Weighted Timed Automata: Model-Checking and Games

Patricia Bouyer [1,2]

LSV – CNRS & ENS de Cachan – France

Abstract

In this paper, we present weighted/priced timed automata, an extension of timed automaton with costs, and solve several interesting problems on that model.

Key words: Weighted/priced timed automata, model-checking, games.

1 Introduction

Model-checking of real-time systems. Timed automata [3,4] are a well-established model for real-time systems. One of the most important properties they enjoy is probably that reachability properties can be decided for that class of systems. This has given rise to multiple works, both on theoretical aspects and on more practical and algorithmic aspects. Indeed, even several tools have been develop for model-checking this model, for instance HyTech [24], Kronos [21] or Uppaal [29], and there are already several success stories, for instance, if we want to only cite one of these examples, the correction and verification of the Bang & Olufsen audio/video protocol [22].

Weighted/priced timed automata. Weighted timed automata [3] extend classical timed automata with cost information both on locations and edges of the automaton. The cost labelling a location represents the price per time unit for staying in that location, whereas the cost labelling an edge represents the price for taking the transition. That way, every run in the automaton has a global cost, which is the accumulated price along the run of every delay and discrete transition. This model is particularly relevant for modelling resource consumption in real-time systems. This is fundamental in the context

1 Work supported by ACI SI Cortos, a program of the French ministry of research.

2 Email: `bouyer@lsv.ens-cachan.fr`

3 In this paper, we follow the terminology of [5], but this model is called priced timed automaton in [9].

of embedded systems, which are faced with several resource constraints (for instance bandwidth, memory, power consumption, etc...).

Model-checking and games with an optimization criterium. Cost can be used as a measure of the "performance" of the runs in a timed automaton, and it can be used for optimizing the performance of the model. For instance, this model can be used for solving scheduling problems, and for computing the optimal cost for scheduling all tasks [31].

In this context, it is interesting to solve problems like "optimizing the cost for reaching some predetermined location of the automaton", or "optimizing the mean-cost per time unit of infinite runs in the automaton". More generally we can consider model-checking problems where the cost is viewed as an observer variable which can be used in formulas. This is for instance the case of the logic WCTL, which extends CTL with cost constraints on modalities. In this logic, we can express properties like "every request is followed by an answer and it costs less than 5" (with the formula "$\mathbf{A}\,\mathbf{G}\ (\mathit{request} \rightarrow \mathbf{A}\,\mathbf{F}_{<5}\ \mathit{answer})$").

Then, systems can be embedded in an environment. It is thus relevant to study similar questions in the presence of an adversary. These last years, there is a prolific literature about optimal reachability timed games.

Plan of the paper. In this paper we survey some recent results on weighted timed automata. In Section 2, we define basic material like the models we consider (weighted timed automata, logic WCTL, and weighted timed games). In Section 3, we present the optimal cost and optimal mean-cost problems in timed automata. In Section 4, we give results concerning model-checking of the logic WCTL. Finally, in Section 5, we briefly give results concerning weighted timed games.

Closest related topics. The cost variable in a weighted timed automaton can be viewed as an hybrid variable [4], thus weighted timed automata are special cases of *linear hybrid automata* [23], in which all variables are clocks, except the cost, which is however never used for the executions in the automaton but only used for an optimality criterion, or as constraints in the formulas.

2 Definitions

2.1 Preliminaries

We consider as time domain the set $\mathbb{R}_{\geq 0}$ of non-negative reals. We consider a finite set X of variables, called *clocks*. A *clock valuation* over X is a mapping $v : X \rightarrow \mathbb{R}_{\geq 0}$ that assigns to each clock a time value. The set of all clock valuations over X is denoted $\mathbb{R}_{\geq 0}^X$. Let $t \in \mathbb{R}_{\geq 0}$, the valuation $v + t$ is defined

[4] An hybrid variable is a variable which can have different slopes on different locations.

by $(v+t)(x) = v(x) + t$ for all $x \in X$. For $Y \subseteq X$, we denote by $v[Y \leftarrow 0]$ the valuation assigning 0 (resp. $v(x)$) for any $x \in Y$ (resp. $x \in X \setminus Y$). We write $\mathbf{0}$ for the valuation which assigns 0 to every clock $x \in X$.

We denote $\mathcal{C}(X)$ the set of *clock constraints* defined as the conjunctions of atomic constraints of the form $x \bowtie c$ with $x \in X$, $c \in \mathbb{N}$ and $\bowtie \in \{<, \leq, =, \geq, >\}$. For $g \in \mathcal{C}(X)$ and $v \in \mathbb{R}_{\geq 0}^X$, we write $v \models g$ if v satisfies g and $[\![g]\!]$ denotes the set $\{v \in \mathbb{R}_{\geq 0}^X \mid v \models g\}$.

2.2 Weighted/priced timed automata

Weighted/priced timed automata are an extension of timed automata [4] with cost information on both locations and edges.

Let X be a finite set of clocks and AP a finite set of atomic propositions.

Definition 2.1 A *weighted* (or *priced*) *timed automaton* over X and AP is a tuple $(L, \ell_0, T, \lambda, \mathsf{cost})$ where L is a finite set of locations, $\ell_0 \in L$ is the initial location, $T \subseteq L \times \mathcal{C}(X) \times 2^X \times L$ is a finite set of transitions, $\lambda : L \to 2^{\mathsf{AP}}$ is a labeling function, and $\mathsf{cost} : L \cup T \to \mathbb{N}$ assigns to each location and each transition a cost.

The semantics of a weighted timed automaton is similar to that of a timed automaton, as there is no constraint on the cost. It is thus given as a timed transition system $(S, s_0, \longrightarrow)$ where $S = L \times \mathbb{R}_{\geq 0}^X$, $s_0 = (\ell_0, \mathbf{0})$, and $\longrightarrow$ contains two types of transitions:

- *delay transitions:* $(\ell, v) \xrightarrow{\delta(d)} (\ell, v+d)$ if $d \in \mathbb{R}_{\geq 0}$
- *discrete transitions:* $(\ell, v) \xrightarrow{t} (\ell', v')$ if there exists a transition $t = (\ell, g, Y, \ell') \in T$ such that $v \models g$ and $v' = [Y \leftarrow 0]v$

To each such step, we associate a cost defined by

$$\begin{cases} \mathsf{cost}\left((\ell, v) \xrightarrow{\delta(d)} (\ell, v+d)\right) = \mathsf{cost}(\ell) \cdot d \\ \mathsf{cost}\left((\ell, v) \xrightarrow{t} (\ell', v')\right) = \mathsf{cost}(t) \end{cases}$$

A *run* ϱ of the weighted timed automaton is a finite or infinite sequence of steps in the transition system (with no time-stuttering). The cost of ϱ, denoted $\mathsf{cost}(\varrho)$, is the accumulated cost of steps along the run.

Example 2.2 Let us consider the weighted timed automaton $\mathcal{A}$ given on Figure 1.

A possible run in $\mathcal{A}$ is:

$$\varrho : (\ell_0, 0) \xrightarrow{\delta(0.1)} (\ell_0, 0.1) \longrightarrow (\ell_1, 0.1) \longrightarrow (\ell_3, 0.1) \xrightarrow{\delta(1.9)} (\ell_3, 2) \longrightarrow (W, 2)$$

The cost of ϱ is $\mathsf{cost}(\varrho) = 5 \cdot 0.1 + 1 \cdot 1.9 + 7 = 9.4$.

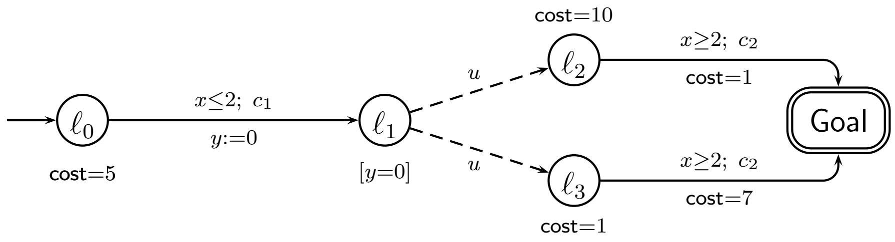


Fig. 1. A weighted timed automaton $\mathcal{A}$

2.3 The logic WCTL

The logic WCTL[5] is a branching-time logic which extends CTL with cost constraints on modalities. It has been first defined in [18].

The syntax of WCTL is given by the following grammar:

$$\mathsf{WCTL} \ni \psi, \varphi ::= p \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \neg \varphi \mid \mathbf{E}\,(\varphi\,\mathbf{U}_{\sim c}\,\psi) \mid \mathbf{A}\,(\varphi\,\mathbf{U}_{\sim c}\,\psi)$$

where $p \in \mathsf{AP}$, $c \in \mathbb{N}$ and $\sim \in \{<, \leq, =, \geq, >\}$.

Formulas of WCTL are interpreted over configurations of a weighted timed automaton $\mathcal{A}$ (*i.e.* pairs (ℓ, v) where ℓ is a location and v a valuation). Its semantics is then defined inductively as follows (atomic propositions and boolean combinations are omitted because they are straightforward):

- $\mathcal{A}, (\ell, v) \models \mathbf{E}\,(\varphi\,\mathbf{U}_{\sim c}\,\psi)$ iff there exists a run ϱ in $\mathcal{A}$ starting in (ℓ, v) and ending in (ℓ', v') such that $\mathcal{A}, (\ell', v') \models \psi$, $\mathsf{cost}(\varrho) \sim c$, and $\mathcal{A}, (\ell'', v'') \models \varphi$ for all configurations (ℓ'', v'') along ϱ.[6]
- $\mathcal{A}, q \models \mathbf{A}\,(\varphi\,\mathbf{U}_{\sim c}\,\psi)$ iff for every infinite run ϱ in $\mathcal{A}$ starting in (ℓ, v) and ending in (ℓ', v') such that $\mathcal{A}, (\ell', v') \models \psi$, $\mathsf{cost}(\varrho) \sim c$, and $\mathcal{A}, (\ell'', v'') \models \varphi$ for all configurations (ℓ'', v'') along ϱ.

Example 2.3 The weighted timed automaton $\mathcal{A}$ of Figure 1 is such that $\mathcal{A}, (\ell_0, \mathbf{0}) \models \mathbf{E}\,\mathbf{F}_{\leq 10} W$ (where we assume that states are labeled by atomic propositions corresponding to their names). This formula expresses that the location W can be reached with a path whose cost is less than or equal to 10, and a witness for that formula is the run ϱ given in Example 2.2.

2.4 Weighted timed games

A *weighted timed game* is a weighted timed automaton in which transitions are decoupled into *controllable* transitions (played by the *controller*) and *uncontrollable* transitions (played by the *environment*).

Let $\mathcal{G} = (L, \ell_0, T, \mathsf{cost})$[7] be a weighted timed game. We assume $\mathcal{G}$ has a distinguished set of Goal locations, which are sink locations with cost 0 per

[5] WCTL stands for "Weighted CTL".

[6] Note that if $(\ell, v) \xrightarrow{\delta(d)} (\ell, v+d)$ is a delay transition of ϱ, then all configurations $(\ell, v+d')$ with $0 \leq d' \leq d$ are "along ϱ".

[7] We abstract away the labeling function as it plays no role in the framework of games.

time unit. A *strategy* for $\mathcal{G}$ from (ℓ, v) is a partial function f from the set of runs in $\mathcal{G}$ starting in (ℓ, v) into the set of controllable transitions of $\mathcal{G}$ plus the symbol λ (which is for "delaying") such that:

- $f((\ell, v))$ is defined,
- if $f(\varrho)$ is defined and ϱ ends in (ℓ', v'), then:
  - either $f(\varrho)$ is a transition t in $\mathcal{G}$ and t is enabled from (ℓ', v'), in which case $f(\varrho \xrightarrow{t})$[8] has to be defined, and for every uncontrollable transition u in $\mathcal{G}$ which is enabled from (ℓ', v'), $f(\varrho \xrightarrow{u})$ has to be defined;
  - or $f(\varrho)$ is λ, and there exists $d > 0$ such that $f(\varrho \xrightarrow{\delta(d')})$ is defined for every $0 < d' \leq d$, and $f(\varrho \xrightarrow{\delta(d')}) = \lambda$ for every $< d' < d$, and for every uncontrollable transition u enabled at some $(\ell', v' + d')$ with $0 \leq d' \leq d$, $f(\varrho \xrightarrow{\delta(d')}\xrightarrow{u})$ has to be defined.

Such a strategy f gives rise in a natural way to a set of *maximal plays* (which means they can not be extended) denoted $\mathit{plays}_{\mathcal{G}}(f, (\ell, v))$. The strategy f is *winning* from (ℓ, v) for the reachability goal Goal iff all (maximal) plays of $\mathit{plays}_{\mathcal{G}}(f, (\ell, v))$ go through Goal at some point.

Classical reachability timed games can be solved and are EXPTIME-complete [8,25]. With weighted timed games, an optimality criterion can be added to those games. The cost of a strategy f from (ℓ, v) is defined by

$$\mathsf{cost}_{\mathcal{G}}(f, (\ell, v)) = \sup\{\mathsf{cost}(\varrho) \mid \varrho \in \mathit{plays}_{\mathcal{G}}(f, (\ell, v))\}$$

Our aim is then to optimize this value and to compute

$$\mathsf{Optcost}_{\mathcal{G}}(\ell, v) = \inf\{\mathsf{cost}_{\mathcal{G}}(f, (\ell, v)) \mid f \text{ winning strategy from } (\ell, v)\}$$

and, when possible, to synthesize *optimal winning strategies* (note that such strategies may not exist).

Example 2.4 [Taken from [16]] We consider the weighted timed automaton of Figure 1. Dashed (resp. plain) arrows are for uncontrollable (resp. controllable) transitions. Depending on the choice of the environment (going to location ℓ_2 or ℓ_3), the accumulated cost along plays of the game is either $5t + 10(2 - t) + 1$ (through ℓ_2) or $5t + (2 - t) + 7$ (through ℓ_3) where t is the delay elapsed in location ℓ_0. The optimal cost the controller can ensure is thus $\inf_{t \leq 2} \max(5t + 10(2 - t) + 1, 5t + (2 - t) + 7) = 14 + \frac{1}{3}$, and the optimal delay is then $t = \frac{4}{3}$. The optimal strategy for the controller is thus to wait in state ℓ_0 until $x = \frac{4}{3}$, and then enter state ℓ_1. Then, the environment chooses to go either to ℓ_2 or to ℓ_3, and finally as soon as $x = 2$, the controller goes to state Goal.

[8] The notation $\varrho \xrightarrow{t}$ is a shortcut for the run ϱ extended by transition t.

3 Optimal Cost and Optimal Mean-Cost Problems

Extending timed automata with cost information gives rise to various interesting optimization problems: is it possible to minimize the global cost for reaching a given goal state? Or, is it possible to stay alive forever while minimizing the mean cost (per time unit for instance)? Such questions are relevant for instance in scheduling problems, where cost can be viewed as resource consumption.

3.1 Decidability results

The *optimal cost problem* asks what is the optimal (e.g. minimal or infimum) cost for reaching a given location in a weighted timed automaton. Given a weighted timed automaton $\mathcal{A} = (L, \ell_0, T, \lambda, \mathsf{cost})$ and a location $\ell \in L$, it is formally defined as

$$\mathsf{Optcost}(\mathcal{A}, \ell) = \inf\{\mathsf{cost}(\varrho) \mid \varrho \text{ finite run from } (\ell_0, \mathbf{0}) \text{ to } \ell\}$$

The first problem which has been solved is the *optimal time problem*, when the cost represents the time elapsing (cost rates in locations are equal to 1 and discrete costs of transitions are equal to 0): the problem is to compute what is the smallest time for reaching a given location.

Theorem 3.1 ([20]) *The optimal time is computable in timed automata.*

Almost ten years after this first result, the general optimal reachability problem has been solved independently in [5] and [9].

Theorem 3.2 ([5,9]) *The optimal cost is computable in weighted timed automata.*

Remark 3.3 Moreover, the problem is PSPACE-complete, as mentioned in [6] and developed in [12]. This is quite surprising as it is the same complexity as simple reachability (without any optimization criterion).

The *optimal mean-cost problem* asks what is the optimal (e.g. minimal or infimum) mean-cost (e.g. cost per time unit) for staying alive in a weighted timed automaton. Given a weighted timed automaton $\mathcal{A} = (L, \ell_0, T, \lambda, \mathsf{cost})$, it is formally defined as

$$\mathsf{Optcost}^{\omega}(\mathcal{A}) = \inf\{\mathsf{cost}(\varrho) \mid \varrho \text{ infinite run from } (\ell_0, \mathbf{0})\}$$

Theorem 3.4 ([14]) *The optimal mean-cost is computable in weighted timed automata.*

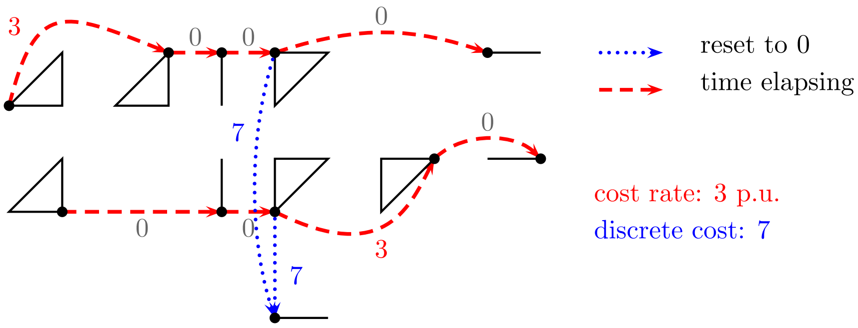
Remark 3.5 As previously, the complexity of this problem is PSPACE-complete.

3.2 The corner-point abstraction

These two decidability results rely on a refinement of the region construction.[9] Indeed, regions are not suitable for computing optimal (mean-)costs because costs of region-equivalent trajectories may have pretty different costs. For example, the cost of run ϱ given in Example 2.2 is 9.4 whereas the cost of the (region-equivalent) run waiting 0.9 time units in ℓ_0 and then 2.1 time units in ℓ_3 is 13.6. The idea is then to record the cost of moving through extremal points of the regions (which have integral coordinates), called *corner-points*, and thus to decorate regions with corner-points. Intuitively, a region R decorated by a corner-point α means that α is an extremal point of R (viewed as a polyhedron), and that we are in region R, close to the point α (then viewed as a valuation). We illustrate this notion in Example 3.6.

Example 3.6 We illustrate the notion of corner-points in a two-dimensional clock space. Classical evolving of regions is depicted in Figure 2(a): while time elapses, regions are visited following time successors (the immediate successor of a triangular region is a flat region while the immediate successor of a flat region is a triangular region), and when firing transitions, clocks may be reset, and regions are then somehow projected into flatter regions.

(a) Classical evolving of regions



(b) Evolving of the regions with corners

Fig. 2. Regions and corner-point abstraction

The corner-point abstraction is depicted on Figure 2(b). Corners decorating regions are indicated with a black bold dot. We consider the top left-most region of the figure decorated with the corner in the bottom. When time

[9] We assume the reader is familiar with the classical notion of regions for timed automata, and better refer to [4].

elapses, it is transformed into the top corner of the same region which is almost one time unit later: thus, as the cost rate in the current location is supposed to be 3 per time unit, the cost of this move is 3 (because almost one time unit has elapsed, the cost has thus increased by almost 3). The next move is then to reach the next region (which is flat) but to stay in the same corner. The cost is thus almost 0 (because almost no time has elapsed), that is why we label the move by 0. And so on. For discrete moves, regions are transformed as usual, and corners are also projected (the projection preserves the property of extremal points of polyhedra). Transitions are then labeled with the cost of the transition (7 in our example).

Given a timed automaton $\mathcal{A}$, we build the so-called *corner-point abstraction of* $\mathcal{A}$, denoted $\Gamma(\mathcal{A})$, which refines the classical region automaton construction by including corner information, as suggested in Example 3.6. The result is a weighted finite graph. The most important property of this graph is that given a finite run $\varrho : (\ell_0, v_0) \to (\ell_1, v_1) \to \ldots \to (\ell_n, v_n)$ in $\mathcal{A}$, there exist two finite paths $\pi : (\ell_0, R_0, \alpha_0) \to (\ell_1, R_1, \alpha_1) \to \ldots \to (\ell_n, R_n, \alpha_n)$ and $\pi' : (\ell_0, R_0, \alpha'_0) \to (\ell_1, R_1, \alpha'_1) \to \ldots \to (\ell_n, R_n, \alpha'_n)$ in $\Gamma(\mathcal{A})$ such that $v_i \in R_i$ for every i, α_i and α'_i are corners of R_i, and $\mathsf{cost}(\pi) \leq \mathsf{cost}(\varrho) \leq \mathsf{cost}(\pi')$. Conversely, for any finite path $\pi : (\ell_0, R_0, \alpha_0) \to (\ell_1, R_1, \alpha_1) \to \ldots \to (\ell_n, R_n, \alpha_n)$ in $\Gamma(\mathcal{A})$, for any $\varepsilon > 0$, it is possible to construct a real path $\varrho : (\ell_0, v_0) \to (\ell_1, v_1) \to \ldots \to (\ell_n, v_n)$ in $\mathcal{A}$ such that $|\mathsf{cost}(\varrho) - \mathsf{cost}(\pi)| < \varepsilon$.

There is thus a strong relation between finite paths in $\mathcal{A}$ and finite paths in $\Gamma(\mathcal{A})$, and computing optimal cost in $\mathcal{A}$ reduces to computing optimal cost in the discrete weighted graph $\Gamma(\mathcal{A})$ [5,9,12].

Though it is not possible to have such a strong relation between infinite paths in $\mathcal{A}$ and infinite paths in $\Gamma(\mathcal{A})$, the corner-point abstraction can be used to compute optimal mean-cost infinite paths [14,15]. Indeed, an optimal mean-cost infinite path in $\Gamma(\mathcal{A})$ is a cycle of optimal mean-cost [26], and its cost is always better than the mean-cost of any infinite path in $\mathcal{A}$.

3.3 Going further: symbolic computation of optimal reachability costs

The computability of optimal reachability cost relies on the construction of a refinement of the region automaton. In practice, this cannot be used for computing optimal cost, and a *symbolic* solution has been proposed in [28]. Analysis of timed automata in practice relies on *zones*, a symbolic representation for state-space of timed automata [11] which is used in many tools like Kronos [21] or Uppaal [29]. A zone is a special kind of polyhedron defined with constraints over clocks of the form $x \bowtie c$ and $x - y \bowtie c$ where x and y are clocks, $\bowtie \in \{<, \leq, =, \geq, >\}$ and $c \in \mathbb{Z}$. Then, the algorithm proposed in [28] relies on an extension of zones, called *priced zones*, which records additional cost information (like the cost of the smallest corner of the zone, and slopes for every clock). The global cost function (representing minimum cost) can be recovered because it is an affine function. The tool Uppaal Cora (a branch of Uppaal) implements the algorithm based on priced zones, and can be downloaded at the address http://www.cs.auc.dk/~behrmann/cora/. The paper [10] reports algorithms and applications of this tool.

4 Model-Checking WCTL

All these positive decidability results on weighted timed automata were very encouraging. It was then very natural to consider more general properties: the logic WCTL has been defined as a natural extension of CTL with cost constraints on modalities [18]. When the cost corresponds to time elapsing, WCTL coincides with the classical logic TCTL [2]. Though WCTL does not extend that much TCTL, it is much harder to model-check. Indeed, the following results have been proved.

Theorem 4.1 ([18,13,17]) *Model-checking* WCTL *is undecidable for weighted timed automata with at least three clocks. It is decidable for weighted timed automata with one clock.*

The undecidability results has been first proved for weighted timed automata with five clocks [18], and further improved for weighted timed automata with three clocks [13]. The decidability result has been proved in [17], and the complexity of the problem lies in 2EXPTIME.

Remark 4.2 Note that the above result only holds for the *dense-time* semantics, *i.e.* when the time domain is $\mathbb{Q}_{\geq 0}$ or $\mathbb{R}_{\geq 0}$. Indeed, it has been proved in [18] that if we restrict to the discrete time domain $\mathbb{N}$, the model-checking of WCTL becomes decidable (the time space can be discretized).

We briefly explain the undecidability result of Theorem 4.1 for weighted timed automata with four clocks, which is a simplified version of the proof presented in [13] for weighted timed automata with three clocks. It is done by reduction from the halting problem for a two-counter machine. Let $\mathcal{M}$ be a two-counter machine. A counter will be encoded using a clock whose value will be $\frac{1}{2^c}$ where c is the value of the counter. Each instruction of $\mathcal{M}$ (test to 0, incrementation, and decrementation) will be simulated with a small widget which will change values of clock according to the nature of the instruction. We consider the incrementation instruction. Its simulation is depicted on Figure 3. When entering the module, the value of the first (resp. second) counter is stored in clock x (resp. y), which means that the value of x (resp. y) is $\frac{1}{2^{c_1}}$ (resp. $\frac{1}{2^{c_2}}$) where c_1 (resp. c_2) is the value of the first (resp. second) counter. Then going through the module takes exactly one time unit (ensured by clock u), and at its end, we expect the value of z be $\frac{1}{2^{c_1+1}}$ while the value of y remains the same. For the value of y to remain the same, we reset clock y when it is equal to 1, and as the total time elapsed within the module is 1, its value at the begin and at the end is the same. Similarly, the value of x is the same at the beginning and at the end of the module. Now, the value of

z is guessed non-deterministically by resetting it at some time in the module, and the module $\mathsf{Test}(x = 2z)$ (that we will present after) will check (together with a WCTL formula) that z stores the correct value, *i.e.* half the value of x.

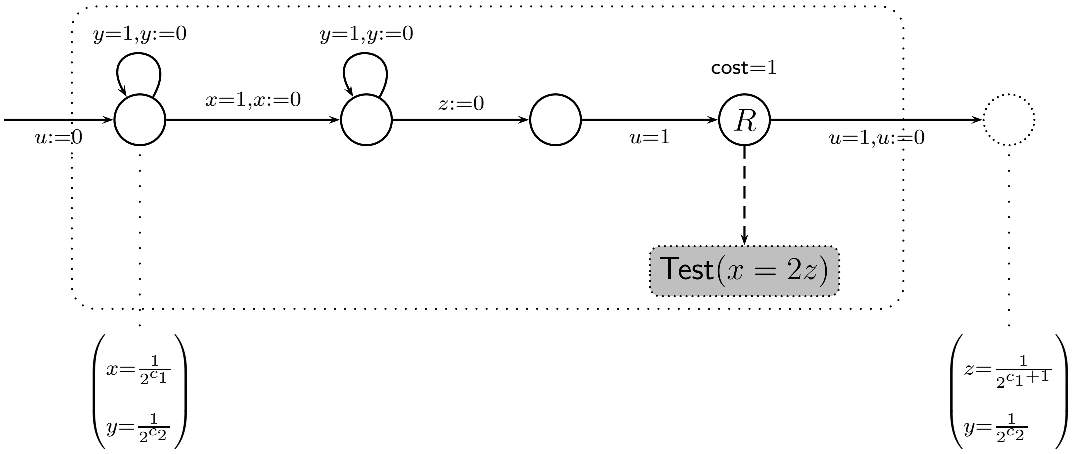
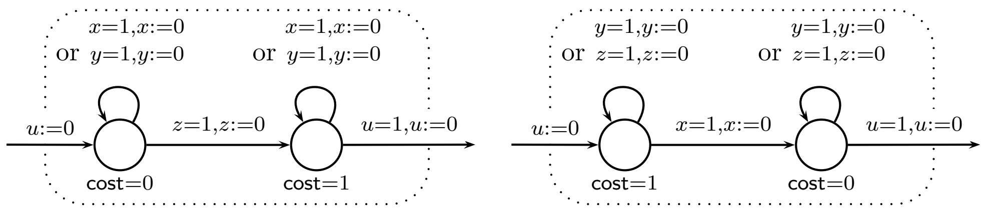


Fig. 3. Simulation of the incrementation instruction



(a) Automaton $\mathsf{Add}(z)$ (b) Automaton $\mathsf{Add}(1 - x)$

Fig. 4. Automata for increasing the cost by z_0 or $1 - x_0$

We first explain the two small automata of Figure 4. Due to cost information on the locations of these automata, the cost along an execution of automaton $\mathsf{Add}(z)$ (resp. $\mathsf{Add}(1 - x)$) increases by the initial value z_0 of z (resp. $1 - x_0$ where x_0 is the initial value of x) when entering the automaton, while the clocks x, y and z have the same values initially and at the end of the automaton. The automaton $\mathsf{Test}(x = 2z)$ of Figure 5 then increases the value of the cost by $2z_0 + (1 - x_0)$. For checking that initially (when entering $\mathsf{Test}(x = 2z)$), it will then be sufficient to check that this cost is equal to 1. That will be done using a WCTL formula.

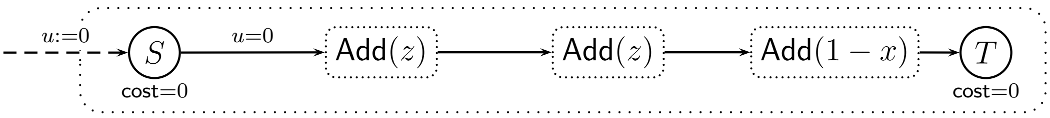


Fig. 5. Automaton $\mathsf{Test}(x = 2z)$

It is pretty similar to simulate a decrementation, and the test to 0 is obvious. The global reduction finally works as follows: we want to reach the halting state (labeled by Halt), and we want the simulation for the counters to be correct all along the halting path. This will be enforced by the following WCTL formula:

$$\mathbf{E}\left(\varphi_{\mathsf{correct}}\,\mathbf{U}\,\mathsf{Halt}\right)$$

where

$$\varphi_{\mathsf{correct}} = R \rightarrow \mathbf{E}\,(R\,\mathbf{U}_{\leq 0}(S \wedge \mathbf{E}\,\mathbf{F}_{=1} T))$$

The formula $\varphi_{\mathsf{correct}}$ checks that the accumulated cost in the $\mathsf{Test}(x = 2z)$ module is equal to 1, which implies that we really have $x_0 = 2z_0$.

5 Optimal Timed Games

In the late 1990's, optimal-time timed games (*i.e.* weighted timed games where cost represents time elapsing) have been considered [7], and the first positive result has been obtained.

Theorem 5.1 ([7]) *Optimal-time timed games are decidable.*

The reason is that regions are sufficient to solve this problem.

Then, in [27], optimal timed games (with general costs) are considered, and a 2EXPTIME algorithm is designed for computing optimal cost (and synthesizing optimal controllers) in *acyclic* timed games. The algorithm somehow extends classical min/max-algorithms for discrete games to timed games.

Optimal timed games have further been studied from 2004 on. In [1], the 2EXPTIME upper bound mentioned above is improved and an EXPTIME upper bound is provided. Note that this algorithm computes for every winning state the optimal cost for winning and provides a (possibly almost) optimal winning strategy. The algorithm which is proposed splits the state-space into polyhedra on which (roughly) optimal winning strategies are uniform, it is pretty involved, and relies on nice geometrical properties of the state-space. Moreover, a family of weighted timed games is given, for which it is unavoidable to split the set of winning states into an exponential number of pieces.

The *bounded cost problem for weighted timed games* asks, given a weighted timed game $\mathcal{G}$ and a threshold c, whether there exists a winning strategy in $\mathcal{G}$ whose cost is less than or equal to c.

Theorem 5.2 ([19,13]) *The bounded cost problem for weighted timed games with more than three clocks is not computable.*

This result has been first obtained for weighted timed games with five clocks [19] and then further improved in [13]. The undecidability result is very close to that for WCTL, which we have presented in Section 4.

In [19], optimal cost in weighted timed games with one clock and *stopwatch cost* is proved to be decidable. Indeed, in this case, the classical region automaton construction can be used. More recently, optimal cost in weighted timed games with one clock (but arbitrary cost) has been proved computable [17] (though in a restricted framework where locations are either controllable — *i.e.* all transitions leaving this location are controllable — or uncontrollable).

6 Conclusion

Timed automata extended with cost information have been extensively studied in the past few years. We have presented here some of the results which have been obtained in the context of model-checking and games. Let us also mention that timed automata extended with several costs have been considered, and an optimal conditional reachability problem has for instance been proved decidable [30].

Acknowledgments. I would like to thank my co-authors for fruitful collaboration and discussions on the subject of timed automata extended with cost information: Thomas Brihaye, Ed Brinksma, Véronique Bruyère, Franck Cassez, Emmanuel Fleury, François Laroussinie, Kim G. Larsen, Nicolas Markey, Jean-François Raskin, and Jacob Illum Rasmussen.

References

[1] Alur, R., M. Bernadsky and P. Madhusudan, *Optimal reachability in weighted timed games*, in: *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, Lecture Notes in Computer Science **3142** (2004), pp. 122–133.

[2] Alur, R., C. Courcoubetis and D. Dill, *Model-checking in dense real-time*, Information and Computation **104** (1993), pp. 2–34.

[3] Alur, R. and D. Dill, *Automata for modeling real-time systems*, in: *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, Lecture Notes in Computer Science **443** (1990), pp. 322–335.

[4] Alur, R. and D. Dill, *A theory of timed automata*, Theoretical Computer Science **126** (1994), pp. 183–235.

[5] Alur, R., S. La Torre and G. J. Pappas, *Optimal paths in weighted timed automata*, in: *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, Lecture Notes in Computer Science **2034** (2001), pp. 49–62.

[6] Alur, R. and P. Madhusudan, *Decision problems for timed automata: A survey*, in: *Proc. 4th International School on Formal Methods for the Design of Computer, Communication and Software Systems: Real Time (SFM-04:RT)*, Lecture Notes in Computer Science **3185** (2004), pp. 122–133.

[7] Asarin, E. and O. Maler, *As soon as possible: Time optimal control for timed automata*, in: *Proc. 2nd International Workshop on Hybrid Systems: Computation and Control (HSCC'99)*, Lecture Notes in Computer Science **1569** (1999), pp. 19–30.

[8] Asarin, E., O. Maler, A. Pnueli and J. Sifakis, *Controller synthesis for timed automata*, in: *Proc. IFAC Symposium on System Structure and Control* (1998), pp. 469–474.

[9] Behrmann, G., A. Fehnker, Th. Hune, K. G. Larsen, P. Pettersson, J. Romijn and F. Vaandrager, *Minimum-cost reachability for priced timed automata*, in: *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, Lecture Notes in Computer Science **2034** (2001), pp. 147–161.

[10] Behrmann, G., K. G. Larsen and J. I. Rasmussen, *Priced timed automata: Decidability results, algorithms, and applications*, in: *Proc. 3rd International Symposium on Formal Methods for Components and Objects (FMCO'04)*, Lecture Notes in Computer Science **3657** (2004), pp. 162–186.

[11] Bouyer, P., *Forward analysis of updatable timed automata*, Formal Methods in System Design **24** (2004), pp. 281–320.

[12] Bouyer, P., Th. Brihaye, V. Bruyère and J.-F. Raskin, *On the optimal reachability problem* (2006), submitted.

[13] Bouyer, P., Th. Brihaye and N. Markey, *Improved undecidability results on weighted timed automata*, Information Processing Letters (2006), to appear.

[14] Bouyer, P., E. Brinksma and K. G. Larsen, *Staying alive as cheaply as possible*, in: *Proc. 7th International Workshop on Hybrid Systems: Computation and Control (HSCC'04)*, Lecture Notes in Computer Science **2993** (2004), pp. 203–218.

[15] Bouyer, P., E. Brinksma and K. G. Larsen, *Optimal infinite scheduling for multi-priced timed automata*, Formal Methods in Systen Design (2005), to appear.

[16] Bouyer, P., F. Cassez, E. Fleury and K. G. Larsen, *Optimal strategies in priced timed game automata*, in: *Proc. 24th Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS'04)*, Lecture Notes in Computer Science **3328** (2004), pp. 148–160.

[17] Bouyer, P., F. Laroussinie, K. G. Larsen, N. Markey and J. I. Rasmussen, *One clock priced timed automata: Model checking and optimal strategies* (2006), submitted.

[18] Brihaye, Th., V. Bruyère and J.-F. Raskin, *Model-checking for weighted timed automata*, in: *Proc. Joint Conference on Formal Modelling and Analysis of Timed Systems and Formal Techniques in Real-Time and Fault Tolerant System (FORMATS+FTRTFT'04)*, Lecture Notes in Computer Science **3253** (2004), pp. 277–292.

[19] Brihaye, Th., V. Bruyère and J.-F. Raskin, *On optimal timed strategies*, in: *Proc. 3rd International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'05)*, Lecture Notes in Computer Science **3821** (2005), pp. 49–64.

[20] Courcoubetis, C. and M. Yannakakis, *Minimum and maximum delay problems in real-time systems*, Formal Methods in System Design **1** (1992), pp. 385–415.

[21] Daws, C., A. Olivero, S. Tripakis and S. Yovine, *The tool Kronos*, in: *Proc. Hybrid Systems III: Verification and Control (1995)*, Lecture Notes in Computer Science **1066** (1996), pp. 208–219.

[22] Havelund, K., A. Skou, K. G. Larsen and K. Lund, *Formal modeling and analysis of an audio/video protocol: An industrial case study using Uppaal*, in: *Proc. 18th IEEE Real-Time Systems Symposium (RTSS'97)* (1997), pp. 2–13.

[23] Henzinger, Th. A., *The theory of hybrid automata*, in: *Proc. 11th Annual Symposim on Logic in Computer Science (LICS'96)* (1996), pp. 278–292.

[24] Henzinger, Th. A., P.-H. Ho and H. Wong-Toi, *HyTech: A model-checker for hybrid systems*, Journal on Software Tools for Technology Transfer **1** (1997), pp. 110–122.

[25] Henzinger, Th. A. and P. W. Kopke, *Discrete-time control for rectangular hybrid automata*, Theoretical Computer Science **221** (1999), pp. 369–392.

[26] Karp, R. M., *A characterization of the minimum mean-cycle in a digraph*, Discrete Mathematics **23** (1978), pp. 309–311.

[27] La Torre, S., S. Mukhopadhyay and A. Murano, *Optimal-reachability and control for acyclic weighted timed automata*, in: *Proc. 2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*, IFIP Conference Proceedings **223** (2002), pp. 485–497.

[28] Larsen, K. G., 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 International Conference on Computer Aided Verification (CAV'01)*, Lecture Notes in Computer Science **2102** (2001), pp. 493–505.

[29] Larsen, K. G., P. Pettersson and W. Yi, *Uppaal in a nutshell*, Journal of Software Tools for Technology Transfer **1** (1997), pp. 134–152.

[30] Larsen, K. G. and J. I. Rasmussen, *Optimal conditional scheduling for multi-priced timed automata*, in: *Proc. 8th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'05)*, Lecture Notes in Computer Science **3441** (2005), pp. 234–249.

[31] Rasmussen, J., K. G. Larsen and K. Subramani, *Resource-optimal scheduling using priced timed automata*, in: *Proc. 10th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04)*, Lecture Notes in Computer Science **2988** (2004), pp. 220–235.