2}$, thus to multiplying clock x by 3 until it possibly equals 1. Consider the following instruction: "$\mathtt{q_k}$: if (c_1==0) goto $\mathtt{q_l}$". We encode this instruction with the automaton of Fig. 7.

Multiplying clock x by 3 is achieved by one pass through the loop with cost exactly 3. Consider the following formula:

$$\varphi_{\text{mult}} = \mathsf{E}\Big(m \Rightarrow \big(m\mathsf{U}_{=0}z \vee m\mathsf{U}_{=0}(\neg m \wedge \neg m\mathsf{U}_{=3}m)\big)\Big)\mathsf{U}z$$

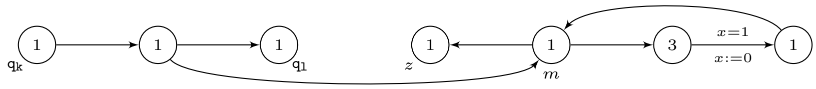

Fig. 7. Testing a counter to 0

It precisely expresses that it is possible to reach z after a finite number of passes through the loop, each pass having total cost 3. This holds iff the original value of clock x when entering the module was of the form $1/3^i$, *i.e.*, iff counter c_1 was equal to 0. Now, from $\mathsf{q_k}$, we simply have to ensure the following property:

$$\varphi_{\text{test1}} = \mathsf{q_k}\mathsf{U}_{=0}\Big(\neg\mathsf{q_k} \wedge \mathsf{E}\left(\neg m\mathsf{U}_{=0}(m \wedge \varphi_{\text{mult}})\right) \wedge \left(\neg\mathsf{q_1}\mathsf{U}_{=0}\mathsf{q_1}\right)\Big)$$

Now, the global reduction consists in building a larger automaton, with one state $\mathsf{q_j}$ per instruction of the two-counter machine, and the intermediary states required by the above modules. The following formula expresses that the halting state can be reached after a finite number of executions of the instructions:

$$\mathsf{E}\Big(\bigwedge_j(\mathsf{q_j} \rightarrow \varphi_{\text{type}(\mathsf{q_j})})\Big)\mathsf{U}\mathsf{q}_{\text{Halt}}$$

where $\text{type}(\mathsf{q_j})$ is the type of instruction $\mathsf{q_j}$ (*i.e.*, "incr1" if $\mathsf{q_j}$ is an incrementation of counter c_1, "test1" is it is a test of counter c_1, and so on). State $\mathsf{q_0}$ satisfies this property iff there exists a computation of the two-counter machine that ends up in state q_{Halt}. □

7 Conclusion

In this paper we have proved that the model-checking of one-clock priced timed automata against WCTL properties is PSPACE-complete. This is rather surprising as model-checking TCTL over one-clock timed automata has the same complexity, though it allows much less features. For proving this result, we have exhibited a sufficient granularity such that truth of formulas over regions defined with this granularity is uniform. Based on this result, we developed a space-efficient algorithm which computes satisfaction of subformulas on-the-fly. This result has to be contrasted with the undecidability result of [5] which establishes that model-checking priced timed automata with three clocks and more against WCTL properties is undecidable.

There are several natural research directions: the decidability of WCTL model-checking for two-clocks priced timed automata is not known, we just know that these models have an infinite bisimulation [10]; another interesting extension is multi-constrained modalities, e.g. $\mathsf{E}\,\phi\mathsf{U}_{P_1\le 5, P_2>3}\phi$?

References

1. Y. Abdeddaïm, E. Asarin, and O. Maler. Scheduling with timed automata. *Theor. Comp. Science*, 354(2):272–300, 2006.

2. R. Alur, M. Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *Proc. 31st Intl. Coll. Automata, Languages and Programming (ICALP'04)*, LNCS 3142, p. 122–133. Springer, 2004.
3. R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th Intl. Workshop Hybrid Systems: Computation and Control (HSCC'01)*, LNCS 2034, p. 49–62. Springer, 2001.
4. G. Behrmann, A. Fehnker, Th. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th Intl. Workshop Hybrid Systems: Computation and Control (HSCC'01)*, LNCS 2034, p. 147–161. Springer, 2001.
5. P. Bouyer, Th. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Inf. Proc. Letters*, 98(5):188–194, 2006.
6. P. Bouyer, E. Brinksma, and K. G. Larsen. Staying alive as cheaply as possible. In *Proc. 7th Intl. Workshop Hybrid Systems: Computation and Control (HSCC'04)*, LNCS 2993, p. 203–218. Springer, 2004.
7. P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Form. Meth. in Syst. Design*, 2006. To appear.
8. P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *Proc. 24th Conf. Found. Softw. Tech. & Theor. Comp. Science (FST&TCS'04)*, LNCS 3328, p. 148–160. Springer, 2004.
9. P. Bouyer, K. G. Larsen, N. Markey, and J. I. Rasmussen. Almost optimal strategies in one-clock priced timed automata. In *Proc. 26th Conf. Found. Softw. Tech. & Theor. Comp. Science (FST&TCS'06)*, LNCS 4337, p. 346–357. Springer, 2006.
10. Th. Brihaye, V. Bruyère, and J.-F. Raskin. Model-checking for weighted timed automata. In *Proc. Joint Conf. Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, LNCS 3253, p. 277–292. Springer, 2004.
11. Th. Brihaye, V. Bruyère, and J.-F. Raskin. On optimal timed strategies. In *Proc. 3rd Intl. Conf. Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, LNCS 3821, p. 49–64. Springer, 2005.
12. E. A. Emerson and J. Y. Halpern. "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *J. ACM*, 33(1):151–178, 1986.
13. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
14. T. A. Henzinger, O. Kupferman, and M. Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. In *Proc. 7th Intl. Conf. Concurrency Theory (CONCUR'96)*, LNCS 1119, p. 514–529. Springer, 1996.
15. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Model checking timed automata with one or two clocks. In *Proc. 15th Intl. Conf. Concurrency Theory (CONCUR'04)*, LNCS 3170, p. 387–401. Springer, 2004.
16. F. Laroussinie, N. Markey, and Ph. Schnoebelen. Efficient timed model checking for discrete-time systems. *Theor. Comp. Science*, 353(1-3):249–271, 2006.
17. K. G. Larsen and J. I. Rassmussen. Optimal conditional reachability for multi-priced timed automata. In *Proc. 8th Intl. Conf. Found. Softw. Science and Computation Structures (FoSSaCS'05)*, LNCS 3441, p. 234–249. Springer, 2005.
18. J. I. Rasmussen, K. G. Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In *Proc. 10th Intl. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, LNCS 2988, p. 220–235. Springer, 2004.