One Clock Priced Timed Automata: Model Checking and Optimal Strategies

Patricia Bouyer
LSV, CNRS & ENS de Cachan, France
bouyer@lsv.ens-cachan.fr

François Laroussinie
LSV, CNRS & ENS de Cachan, France
fl@lsv.ens-cachan.fr

Kim G. Larsen
Aalborg University, Denmark
kgl@cs.aau.dk

Nicolas Markey
LSV, CNRS & ENS de Cachan, France
markey@lsv.ens-cachan.fr

Jacob Illum Rasmussen
Aalborg University, Denmark
illum@cs.aau.dk

Abstract

In this paper, we consider priced (or weighted) timed automata, and prove various decidability results when the automaton has only one clock: we prove that model checking of WCTL is decidable and that optimal costs in priced timed games are computable. In contrast, it has recently been proved that these problems are undecidable for this model as soon as the system has three clocks.

1 Introduction

An interesting direction of real-time model checking that has recently recieved substantial attention is to extend and re-target timed automata technology towards optimal scheduling and planning [1, 19, 9]. In particular, as part of this effort, the notion of priced timed automata [6, 5] has been promoted as a useful extension of the classical model of timed automata [4]. In this extended model each location q is associated with a cost c_l giving the cost of a unit of time spent in q. Thus, each run of a priced timed automaton has an accumulated cost, based on which a variety of optimization problems may be formulated.

Several of the established results concerning priced timed automata are concerned with reachability questions. In [3] cost-bound reachability was shown decidable. [6] and [5] independently show computability of the cost-optimal reachability for priced (or weighted) timed automata using different adaptations of the so-called region technique. In [17] the notion of priced zone was put forward enabling efficient implementation of cost-optimal reachability. The manipulation of priced zones was further improved in [14] and has now found its way into the competitive tool UPPAAL Cora [20]. Also the problem of computing optimal *infinite* schedules (in terms of minimal limit-ratios) has been shown computable [8]. Finally cost-optimal reachability has been shown decidable in a setting with multiple cost-variables [18].

In this paper we consider the more challenging problems of the computation of *cost-optimal winning strategies* for priced timed *game* automata and of *model checking* priced timed automata (PT(G)A) with respect to WCTL (an extension of CTL where modalities have constraints on cost). Consider the priced timed (game) automata A in Fig.1. Let q_3 satisfy the property ϕ, now all states (q_1, x) and (q_2, x) with $x \geq \frac{2}{3}$ satisfy the WCTL property $\mathsf{EF}_{\leq 3}\phi$, meaning that there is a path leading to q_3 with an accumulated cost not greater than three. For the priced timed game, let q_1 be controllable, q_2 be uncontrollable, and q_3 be the winning location, now the optimal winning strategy for q_2 is: move to q_1 when $x \leq \frac{2}{3}$, delay when $\frac{2}{3} < x < 1$, and move to q_3 when $x = 1$.

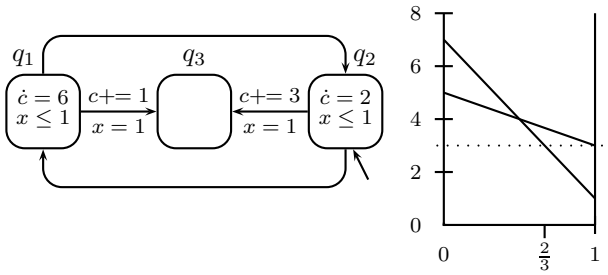

Figure 1. A PT(G)A A.

In [15] the problem of computing cost-optimal winning strategies has been studied and shown computable for *acyclic* priced timed games. Furthermore, in [12], it is proved that computing optimal winning strategies for one-clock PTGA with stopwatch cost (*i.e.* cost are either zero or one) are decidable. [2] and [10] provide (quite similar) partial solutions to the general case of non-acyclic games: under the assumption of certain non-Zenoness behaviour of the underlying priced timed automata it is shown that it suffices only to consider strategies guaranteed to win within

some given number k of steps, or alternatively to unfold the given game k times and reduce the problem to solving an acyclic game. To see how restricted these results are, it may be observed that the priced timed automaton A in Fig.1 does not belong to any of the above classes. In fact, in [12] 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 wrt. WCTL is undecidable [11]. Most recently, it has been shown that these negative results hold even for priced timed (game) automata with no more than three clocks [7].

In this paper we completely solve the computation of cost-optimal winning strategies for arbitrary priced timed (game) automata *with one clock* and the WCTL model checking problem by providing suitable algorithm solutions. We first provide an algorithm for computing cost-optimal winning strategies and we show as a consequence that the iterative semi-algorithm proposed in [10] always terminates. We then provide an EXPSPACE algorithm for solving the model checking problem of WCTL. We finally give indications why this is a difficult problem by proving that a slight extension of WCTL is undecidable, even in the one-clock priced timed automata framework.

2 Definitions

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 $\mathcal{X} \backslash R$.

Definition 2.1. *A* priced timed automaton *(PTA for short) is a tuple $A = (Q, q_0, \mathcal{X}, T, \eta, P)$ 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;*
- *$P \colon Q \cup T \to \mathbb{N}$ is the* cost (or price) function.

*A k-*clock PTA *(k-PTA for short) is a PTA where $|\mathcal{X}| = k$.*

In the remainder of the paper, we focus on 1-PTA. The semantics of a 1-PTA A is given as a labeled timed transition $\mathcal{T} = (S, s_0, \to)$ where $S \subseteq Q \times \mathbb{R}_+$ is the set of states, $s = (q_0, v_0)$[1] is the initial state, and the transitions relation $\to \subseteq S \times \mathbb{R}_+ \times S$ is defined as:

1. (discrete transition) $(q, v) \xrightarrow{c} (q', v')$ if there exists $(q, g, R, q') \in E$ s.t. $v(x) \models g$, $v' = [R \leftarrow 0]v$, $v'(x) \models \eta(q')$, and $c = P(q, g, R, q')$;
2. (delay transition) $(q, v) \xrightarrow{c} (q, v + t)$ if $\forall 0 \leq t' \leq t$, $v + t' \models \eta(q)$, and $c = t \cdot P(q)$.

A *run* (or equivalently *trajectory*) of a 1-PTA is a path in the underlying transition system. Given a run

$$\varrho = s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \cdots \xrightarrow{c_{n-1}} s_n,$$

the cost of ϱ, denoted $\mathsf{cost}(\varrho)$, is $\sum_{i=0}^{n-1} c_i$.

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 $\varrho_{\leq \pi}$ denotes the finite prefix of ϱ ending at position π.

In the sequel, we assume without loss of generality that clocks are bounded.

2.1 Priced Timed Game Automata

A *priced timed game automata* (PTGA) is a PTA where the set of locations is divided into two disjoint sets $Q = Q_c \cup Q_u$ called *controllable* and *uncontrollable* locations, respectively. Furthermore, there is a special goal location q_{goal} with $P(q_{\text{goal}}) = 0$ and no outgoing edges. When describing game automata we use G as opposed to A.

A strategy is a function: $\sigma : Q_c \times \mathbb{R}_+ \to \{\lambda\} \cup Q$. The special symbol λ indicates a strategy to delay.

A run $\varrho = (q_0, x_0) \xrightarrow{c_0} (q_1, x_1) \xrightarrow{c_1} \cdots \xrightarrow{c_{n-1}} (q_n, x_n)$ in a PTGA is a σ-run according to some strategy σ if for all delay- (respectively, discrete-) transitions $(q_i, x_i) \xrightarrow{c_i} (q_{i+1}, x_{i+1})$ where $q_i \in Q_c$, we have $\forall\ x \in [x_i, x_{i+1}[\ : \sigma(q_i, x) = \lambda$ (respectively, $\sigma(q_i, x_i) = q_{i+1}$).

Given a PTGA G, a state, $s \in Q \times [0, e]$, and a strategy σ, we write $\mathsf{Run}_G(s, \sigma)$ to denote all σ-runs in G starting in s. The cost of σ from s in G is given as:

$$\mathsf{Cost}_G(s, \sigma) = \sup\{\mathsf{cost}(\varrho) \mid \varrho \in \mathsf{Run}_G(s, \sigma)\}.$$

A strategy σ is winning if all σ-runs end in q_{goal}. As q_{goal} has no outgoing edges we write $(q_{\text{goal}}, -)$ for the goal state. The optimal cost of a state s in A is defined as:

$$\mathsf{OptCost}_G(s) = \inf\{\mathsf{Cost}_G(s, \sigma) \mid \sigma \text{ is a winning strategy}\}$$

Strategy σ^* is optimal if $\mathsf{OptCost}_G(s) = \mathsf{Cost}_G(s, \sigma^*)$ for all s.

[1] v_0 assigns zero to all clocks.

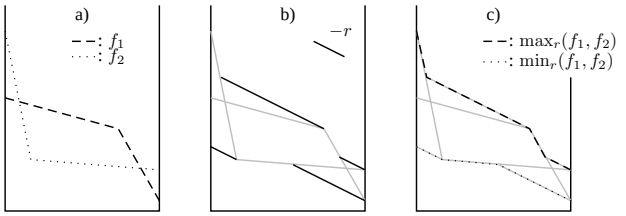

Figure 2. The $\min_r$ and $\max_r$ operations over two functions f_1 and f_2.

Given a cost function, $f : [a, b] \to \mathbb{R}_+$, over an interval $[a, b]$, we define the operations $\min_r$ and $\max_r$ on f with respect to some rate r as functions:

$$\min{}_r(f) = \lambda x \in [a, b].\ \min_{x \le x' \le b} r \cdot (x' - x) + f(x')$$
$$\max{}_r(f) = \lambda x \in [a, b].\ \max_{x \le x' \le b} r \cdot (x' - x) + f(x')$$

We extend the definition of $\min_r$ and $\max_r$ to sets of functions in the obvious way, see Figure 2.

2.2 The Logic WCTL

Let AP be a set of atomic propositions. The logic WCTL [11] extends CTL with cost constraints. Its syntax is given by the following grammar:

$$\text{WCTL} \ni \phi ::= p \mid \neg\phi \mid \phi \vee \phi \mid \mathsf{E}\phi\mathsf{U}_{\sim c}\phi \mid \mathsf{A}\phi\mathsf{U}_{\sim c}\phi$$

where p ranges over AP, c ranges over $\mathbb{N}$, and $\sim\ \in \{<, \le, =, \ge, >\}$.

We interpret formulas of WCTL over labeled 1-PTA, *i.e.* 1-PTA having a labeling function ℓ which associates with every location q a subset of AP.

Definition 2.2. *Let A be a labeled 1-PTA. The satisfaction relation of WCTL is defined over configurations (q, x) of A as follows:*

$$\begin{aligned}
(q,x) \models p &\Leftrightarrow p \in \ell(q)\\
(q,x) \models \neg\phi &\Leftrightarrow (q,x) \not\models \phi\\
(q,x) \models \phi_1 \vee \phi_2 &\Leftrightarrow (q,x) \models \phi_1 \text{ or } (q,x) \models \phi_2\\
(q,x) \models \mathsf{E}\phi_1\mathsf{U}_{\sim c}\phi_2 &\Leftrightarrow \text{there is an infinite run } \varrho \text{ in } A\\
&\quad \text{from } (q,x) \text{ s.t. } \varrho \models \phi_1\mathsf{U}_{\sim c}\phi_2\\
(q,x) \models \mathsf{A}\phi_1\mathsf{U}_{\sim c}\phi_2 &\Leftrightarrow \text{any infinite run } \varrho \text{ in } A \text{ from } (q,x)\\
&\quad \text{satisfies } \varrho \models \phi_1\mathsf{U}_{\sim c}\phi_2\\
\varrho \models \phi_1\mathsf{U}_{\sim c}\phi_2 &\Leftrightarrow \text{there exists } \pi \text{ position along } \varrho \text{ s.t.}\\
&\quad \varrho[\pi] \models \phi_2, \text{ for all position } \pi'\\
&\quad \text{before } \pi \text{ on } \varrho,\ \varrho[\pi'] \models \phi_1,\\
&\quad \text{and } \mathsf{cost}(\varrho_{\le\pi}) \sim c
\end{aligned}$$

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

Note that formulas of the form $\mathsf{E}\phi\mathsf{U}_{[a,b]}\psi$, with the obvious semantics, can easily be expressed with our definition of WCTL. For instance, $\mathsf{E}\phi\mathsf{U}_{[a,b]}\psi$ is equivalent to $\mathsf{E}(\phi\mathsf{U}_{=a}\mathsf{E}(\phi\mathsf{U}_{\le b-a}\psi))$. Since ϕ is duplicated, the size of the equivalent formula could be exponential, but the complexity of model checking would not suffer from this explosion since it depends on the number of subformulas (*i.e.*, the DAG-size of the formula).

We write $\text{WCTL}_{\le,\ge}$ for the restriction of WCTL not involving punctual constraints, and WCTL^* for the extension of WCTL corresponding to CTL^*.

3 Optimal Strategies for 1-PTGA

In this section, we prove that optimal costs are computable in 1-PTGA, and that optimal strategies always exist and can be computed.

To simplify the proof of computability of optimal winning strategies, we use a restricted 1-PTGA model where all invariants are of the form $x \le 1$ and all guards are either *true* or $x = 1$. We use this model without loss of generality as any 1-PTGA can be transformed to satisfy these constraints[2].

Note that runs of an optimal winning strategy never take the same resetting edge twice. This follows from the fact that strategies are deterministic and we only have one clock, thus, following a resetting edge twice means that the resetting edge is taken infinitely many times contradicting that the strategy is winning. Based on this observation we make a transformation of the game that has no resetting cycles, while maintaining optimal winning strategies.

We assume the game has set of locations Q, and has p resetting transitions. We construct a new game as a $(p+1)$-unfolding of the original game. More formally, its set of locations is $\cup_{0 \le i \le p} Q_i$ where $Q_i = \{q_i \mid q \in Q\}$. If there is a transition $q \xrightarrow{g} q'$ which does not reset clock x, then there are transitions $q_i \xrightarrow{g} q'_i$ for every $0 \le i \le p$. If there is a transition $q \xrightarrow{g,x:=0} q'$ which resets clock x, then there are transitions $q_i \xrightarrow{g,x:=0} q'_{i+1}$ for every $0 \le i < p$.

Proposition 3.1. *The optimal costs in the original game and in locations Q_0 of the constructed game are equal.*

Note that the unfolding is at most polynomial in the size of the game. The combination of the above transformations results in a game with a finite set of strongly connected components (SCCs) without resets, *i.e.* resets only occur on edges from one SCC to another. Each SCC can easily be further transformed into SCCs with either $x = 1$ as guards on all internal edges, or guards *true* on all internal edges (but with invariant $x \le 1$).

[2] We leave the transformation to the keen reader.

We further require that all edges have zero cost, but this requirement is later lifted.

Without resets on interval $[0, e]$**.** We consider a 1-PTGA G, based on an SCC of locations with edges with *outside cost functions* giving in every location, the optimal cost for winning from this location when leaving the SCC (which has already been computed). For use in the proof of the following theorem, some locations in the SCC can be made *urgent*, meaning that no delay is possible in these locations. Furthermore, for all locations in the SCC, the optimal cost of winning is assumed for $x = e$. Initially, we consider $e = 1$.

For SCCs with all guards being $x = e$, *i.e.* no delay is possible, optimal winning strategies are easily computed using a classical min/max-algorithm.

To prove computability of optimal winning strategies in SCCs with all guards being *true* (but invariant $x \leq 1$), we need the following lemma that follows from the definition of cost of runs:

Lemma 3.2. *Let σ be a strategy and let*

$$\varrho = s_0 \xrightarrow{c_0} s_1 \xrightarrow{c_1} \cdots \xrightarrow{c_{i-1}} s_i \xrightarrow{c_i} \cdots \xrightarrow{c_{n-1}} s_n \in G$$

be a σ-run, then

$$\mathsf{Cost}(s_0, \sigma) \geq \mathsf{Cost}(s_i, \sigma) + \sum_{0 \leq j < i} c_j$$

Given the above assumptions we may now state and prove the following main theorem.

Theorem 3.3. *For a 1-PTGA $G = (Q, q_0, \mathcal{X}, T, \eta, P)$ based on an SCC and with piecewise affine outside functions:*

1. *$\mathsf{OptCost}_G(s)$ is computable for all $s \in Q \times [0, e]$,*
2. *σ^* is computable,*
3. *for a given location $q \in Q$ the cost function $cost_q^{[0,e]} : \lambda x \in [0, e] \to \mathsf{OptCost}_G(q, x)$ is a continuous, monotonically decreasing, piecewise affine function whose finitely many segments $f_1, ..., f_n$ either have slope $-c$, where c is the cost rate of some location in G, or are fragments of outside functions.*

Proof. We prove the theorem by induction on the number of non-urgent locations in the SCC.
Basis: If all locations are urgent the solution is the classical iterative min/max algorithm. It may be proved that this procedure converges after at most $n^2 + 1$ iteration, where n is the number of locations.
Step: Now let $q_{\min}$ be the location of the SCC with the smallest cost rate of the non-urgent locations. We consider the cases where $q_{\min}$ is controllable and uncontrollable.

Case: $q_{\min}$ *uncontrollable:* Let G' be identical to G except $q_{\min}$ has been made urgent. By the induction hypothesis we know that the optimal winning strategy, σ^*, can be computed in G'. Furthermore, $\mathsf{OptCost}_{G'}(s) \leq \mathsf{OptCost}_G(s)$ for all states $s \subseteq Q \times [0, e]$ as G' allows fewer runs than G for any strategy.
Let $f = \lambda x \in [0, e].\mathsf{Cost}_{G'}((q_{\min}, x), \sigma^*)$ and let $f_1, ..., f_n$ be the piecewise affine functions constituting f. Choose the largest i such that for all $j > i$ f_j has strictly larger slope than $P(q_{\min})$[3], and let $[u, v]$ denote the interval ranged over by f_i. Now, according to the induction hypothesis f_i has slope corresponding to a fragment of some outside function. Our claim to prove now is that for the interval $[u, e]$, σ^* is also optimal for G.
Consider the interval $[v, e]$: Let ϱ be a σ^*-run in G starting in (q_0, x_0) for some $x_0 \in [v, e]$:

$$\begin{aligned}\varrho : (q_0, x_0) \rightsquigarrow^{\Delta} (q_{\min}, x_1) &\xrightarrow{c_1} (q_{\min}, x_1 + d_1) \\ &\cdots \\ \rightsquigarrow^{\Delta} (q_{\min}, x_n) &\xrightarrow{c_n} (q_{\min}, x_n + d_n) \\ \rightsquigarrow^{\Delta} (q_{\text{goal}}, -)&\end{aligned}$$

where $c_i = P(q_{\min}) \cdot d_i$ and $\rightsquigarrow^{\Delta}$ denotes a sequence where $q_{\min}$ is not visited. Using Lemma 3.2 we have:

$$\begin{aligned}\mathsf{cost}(\varrho) \leq\ & (\mathsf{Cost}_G((q_0, x_0), \sigma^*) - f(x_1)) + c_1 \\ & + (f(x_1 + d_1) - f(x_2)) + c_2 \\ & \cdots \\ & + (f(x_{n-1} + d_{n-1}) - f(x_n)) + c_n + f(x_n + d_n) \\ \leq\ & \mathsf{Cost}_G((q_0, x_0), \sigma^*)\end{aligned}$$

where the first inequality holds because $P(q_{\min})$ is smaller than any slope in f over $[v, e]$.
Now, consider the interval $[u, v]$ corresponding to f_i with smaller slope than $P(q_{\min})$. For $x \in [u, v]$ we get:

$$\begin{aligned}\mathsf{OptCost}_G(q_{\min}, x) \geq & (v - x) \cdot P(q_{\min}) + \mathsf{OptCost}_G(q_{\min}, v) \\ = & (v - x) \cdot P(q_{\min}) + f(v)\end{aligned}$$

where '$\geq$' holds as $q_{\min}$ may delay to $x = v$ regardless of controller strategy and '$=$' holds as we just proved that σ^* is optimal in G for $x \in [v, e]$. We now prove that the inequality is, actually, an equality. Consider a σ^*-run, ϱ, in G from $(q_{\min}, x)$, e.g.

$$\begin{aligned}\varrho : (q_{\min}, x) \rightsquigarrow^{\Delta} (q_{\min}, x_1) &\xrightarrow{c_1} (q_{\min}, x_1 + d_1) \\ &\cdots \\ \rightsquigarrow^{\Delta} (q_{\min}, x_n) &\xrightarrow{c_n} (q_{\min}, x_n + d_n) \\ \rightsquigarrow^{\Delta} (q_{\text{goal}}, -)&\end{aligned}$$

[3] If such an i does not exist, we are already done.

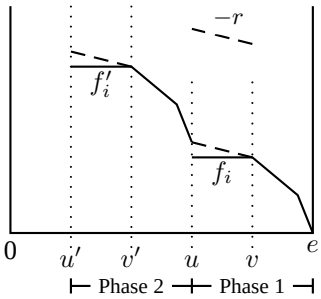

Figure 3. Two phases ($r = P(q_{\min})$)

Again, according to Lemma 3.2 we get:

$$\begin{aligned}\mathsf{cost}(\varrho) &\leq (f(x) - f(x_1)) + c_1 \\ &\quad +(f(x_1 + d_1) - f(x_2)) + c_2 \\ &\quad \dots \\ &\quad +(f(x_{n-1} + d_{n-1}) - f(x_n)) + c_n + f(x_n + d_n) \\ &\quad \leq (v - x) \cdot P(q_{\min}) + f(v)\end{aligned}$$

Thus, delaying is the optimal uncontrollable strategy in $q_{\min}$ over $[u, v]$. Now, for all other $q \in Q$ over $[u, v]$ we need to compute the optimal strategy. Let $G^{\div}$ be the game over $[u, v]$ where $q_{\min}$ has been replaced by an outside function $\lambda x \in [u, v].(v - x) \cdot P(q_{\min}) + \mathsf{Cost}_{G'}((q_{\min}, v), \sigma^*)$. The induction hypothesis gives that the optimal strategy can be computed for all locations in $G^{\div}$.

Finally, consider (q, x) for all $q \in Q$ and $x \in [0, u]$. We have concluded that the optimal cost can be computed for (q, u) regardless of q. We can now apply the same procedure over the interval $[0, u]$ (giving us a cost function f', see Figure 3), that we have just applied for $[0, e]$, and we need to show that this process eventually terminates. In this second iteration we select the appropriate interval $[u', v']$ where f_i' has a slope less than $P(q_{\min})$ and $f'[v', u]$ has a strictly greater slope than $P(q_{\min})$. $f'[u', v']$ corresponds to some fragment of an outside cost function $g'[u', v']$ and $f[u, v]$ corresponded to some fragment of an outside function $g[u, v]$. But since the rates of f' in $[v', u]$ are greater than $P(q_{\min})$ (thus, greater than both the slopes of g and g'), g and g' are part to different affine segments of outside cost functions. As there are finitely such affine segments of outside cost functions, continuing this procedure obtaining $f'', f''', ...$ eventually terminates.

Case $q_{\min}$ *controllable:* We prove this case by an unfolding of G into an equivalent game[4], $G_{q_{\min}}$, that is solvable by the induction hypothesis. We use the knowledge that for any strategy that visits $q_{\min}$ more than once, there is a strategy that visits $q_{\min}$ only once and is at least as good. Consider a strategy σ and a σ-run that visits $q_{\min}$ twice, e.g.

$$\varrho : \underbrace{(q_{\min}, x) \rightsquigarrow^{\Delta} (q_{\min}, x')}_{\varrho'} \rightsquigarrow^{\Delta} (q_{\text{goal}}, -)$$

Using Lemma 3.2 we get:

$$\begin{aligned}\mathsf{Cost}((q_{\min}, x), \sigma) &\geq \mathsf{cost}(\varrho') + \mathsf{Cost}((q_{\min}, x'), \sigma) \\ &\geq P(q_{\min}) \cdot (x' - x) + \mathsf{Cost}((q_{\min}, x'), \sigma)\end{aligned}$$

Thus, the strategy of delaying in $q_{\min}$ until x' would yield a strategy that is at least as good. As an unfolding, we now construct the new game $G_{q_{\min}}$ as follows. Make two identical copies of the SCC, called SCC_1 and SCC_2. Replace $q_{\min}$ by a single $q^*_{\min}$ between the two SCCs and let $q^*_{\min}$ have incoming edges from SCC_2 only and outgoing edges to SCC_1 only. In SCC_1, direct all incoming edges to $q_{\min}$ to an outside cost function with infinite cost (see Figure 4). Now, by the induction hypothesis we can compute the optimal cost of winning in SCC_1. Now, we can use the computed cost functions for successors of $q_{\min}$ and any outside cost functions directly connected to $q_{\min}$ to compute the optimal cost of winning in $q^*_{\min}$. By the induction hypothesis we can compute the optimal cost of winning in SCC_2 using $q^*_{\min}$ as an external cost function, which for all locations, $q \in G \backslash \{q_{\min}\}$, corresponds to $\mathsf{OptCost}_G((q, x))$ for all $x \in [0, e]$. σ^* is then defined as the optimal strategy computed in SCC_2. □

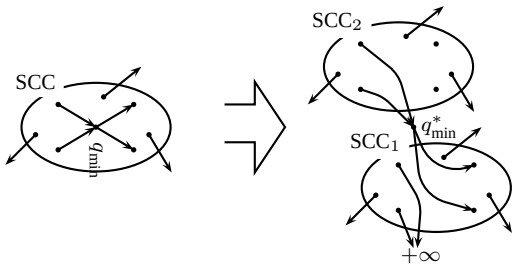

Figure 4. Unfolding

The complexity of this approach is at least exponential, and results from the case when all locations are controllable where an SCC with n location is split into two subproblems of size $n - 1$, etc. Previously, we imposed the restriction of no discrete cost on transitions, however, this is easily lifted by adding the edge cost constant to the respective cost functions.

Fix-Point Characterization. In this section we provide a fix-point characterization of optimal winning strategies, analogous to the previous run-based characterization. Let $Q = Q_c \cup Q_u$ be the locations of a 1-PTGA, and $succ(q_i)$ be the successor locations of $q_i \in Q$, then define the cost

[4] W.r.t. optimal cost.

functions $\mathsf{cost}^i_{G,q}$ inductively as:

$$\mathsf{cost}^0_{G,q} = \begin{cases} \mathbf{0} : q = q_{\text{goal}} \\ +\infty : \text{otherwise} \end{cases}$$

$$\mathsf{cost}^i_{G,q} = \begin{cases} \max_{P(q)}\{\mathsf{cost}^{i-1}_{G,q'} \mid q' \in \mathit{succ}(q)\} : q \in Q_u \\ \min_{P(q)}\{\mathsf{cost}^{i-1}_{G,q'} \mid q' \in \mathit{succ}(q)\} : q \in Q_c \end{cases}$$

where $i > 0$.

Theorem 3.4. *For a* 1*-PTGA,* $\langle \mathsf{cost}^i_G \rangle^\infty_{i=1}$ *(the semi-algorithm of [10]) converges in a finite number of steps towards the optimal costs for winning. Moreover, this computation provides an optimal winning strategy.*

Proof. Due to lack of space we only give the structure of the proof. It is straightforward to see that $\mathsf{cost}^i_G \geq \mathsf{cost}^{i+1}_G$ (where $\geq$ indicates pointwise ordering of functions, *i.e.* for all locations q and all x, $\mathsf{cost}^i_{G,q}(x) \geq \mathsf{OptCost}_G(q,x)$).

To prove convergence of the decreasing sequence $\langle \mathsf{cost}^i_G \rangle^\infty_{i=1}$, the induction proof from Theorem 3.3 is followed. In both case of the induction step the 1-PTGA is essentially divided into two components G_1 and G_2 (either by dividing the interval $[0, e]$ or an unfolding) such that $\mathsf{OptCost}_{G_1}$ depends on $\mathsf{OptCost}_{G_2}$, but not *vice versa*. Due to standard fix-point theory the convergence of the simultaneous fix-point computation, given by the sequence $\langle \mathsf{cost}^i_G \rangle^\infty_{i=1}$, converges at least as fast and the convergence of the sequence $\langle \mathsf{cost}^i_{G_2} \rangle^\infty_{i=1}$ followed by the convergence of $\langle \mathsf{cost}^i_{G_1} \rangle^\infty_{i=1}$, both guaranteed by the induction hypothesis. □

The complexity of computing optimal winning strategies for 1-PTGA remains an open problem.

4 Model Checking WCTL

In this section, we provide an algorithm for solving the model checking problem for WCTL on 1-PTA. Our algorithm is EXPSPACE, with two sub-cases in EXPTIME. The best known lower bound is PSPACE [16]. However it is worth reminding that the model checking of WCTL on 3-PTA (*i.e.* PTA with 3 clocks) is undecidable [7]. Finally we also provide hints that partially explain the high complexity of our algorithm compared to TCTL (see [16]).

4.1 Model Checking WCTL on 1-PTA

Theorem 4.1. *Model checking WCTL on* 1*-PTA is decidable. It is in* EXPSPACE *in general for* 1*-PTA,* EXPTIME *for the restriction to reset-free* 1*-PTA and in* EXPTIME *also for* 1*-PTA when the logic is restricted to WCTL*$_{\leq,\geq}$.

Let A be a 1-PTA and φ a WCTL formula. Our algorithm will compute for every sub-formula of φ all states satisfying the sub-formula. We thus assume we have already computed states that satisfy formulas ϕ and ψ, and we assume in addition that for every location q, the set of x such that (q, x) satisfies ϕ (resp. ψ) is a finite union of intervals. We now describe the algorithm for computing all states satisfying $\mathsf{E}\phi\mathsf{U}_{\sim p}\psi$.

We split the time line into *regions* as follows: let $0 = a_0 < a_1 < \ldots < a_n < +\infty$ be the original constants of the automaton and the constants defining the intervals on which truth of ϕ and ψ is uniform. A *region* is then either an open interval (a_i, a_{i+1}) or a single point $\{a_i\}$. Like classically, these regions define a time-abstract bisimulation [16]. Note that initially, the constants are the constants appearing in the constraints of the automaton.

For computing the set of states that satisfy $\mathsf{E}\phi\mathsf{U}_{\sim p}\psi$, we can compute for every state (q, x) all costs of paths from (q, x), along which ϕ always holds, and ending in some region (q', r) satisfying ψ, and then check whether we can achieve a cost satisfying constraint "$\sim p$".

We thus explain how we compute the set of possible costs between a state (q, x) and a region (q', r) in a 1-PTA having non-strict guards. It is then just a technical matter to adapt this algorithm for model checking formulas $\mathsf{E}\phi\mathsf{U}_{\sim p}\psi$ (by handling strict guards, and taking care of the truth value of ϕ and ψ).

We first consider the simpler case when the timed automaton A has no reset transition, and no discrete cost. We restrict the automaton A to transitions whose guards contain the interval (a_i, a_{i+1}). We denote by A_i this new restricted automaton. Since we have assumed that guards are non-strict, all transitions are enabled on the whole interval $[a_i, a_{i+1}]$. Let q and q' be two locations in A_i. Then, we have:

Lemma 4.2. *Let* $S_i(q, q')$ *be the set of locations that are accessible from* (q, a_i) *and co-accessible from* (q', a_{i+1}) *in* A_i*. Then the set of all possible costs of paths going from* (q, a_i) *to* (q', a_{i+1}) *is* $(a_{i+1} - a_i) \cdot [c^{S_i(q,q')}_{\min}, c^{S_i(q,q')}_{\max}]$*, where* $c^{S_i(q,q')}_{\min}$ *and* $c^{S_i(q,q')}_{\max}$ *are the minimum and maximum costs among the costs of locations in* $S_i(q, q')$*.*

Proof of the lemma. Obviously the costs of all paths in A_i belong to the interval $(a_{i+1} - a_i) \cdot [c^{S_i(q,q')}_{\min}, c^{S_i(q,q')}_{\max}]$.

We assume that some value in $(a_{i+1} - a_i) \cdot [c^{S_i(q,q')}_{\min}, c^{S_i(q,q')}_{\max}]$ does not correspond to the cost of a path from (q, a_i) to (q', a_{i+1}) in A_i. Let $\rho_{\min}$ (resp. $\rho_{\max}$) be a path that achieves cost $(a_{i+1} - a_i) \cdot c^{S_i(q,q')}_{\min}$ (resp. $c^{S_i(q,q')}_{\max}$). Let $\tau_{\min}$ (resp. $\tau_{\max}$) be the corresponding sequence of transitions. The set of costs of all the paths that follow the sequence of transitions $\tau_{\min}$ should be disjoint from the set

of costs of all paths that follow the sequence of transitions $\tau_{\max}$. This is not possible as both sets contain the value $c \cdot (a_{i+1} - a_i)$ where c is the cost rate of location q. □

From these computations, we build a graph G that will store all possible costs between symbolic states (*i.e.* pairs (q, r) where q is a location and r a region) in A. Vertices of G are pairs (q, a_i), pairs $(q, x, (a_i, a_{i+1}))$, and pairs $(q, (a_i, a_{i+1}))$ where q is a location of A. There is an edge from (q, a_i) to (q', a_k) iff

- either $k = i + 1$ and there is a path from (q, a_i) to (q', a_k), the edge is then labeled with the interval $(a_{i+1} - a_i) \cdot [c_{\min}^{S_i(q,q')}, c_{\max}^{S_i(q,q')}]$;
- or $k = i$ and there is a path from (q, a_i) to (q', a_i) in A, the edge is then labeled with the interval $[0, 0]$.

Other edges of G are defined below.

- there is an edge from (q, a_i) to $(q', (a_i, a_{i+1}))$ labeled with the interval $(0, (a_{i+1} - a_i) \cdot c_{\max}^{S_i(q,q')})$ (with the same notations as above);
- there is an edge from $(q, x, (a_i, a_{i+1}))$ to (q', a_{i+1}) labeled with $(a_{i+1} - x) \cdot [c_{\min}^{S_i(q,q')}, c_{\max}^{S_i(q,q')}]$;
- there is an edge $(q, x, (a_i, a_{i+1}))$ to $(q', (a_i, a_{i+1}))$ labeled with $[0, (a_{i+1} - x) \cdot c_{\max}^{S_i(q,q')})$.

Roughly, the vertex $(q, x, (a_i, a_{i+1}))$ will be used to initiate the computation of the costs of paths starting in some (q, x) with $x \in (a_i, a_{i+1})$, whereas the vertex $(q', (a_i, a_{i+1}))$ will be used as a final state where we will end up the computations of the costs of paths ending in some (q', y) with $y \in (a_i, a_{i+1})$.

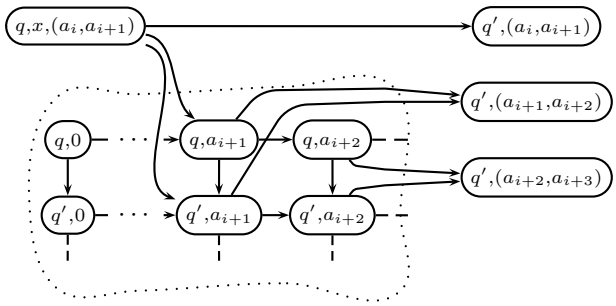

Figure 5. The graph $\mathcal{G}$ (intervals omitted)

Let τ be a finite sequence of consecutive edges in G. The interval cost of this sequence of transitions is the sum of all intervals labeling the edges, which is defined in a natural way. For instance $[a, b] + [c, d] = [a + c, b + d]$.

We can now state the following correctness lemma, which follows from what precedes.

Lemma 4.3. *Let q and q' be two locations of A. Let $x \in (a_i, a_{i+1})$ (resp. $x = a_i$), and let r be a region (either a singleton a_j or an open interval (a_j, a_{j+1})). Then the union of all costs of paths from $(q, x, (a_i, a_{i+1}))$ (resp. (q, a_i) to (q', r) in G is the set of all costs of paths from (q, x) to (q', r) in A.*

Note that, since A has no reset, the graph G is "almost" acyclic, and all cycles are labeled by intervals $[0, 0]$. It is then easy to compute the cost between $(q, x, (a_i, a_{i+1}))$ (resp. (q, a_i)) and (q', r). In the first case, it is a finite union of intervals of the form $[\alpha - \beta x, \alpha' - \beta' x]$. In the second case, it is a finite union of intervals $[\alpha, \beta]$. It then suffices to compute from which configurations (q, x) it is possible to reach a given "region" (q', r) with cost satisfying $\sim p$.

We now explain how we handle resets. The important property of 1-PTA is that, when taking a transition that resets the clock and goes to location q', the configuration is always $(q', 0)$, no matter what the past of the path is. Thus, we extend the graph G, computed previously without resets, by adding edges from (q, r) (r is a region) to $(q', 0)$ labeled by $[0, 0]$ whenever there is a transition $q \xrightarrow{g, x:=0} q'$ such that $r \subseteq g$. As previously, this graph contains all possible costs between states of the automaton, and Lemma 4.3 still holds. The graph has now cycles, but to compute all possible costs from a state $(q, x, (a_i, a_{i+1}))$ (or (q, a_i)) to a state (q', a_j) or $(q', (a_j, a_{j+1}))$, we can use regular expressions over intervals [13], and applying [13, Theo. 3.5], we get that the above sets of costs are finite unions of intervals. Thus, given a final region (q', r) and a location q, the set of x such that we can reach (q', r) from (q, x) with a cost satisfying $\sim p$ is a finite union of intervals.

We now go back to the initial problem, which is to compute the set of states (q, x) that satisfy formula $\mathsf{E}\phi\mathsf{U}_{\sim p}\psi$. We first restrict the automaton so that all states of the restricted automaton satisfy the formula $\mathsf{E}\phi\mathsf{U}\psi$, and then compute all costs between (q, x) and a region satisfying ψ while staying in states satisfying ϕ. However regions satisfying formulas ϕ and ψ may be non-closed (it may be the case that ϕ holds in $(q, (a_i, a_{i+1}))$ but not in (q, a_i)), we can thus not directly apply the previous construction. A slight modification of the construction is sufficient to overcome the problem. It roughly consists in changing bounds (strict instead of non-strict of some intervals labeling the edges of the graph, but we will not enter into the details here).

Analysis of trajectories. The size of the graph G is linear in the number of locations of the original automaton and in the number of constants that have been computed at the previous steps. We will compute the size of the set of new constants that are required for defining the set of configurations satisfying formula $\mathsf{E}\phi\mathsf{U}_{=p}\psi$. Fix a location q and

$\alpha \in (a_i, a_{i+1})$ such that the formula $\mathsf{E}\phi\mathsf{U}_{=p}\psi$ holds just on the right of α but not just on the left of α (the converse case can be handled similarly) —this is possible as we have seen that the set of x such that $(q, x) \models \mathsf{E}\phi\mathsf{U}_{=p}\psi$ is a finite union of intervals. We will show that α is a rational with a particular form.

Let τ be the sequence of transitions that allows to achieve cost p just on the left of α. The cost of all possible runs that go through the same sequence of (q_i, a_i)'s as τ is, as stated before, a set of one of the following forms:

$$\iota_1(x) = (a - x) \cdot [c, d] + \sum_{j \in J} (a_{j+1} - a_j) \cdot [c_j, d_j]$$
$$\iota_2(x) = \iota_1(x) + (0, (a_{k+1} - a_k) \cdot c_k)$$
$$\iota_3(x) = [0, (a - x) \cdot c)$$

where c's and d's are cost rates of locations of the automaton and a's are constants computed at the previous steps.

Several sub-cases should be considered, we only deal with one of them, others are very similar and lead to the same results. We assume that the constraint "$\sim p$" is "$\leq p$" and that the path thanks to which $\mathsf{E}\phi\mathsf{U}_{\leq p}\psi$ holds on the right of α has cost of the form $\iota_1(x)$, assuming $c \neq 0$. On the right of α, interval $\iota_1(x)$ intersects the interval $[0, p]$ whereas it is not the case on the left of α. Thus, it means that the lower bound of the interval $\iota_1(\alpha)$ is precisely equal to p. Thus,

$$\alpha = a + \frac{\sum_{j \in J} (a_{j+1} - a_j) \cdot c_j - p}{c}$$

which means that if previously computed constants are rationals of granularity $1/C^n$, where C is the lcm of all non-null costs of locations, then α is a rational of granularity $1/C^{n+1}$.

Thus, by induction, we can prove that at the n-th step of the computation, if all constants in the original automaton are integers, interval bounds are rationals of granularity $1/C^n$, where C is the lcm of all cost rates of locations.

Algorithm and complexity. We first explain the algorithm for automata without reset. Assume we have computed all the constants up to the n-th step. We compute the new constants at the $n+1$-st step by simply testing all the possible constants of the form $m/(2C^{n+1})$, *i.e.* all the possible constants and the middles of the intervals they define, against the formula $\mathsf{E}\phi\mathsf{U}_{=p}\psi$. Indeed, by the previous study, we know that truth of this formula is uniform in all intervals of the form $(m/C^{n+1}, (m+1)/C^{n+1})$, it is sufficient to check for one point of this interval. For each state q and each value $x = m/2C^{n+1}$, we thus nondeterministically guess a trajectory in the graph G, starting in (q, x), satisfying $\phi\mathsf{U}\psi$, and we compute its interval of possible costs (this can be achieved in polynomial space, since the graph is (almost) acyclic). The formula is satisfied iff p belongs to that interval. Each step of this algorithm can thus be achieved by an exponential number of independant calls to a PSPACE algorithm, and the whole algorithm is in EXPTIME.

We now turn to the general case. The idea will be similar, but we have to bound the length of the trajectory we will guess, since the graph now contains cycles. Let S be the minimal positive cost of a location in the automaton, and $N = \lceil p \cdot C^n / S \rceil$. Let τ be a cycle in the graph G. The cost interval of τ is either $[0, 0]$ (we call such a cycle a *free* cycle), or $[a, b]$ with $b > 0$ (a *non-free* cycle). But in that case, from Lemma 4.2, $b \geq S/C^n$, and N non-free cycles are sufficient for the maximal bound of the cost interval to go beyond cost p. Take now a path in graph G such that the interval of costs along this path has its upper bound above p. Then, if the lower bound of this interval is below p, we are done, otherwise there is no reason to continue guessing a longer path, the cost will always be above cost p. From these remarks, we deduce that, if there is a trajectory from (q, x) satisfying $\phi\mathsf{U}_{=p}\psi$, we can build one along which each state of G is visited at most N times.

For each state (q, x), the worst-case trajectory is too long to be guessed in polynomial space. However, it can be guessed in nondeterministic exponential time, and the i-th step can thus be achieved through an exponential number (one for every choice of $x = m/2C^i$) of independant calls to an NEXPTIME algorithm. The global algorithm thus lies in the exponential-time hierarchy (and is thus in EXPSPACE).

The complexity for the fragment $\mathrm{WCTL}_{\leq,\geq}$ is lower. For the modality $\mathsf{EU}_{\leq c}$, we only need to consider shortest trajectories, thus we can bound their length by the number of nodes in the graph G. For the modality $\mathsf{EU}_{\geq c}$, we are interested in the longest trajectories, it is sufficient to look at acyclic trajectories in the graph G and to cycles that can be iterated. Globally, the complexity of our algorithm for the fragment $\mathrm{WCTL}_{\leq,\geq}$ is thus EXPTIME (we need to construct graph G and then to nondeterministically guess paths of length polynomial in the size of G).

Handling discrete costs. We now explain how we handle discrete costs on transitions. We will simplify the problem and reduce it to the computation of states satisfying a formula in an automaton without discrete cost. The transformation we present will thus be a preliminary step of the computation. We note T the set of transitions that have a positive discrete cost. We unfold the automaton as follows: there is a copy of A for every integer smaller than p. Copy of location q in the i-th copy is denoted $q_{(i)}$. There is a transition from $q_{(i)}$ to $q'_{(j)}$ if: either $i = j$ and there is a transition from q to q' not in T; or $j = i + k$ and there is a transi-

tion in T with discrete cost k from q to q'. Then, $(q,x), A \models \mathsf{E}\phi\mathsf{U}_{=p}\psi$ iff $(q,x), A_0 \models \bigvee_i \mathsf{E}\phi\mathsf{U}_{=p-i}(\psi \wedge \mathsf{copy}_i)$ where copy_i is an atomic proposition labeling all locations of A_i. The correctness of this construction is obvious. The size of the unfolding is linear in the value p, thus exponential in the size of p if it is encoded in binary.

Note that this unfolding does not increase the complexities we have computed before. Indeed, the size of the graph G for the unfolding will be exponential both in the cost rates of the locations and in the discrete costs of the transitions, instead of only exponential in the cost rates of the locations. New constants necessary to define the solution set are still rational numbers of granularity $1/C^n$.

Handling further modalities. Modalities with non-punctual constraints can be easily expressed from $\mathsf{EU}_{\leq p}$ and $\mathsf{EU}_{\geq p}$ [16].

We now explain how we can handle modalities $\mathsf{EG}_{=p}$ and $\mathsf{AU}_{=p}$. We first explain modality $\mathsf{EG}_{=0}$. We let $\mathsf{c}_{>0}$ be a predicate labeling locations with a positive cost rate, and $\mathsf{r}_{\mathsf{open}}$ (resp. $\mathsf{r}_{\mathsf{closed}}$) be a predicate true on intervals (a_i, a_{i+1}) (resp. on singletons $\{a_i\}$). We note $\mathsf{d}_{>0}$ a predicate that labels states from which it is possible to fire a transition with positive discrete cost. It is easy to prove that the model checking of formula $\mathsf{EG}_{=0}\phi$ can be reduced to the model checking of property $\mathsf{E}\phi\mathsf{U}_{=0}\psi \vee \mathsf{EG}\phi$ where $\psi = \phi \wedge \Big(\mathsf{d}_{>0} \vee (\mathsf{c}_{>0} \wedge \mathsf{r}_{\mathsf{open}})\Big)$. Similarly, model checking formula $\mathsf{EG}_{=p}\phi$ can be handled by distinguishing the different possible ways of satisfying such a formula. The hard case is when the "$= p$" constraint becomes true precisely when entering a closed region. This can be handled by refining our algorithm for detecting that date, and then checking $\mathsf{EG}_{=0}\phi$ from there. We can then handle $\mathsf{A}\phi\mathsf{U}_{=p}\psi$ since it is equivalent to $\mathsf{A}\phi\mathsf{U}_{\geq p}\psi \wedge \mathsf{AF}_{=p}\psi$.

4.2 Costs Are Expensive...

We did not manage to prove optimality of our algorithm, and the best lower bound we know is PSPACE-hardness for WCTL and PTIME-hardness for $\mathrm{WCTL}_{\leq,\geq}$ [16]. However, we believe that "costs are more expensive than time", even with only one clock.

Indeed, we prove that it is possible to encode the halting problem for two-counter machines as a model-checking problem for WCTL*. The two counters c_1 and c_2 are encoded by the clock x by the value $1/(2^{c_1} \cdot 3^{c_2})$.

We first show how to encode an instruction incrementing counter c_1: "$\mathtt{q_j}$: 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). Again, we will require that the price between the date at which we exit $\mathtt{q_j}$ and the date at which we enter $\mathtt{q_k}$ is exactly 1. This is enforced by checking the following formula in q_j:

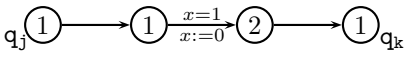

Figure 6. Incrementing a counter

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

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, as well as the same operations on counter c_2 (in that case, the cost of the third location is 3 instead of 2).

Testing if counter c_1 equals 0 requires nesting path modalities: it consists in 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. Mutliplying clock x by 3

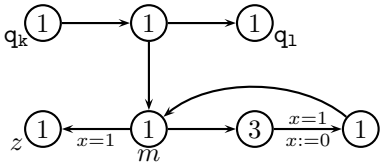

Figure 7. Incrementing a counter

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

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

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 was of the form $1/3^i$, *i.e.*, iff counter c_1 was equal to 0. Now, from $\mathtt{q_k}$, we simply have to ensure the following property:

$$\varphi_{\text{test1}} = \mathsf{E}(\mathtt{q_k}\mathsf{U}_{=0}(\neg\mathtt{q_k} \wedge \mathsf{E}(\neg m\mathsf{U}_{=0}m \wedge \varphi_{\text{mult}}) \wedge \mathsf{E}(\neg\mathtt{q_l}\mathsf{U}_{=0}\mathtt{q_l}))).$$

Now, the global reduction consists in building a larger automaton, with one state $\mathtt{q_j}$ per instruction of the Minsky machine, and the intermediary states. We then need to express that the halting state can be reached after a finite number of executions of the instructions. This is captured by the following formula:

$$\mathtt{q_0} \models \mathsf{E}\Big(\bigwedge_j (\mathtt{q_j} \rightarrow \varphi_{\text{type}(\mathtt{q_j})})\Big)\mathsf{U}\mathtt{q}_{\text{Halt}}$$

where $\text{type}(\mathtt{q_j})$ is the type of instruction $\mathtt{q_j}$ (*i.e.*, "incr1" if $\mathtt{q_j}$ is an incrementation of counter c_1, "test1" is it is a test

of counter c_1, and so on). State q_0 satisfies this property iff there exists a computation of the Minsky machine that ends up in state q_{Halt}.

Theorem 4.4. *Model checking WCTL* on* 1*-PTA is undecidable.*

Note that this reduction only requires the existential fragment of $\text{WCTL}^*_{\leq,\geq}$. It can also easily be adapted to existential $\text{WCTL}_{\leq,\geq}$ augmented with the least fixpoint operator, whose model-checking on 1-PTA is thus also undecidable.

5 Conclusion

In this paper we have proved two decidability results for the class of one-clock priced timed automata: we have first proved that optimal cost in games is computable, and that model checking of WCTL, an extension of CTL with cost constraints, is decidable.

The algorithms we propose are quite involved and have a pretty high complexity. However, this is not really surprising since slight extensions of WCTL already lead to undecidability of model checking, and also since the two problems we consider are undecidable as soon as priced timed automata have three clocks [7].

As further work we would like, of course, to determine what happens with priced timed automata using two clocks, but this seems really difficult as our two algorithms heavily rely on the fact that there is only one clock. Another challenging direction is to determine the precise complexity of the two problems.

References

[1] Y. Abdeddaim, E. Asarin, and O. Maler. Scheduling with timed automata. Submitted to Theor. Comp. Science, 2004.

[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)*, vol. 3142 of *LNCS*, p. 122–133. Springer, 2004.

[3] R. Alur, C. Courcoubetis, and Th. A. Henzinger. Computing accumulated delays in real-time systems. In *Proc. 5th Intl Conf. Computer Aided Verification (CAV'93)*, vol. 697 of *LNCS*, p. 181–193. Springer, 1993.

[4] R. Alur and D. Dill. A theory of timed automata. *Theor. Comp. Science*, 126(2):183–235, 1994.

[5] 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)*, vol. 2034 of *LNCS*, p. 49–62. Springer, 2001.

[6] 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)*, vol. 2034 of *LNCS*, p. 147–161. Springer, 2001.

[7] P. Bouyer, Th. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Inf. Proc. Letters*, 2006. To appear.

[8] 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)*, vol. 2993 of *LNCS*, p. 203–218. Springer, 2004.

[9] P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Form. Meth. in Syst. Design*, 2005. To appear.

[10] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *Proc. 24th Conf. Foundations of Software Technology & Theoretical Computer Science (FST&TCS'04)*, vol. 3328 of *LNCS*, p. 148–160. Springer, 2004.

[11] 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)*, vol. 3253 of *LNCS*, p. 277–292. Springer, 2004.

[12] 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)*, vol. 3821 of *LNCS*, p. 49–64. Springer, 2005.

[13] C. Dima. Real-time automata. *Journal of Automata, Languages and Combinatorics*, 6(1):3–24, 2001.

[14] Th. Hune, K. G. Larsen, and P. Pettersson. Guided synthesis of control programs using UPPAAL. In *Proc. IEEE ICDS Intl Workshop Distributed Systems Verification and Validation*, p. E15–E22. IEEE Comp. Soc. Press, 2000.

[15] S. La Torre, S. Mukhopadhyay, and A. Murano. Optimal-reachability and control for acyclic weighted timed automata. In *Proc. 2nd IFIP Intl Conf. Theoretical Computer Science (TCS 2002)*, vol. 223 of *IFIP Conf. Proc.*, p. 485–497. Kluwer, 2002.

[16] 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)*, vol. 3170 of *LNCS*, p. 387–401. Springer, 2004.

[17] K. G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, Th. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. 13th Intl Conf. Computer Aided Verification (CAV'01)*, vol. 2102 of *LNCS*, p. 493–505. Springer, 2001.

[18] K. G. Larsen and J. I. Rassmussen. Optimal conditional reachability for multi-priced timed automata. In *Proc. 8th Intl Conf. Foundations of Software Science and Computation Structures (FoSSaCS'05)*, vol. 3441 of *LNCS*, p. 234–249. Springer, 2005.

[19] 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)*, vol. 2988 of *LNCS*, p. 220–235. Springer, 2004.

[20] UPPAAL CORA. http://www.cs.aau.dk/~behrmann/cora/, Jan. 2006.