# Symbolic Optimal Reachability in Weighted Timed Automata*


Patricia Bouyer, Maximilien Colange, and Nicolas Markey

LSV – CNRS, ENS Cachan, Université Paris Saclay

**Abstract.** Weighted timed automata have been defined in the early 2000's for modelling resource-consumption or -allocation problems in real-time systems. Optimal reachability is decidable in weighted timed automata, and a symbolic forward algorithm has been developed to solve that problem. This algorithm uses so-called *priced zones*, an extension of standard zones with *cost functions*. In order to ensure termination, the algorithm requires clocks to be bounded. For unpriced timed automata, much work has been done to develop sound abstractions adapted to the forward exploration of timed automata, ensuring termination of the model-checking algorithm without bounding the clocks. In this paper, we take advantage of recent developments on abstractions for timed automata, and propose an algorithm allowing for symbolic analysis of all weighted timed automata, without requiring bounded clocks.

## 1 Introduction

Timed automata [AD94] have been introduced in the early 1990's as a powerful model to reason about (the correctness of) real-time computerized systems. Timed automata extend finite-state automata with several clocks, which can be used to enforce timing constraints between various events in the system. They provide a convenient formalism and enjoy reasonably-efficient algorithms (e.g. reachability can be decided using polynomial space), which explains the enormous interest that they raised in the community of formal verification.

Hybrid automata [ACHH93] can be viewed as an extension of timed automata, involving hybrid variables: those variables can be used to measure other quantities than time (e.g. temperature, energy consumption, ...). Their evolution may follow differential equations, depending on the state of the system. Those variables unfortunately make the reachability problem undecidable [HKPV98], even in the restricted case of stopwatches (i.e., clocks that can be stopped and restarted).

Weighted (or priced) timed automata [ALP01,BFH+01] have been proposed in the early 2000's as an intermediary model for modelling resource-consumption or -allocation problems in real-time systems (*e.g.* optimal scheduling [BLR05]). Figure 1 displays an example of a weighted timed automaton, modelling aircrafts (left) that have to land on runways (right). In (single-variable) weighted

* This work was partly supported by ERC project EQualIS (FP7-308087) and FET project Cassting (FP7-601148).

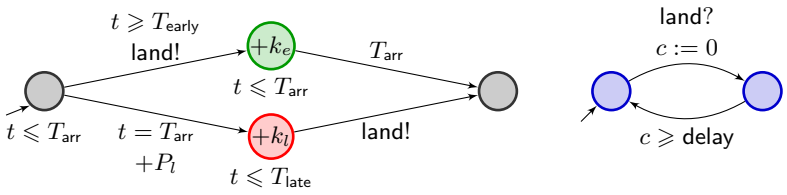



**Fig. 1.** A (simplified) model of the Aircraft Landing System [LBB+01]: aircrafts (left) have an optimal landing time $T_{\text{arr}}$ within a possible landing interval $[T_{\text{early}}, T_{\text{late}}]$. The aircraft can speed up (which incurs some extra cost, modelled by $k_e$) to land earlier than $T_{\text{arr}}$, or can delay landing (which also entails some penalties, modelled by $P_l$ and $k_l$). Some delay has to occur between consecutive landings on the same runway, because of wake turbulence; this is taken into account by the model of the runways (right).

timed automata, each location carries an integer, which is the rate by which the hybrid variable (called *cost* variable hereafter) increases when time elapses in that location. Edges may also carry a value, indicating how much the cost increases when crossing this edge. Notice that, as opposed to (linear) hybrid systems, the constraints on edges (a.k.a. *guards*) only involve clock variables: the extra quantitative information measured by the *cost* is just an observer of the system, and it does not interfere with the behaviors of the system.

Optimal cost for reaching a target, and associated almost-optimal schedules, can be computed in weighted timed automata [ALP01,BFH+01,BBBR07]. The proofs of these results rely on region-based algorithms (either priced regions [BFH+01], or corner-point refinements [ALP01,BBBR07]). Similarly to standard regions for timed automaton [AD94], such refinements of regions are not adapted to a real implementation. A symbolic approach based on priced zones has been proposed in [LBB+01], and later improved in [RLS06]. Zones are a standard symbolic representation for the analysis of timed-automata [BY03,Bou04], and priced zones extend zones with cost functions recording, for each state of the zone, the optimal cost to reach that state. A forward computation in a weighted timed automaton can be performed using priced zones [LBB+01]: it is based on a single-step Post-operation on priced zones, and on a basic inclusion test between priced zones (inclusion of zones, and point-to-point comparison of the cost function on the smallest zone). The algorithmics has been improved in [RLS06], and termination and correctness of the forward computation is obtained for weighted timed automata *in which all clocks are bounded*. Bounding clocks of a weighted timed automaton can always be achieved (while preserving the cost), but it may increase the size of the model. We believe that a better solution is possible: for timed automata and zones, a lot of efforts have been put into the development of sound abstractions adapted to the forward exploration of timed automata, ensuring termination of the model-checking algorithms without bounding clocks [BY03,BBFL03,BBLP06,HKSW11,HSW12].

In this paper, we build on [LBB+01,RLS06], and extend the symbolic algorithm to general weighted timed automata, without artificially bounding the clocks of the model. The keypoint of our algorithm is an inclusion test between *abstractions* of priced zones, computable from the (non abstracted) priced zones themselves. It can be seen as a priced counterpart of a recently-developed inclusion test over standard zones [HSW12]: it compares abstractions of zones without explicitly computing them, which has shown its efficiency for the analysis of timed automata. We prove that the forward-exploration algorithm using priced zones with this inclusion test indeed computes the optimal cost, and that it terminates. We also propose an algorithm to effectively decide inclusion of priced zones. We implemented our algorithm, and we compare it with that of [RLS06].

*Related work.* The approach of [LBB+01,RLS06] is the closest related work. Our algorithm applies to a more general class of systems (unbounded clocks), and always computes fewer symbolic states on bounded models (see Remark 1); also, while the inclusion test of [RLS06] reduces to a mincost flow problem, for which efficient algorithms exist, we had to develop specific algorithms for checking our new inclusion relation. We develop this comparison with [RLS06] further in Section 6, including experimental results.

Our algorithm can be used in particular to compute best- and worst-case execution times. Several tools propose WCET analysis based on timed automata: TIMES [AFM+03] uses binary-search to evaluate WCET, while Uppaal [GELP10] and METAMOC [DOT+10] rely on the algorithm of [RLS06] mentioned above; in particular they require bounded clocks to ensure termination. A tentative workaround to this problem has been proposed in [ARF14], but we are uncertain about its correctness (as we explain with a counter-example in [BCM16]).

All proofs are available in the research report [BCM16].

## 2 Weighted timed automata

In this section we define the weighted (or priced) timed automaton model, that has been proposed in 2001 for representing resource consumption in real-time systems [ALP01,BFH+01]

We consider as time domain the set $\mathbb{R}_{\geqslant 0}$ of non-negative reals. We let $X$ be a finite set of variables, called *clocks*. A *(clock) valuation* over $X$ is a mapping $v\colon X \to \mathbb{R}_{\geqslant 0}$ that assigns to each clock a time value. The set of all valuations over $X$ is denoted $\mathbb{R}_{\geqslant 0}^X$. Let $t \in \mathbb{R}_{\geqslant 0}$, the valuation $v+t$ is defined by $(v+t)(x) = v(x)+t$ for every $x \in X$. For $Y \subseteq X$, we denote by $[Y \leftarrow 0]v$ the valuation assigning 0 (respectively $v(x)$) to every $x \in Y$ (respectively $x \in X \setminus Y$). We write $\mathbf{0}_X$ for the valuation which assigns 0 to every clock $x \in X$.

The set of *clock constraints* over $X$, denoted $\mathcal{C}(X)$, is defined by the grammar $g ::= x \sim c \mid g \wedge g$, where $x \in X$ is a clock, $c \in \mathbb{N}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$.

Clock constraints are evaluated over clock valuations, and the satisfaction relation, denoted $v \models g$, is defined inductively by $v \models (x \sim c)$ whenever $v(x) \sim c$, and $v \models g_1 \wedge g_2$ whenever $v \models g_1$ and $v \models g_2$.

**Definition 1.** *A* weighted timed automaton *is a tuple* $\mathcal{A} = (X, L, \ell_0, \mathsf{Goal}, E, \mathsf{weight})$ *where* $X$ *is a finite set of clocks,* $L$ *is a finite set of locations,* $\ell_0 \in L$ *is the initial location,* $\mathsf{Goal} \subseteq L$ *is a set of goal (or final) locations,* $E \subseteq L \times \mathcal{C}(X) \times 2^X \times L$ *is a finite set of edges (or transitions), and* $\mathsf{weight} : L \cup E \to \mathbb{Z}$ *is a* weight function *which assigns a value to each location and to each transition.*

In the above definition, if we omit the weight function, we obtain the well-known model of *timed automata* [AD90,AD94]. The semantics of a weighted timed automaton is that of the underlying timed automaton, and the weight function provides quantitative information about the moves and executions of the system.

The semantics of a timed automaton $\mathcal{A} = (X, L, \ell_0, \mathsf{Goal}, E)$ is given as a timed transition system $\mathcal{T}_\mathcal{A} = (S, s_0, \to)$ where $S = L \times \mathbb{R}^X_{\geqslant 0}$ is the set of configurations (or states) of $\mathcal{A}$, $s_0 = (\ell_0, \mathbf{0}_X)$ is the initial configuration, and $\to$ contains two types of moves:

- delay moves: $(\ell, v) \xrightarrow{t} (\ell, v + t)$ if $t \in \mathbb{R}_{\geqslant 0}$;
- discrete moves: $(\ell, v) \xrightarrow{e} (\ell', v')$ if there exists an edge $e = (\ell, g, Y, \ell')$ in $E$ such that $v \models g$, $v' = [Y \leftarrow 0]v$.

A run $\varrho$ in $\mathcal{A}$ is a finite sequence of moves in the transition system $\mathcal{T}_\mathcal{A}$, with a strict alternation of delay moves (though possibly 0-delay moves) and discrete moves. In the following, we may write a run $\varrho = s \xrightarrow{t_1} s'_1 \xrightarrow{e_1} s_1 \xrightarrow{t_2} s'_2 \xrightarrow{e_2} s_2 \dots$ more compactly as $\varrho = s \xrightarrow{t_1, e_1} s_1 \xrightarrow{t_2, e_2} s_2 \cdots$. If $\varrho$ ends in some $s = (\ell, v)$ with $\ell \in \mathsf{Goal}$, we say that $\varrho$ is accepting. For a configuration $s \in S$, we write $\mathsf{Runs}(\mathcal{A}, s)$ the set of accepting runs that start in $s$.

In the following we will assume timed automata are non-blocking, that is, from every reachable configuration $s$, there exist some delay $t$, some edge $e$ and some configuration $s'$ such that $s \xrightarrow{t,e} s'$ in $\mathcal{A}$.

We can now give the semantics of a weighted timed automaton $\mathcal{A} = (X, L, \ell_0, \mathsf{Goal}, E, \mathsf{weight})$. The value $\mathsf{weight}(\ell)$ given to location $\ell$ represents a cost rate, and delaying $t$ time units in a location $\ell$ will then cost $t \cdot \mathsf{weight}(\ell)$. The value $\mathsf{weight}(e)$ given to edge $e$ represents the cost of taking that edge. Formally, the cost of the two types of moves is defined as follows:

$$\begin{cases} \mathsf{cost}\Big((\ell, v) \xrightarrow{t} (\ell, v + t)\Big) = t \cdot \mathsf{weight}(\ell) \\ \mathsf{cost}\Big((\ell, v) \xrightarrow{e} (\ell', v')\Big) = \mathsf{weight}(e) \end{cases}$$

A *run* $\varrho$ of a weighted timed automaton is a run of the underlying timed automaton. The cost of $\varrho$, denoted $\mathsf{cost}(\varrho)$, is the sum of the costs of all the simple moves along $\varrho$.

*Example 1.* We consider the weighted timed automaton $\mathcal{A}$ depicted in Figure 2 (left). When a weight is non-null, we add a corresponding decoration to the location or to the transition. A possible run in $\mathcal{A}$ is:

$$\varrho \;=\; (\ell_0, 0) \xrightarrow{0.1} (\ell_0, 0.1) \xrightarrow{e_1} (\ell_1, 0.1) \xrightarrow{e_3} (\ell_3, 0.1) \xrightarrow{1.9} (\ell_3, 2) \xrightarrow{e_5} (\text{☺}, 2)$$

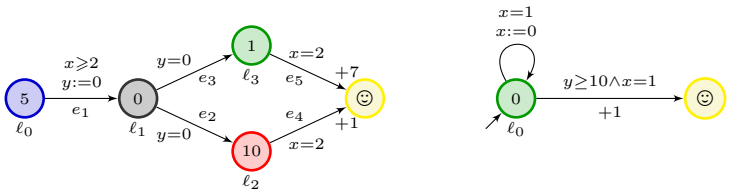



**Fig. 2.** Examples of weighted timed automata

The cost of $\varrho$ is $\mathsf{cost}(\varrho) = 5 \cdot 0.1 + 1 \cdot 1.9 + 7 = 9.4$ (the cost per time unit is 5 in $\ell_0$, 1 in $\ell_3$, and the cost of transition $e_5$ is 7).

### The optimal-reachability problem

For this model we are interested in the optimal-reachability problem, and in the synthesis of almost-optimal schedules. Given a weighted timed automaton $\mathcal{A} = (X, L, \ell_0, \mathsf{Goal}, E, \mathsf{weight})$, the optimal cost from $s = (\ell, v)$ is defined as:

$$\mathit{Optcost}_{\mathcal{A}}(s) = \inf_{\varrho \in \mathsf{Runs}(\mathcal{A}, s)} \mathsf{cost}(\varrho)$$

If $\epsilon > 0$, a run $\varrho \in \mathsf{Runs}(\mathcal{A}, s)$ is *$\epsilon$-optimal* whenever $\mathsf{cost}(\varrho) \leq \mathit{Optcost}_{\mathcal{A}}(s) + \epsilon$.

We are interested in $\mathit{Optcost}_{\mathcal{A}}(s_0)$, simply written as $\mathit{Optcost}_{\mathcal{A}}$, when $s_0$ is the initial configuration of $\mathcal{A}$. It is known that $\mathit{Optcost}_{\mathcal{A}}$ can be computed in polynomial space [ALP01,BFH+01,BBBR07], and that almost-optimal schedules (that is, for every $\epsilon > 0$, $\epsilon$-optimal schedules) can also be computed.

The solutions developed in the aforementioned papers are based on refinements of regions, and a symbolic approach has been proposed in [LBB+01,RLS06], which extends standard zones with cost functions: this algorithm computes the optimal cost in weighted timed automata with nonnegative weights, assuming the underlying timed automata are *bounded*, that is, there is a constant $M$ such that no clock can go above $M$. This is without loss of generality w.r.t. optimal cost, since any weighted timed automaton can be transformed into a bounded weighted timed automaton with the same optimal cost; it may nevertheless increase the size of the model, and more importantly of the state-space which needs to be explored (it can be exponentially larger). We believe that a better solution is possible: for timed automata and zones, a lot of efforts have been put into the development of sound abstractions adapted to the forward exploration of timed automata, ensuring termination of the model-checking algorithm without bounding clocks [BY03,BBFL03,BBLP06,HKSW11,HSW12].

Building on [LBB+01,RLS06], we extend the symbolic algorithm to general weighted timed automata, without assuming bounded clocks. The keypoint of our algorithm is an *abstract* inclusion test between priced zones. It can be seen as a priced counterpart of a recently-developed abstract inclusion test over standard zones [HSW12]; this test compares abstractions of zones without explicitly computing them, and has shown its efficiency for the analysis of timed

automata. We prove that the symbolic algorithm using priced zones and this inclusion test indeed computes the optimal cost, and that it terminates.

## 3 Symbolic algorithm

In this section we briefly recall the approach of [LBB+01,RLS06], and explain how we extend it to the general model, explaining which extra operation is required. The rest of the paper is devoted to proving correctness, effectiveness and termination of our algorithm.

### 3.1 The symbolic representation: priced zones

Let $X$ be a finite set of clocks. A *zone* is a set of valuations defined by a generalized constraint over clocks, given by the grammar $\gamma ::= x \sim c \mid x - y \sim c \mid \gamma \wedge \gamma$, where $x, y \in X$ are clocks, $c \in \mathbb{Z}$, and $\sim \in \{<, \leqslant, =, \geqslant, >\}$. Zones and their representation using Difference Bound Matrices (DBMs in short) are the standard symbolic data structure used in tools implementing timed systems [BY03,Bou04].

To deal with weighted timed automata, zones have been extended to priced zones in [LBB+01]. A *priced zone* is a pair $\mathcal{Z} = (Z, \zeta)$ where $Z$ is a zone, and $\zeta \colon \mathbb{R}^X_{\geqslant 0} \to \mathbb{R}$ is an affine function. In a symbolic state $(\ell, \mathcal{Z})$, the cost function $\zeta$ is meant to represent the optimal cost so far (that is, $\zeta(v)$ is the optimal cost so far for reaching configuration $(\ell, v)$). In [LBB+01], it is shown how one can simply represent priced zones, and how these can be used in a forward-exploration algorithm. The algorithm is shown as Algorithm 1, and we parametrize it by an inclusion test $\preceq$ between priced zones.

Let $\mathcal{A} = (X, L, \ell_0, \mathsf{Goal}, E, \mathsf{weight})$ be a weighted timed automaton. The algorithm makes a forward exploration of $\mathcal{A}$ from $(\ell_0, \mathcal{Z}_0)$ with $\mathcal{Z}_0 = (Z_0, \zeta_0)$, where $Z_0$ is the initial zone defined by $\bigwedge_{x \in X} x = 0$ and $\zeta_0$ is identically 0 everywhere. Then, symbolic successors are iteratively computed, and when the target location is reached, the minimal cost given by the priced zone is computed

**Algorithm 1:** Symbolic algorithm for optimal cost, with inclusion test $\preceq$

```
1  Cost ← ∞
2  Passed ← ∅
3  Waiting ← {(ℓ0, Z0)}
4  while Waiting ≠ ∅ do
5      select (ℓ, Z) from Waiting
6      if ℓ ∈ Goal and infCost(Z) < Cost then
7          Cost ← infCost(Z)
8      if for all (ℓ, Z') ∈ Passed, Z ⋠ Z' then
9          add (ℓ, Z) to Passed
10         add Post(ℓ, Z) to Waiting
11 return Cost
```

(for a priced zone $\mathcal{Z} = (Z, \zeta)$, we note $\mathit{infCost}(\mathcal{Z}) = \inf_{v \in Z} \zeta(v)$), and compared to the current optimal value (variable Cost). An inclusion test between priced zones is performed, which allows to stop the exploration from $(\ell, \mathcal{Z})$ when $\mathcal{Z} \preceq \mathcal{Z}'$ and $(\ell, \mathcal{Z}')$ already appears in the set of symbolic states that have already been explored. In [RLS06], the algorithm uses the following inclusion test $\Subset$, which refines the inclusion test of [LBB+01]: inclusion $\mathcal{Z} \Subset \mathcal{Z}'$ holds whenever $Z \subseteq Z'$ and $\zeta(v) \geq \zeta'(v')$ for every $v \in Z$. As shown in [RLS06], this algorithm computes the optimal cost in $\mathcal{A}$, provided it terminates, and this always happens when the weights in $\mathcal{A}$ are nonnegative, and when all clocks in $\mathcal{A}$ are bounded.

In the present paper, we define a refined inclusion test $\sqsubseteq$ between priced zones, which will enforce termination of Algorithm 1 even when clocks are not upper-bounded, and, to some extent, when costs are negative.

We now give some definitions which will allow to state the correctness of the algorithm. Given a timed automaton $\mathcal{A}$, a location $\ell$ and a priced zone $\mathcal{Z} = (Z, \zeta)$, we say that $(\ell, \mathcal{Z})$ is *realized* in $\mathcal{A}$ whenever for every valuation $v \in Z$, and for every $\epsilon > 0$, there exists a run $\varrho$ from the initial state $(\ell_0, \mathbf{0}_X)$ to $(\ell, v)$, such that $\zeta(v) \leqslant \mathsf{cost}(\varrho) \leqslant \zeta(v) + \epsilon$. For a location $\ell$, a priced zone $\mathcal{Z} = (Z, \zeta)$ and a run $\varrho$ starting in a configuration $s$, we say that $\varrho$ *ends in* $(\ell, \mathcal{Z})$ if $\varrho$ leads from $s$ to a configuration $(\ell, v)$ with $v \in Z$ and $\mathsf{cost}(\varrho) \geqslant \zeta(v)$. The post operation $\mathsf{Post}$ on priced zones used in Algorithm 1 is described in [LBB+01]. Its computation is effective (see [LBB+01]), and is such that (see [RLS06]):

- every $(\ell, \mathcal{Z}) \in \mathsf{Post}^*(\ell_0, \mathcal{Z}_0)$ is realized in $\mathcal{A}$, where $\mathsf{Post}^*$ denotes the iteration of the $\mathsf{Post}$ operator;
- for every run $\varrho$ from a configuration $s$ to a configuration $s'$, and every mixed move $\tau$ from $s'$, if $\varrho$ ends in $(\ell, \mathcal{Z})$, then $\varrho\tau$ ends in an element of $\mathsf{Post}(\ell, \mathcal{Z})$.
- for every run $\varrho$ from $(\ell_0, \mathbf{0}_X)$, there exists $(\ell, \mathcal{Z}) \in \mathsf{Post}^*(\ell_0, \mathcal{Z}_0)$ such that $\varrho$ ends in $(\ell, \mathcal{Z})$ (this is a consequence of the previous property).

The purpose of this work is to propose an inclusion test $\sqsubseteq$ such that the following three properties are satisfied:

1. *(Termination)* Algorithm 1 with inclusion test $\sqsubseteq$ terminates;
2. *(Soudness w.r.t. optimal reachability)* Algorithm 1 with inclusion test $\sqsubseteq$ computes the optimal cost for reaching $\mathsf{Goal}$;
3. *(Effectiveness)* There is an algorithm deciding $\sqsubseteq$ on priced zones.

We now present our inclusion test, and show its soundness for optimal reachability. We then turn to effectiveness (Sect. 4), and then to termination (Sect. 5).

### 3.2 The inclusion test

Our inclusion test is inspired by the inclusion test on (pure) zones proposed in [HSW12].[1] We start by recalling an equivalence relation on valuations. We

[1] Contrary to pure reachability, we cannot use the preorder $\preceq_{LU}$ (which distinguishes between lower-bounded constraints and upper-bounded constraints) [BBLP06], since it does not preserve optimal cost (not even optimal time).

assume a function $M\colon X \mapsto \mathbb{N} \cup \{-\infty\}$ such that $M(x)$ is larger than any constant against which clock $x$ is compared to in the (weighted) timed automata under consideration. Let $v$ and $v'$ be two valuations in $\mathbb{R}^X_{\geq 0}$. Then, $v \equiv_M v'$ iff for every clock $x \in X$, either $v(x) = v'(x)$, or $v(x) > M(x)$ and $v'(x) > M(x)$. We note $[v]_M$ the equivalence class of $v$ under $\equiv_M$.

**Lemma 2.** *If $v \equiv_M v'$, then, for any $\ell \in L$, $Optcost_{\mathcal{A}}(\ell, v) = Optcost_{\mathcal{A}}(\ell, v')$.*

We now define our inclusion test for two priced zones $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$; it is parameterized by $M$, which gives upper bounds on clocks:

$$\mathcal{Z} \sqsubseteq_M \mathcal{Z}' \text{ iff } \forall v \in Z,\ \forall \epsilon > 0,\ \exists v' \in Z' \text{ s.t. } v \equiv_M v' \text{ and } \zeta'(v') \leq \zeta(v) + \epsilon.$$

**Theorem 3.** *When using $\sqsubseteq_M$, provided Algorithm 1 terminates, it is sound w.r.t. optimal reachability (the returned cost is the optimal one).*

*Remark 1.* Remember that the inclusion test $\Subset$ of [RLS06] requires $Z \subseteq Z'$ and, for every $v \in Z$, $\zeta(v) \geq \zeta'(v)$. It is easily seen that $\mathcal{Z} \Subset \mathcal{Z}'$ implies $\mathcal{Z} \sqsubseteq_M \mathcal{Z}'$ for any $M$; hence the branches are always stopped earlier in our algorithm (which uses $\sqsubseteq_M$) than in the original algorithm of [RLS06] (which uses $\Subset$). Moreover, $\Subset$ does not ensure termination of the forward exploration when clocks are not bounded: on the automaton of Figure 2 (right), where the optimal time to reach the right state is 10, the forward algorithm successively computes zones $x \leq 1 \wedge n \leq y - x \leq n + 1$, for every integer $n$. Any two such zones are always incomparable (for $\Subset$).

## 4 Effective inclusion check

In this section we show that we can effectively check the inclusion test $\sqsubseteq_M$ of priced zones. For the rest of this section, we fix two priced zones $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$, and a function $M$. To improve readability, we write $\equiv$ and $\sqsubseteq$ in place of $\equiv_M$ and $\sqsubseteq_M$.

### 4.1 Formulation of the optimization problem

We first express the inclusion of the two priced zones as an optimization problem.

**Lemma 4.** $\mathcal{Z} \sqsubseteq \mathcal{Z}' \iff \sup_{v \in Z} \inf_{\substack{v' \in Z' \\ v' \equiv v}} \zeta'(v') - \zeta(v) \leqslant 0.$

Note that $\mathcal{Z} \sqsubseteq \mathcal{Z}'$ already requires some relation between zones $Z$ and $Z'$: indeed, for the above inclusion to hold, it should be the case that for every $v \in Z$, there exists some $v' \in Z'$ such that $v \equiv v'$. Interestingly, this corresponds to the test on (unpriced) zones developed in [HSW12] (with $L = U = M$); this can be done efficiently (in time quadratic in the number of clocks) as a preliminary test [HSW12, Theorem 34].

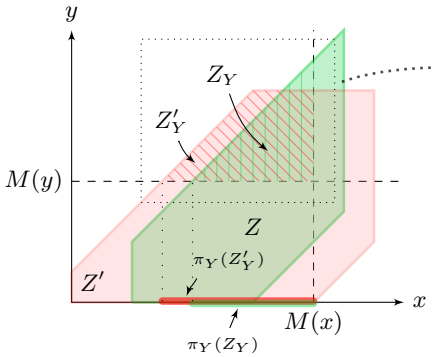



**Fig. 3.** Two-dimensional zones $Z$ and $Z'$, and sub-zones $Z_Y$ and $Z'_Y$ for $Y = \{x\}$.

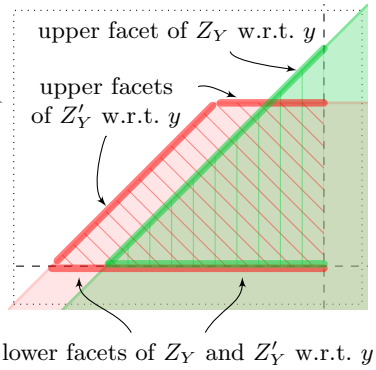



**Fig. 4.** Simple facets of $Z_Y$ and $Z'_Y$ w.r.t. clock $y$.

*Remark 2.* The constraint $v \equiv v'$ is not convex, and we have a bi-level optimization problem to solve. Hence common techniques for convex optimization, such as dualization [BV04], do not directly apply to the above problem. Still, it is possible to transform it into finitely many so-called *generalized semi-infinite optimization problems* (GSIPs) [RS01] (using $Z_Y$'s as defined later in this section). As far as we know, such problems do not have dedicated efficient algorithmic solutions. We thus propose a more direct solution, that benefits from the specific structure of our problem (see for instance Section 4.3); it provides a feasible way to solve our optimization problems, hence to decide $\sqsubseteq$ on priced zones.

In order to compute the above optima, we transform our problem into a finite number of optimization problems that are easier to solve. Let $Y \subseteq X$. A zone $Z$ is $M$-bounded on $Y$ if, for every $v \in Z$, $\{x \mid v(x) \leqslant M(x)\} = Y$. We note $Z_Y$ the restriction of $Z$ to its $M$-bounded-on-$Y$ component: $Z_Y = Z \cap \bigcap_{x \in Y}(x \leqslant M(x)) \cap \bigcap_{x \notin Y}(x > M(x))$. Note that $Z_Y$ may be empty, and that the family $(Z_Y)_{Y \subseteq X}$ forms a partition of $Z$. We also define $\mathcal{Z}_Y$ as the priced zone $(Z_Y, \zeta)$. We define the natural projection $\pi_Y \colon \mathbb{R}^X_{\geq 0} \to \mathbb{R}^Y_{\geq 0}$, which associates with $v \in \mathbb{R}^X_{\geq 0}$ the valuation $v' \in \mathbb{R}^Y_{\geq 0}$ that coincides with $v$ on $Y$.

**Lemma 5.** *The following two properties are equivalent:*

*(i) for every $v \in Z$, there is $v' \in Z'$ such that $v' \equiv v$*
*(ii) for every $Y \subseteq X$, $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$.*

This allows to transform the initial optimization problem into finitely many optimization problems.

**Lemma 6.** $$\sup_{v \in Z} \inf_{\substack{v' \in Z' \\ v' \equiv v}} \zeta'(v') - \zeta(v) = \max_{Y \subseteq X} \sup_{v \in Z_Y} \inf_{\substack{v' \in Z'_Y \\ v' \equiv v}} \zeta'(v') - \zeta(v).$$

**Corollary 7.** *$\mathcal{Z} \sqsubseteq \mathcal{Z}'$ iff for every $Y \subseteq X$, $\mathcal{Z}_Y \sqsubseteq \mathcal{Z}'_Y$*

In the sequel, we write

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \sup_{v \in Z_Y} \inf_{\substack{v' \in Z'_Y \\ v' \equiv v}} \zeta'(v') - \zeta(v)$$

Lemma 4 and Corollary 7 suggest an algorithm for deciding whether $\mathcal{Z} \sqsubseteq \mathcal{Z}'$: enumerate the subsets $Y$ of $X$, and prove that $S(\mathcal{Z}, \mathcal{Z}', Y) \leq 0$. We now show how to solve the latter optimization problem (for a fixed $Y$), and then show how we can drive the choice of $Y$ so that not all subsets of $X$ have to be analyzed.

### 4.2 Computing $S(\mathcal{Z}, \mathcal{Z}', Y)$

We show the following main result to compute $S(\mathcal{Z}, \mathcal{Z}', Y)$, which produces a simpler optimization problem, allowing to decide the inclusion of two priced zones, on parts where cost functions are lower-bounded.

**Theorem 8.** *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two non-empty priced zones, and let $Y \subseteq X$ be such that $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$ and $\zeta$ and $\zeta'$ are lower-bounded on $Z_Y$ and $Z'_Y$ respectively. Then we can compute finite sets $\mathcal{K}_Y$ and $\mathcal{K}'_Y$ of zones over $Y$, and affine functions $\zeta_F$ and $\zeta'_{F'}$ for every $F \in \mathcal{K}_Y$ and $F' \in \mathcal{K}'_Y$ s.t.:*

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{F \in \mathcal{K}_Y} \max_{F' \in \mathcal{K}'_Y} \sup_{u \in F \cap F'} \zeta'_{F'}(u) - \zeta_F(u). \tag{1}$$

The idea behind this result is to first rewrite $S(\mathcal{Z}, \mathcal{Z}', Y)$ into:

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \sup_{u \in \pi_Y(Z_Y)} \Big[ \Big( \inf_{\substack{v' \in Z'_Y \\ \pi_Y(v') = u}} \zeta'(v') \Big) - \Big( \inf_{\substack{v \in Z_Y \\ \pi_Y(v) = u}} \zeta(v) \Big) \Big]$$

which decouples the dependency of $v'$ on $v$. The algorithm then uses the notion of facets (introduced in [LBB+01]), which corresponds to the boundary of the zone w.r.t. a clock (if $W$ is the zone, a facet of $W$ w.r.t. $x$ is $\overline{W} \cap (x = n)$ or $\overline{W} \cap (x - y = m)$ whenever $x \bowtie n$ or $x - y \bowtie m$ is a constraint defining $W$). Given a clock $x \in X \setminus Y$, we consider the facets of $Z_Y$ w.r.t. $x$ that minimize, for any $w \in \pi_{X \setminus \{x\}}(Z_Y)$, the function $v \mapsto \zeta(v)$ when $\pi_{X \setminus \{x\}}(v) = w$. The restriction of $\zeta$ on such a facet is a new affine function, which we can compute. We then iterate the process for all clocks in $X \setminus Y$. We do the same for $\zeta'$. This yields the result claimed above: sets $\mathcal{K}_Y$ and $\mathcal{K}'_Y$ are sets of projections of facets over $Y$.

Facets are zones, and so are their projections on $Y$ and intersections thereof. Additionally, all functions $\zeta_F$ and $\zeta'_{F'}$ are affine; hence the supremum in Eq. (1) is reached at some vertex $u_0$ of zone $F \cap F'$, for some facets $F$ and $F'$. By construction of $\zeta_F$ and $\zeta'_{F'}$, we get

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \inf_{\substack{v' \in Z'_Y \\ \pi(v') = u_0}} \zeta'(v') - \inf_{\substack{v \in Z_Y \\ \pi(v) = u_0}} \zeta(v)$$

In particular, $u_0$ has integral coordinates. We end up with the following result, which will be useful for proving the termination of Algorithm 1:

**Corollary 9.** *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two non-empty priced zones, and let $Y \subseteq X$ be such that $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$ and $\zeta$ and $\zeta'$ are lower-bounded on $Z_Y$ and $Z'_Y$ respectively. Then the following holds:*

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{\substack{u_0 \in \pi_Y(Z_Y) \\ u_0 \in \mathbb{N}^Y}} \Big[ \min_{\substack{v' \in Z'_Y \\ v' \equiv u_0}} \zeta'(v') - \min_{\substack{v \in Z_Y \\ v \equiv u_0}} \zeta(v) \Big]$$

The requirement for lower-bounded priced zones in Theorem 8 is crucial in the proof. But the case when this requirement is not met can easily be handled separately, so that $\sqsubseteq$ can always be effectively decided:

**Lemma 10.** *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two non-empty priced zones.*

- *If $\zeta$ is not lower-bounded on $Z$ but $\zeta'$ is lower-bounded on $Z'$, then $\mathcal{Z} \not\sqsubseteq \mathcal{Z}'$.*
- *Let $Y \subseteq X$ such that $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$. If $\zeta'$ is not lower-bounded on $Z'_Y$, then $\mathcal{Z}_Y \sqsubseteq \mathcal{Z}'_Y$.*

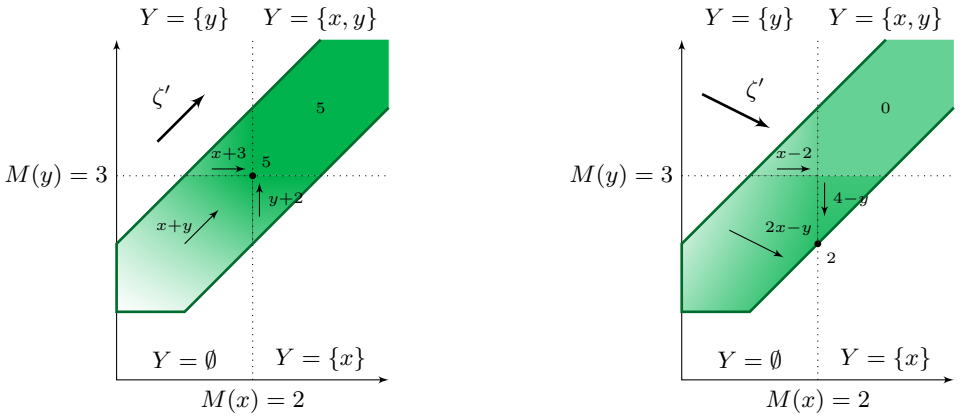
**Corollary 11.** *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two priced zones. Then we can effectively decide whether $\mathcal{Z} \sqsubseteq \mathcal{Z}'$.*

### 4.3 Finding the right $Y$

Applying Lemma 6, the main obstacle to efficiently decide $\sqsubseteq_M$ is to find the appropriate $Z_Y$ in which the sought supremum is reached. Unless good arguments can be found to guide the search towards the best choice for $Y$, an exhaustive enumeration of all the $Y$'s will be required.

*Example 2.* We consider the zone $Z$ defined by the constraints $x \geqslant 0, y \geqslant 1, x \leqslant y$ and $y \leqslant x + 2$. We fix $M(x) = 2$ and $M(y) = 3$. We then consider $Z' = Z$. The zone $Z$ is equipped with a constant cost function $\zeta$. In Figure 5(a), $Z'$ is attached $\zeta'(x, y) = x + y$, and the expression of the function $f(v) = \inf_{v' \in Z',\ v' \equiv_M v} \zeta'(v')$ is given in each $Z_Y$, for $Y \subseteq X$. It is then easy to see that the supremum of $f$ is reached at the point $(2, 3)$, in the middle of the zone. In Figure 5(b), we take $\zeta'(x, y) = 2x - y$, and the expression of the function $f(v) = \inf_{v' \in Z'.\ v' \equiv_M v} \zeta'(v')$ is given in each $Z_Y$. The supremum of $f$ is then reached at the point $(2, 2)$, on the border, but not at a corner of the zone. The latter example also shows that $f$ is not continuous on the whole zone $Z$.

Nevertheless, in many cases, we will be able to guide the search of the $Z_Y$ where the sought optimal is to be found. The following development focuses on the zone, not on the cost function. Given a zone $Z$, we define a preorder $\preceq$ on the clocks, such that if $Z_Y \neq \emptyset$, then $Y$ is downward-closed for $\preceq$. In other words, whenever $x \preceq y$, $y \in Y$ and $Z_Y \neq \emptyset$, then $x \in Y$. The knowledge of $\preceq$ can be a precious help to guide the enumeration of non-empty $Z_Y$'s. Indeed, if $Z_Y \neq \emptyset$, $Y$ is downward-closed for $\preceq$, and candidates for $Y$ are thus found by enumerating the antichains of $\preceq$. In particular, if $\preceq$ is total, then there are at most $|X| + 1$ sets $Y$ such that $Z_Y \neq \emptyset$.




(a) The sought supremum is reached in the middle of the zone.

(b) The sought supremum is reached on the border of the zone.

**Fig. 5.** The supremum may lie in the middle of zones or facets

To be concrete, let $X_{\leq M}$ and $X_{>M}$ be the (disjoint) sets of clocks $x$ such that $Z \subseteq (x \leq M(x))$ and $Z \subseteq (x > M(x))$, respectively. We define the relation $\preceq_Z$ as the least relation satisfying the following conditions:

- for each $x \in X_{\leq M}$, for each $y \in X$, $x \preceq_Z y$;
- for each $y \in X_{>M}$, for each $x \in X$, $x \preceq_Z y$;
- for all $x, y \in X \setminus (X_{\leq M} \cup X_{>M})$, $Z \subseteq (x - y \leq M(x) - M(y))$ implies $x \preceq_Z y$.

Note that, since $\preceq_Z$ is the least relation satisfying the above conditions, we have $x \not\preceq_Z y$ when (a) $x \in X_{>M}$ and $y \in X \setminus X_{>M}$, and when (b) $x \in X \setminus X_{\leq M}$ and $y \in X_{\leq M}$. It is then not difficult to show that $\preceq_Z$ is a preorder such that: $y \in X_{\leq M}$ and $x \preceq_Z y$ implies $x \in X_{\leq M}$, and $x \in X_{>M}$ and $x \preceq_Z y$ implies $y \in X_{>M}$.

**Lemma 12.** *Let $Y \subseteq X$ such that $Z_Y \neq \emptyset$. Then $Y$ is downward-closed for $\preceq_Z$.*

The preorder $\preceq_Z$ can be computed in polynomial time, since it only requires to check emptiness of zones, which can be done in time polynomial in $|X|$ (cubic in $|X|$ with DBMs for instance).

We recall that, if $Z$ is a zone generated in a timed automaton where only resets of clocks to 0 are allowed, for any pair of clocks $x, y$, it cannot be the case that $Z$ crosses the diagonal hyperplane of equation $x = y$.

**Proposition 13.** *If $Z$ is generated by a timed automaton, and all clocks have the same bound $M$, then $\preceq_Z$ is total.*

*Proof.* Let $x$ and $y$ be two clocks. Since $Z$ is generated by a timed automaton, it is contained either in the half-space of equation $[x \leqslant y]$, or in the one of equation $[x \geqslant y]$. By definition of $\preceq_Z$, and since $M(x) = M(y)$, the former entails $x \preceq_Z y$, and the latter $y \preceq_Z x$. Any two clocks are thus always comparable, and $\preceq_Z$ is therefore total. □

Under the assumptions of Proposition 13, there are polynomially many subsets $Y \subseteq X$ to try. Note that these assumptions are easily realized by taking $\widetilde{M} = \max_{x \in X} M(x)$ as the unique maximal constant for all the clocks. Formally, $\sqsubseteq_{\widetilde{M}}$ is an under-approximation of the exact version of $\sqsubseteq_M$. This approximation does not hinder correctness, and illustrate the trade-off between the complexity of the inclusion procedure and the number of priced zones that will be explored.

## 5 Termination of the computation

In this section we prove termination of our algorithm, by exhibiting an appropriate well-quasi-order. We fix a timed automaton $\mathcal{A}$ and a maximal-constant function $M$ (for every clock $x \in X$, the integer $M(x)$ is larger than any constant with which clock $x$ is compared in $\mathcal{A}$).

**Proposition 14.** *$\sqsubseteq$ is a preorder (or quasi-ordering).*

We now consider the "converse" preorder $\sqsupseteq$, defined over priced zones by $\mathcal{Z}' \sqsupseteq \mathcal{Z}$ iff $\mathcal{Z} \sqsubseteq \mathcal{Z}'$. We show that $\sqsupseteq$ is a *well quasi-ordering (wqo)*. Thus the relation $\sqsupseteq$ has no infinite antichain, which entails termination of Algorithm 1.

We now gather the results to exhibit a sufficient condition for $\sqsupseteq$ to be a wqo.

**Theorem 15.** *For every $\mu \in \mathbb{Z}$, $\sqsupseteq$ is a well-quasi-order on (non-empty) priced zones whose cost functions are either not lower-bounded, or lower-bounded by $\mu$.*

**Corollary 16.** *Algorithm 1 terminates on weighted timed automata, which generate priced zones with a uniform lower bound on the cost functions,*

We can argue that infinite antichains for $\sqsupseteq$ generated by a forward exploration of $\mathcal{A}$ actually corresponds to infinite paths in $\mathcal{A}$ with cost $-\infty$. While this condition can be decided (using the corner-point abstraction of [BBL08]), we do not want to check this as a preliminary step, since this is as complex as computing the optimal cost. Furthermore, symbolically, this would amount to finding a cycle of symbolic states which is both $\omega$-iterable [JR11,DHS+14] and cost-divergent; this is a non-trivial problem. We can nevertheless give simple syntactic conditions for the condition to hold: this is the case of weighted timed automata with non-negative weights (this is the class considered in [LBB+01,RLS06]); let $T_\ell$ be the minimum (resp. maximum) delay that can be delayed in $\ell$ if location $\ell$ has positive (resp. negative) cost: if along any cycle of the weighted timed automaton, the sum of the discrete weights and of each $T_\ell.\mathsf{weight}(\ell)$ is nonnegative, then the above condition will be satisfied; this last condition encompasses all the acyclic weighted timed automata, like all scheduling problems [BLR05].

## 6 Experimental Results

We have implemented a prototype, TiAMo, to test our new inclusion test. It is based on the DBM library of Uppaal (in C++),[2] which features the inclusion

[2] `http://people.cs.aau.dk/~adavid/UDBM/`

test of [RLS06]. We added our inclusion test (also in C++). This core is then wrapped in OCaml code, in which the main algorithm is written. The source code is publicly available online: `http://git.lsv.fr/colange/tiamo`.

As we have seen, termination in presence of negative costs is not guaranteed. We thus limited our experiments to models with positive costs only.

TiAMo is able to prune the state space using the best cost so far. Concretely, it would not explore states whose cost exceeds the current optimal cost. This can dramatically reduce the state space to explore, but is sound only when all costs in the model are non-negative. On such models, the user can provide a hint, a known cost to TiAMo (obtained for example by a reachability analysis, or by other independent techniques) to be used to prune the model. Moreover, TiAMo reports, during the computation, the best known cost so far. Such values are upper bounds on the sought optimum, and may be interesting to get during long computations.

A direct comparison between TiAMo and Uppaal[3] (or Uppaal-CORA[4]) is difficult: the source code of Uppaal (and Uppaal-CORA) is not open, and it is often hard to know what is precisely implemented. For instance, on the unbounded automaton of Figure 2, the algorithm described in [LBB+01,RLS06] does not terminate. Depending on the way it is queried (asking for the fastest trace, or with an `inf` query), Uppaal terminates or runs forever on this model.

In order to measure the impact of the inclusion test on the algorithm, we decided to compare the performance of TiAMo running one or the other inclusion test ($\Subset$ or $\sqsubseteq$). Our primary concern is to compare the number of (symbolic) states explored, and the number of inclusion tests performed.

We run our experiments with and without pruning activated. Deactivated pruning allows to measure the impact of the choice of the inclusion test itself. It is also more representative of the behavior that can be expected on models with negative costs, for which pruning is not sound.

*The models.* We briefly describe the models used in our experiments. The first two are case studies described on the web page of Uppaal-CORA.

The Aircraft Landing System (ALS) problem has been described in Figure 1: it consists in scheduling landings of aircrafts arriving to an airport with several runways, subject to timing constraints. Early and late arrivals induce a cost, which is to be minimized globally. We use the original version form Uppaal-CORA, with two runways and 10 aircrafts. The model has 5 clocks (one global clock, plus two per runway) and $14,000$ discrete states.

In the Energy-optimal Task-graph Scheduling (ETS) problem, several processors having different speeds and powers are to be used to perform interdependent tasks. The aim is to optimize energy consumption for performing the given set of tasks within a certain delay. The model we used for our experiments is the one described in [BFLM11, Example 3]. It has 2 clocks (one per CPU) and 55 discrete states.

[3] `http://www.uppaal.org/`
[4] `http://people.cs.aau.dk/~adavid/cora/`

| | | | # Waiting | # Passed | # stored | # tests | # succ. tests | time (s.) |
|---|---|---|---|---|---|---|---|---|
| ASL | +P | $\sqsubseteq$ | 11,820 | 4,785 | 9,324 | $3.7 \times 10^{05}$ | 13,676 | 0.3 |
| | | $\Subset$ | 32,322 | 13,036 | 26,555 | $2.9 \times 10^{06}$ | 32,263 | 0.7 |
| | -P | $\sqsubseteq$ | $1.7 \times 10^{06}$ | $1.5 \times 10^{06}$ | $6.9 \times 10^{05}$ | $8.1 \times 10^{08}$ | $1.2 \times 10^{07}$ | 312.7 |
| | | $\Subset$ | TO | TO | TO | TO | TO | TO |
| ETS | +P | $\sqsubseteq$ | 107 | 84 | 83 | 174 | 66 | 0.0 |
| | | $\Subset$ | 664 | 606 | 590 | 17,684 | 455 | 0.0 |
| VRPTW | +P | $\sqsubseteq$ | $6.0 \times 10^{05}$ | $4.8 \times 10^{05}$ | $5.6 \times 10^{05}$ | $6.2 \times 10^{06}$ | $1.7 \times 10^{05}$ | 11.3 |
| | | $\Subset$ | $1.5 \times 10^{06}$ | $1.3 \times 10^{06}$ | $1.4 \times 10^{06}$ | $9.1 \times 10^{07}$ | $7.0 \times 10^{05}$ | 27.5 |
| | -P | $\sqsubseteq$ | $1.3 \times 10^{06}$ | $1.3 \times 10^{06}$ | $1.3 \times 10^{06}$ | $2.5 \times 10^{07}$ | $7.0 \times 10^{05}$ | 23.9 |
| | | $\Subset$ | $5.8 \times 10^{06}$ | $5.8 \times 10^{06}$ | $5.4 \times 10^{06}$ | $1.1 \times 10^{09}$ | $1.9 \times 10^{06}$ | 111.2 |
| unbound. | +P | $\sqsubseteq$ | 14 | 13 | 14 | 135 | 3 | 0.0 |
| | | $\Subset$ | TO | TO | TO | TO | TO | TO |
| | -P | $\sqsubseteq$ | 14 | 14 | 14 | 135 | 3 | 0.0 |
| | | $\Subset$ | TO | TO | TO | TO | TO | TO |

**Table 1.** Experimental results

In the Vehicle Routing Problem with Time Windows (VRPTW) problem, a fleet of vehicles with limited capacity is to be scheduled to deliver goods to customers. Deliveries should respect the customers preferred time windows. We use the version downloadable from the Uppaal-CORA website, with a few syntactical modifications to account for the limits of the parser of TiAMo. The cost function used in this example is a combination of the distance travelled by the vehicles, the time to achieve deliveries, and the demand satisfied on time. The model considers 3 vehicles and 7 customers, and has 4 clocks (one for each vehicle and a global clock) and about $150,000$ discrete states.

Finally, we also ran TiAMo on the model Figure 2, to illustrate that $\sqsubseteq$ handles unbounded models. This model has two clocks and two discrete states.

*Exploration strategies.* TiAMo implements several strategies to explore the symbolic state space. We retain here only the one called SBFS, a modification of BFS based on the observation that, if $s$ subsumes $s'$, the successors of $s'$ are subsumed by successors of $s$. Successors of $s$ are thus explored first, until all successors of $s'$ in the Waiting list are subsumed. This is a very naive implementation of a strategy proposed in [HT15]. The strategy has two variants, depending on whether pruning is activated (+P) or not (-P). For the ETS problem, both yield very similar results, so we chose to only present +P.

*Experimental results.* The results are summed up in Table 1. For each model, and for different combinations of inclusion test and exploration strategy, we indicate the number of symbolic states added to the Waiting list, added to the Passed list, as well as the number of tests (successful or not) that have been performed. We also indicate the maximal size of the list Passed; although not detailed in Algorithm 1, the tool ensures that Passed remains an antichain. This minimizes the number of inclusion tests. When a new element is added to the Passed list, all elements of Passed subsumed by the new one are removed, so that the size of Passed does not necessarily increase.

The mention "TO" means that the computation does within the time bound of 120 minutes. We observe that $\sqsubseteq$ always explores fewer states than $\Subset$, for any given exploration strategy. Though this was expected (recall Remark 1), we believe the reduction is impressive. It is significant even for small models (such as ETS). The case of ALS with no pruning shows that the higher complexity of $\sqsubseteq$ can be largely compensated by the reduction in the size of the state space to explore. On the model of Figure 2, our inclusion $\sqsubseteq$ ensures termination, while $\Subset$ does not.

## 7 Conclusion

In this paper we have built over a symbolic approach to the computation of optimal cost in weighted timed automata [LBB+01,RLS06], by proposing an inclusion test between priced zones. Using that inclusion test, the forward symbolic exploration terminates and computes the optimal cost for all weighted timed automata, regardless whether clocks are bounded or not. The idea of this approach is based on recent works on pure timed automata [HSW12], where a clever inclusion test "replaces" any abstraction computation during the exploration.

We will pursue our work with extensive experimentations using our tool TiAMo. We will also look for more dedicated methods for specific application domains, like planning problems.

## References

[ACHH93] Rajeev Alur, Costas Courcoubetis, Thomas A. Henzinger, and Pei-Hsin Ho. Hybrid automata: an algorithmic approach to specification and verification of hybrid systems. In *Proc. Workshop on Hybrid Systems (1991 & 1992)*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer, 1993.

[AD90] Rajeev Alur and David L. Dill. Automata for modeling real-time systems. In *Proc. 17th International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *Lecture Notes in Computer Science*, pages 322–335. Springer, 1990.

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.

[AFM+03] Tobias Amnell, Elena Fersman, Leonid Mokrushin, Paul Pettersson, and Wang Yi. TIMES: A tool for schedulability analysis and code generation of real-time systems. In *Proc. 1st International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *Lecture Notes in Computer Science*, pages 60–72. Springer, 2003.

[ALP01] Rajeev Alur, Salvatore La Torre, and George J. Pappas. Optimal paths in weighted timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 49–62. Springer, 2001.

[ARF14] Omar Al-Bataineh, Mark Reynolds, and Tim French. Finding best and worst case execution times of systems using Difference-Bound Matrices. In

*Proc. 12th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'14)*, volume 8711 of *Lecture Notes in Computer Science*, pages 38–52. Springer, 2014.

[BBBR07] Patricia Bouyer, Thomas Brihaye, Véronique Bruyère, and Jean-François Raskin. On the optimal reachability problem. *Formal Methods in System Design*, 31(2):135–175, 2007.

[BBFL03] Gerd Behrmann, Patricia Bouyer, Emmanuel Fleury, and Kim G. Larsen. Static guard analysis in timed automata verification. In *Proc. 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *Lecture Notes in Computer Science*, pages 254–277. Springer, 2003.

[BBL08] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):2–23, 2008.

[BBLP06] Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelànek. Zone based abstractions for timed automata exploiting lower and upper bounds. *International Journal on Software Tools for Technology Transfer*, 8(3):204–215, 2006.

[BCM16] Patricia Bouyer, Maximilien Colange, and Nicolas Markey. Symbolic optimal reachability in weighted timed automata. Technical Report abs/1602.00481, CoRR, 2016. Available at http://arxiv.org/abs/1602.00481.

[BFH+01] Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim G. Larsen, Paul Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[BFLM11] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, and Nicolas Markey. Quantitative analysis of real-time systems using priced timed automata. *Communication of the ACM*, 54(9):78–87, 2011.

[BLR05] Gerd Behrmann, Kim G. Larsen, and Jacob I. Rasmussen. Optimal scheduling using priced timed automata. *ACM Sigmetrics Performancs Evaluation Review*, 32(4):34–40, 2005.

[Bou04] Patricia Bouyer. Forward analysis of updatable timed automata. *Formal Methods in System Design*, 24(3):281–320, 2004.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[BY03] Johan Bengtsson and Wang Yi. On clock difference constraints and termination in reachability analysis of timed automata. In *Proc. 5th International Conference on Formal Engineering Methods (ICFEM'03)*, volume 2885 of *Lecture Notes in Computer Science*, pages 491–503. Springer, 2003.

[DHS+14] Aakash Deshpande, Frédéric Herbreteau, B. Srivathsan, Thanh-Tung Tran, and Igor Walukiewicz. Fast detection of cycles in timed automata. Technical Report abs/1410.4509, CoRR, 2014.

[DOT+10] Andreas E. Dalsgaard, Mads Chr. Olesen, Martin Toft, René Rydhof Hansen, and Kim Guldstrand Larsen. METAMOC: Modular execution time analysis using model checking. In *Proc. 10th International Workshop on Worst-Case Execution Time Analysis (WCET'10)*, volume 15 of *OpenAccess Series in Informatics (OASIcs)*, pages 113–123. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[GELP10] Andreas Gustavsson, Andreas Ermedahl, Björn Lisper, and Paul Pettersson. Towards WCET analysis of multicore architectures using UPPAAL. In *Proc. 10th International Workshop on Worst-Case Execution Time Analysis (WCET'10)*, volume 15 of *OpenAccess Series in Informatics (OASIcs)*, pages 101–112. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

[HKPV98] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. What's decidable about hybrid automata? *Journal of Computer and System Sciences*, 57(1):94–124, 1998.

[HKSW11] Frédéric Herbreteau, Dileep Kini, B. Srivathsan, and Igor Walukiewicz. Using non-convex approximations for efficient analysis of timed automata. In *Proc. 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'11)*, volume 13 of *LIPIcs*, pages 78–89. Leibniz-Zentrum für Informatik, 2011.

[HSW12] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. In *Proc. 27th Annual Symposium on Logic in Computer Science (LICS'12)*, pages 375–384. IEEE Computer Society Press, 2012.

[HT15] Frédéric Herbreteau and Thanh-Tung Tran. Improving search order for reachability testing in timed automata. In *Proc. 13th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'15)*, pages 124–139. Springer, 2015.

[JR11] Rémi Jaubert and Pierre-Alain Reynier. Quantitative robustness analysis of flat timed automata. In *Proc. 14th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS'11)*, volume 6604 of *Lecture Notes in Computer Science*, pages 229–244. Springer, 2011.

[LBB+01] Kim G. Larsen, Gerd Behrmann, Ed Brinksma, Angskar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer, 2001.

[RLS06] Jacob I. Rasmussen, Kim G. Larsen, and K. Subramani. On using priced timed automata to achieve optimal scheduling. *Formal Methods in System Design*, 29(1):97–114, 2006.

[RS01] Jan-J Rückmann and Oliver Stein. On linear and linearized generalized semi-infinite optimization problem. *Annals of Operations Research*, 101(1-4):191–208, 2001.

# A A counter-example to the algorithm of [ARF14]

Algorithm 2 is the algorithm proposed in [ARF14] to compute optimal-time reachability in timed automata. Briefly, the approach considers a global clock, denoted by $x_i$, to measure the duration of the current run. The algorithm explores the state space in a forward manner. Newly encountered zones are compared against those already explored using the abstract inclusion test described in [HSW12], with the particularity that the global clock $x_i$ is *not* considered in this test. The clock $x_i$ keeps track of the time elapsed so far, and plays no role in the transition relation, and this is why it does not need to be used in the comparison of zones.

The abstract test between $Z$ and $Z'$ is noted $Z \subseteq closure_\alpha(Z')$. Zones are represented as sets of clock constraints, $UC$ denotes the set of constraints involving the global clock $x_i$. Thus, $Z \setminus UC$ denotes the zone $Z$ where constraints on $x_i$ are ignored (though this is not formally defined in [ARF14]).

**Algorithm 2:** Algorithm for optimal time reachability from [ARF14]

```
1  BCET ← ∞
2  PASSED ← ∅
3  WAITING ← {(ℓ0, Z0)}
4  while WAITING ≠ ∅ do
5      select (ℓ, Z) from WAITING
       // In [ARF14], Goal is the set of final states, i.e. states with
          no successors
6      if (ℓ, Z) ∈ Goal and lowerBound(Z, xi) < BCET then
7          BCET ← lowerBound(Z, xi)
8      add (ℓ, Z) to PASSED
9      forall (ℓ', Z') ∈ Post(ℓ, Z) do
           // check if lower bound of xi in the new zone is less than
              the best known BCET
10         if lowerBound(Z', xi) < BCET then
11             if (Z' \ UC) ⊈ closure_α(Z'' \ UC) for all (ℓ', Z'') ∈ PASSED then
12                 add (ℓ', Z') to WAITING
13 return BCET
```

We claim that the algorithm fails to find the BCET of the automaton shown in Figure 6. We show this by describing the run of Algorithm 2 on this automaton.

Assume that the state space is explored depth-first. The initial state is $s_0 = (\ell_0, (x = x_i = 0))$. Initially, $\mathsf{BCET} = \infty$, $\textsc{Passed} = \emptyset$ and $\textsc{Waiting} = \{s_0\}$. State $s_0$ has two successors $s_1 = (\ell_1, (x = x_i \wedge x > 1))$ and $s_2 = (\ell_2, (x = x_i \wedge x \leqslant 1))$. It holds $\mathsf{lowerBound}(s1, x_i) = 1 < \mathsf{BCET}$ and $\mathsf{lowerBound}(s2, x_i) = 2 < \mathsf{BCET}$, so both states are added to WAITING, while $s_0$ is added to PASSED.

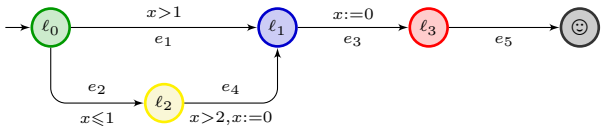



**Fig. 6.** Counter-example to Algorithm 2

If $s_2$ is the next state being picked out of WAITING, it is easy to see that its successors $s_3 = (\ell_1, (x_i > 2 \wedge x = 0))$ and $s_4 = (\ell_3, (x_i > 2 \wedge x = 0))$ are added to PASSED. At the end of this branch, state $s_5 = (\ell_f, (x_i > x+2))$ is final, and its lower value for $x_i$ is used to update the value of BCET. We now have $\mathsf{BCET} = 2$.

WAITING now contains a single state $s_1$. Its single successor is $s_6 = (\ell_3, (x_i > 1 \wedge x = 0))$. Its lower value for $x_i$ is 1, which is smaller than the current value for BCET. But $((x_i > 1 \wedge x = 0) \setminus UC) \subseteq closure_\alpha((x_i > 2 \wedge x = 0) \setminus UC)$. Indeed, "$UC$ represents the set of constraints involving the extra clock $x_i$", so $((x_i > 1 \wedge x = 0) \setminus UC) = (x = 0) = ((x_i > 2 \wedge x = 0) \setminus UC)$. Since $Z \subseteq closure_\alpha(Z)$ for every $Z$, we obtain the inclusion above. The paper further states that "it is necessary to check inclusion between zones with respect only to the automaton clocks", which confirms our understanding of the operation "$\setminus UC$".

Therefore, the state $s_6$ is *not* added to WAITING, which is now empty, and the algorithm terminates, returning the value $\mathsf{BCET} = 2$.

It is clear however that the BCET of the given automaton is 1, a value that would have been discovered by the algorithm if $s_1$ had been picked out of WAITING before $s_2$. However the paper does not mention any assumptions regarding the order of exploration that would invalidate our counter-example. We contacted the authors of [ARF14], asking to get access to the prototype implementation mentioned in the paper, but did not get an answer.

## B Details for Section 3

### B.1 Proof of Lemma 2

**Lemma 2.** *If $v \equiv_M v'$, then, for any $\ell \in L$, $Optcost_{\mathcal{A}}(\ell, v) = Optcost_{\mathcal{A}}(\ell, v')$.*

*Proof.* Pick a run $\varrho$ starting at $s = (\ell, v)$, and assume $\varrho = s \xrightarrow{t_1, e_1} s_1 \ldots \xrightarrow{t_p, e_p} s_p$. Then $\varrho' = s' \xrightarrow{t_1, e_1} s'_1 \ldots \xrightarrow{t_p, e_p} s'_p$ is also a run of $\mathcal{A}$ from $s'$ such that $s_p \equiv_M s'_p$, and $\mathsf{cost}(\varrho) = \mathsf{cost}(\varrho')$. This can be easily shown by induction on $p$. This reinforces the property stated in [BBLP06] saying that $\equiv_M$ is a strong timed bisimulation. □

### B.2 Proof of Theorem 3 (soundness of Algorithm 1)

**Theorem 3.** *When using $\sqsubseteq_M$, provided Algorithm 1 terminates, it is sound w.r.t. optimal reachability (the returned cost is the optimal one).*

*Proof.* We assume Algorithm 1 terminates. Remark that all elements added to the sets Waiting or Passed are obtained through the $\mathsf{Post}$ operation from elements of Waiting. Initially, Waiting contains the initial priced zone $(\ell_0, \mathcal{Z}_0)$, hence Waiting only contains elements from $\mathsf{Post}^*(\ell_0, \mathcal{Z}_0)$. Such elements are necessarily realized in $\mathcal{A}$, hence the computed Cost is not smaller than the actual optimal cost $C = \mathit{Optcost}_{\mathcal{A}}(s_0)$: Cost $\geq C$.

- Assume that $C$ is finite. Let $\epsilon > 0$ and $\rho = (\ell_0, v_0) \xrightarrow{t_1,e_1} (\ell_1, v_1) \dots \xrightarrow{t_p,e_p} (\ell_p, v_p)$ be an $\epsilon/3$-optimal run ($\ell_p$ is a target location): $\mathsf{cost}(\rho) \leq C + \epsilon/3$. Unless it is already the case for $\rho$, we will build a run $\widehat{\rho}$ such that all symbolic states along $\widehat{\rho}$ are added to Passed, and $\mathsf{cost}(\widehat{\rho}) \leq C + \epsilon$. In particular, if $(\widehat{\ell}, \widehat{\mathcal{Z}})$ is the last symbolic state obtained along $\widehat{\rho}$, and $\widehat{v}$ is the last valuation along $\widehat{\rho}$, $\mathit{infCost}(\widehat{\mathcal{Z}}) \leq \widehat{\zeta}(\widehat{v}) \leq \mathsf{cost}(\widehat{\rho}) \leq C + \epsilon$. Hence the computed cost Cost is bounded by $C + \epsilon$. As $\epsilon > 0$ was arbitrary, we conclude that the algorithm really computes the optimal cost.
  Assume that not all symbolic states along $\rho$ are added to Passed, and write $\rho_1$ for the longest prefix of $\rho$ along which all symbolic states are added to Passed. Write $\tau$ for the next move along $\rho$, and $\rho_2$ for the suffix after $\rho_1\tau$. Let $(\ell, v)$ be the configuration at the end of $\rho_1\tau$, and $(\ell, \mathcal{Z})$ be the corresponding symbolic state (which we know, by hypothesis, has not been added to Passed). Hence, at the time when $(\ell, \mathcal{Z})$ is handled, there exists some $(\ell, \mathcal{Z}')$ in Passed, such that $\mathcal{Z} \sqsubseteq_M \mathcal{Z}'$. Thus, there is some valuation $v' \in Z'$ such that $v' \equiv_M v$ and $\zeta'(v') \leq \zeta(v) + \epsilon/3$. Since $(\ell, \mathcal{Z}') \in$ Passed, there is a run $\rho'$ from $(\ell_0, v_0)$ to $(\ell, v')$ such that (a) all symbolic states along $\rho'$ have been added to Passed, and (b) $\mathsf{cost}(\rho') \leq \zeta'(v') + \epsilon/3 \leq \zeta(v) + 2\epsilon/3$. Now, since $v' \equiv_M v$, thanks to the proof of Lemma 2, there is a run $\rho''$ from $(\ell, v')$ to the target state, with the same length as $\rho_2$, such that $\mathsf{cost}(\rho'') = \mathsf{cost}(\rho_2)$. The new run $\widetilde{\rho} = \rho' \cdot \rho''$ reaches the target location and its cost is: $\mathsf{cost}(\widetilde{\rho}) = \mathsf{cost}(\rho') + \mathsf{cost}(\rho'') \leq \zeta(v) + 2\epsilon/3 + \mathsf{cost}(\rho_2) \leq \mathsf{cost}(\rho_1\tau) + 2\epsilon/3 + \mathsf{cost}(\rho_2) = \mathsf{cost}(\rho) + 2\epsilon/3 \leq C + \epsilon$. We repeat the argument on $\rho_2$, which is smaller than the initial run $\rho$, and inductively we are able to build the expected run $\widehat{\rho}$ that we described earlier.
- Assume that $C = \infty$. This means that the target location is not reachable, and the algorithm never updates variable Cost; its value is therefore $\infty$ as well.
- Assume that $C = -\infty$. A reasoning similar to the first case can be done, with a slight adaptation: we fix some arbitrary bound $K$ and $\epsilon > 0$, and we pick the run $\rho$ such that $\mathsf{cost}(\rho) \leq K$. The rest is identical: we will build a run $\widehat{\rho}$ such that all symbolic states along $\widehat{\rho}$ are added to Passed, and $\mathsf{cost}(\widehat{\rho}) \leq \mathsf{cost}(\rho) + \epsilon$. □

# C Details for Section 4

## C.1 Formulation of the optimization problem

**Lemma 4.** $\mathcal{Z} \sqsubseteq \mathcal{Z}' \iff \sup_{v \in Z} \inf_{\substack{v' \in Z' \\ v' \equiv v}} \zeta'(v') - \zeta(v) \leqslant 0.$

*Proof.* ($\Rightarrow$) By definition of $\sqsubseteq$, for every $\epsilon > 0$ and $v \in Z$, there is some $v' \in Z'$ such that $v \equiv v'$ and $\zeta'(v') \leqslant \zeta(v) + \epsilon$. The existence of such a $v'$ guarantees that the infimum is not taken over an empty set, and hence is not $+\infty$. Thus, for every $v \in Z$, $\inf_{\substack{v' \in Z' \\ v' \equiv v}} \zeta'(v') - \zeta(v) \leqslant 0$, from which the result is straightforward. Note that if $Z$ is empty, then the supremum is $-\infty$, and the result still holds.

($\Leftarrow$) Let $v \in Z$ and $\epsilon > 0$. $\inf_{\substack{v' \in Z' \\ v' \equiv v}} \zeta'(v') - \zeta(v) \leqslant 0$, which guarantees the existence of $v' \in Z'$ such that $v' \equiv v$ (otherwise this infimum, and the above supremum also, would be $+\infty$). Furthermore, $v'$ can be chosen so that $\zeta'(v') - \zeta(v) \leqslant \epsilon$. □

**Lemma C.1.** *Let $Y \subseteq X$, $v \in Z_Y$ and $v' \in Z'$. Then $v \equiv v'$ if, and only if, $v' \in Z'_Y$ and $\pi_Y(v) = \pi_Y(v')$.*

*Proof.* If $v \equiv v'$, then for all $x \in X$, we have $v(x) = v'(x)$ or both $v(x)$ and $v'(x)$ are larger than $M$. As $v \in Z_Y$, it holds $v(x) \leqslant M$ for $x \in Y$ and $v(x) > M$ for $x \notin Y$. Hence $v'(x) = v(x)$ for $x \in Y$ and $v'(x) > M$ for $x \notin Y$. It follows $v' \in Z'_Y$ and $\pi_Y(v) = \pi_Y(v')$.

Conversely, if $v' \in Z'_Y$ and $\pi_Y(v) = \pi_Y(v')$, then $v'(x) > M$ if $x \notin Y$, and $v(x) = v'(x)$ for $x \in Y$. Hence $v \equiv v'$. □

Lemma C.1 entails the following two lemmas:

**Lemma 5.** *The following two properties are equivalent:*

*(i) for every $v \in Z$, there is $v' \in Z'$ such that $v' \equiv v$*
*(ii) for every $Y \subseteq X$, $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$.*

*Proof.* Assume *(i)* holds, and pick $Y \subseteq X$. If $Z_Y$ is empty, the inclusion $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$ is trivial. Otherwise, pick $v \in Z_Y$. Applying *(i)*, there exists $v' \in Z'$ s.t. $v \equiv v'$. From Lemma C.1, $v' \in Z'_Y$ and $\pi_Y(v) = \pi_Y(v')$. Hence $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$.

Conversely, assuming *(ii)*, pick $v \in Z$. Then for some (unique) $Y$, we have $v \in Z_Y$, so that there exists $v' \in Z'_Y$ for which $\pi_Y(v) = \pi_Y(v')$. Applying Lemma C.1, it also holds $v' \equiv v$, which concludes the proof. □

**Lemma 6.** $\displaystyle\sup_{v \in Z} \inf_{\substack{v' \in Z' \\ v' \equiv v}} \zeta'(v') - \zeta(v) = \max_{Y \subseteq X} \sup_{v \in Z_Y} \inf_{\substack{v' \in Z'_Y \\ v' \equiv v}} \zeta'(v') - \zeta(v).$

*Proof.* It suffices to notice that when $v \in Z_Y$, the sets $\{v \in Z \mid v \equiv v'\}$ and $\{v \in Z_Y \mid v \equiv v'\}$ coincide. □

### C.2 Proof of Theorem 8

**Theorem 8.** *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two non-empty priced zones, and let $Y \subseteq X$ be such that $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$ and $\zeta$ and $\zeta'$ are lower-bounded*

*on $Z_Y$ and $Z'_Y$ respectively. Then we can compute finite sets $\mathcal{K}_Y$ and $\mathcal{K}'_Y$ of zones over $Y$, and affine functions $\zeta_F$ and $\zeta'_{F'}$ for every $F \in \mathcal{K}_Y$ and $F' \in \mathcal{K}'_Y$ s.t.:*

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{F \in \mathcal{K}_Y} \max_{F' \in \mathcal{K}'_Y} \sup_{u \in F \cap F'} \zeta'_{F'}(u) - \zeta_F(u). \tag{1}$$

For the rest of this section, we fix $Y \subseteq X$ such that $Z_Y \neq \emptyset$ (otherwise $S(\mathcal{Z}, \mathcal{Z}', Y) = -\infty$, and we can take $\mathcal{K} = \mathcal{K}' = \emptyset$). We assume that $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$ (otherwise by Lemma 5, for some $v \in Z_Y$, the set $\{v' \in Z'_Y \mid v' \equiv v\}$ would be empty, and $S(\mathcal{Z}, \mathcal{Z}', Y) = +\infty$). In this section, we furthermore require that $\zeta$ and $\zeta'$ are bounded from below over $Z_Y$ and $Z'_Y$, respectively. Under these assumptions, we describe an algorithm to compute $S(\mathcal{Z}, \mathcal{Z}', Y)$.

First remark that, for $v \in Z_Y$ and $v' \in Z'_Y$, it holds $v \equiv v'$ if, and only if, $\pi_Y(v) = \pi_Y(v')$. In this subsection, we frequently use this characterization, which can be extended by continuity to the boundaries of $Z_Y$ and $Z'_Y$.

*Facets and decomposition using facets.* We recall the notion of facets, originally defined in [LBB+01]. Whenever $x \bowtie n$ (resp. $x - y \bowtie m$) is a tight constraint defining a *closed* zone $W$, the strengthened zone $W \wedge (x = n)$ (resp. $W \wedge (x - y = m)$) is called a *simple facet* (w.r.t. $x$) of $W$. A simple facet of a zone is itself a zone, hence we can look at its simple facets as well, and so on iteratively we call them the *facets* of $W$, and we say that $W$ is a facet of $W$ as well. Facets of a non-closed zone are those of its closure.

We note $\mathsf{LF}_x(W)$ (resp. $\mathsf{UF}_x(W)$) the set of *lower simple facets w.r.t $x$* (resp. *upper simple facets w.r.t. $x$*) of $W$, i.e. those obtained from a lower bound $x \geq n$ or $x - y \geqslant m$ (resp. from an upper bound $x \leqslant n$ or $x - y \leqslant m$). We write $e_x$ for the unit vector in the direction of the $x$-axis. If $F$ is a lower (resp. upper) simple facet of $W$ w.r.t. $x$, and $v_F \in F$, we note $W^F(v_F)$ the set $\{v \in W \mid \exists \lambda \in \mathbb{R}_{\geqslant 0}.\ v - \lambda.e_x = v_F\}$ (resp. $\{v \in W \mid \exists \lambda \in \mathbb{R}_{\geqslant 0}.\ v + \lambda.e_x = v_F\}$), the set of points of $W$ that lie above (resp. below) $v_F$ in the direction of the $x$-axis. Similarly, we note $W^F = \bigcup_{v_F \in F} W^F(v_F)$, the set of points of $W$ that lie above (resp. below) $F$ in the direction of the $x$-axis. Note that it holds $\bigcup_{F \in \mathsf{LF}_x(W)} W^F = W$; in case $W$ contains no half-line of the form $v + \mathbb{R}_{\geqslant 0}.e_x$ for any $v \in W$, then we also have $\bigcup_{F \in \mathsf{UF}_x(W)} W^F = W$.

*Reduction of one variable in the optimization problem.* We recall that $\zeta$ and $\zeta'$ are affine functions over $Z$ and $Z'$ respectively. We note $\zeta_x = \frac{\partial}{\partial x}\zeta$ and $\zeta'_x = \frac{\partial}{\partial x}\zeta'$ for every $x \in X$. We note $c$ the constant term of $\zeta$.

Let $x \in X \setminus Y$. We define $\mathcal{F}_x(Z_Y)$ as $\mathsf{LF}_x(Z_Y)$ if $\zeta_x \geqslant 0$, and as $\mathsf{UF}_x(Z_Y)$ if $\zeta_x < 0$. In the former case, $Z_Y = \bigcup_{F \in \mathcal{F}_x(Z_Y)} Z_Y^F$ always holds. In the latter case, recall that $\zeta$ is bounded from below on $Z$, so $Z$ contains no half-line $v + \mathbb{R}_{\geqslant 0}.e_x$ for any $v \in Z$, otherwise $\zeta$ would take arbitrarily low values along this half-line. This ensures that $Z_Y = \bigcup_{F \in \mathcal{F}_x(Z_Y)} Z_Y^F$ in both cases.

Now, we have:

$$\begin{aligned}
S(\mathcal{Z},\mathcal{Z}',Y) &= \sup_{v\in Z_Y} \inf_{\substack{v'\in Z'_Y\\ v'\equiv_M v}} \zeta'(v')-\zeta(v)\\
&= \sup_{u\in\pi_Y(Z_Y)} \sup_{\substack{v\in Z_Y\\ \pi_Y(v)=u}} \inf_{\substack{v'\in Z'_Y\\ \pi_Y(v')=u}} (\zeta'(v')-\zeta(v))\\
&= \sup_{u\in\pi_Y(Z_Y)} \Big[\Big(\inf_{\substack{v'\in Z'_Y\\ \pi_Y(v')=u}} \zeta'(v')\Big)+\Big(\sup_{\substack{v\in Z_Y\\ \pi_Y(v)=u}} (-\zeta(v))\Big)\Big]\\
&\qquad\qquad \text{(because the constraint on } v' \text{ is now independent of } v\\
&\qquad\qquad\qquad \text{and both } \zeta \text{ and } \zeta' \text{ are bounded from below)}\\
&= \sup_{u\in\pi_Y(Z_Y)} \Big[\Big(\inf_{\substack{v'\in Z'_Y\\ \pi_Y(v')=u}} \zeta'(v')\Big)-\Big(\inf_{\substack{v\in Z_Y\\ \pi_Y(v)=u}} \zeta(v)\Big)\Big] \qquad (2)\\
&= \max_{F\in\mathcal{F}_x(Z_Y)} \sup_{u\in\pi_Y(Z_Y^F)} \Big[\Big(\inf_{\substack{v'\in Z'_Y\\ \pi_Y(v')=u}} \zeta'(v')\Big)-\Big(\inf_{\substack{v\in Z_Y\\ \pi_Y(v)=u}} \zeta(v)\Big)\Big].
\end{aligned}$$

For $F\in\mathcal{F}_x(Z_Y)$, write $\gamma_F$ for the extra constraint defining $F$ in $\overline{Z_Y}$ (of the form $x=n$ or $x-y=m$). For $v\in F$, the value of $v(x)$ is entirely determined by the values $(v(y))_{y\neq x}$ and by the constraint $\gamma_F$: $v(x)=n$ or $v(x)=v(y)+m$. Thus, $\pi_{X\setminus\{x\}}$ is a bijection between $F$ and $\pi_{X\setminus\{x\}}(F)$. For $w\in\pi_{X\setminus\{x\}}(F)$, we note $\iota_F(w)$ the unique point of $F$ such that $w=\pi_{X\setminus\{x\}}(\iota_F(w))$.

We define a new cost function $\zeta_F$ on $\pi_{X\setminus\{x\}}(F)$ by $\zeta_F(w)=\zeta(\iota_F(w))$. We remark that $\zeta_F(w)=\zeta_x\cdot(\iota_F(w))(x)+\sum_{y\in X\setminus\{x\}}\zeta_y\cdot w(y)+c$. Since $(\iota_F(w))(x)$ is an affine function of $w$, so is $\zeta_F$.

By definition of $\mathcal{F}_x(Z_Y)$, we have $\zeta_F(w)=\inf_{v\in Z_Y^F(\iota_F(w))}\zeta(v)$. Now, remark that $Z_Y^F(\iota_F(w))=\{v\in Z_Y\mid \pi_{X\setminus\{x\}}(v)=w\}$, so that $\zeta_F(w)=\inf_{v\in Z_Y,\ \pi_{X\setminus\{x\}}(v)=w}\zeta(v)$. For $u\in\pi_Y(Z_Y^F)$, we can then write:

$$\begin{aligned}
\inf_{\substack{v\in Z_Y\\ \pi_Y(v)=u}} \zeta(v) &= \inf_{\substack{w\in\pi_{X\setminus\{x\}}(F)\\ \pi_Y(w)=u}} \inf_{\substack{v\in Z_Y\\ \pi_{X\setminus\{x\}}(v)=w}} \zeta(v)\\
&= \inf_{\substack{w\in\pi_{X\setminus\{x\}}(F)\\ \pi_Y(w)=u}} \zeta_F(w)
\end{aligned}$$

The above being true for every $F\in\mathcal{F}_x(Z_Y)$, we get:

$$S(\mathcal{Z},\mathcal{Z}',Y)=\max_{F\in\mathcal{F}_x(Z_Y)}\sup_{u\in\pi_Y(Z_Y^F)}\Big[\Big(\inf_{\substack{v'\in Z'_Y\\ \pi_Y(v')=u}}\zeta'(v')\Big)-\Big(\inf_{\substack{w\in\pi_{X\setminus\{x\}}(F)\\ \pi_Y(w)=u}}\zeta_F(w)\Big)\Big].$$

Finally, note that $\overline{\pi_Y(F)}=\overline{\pi_Y(\pi_{X\setminus\{x\}}(F))}=\overline{\pi_Y([\pi_{X\setminus\{x\}}(F)]_Y)}=\overline{\pi_Y(Z_Y^F)}$ (because $x\in X\setminus Y$ and $F\subseteq\overline{Z_Y}$), so that we conclude:

$$S(\mathcal{Z},\mathcal{Z}',Y)=\max_{F\in\mathcal{F}_x(Z_Y)}\sup_{u\in\pi_Y([\pi_{X\setminus\{x\}}(F)]_Y)}\Big[\Big(\inf_{\substack{v'\in Z'_Y\\ \pi_Y(v')=u}}\zeta'(v')\Big)-\Big(\inf_{\substack{w\in\pi_{X\setminus\{x\}}(F)\\ \pi_Y(w)=u}}\zeta_F(w)\Big)\Big].$$

This can be rewritten as[5]

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{F \in \mathcal{F}_x(Z_Y)} S((\pi_{X\setminus\{x\}}(F), \zeta_F), \mathcal{Z}', Y)$$

where $(\pi_{X\setminus\{x\}}(F), \zeta_F)$ is a new priced zone.

*Iteration of the construction.* For each sequence $(x_i)_{1\leq i\leq p}$ of distinct elements of $X \setminus Y$, we define $\mathcal{F}_{x_1,\ldots,x_p}(Z_Y)$ inductively as follows:

$$\mathcal{F}_{x_1,\ldots,x_p}(Z_Y) = \bigcup_{F \in \mathcal{F}_{x_1,\ldots,x_{p-1}}(Z_Y)} \mathcal{F}_{x_p}(\pi_{X\setminus\{x_1,\ldots,x_{p-1}\}}(F)).$$

Note that $F \in \mathcal{F}_{x_1,\ldots,x_{p-1}}(Z_Y)$ is a zone over $X \setminus \{x_1, \ldots, x_{p-1}\} \supseteq Y$.

By iteratively applying the above construction, we can compute, for each $F \in \mathcal{F}_{x_1,\ldots,x_{p-1}}(Z_Y)$, an affine function $\zeta_F$ such that

$$\forall w \in \overline{\pi_{X\setminus\{x_1,\ldots,x_p\}}(F)}. \quad \zeta_F(w) = \inf_{\substack{v \in Z_Y \\ \pi_{X\setminus\{x_1,\ldots,x_p\}}(v) = w}} \zeta(v). \tag{3}$$

The previous analysis also entails:

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{F \in \mathcal{F}_{x_1,\ldots,x_p}(Z_Y)} \sup_{u \in \pi_Y(F)} \inf_{\substack{v' \in Z'_Y \\ \pi_Y(v') = u}} \zeta'(v') - \inf_{\substack{w \in \pi_{X\setminus\{x_1,\ldots,x_p\}}(F) \\ \pi_Y(w) = u}} \zeta_F(w).$$

Then, when $\{x_1, \ldots, x_p\} = X \setminus Y$, we have $X \setminus \{x_1, \ldots, x_p\} = Y$, so that

$$\forall u \in \pi_Y(F). \quad \inf_{\substack{w \in \pi_{X\setminus\{x_1,\ldots,x_p\}}(F) \\ \pi_Y(w) = u}} \zeta_F(w) = \inf_{\substack{w \in \pi_Y(F) \\ \pi_Y(w) = u}} \zeta_F(w) = \zeta_F(u).$$

It follows:

$$\begin{aligned} S(\mathcal{Z}, \mathcal{Z}', Y) &= \max_{F \in \mathcal{F}_{X\setminus Y}(Z_Y)} \sup_{u \in \pi_Y(F)} \inf_{\substack{v' \in Z'_Y \\ \pi_Y(v') = u}} \zeta'(v') - \zeta_F(u) \\ &= \max_{F \in \pi_Y(\mathcal{F}_{X\setminus Y}(Z_Y))} \sup_{u \in F} \inf_{\substack{v' \in Z'_Y \\ \pi_Y(v') = u}} \zeta'(v') - \zeta_F(u) \end{aligned}$$

where we write $\mathcal{F}_{X\setminus Y}(Z_Y)$ for $\mathcal{F}_{x_1,\ldots,x_p}(Z_Y)$. Elements thereof are facets of zones over $Y \cup \{x_p\}$, and elements of $\pi_Y(\mathcal{F}_{X\setminus Y}(Z_Y))$ are thus zones over $Y$.

A similar construction of facets for $Z'_Y$ can be performed: for $x \in X \setminus Y$, we consider the set $\mathcal{F}'_x(Z'_Y)$ of upper or lower (depending on $\zeta'_x$) facets of $Z'_Y$ w.r.t. $x$. For each $F' \in \mathcal{F}'_x(Z'_Y)$, we define the affine function $\zeta'_{F'}$ over $\pi_{X\setminus\{x\}}(F')$ using the "inverse" projection on $F'$. By construction of $\mathcal{F}'_X(Z'_Y)$, it satisfies

[5] Notice that the definition of $S(\mathcal{Z}, \mathcal{Z}', Y)$ requires $\mathcal{Z}$ and $\mathcal{Z}'$ to have the same dimension (because $\equiv$ does). However, using Eq. (2), we can extend the definition to any zones involving at least the clocks of $Y$.

$\zeta'_{F'}(w') = \inf_{v' \in Z'_Y,\ \pi_{X\setminus\{x\}}(v')=w'} \zeta'(v')$ for all $w' \in \pi_{X\setminus\{x\}}(F')$. The situation is similar as previously, and we end up with

$$\forall u \in \pi_Y(Z'^{F'}_Y).\quad \inf_{\substack{v' \in Z'_Y \\ \pi_Y(v')=u}} \zeta'(v') = \inf_{\substack{w' \in \pi_{X\setminus\{x\}}(F') \\ \pi_Y(w')=u}} \zeta'_{F'}(w').$$

It follows:

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{F \in \pi_Y(\mathcal{F}_{X\setminus Y}(Z_Y))} \max_{F' \in \mathcal{F}_x(Z'_Y)} \sup_{u \in F \cap \pi_Y(Z'^{F'}_Y)} \inf_{\substack{w' \in \pi_{X\setminus\{x\}}(F') \\ \pi_Y(w')=u}} \zeta'_{F'}(w') - \zeta_F(u).$$

By iteratively applying the same transformation for all clocks in $X \setminus Y$, and using the same notations as above, we finally get

$$S(\mathcal{Z}, \mathcal{Z}', Y) = \max_{F \in \pi_Y(\mathcal{F}_{X\setminus Y}(Z_Y))} \max_{F' \in \pi_Y(\mathcal{F}'_{X\setminus Y}(Z'_Y))} \sup_{u \in F \cap F'} \zeta'_{F'}(u) - \zeta_F(u). \quad (4)$$

This concludes the proof of Theorem 8.

### C.3 Effectiveness of $\sqsubseteq$

**Lemma 10.** *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two non-empty priced zones.*

- *If $\zeta$ is not lower-bounded on $Z$ but $\zeta'$ is lower-bounded on $Z'$, then $\mathcal{Z} \not\sqsubseteq \mathcal{Z}'$.*
- *Let $Y \subseteq X$ such that $\pi_Y(Z_Y) \subseteq \pi_Y(Z'_Y)$. If $\zeta'$ is not lower-bounded on $Z'_Y$, then $\mathcal{Z}_Y \sqsubseteq \mathcal{Z}'_Y$.*

*Proof.* Assume $\zeta$ is not lower-bounded on $Z$ but $\zeta'$ is lower-bounded on $Z'$, then it is easy to see that $\mathcal{Z} \not\sqsubseteq \mathcal{Z}'$ (picking $v \in Z$ such that $\zeta(v) < \inf_{v' \in Z'} \zeta'(v') - 1$, it is easy to see that we cannot find any $v' \in Z'_Y$, $v' \equiv v$, such that $\zeta'(v') \leq \zeta(v) + 1$).

Now assume that $\zeta'$ is not lower-bounded on $Z'$. Let $Y \subseteq X$. If $\zeta'$ is lower-bounded on $Z'_Y$, $Z_Y \subseteq Z'_Y$ is decided with Theorem 8 or with the result above (depending on whether $\zeta$ is lower-bounded or not on $Z_Y$).

Otherwise, there exists a direction $d$ and a valuation $v'_0 \in Z'_Y$ such that the half-line $v'_0 + \mathbb{R}_{\geqslant 0} \cdot d$ is included in $Z'_Y$, and $\zeta'$ decreases strictly along this half-line. We have $d \cdot e_y = 0$ for every $y \in Y$ (otherwise the whole half-line would not be in $Z'_Y$), and $d \cdot e_x \geqslant 0$ for every $x \in X \setminus Y$. Furthermore, as explained below, the half-lines $v' + \mathbb{R}_{\geqslant 0} \cdot d$ are all included in $Z'_Y$, for and $v' \in Z'_Y$. For $v \in Z_Y$, there exists $v' \in Z'_Y$ such that $v \equiv_M v'$. But $d$ is such that the half-line $v' + \mathbb{R}_{\geqslant 0} \cdot d$ is included in $[v]_M$. The affine function $\zeta'$ takes arbitrarily low values along this half-line, so we can always find $v' \equiv_M v$ with $\zeta'(v') \leqslant \zeta(v)$. This holds even if $\zeta$ is not lower-bounded on $Z_Y$.

We now prove our claim on half-lines: more precisely, we show that for any $v$ and $v'$ in some zone $Z$, and any vector $d$ such that $v + \mathbb{R}_{\geqslant 0} \cdot d \subseteq Z$, it holds $v' + \mathbb{R}_{\geqslant 0} \cdot d \subseteq Z$.

We first characterize the directions in which $Z$ is unbounded. Let $C$ be the set of clocks that are unbounded in $Z$, i.e., those clocks $x$ for which the entry $(x, 0)$

of the normalized DBM of $Z$ is $+\infty$. Let $D$ be the set of vectors $\sum_{x\in C} \alpha_x \cdot e_x$, in which $\alpha_x \geqslant 0$ for all $x \in C$, and $\alpha_x \leqslant \alpha_y$ in case the entry $(x, y)$ of the normalized DBM of $Z$ is finite. Now, pick $v \in Z$, and $d \in D$. Then assume that for some $\lambda > 0$, $w = v + \lambda \cdot d$ is not in $Z$: in particular, for some $\lambda_0 > 0$, $v + \lambda_0 \cdot d$ is on the border of $Z$, hence it satisfies some constraint $x = n$ or $x - y = n$, for which $x \leqslant n$ or $x - y \bowtie n$ is a tight constraint defining $Z$. The former case ($x = n$) is impossible, since the fact that the value of $x$ changes in direction $d$ indicates that $x \in C$, so that $x$ is unbounded. Now, assume that $v + \lambda_0 \cdot d$ satisfies $x - y = n$ for some tight constraint $x - y \bowtie n$ defining $Z$ ($x - y \leqslant n$, say). Necessarily at least one of $x$ and $y$ must be in $C$ (otherwise the value of $x - y$ would not change in direction $d$). In case only one of them were in $C$, since in $Z$ it holds $x \leqslant y + n$, it must be $y$. But then in the direction of $d$, the value of $x - y$ would decrease, so it cannot be the case that $x - y \leqslant n$ gets violated. Hence both $x$ and $y$ are in $C$, and since $Z$ contains constraint $x - y \leqslant n$, it holds $\alpha_x \leqslant \alpha_y$. But then again, $x - y$ would decrease in the direction of $d$, so it cannot be the case that $x - y \leqslant n$ gets violated.

Conversely, if for some direction $d = \sum_{x\in X} \alpha_x \cdot e_x$ it holds $v + \mathbb{R}_{\geqslant 0} \cdot d \subseteq Z$, then any clock $x$ for which the vector $e_x$ has a non-zero coefficient in $d$ is unbounded. Moreover, if for two clocks $x$ and $y$ the coefficients in $d$ are such that $\alpha_x > \alpha_y$, then the difference $x - y$ cannot be bounded in $Z$. Hence if $x - y$ is bounded, we must have $\alpha_x \leqslant \alpha_y$, which entails that $d \in D$. Thus $D$ characterizes all directions in which $Z$ is infinite.

Now, pick a point $v$ in $Z$, hence satisfying the constraints in $Z$, and a direction $d$ in $D$. Assume that $v + \lambda \cdot d$ is outsize $Z$, which means that there is some value $\lambda_0 > 0$ such that $v + \lambda_0 \cdot d$ is outside of $Z$. Hence there is some constraint, $x \bowtie n$ or $x - y \bowtie n$, defining $Z$ that holds true of $v$ and false of $v + \lambda_0 \cdot d$. Applying the same arguments as above, we can prove that this leads to a contradiction. □

### C.4 Complements for Section 4.3

**Lemma 12.** *Let $Y \subseteq X$ such that $Z_Y \neq \emptyset$. Then $Y$ is downward-closed for $\preceq_Z$.*

*Proof.* First notice that $X_{\leq M} \subseteq Y$ and $Y \cap X_{>M} = \emptyset$, since $Z_Y \neq \emptyset$. Towards a contradiction, assume $Y$ is not downward-closed for $\preceq_Z$. There is $y \in Y$ and $x \in X \setminus Y$ such that $x \preceq_Z y$. Clock $y$ cannot belong to $X_{\leq M}$, as otherwise $x$ would also be an element of $X_{\leq M}$, thus of $Y$. Also, $y$ cannot belong to $X_{>M}$, as otherwise it would not be in $Y$. Hence, $Z \cap (y \leq M(y)) \neq \emptyset$ and $Z \cap (y > M(y)) \neq \emptyset$. So it must be the case that $Z \subseteq (x - y \leq M(x) - M(y))$. Now $\emptyset \neq Z_Y \subseteq (y \leq M(y)) \cap (x > M(x))$, which yields a contradiction with $Z_Y \subseteq Z \subseteq (x - y \leq M(x) - M(y))$. □

## D Details for Section 5

**Proposition 14.** *$\sqsubseteq$ is a preorder (or quasi-ordering).*

*Proof.* $\sqsubseteq$ is obviously reflexive; we prove transitivity. Assume $(Z,\zeta) \sqsubseteq (Z',\zeta')$ and $(Z',\zeta') \sqsubseteq (Z'',\zeta'')$. Let $v \in Z$ and $\epsilon > 0$. Since $(Z,\zeta) \sqsubseteq (Z',\zeta')$, there exists $v' \in Z'$ such that $v \equiv v'$ and $\zeta'(v') \leqslant \zeta(v) + \epsilon/2$. Since $(Z',\zeta') \sqsubseteq (Z'',\zeta'')$, there exists $v'' \in Z''$ such that $v'' \equiv v'$ and $\zeta''(v'') \leqslant \zeta'(v') + \epsilon/2$. Thus, $v'' \equiv v$ and $\zeta''(v'') \leqslant \zeta(v) + \epsilon$. □

**Theorem 15.** *For every $\mu \in \mathbb{Z}$, $\sqsupseteq$ is a well-quasi-order on (non-empty) priced zones whose cost functions are either not lower-bounded, or lower-bounded by $\mu$.*

*Proof.* Fix $Y \subseteq X$; we first show that $\sqsupseteq$ is a well-quasi-order on $M$-bounded-on-$Y$ priced zones. Pick an infinite sequence of $M$-bounded-on-$Y$ priced zones $(\mathcal{Z}_i)_{i\geq 0}$, with $\mathcal{Z}_i = (Z_i, \zeta_i)$. $\pi_Y(Z)$ is a zone, whose corners have integer coordinates, and is included in the bounded space $\prod_{x\in Y}[0, M(y)]$. Thus, there are finitely many possible values for $\pi_Y(Z)$. We cant thus assume without loss of generality that for every $i \geq 0$, $\pi_Y(Z_{i+1}) \subseteq \pi_Y(Z_i)$.

From Lemma 10, if there is $i \geq 0$ such that $\zeta_i$ is not lower-bounded on $Z_i$, then for every $j \geq i$, $\mathcal{Z}_j \sqsubseteq \mathcal{Z}_i$. We now assume that all $\zeta_i$ are lower-bounded on $Z_i$ by $\mu$.

With every $i$, we can associate the tuple

$$\kappa_i = \Big( \min_{\substack{v\in\overline{Z_i}\\ \pi_Y(v)=u}} \zeta_i(v) \Big)_{\substack{u\in\prod_{y\in Y}[0,M(y)]\\ u \text{ has integral coord.}}}$$

Each such coordinate has integral value larger than or equal to $\mu$, or $+\infty$ (in case the minimum is taken on an empty set). There are finitely many $u \in \prod_{y\in Y}[0, M(y)]$ with integral coordinates, hence the order on such tuples is a well-quasi-order (by Higman's lemma [Hig52]): there exists $i < j$ such that $\kappa_i \leq \kappa_j$ (pointwise order). Now, using Corollary 9, we get that $\mathcal{Z}_j \sqsubseteq \mathcal{Z}_i$, which implies the expected result for $M$-bounded-on-$Y$ priced zones.

By application of Higman's lemma again and of Corollary 7, we get that $\sqsupseteq$ is a well-quasi-order over priced zones which are either not lower-bounded or lower-bounded by $\mu$. □

### D.1 When does a weighted timed automaton generate only priced zones either non lower-bounded or lower-bounded by some fixed $\mu$?

We will characterize such weighted timed automata using an extension of the corner-point abstraction defined in [BBL08] for (clock-)bounded weighted timed automata.

Let $\mathcal{A} = (X, L, \ell_0, \mathsf{Goal}, E, \mathsf{weight})$ be a weighted timed automaton. Let $\mathcal{R}$ be the set of regions for $\mathcal{A}$ with regards to maximal constants $M$. Given $R \in \mathcal{R}$, it is $M$-bounded on $Y$ for a maximal set $Y$, and a corner of $R$ is a point of $\mathbb{N}^Y$ which belongs to $\overline{\pi_Y(R)}$: somehow, once a clock is larger than the maximal constant, then it becomes inactive, hence we forget about those clocks when recording the corners; this allows to have finitely many reachable corners! We write $\mathcal{R}^\star$ for the set of pairs $(R, \alpha)$, where $R \in \mathcal{R}$ and $\alpha$ is a corner of $R$.

We construct corner-point abstraction $\mathcal{A}_{\mathsf{cp}}$ of $\mathcal{A}$ as the following weighted graph: its set of states is $L \times \mathcal{R}^\star$, and its set of transitions is defined by:

- There are transitions $e = (\ell, (R, \alpha)) \to (\ell, (R, \alpha'))$ where $\alpha$ and $\alpha'$ are distinct corners of $R$ and $\alpha'$ is the time successor of $\alpha$ (in which case, $\alpha' = \alpha + 1$). We then set $\mathsf{weight}(e') = \mathsf{weight}(\ell)$ (intuitively the delay between the corner-points $\alpha$ and $\alpha'$ is one time unit).
- There are transitions $e = (\ell, (R, \alpha)) \to (\ell, (R, \alpha))$ when $R$ is $M$-bounded on $X$ (hence $R$ has a single corner), and $\mathsf{weight}(e) = \mathsf{weight}(\ell)$
- There are transitions $e = (\ell, (R, \alpha)) \to (\ell, (R', \alpha))$ where $R'$ is the time successor region of $R$ and $\alpha$ is a corner-point associated with both $R$ and $R'$. We then set $\mathsf{weight}(e) = 0$(intuitively, there is no delay between the corner-point of the two distinct regions).
- If $e = \ell \xrightarrow{g,Y} \ell'$ is a transition of $\mathcal{A}$, there will be transitions $e' = (\ell, (R, \alpha)) \to (\ell', (R', \alpha'))$ in $\mathcal{A}_{\mathsf{cp}}$ with $R \subseteq g$, $R' = [Y \leftarrow 0]R$, $\alpha$ corner-point associated with $R$, $\alpha'$ corner associated with $R'$ and $\alpha' = [Y \leftarrow 0]\alpha$. We then set $\mathsf{weight}(e') = \mathsf{weight}(e)$.

The first three types of transitions are delay transitions whereas the last are discrete transitions. It is just an extension of [BBL08, Prop. 3, Prop. 5] to get that optimal cost in $\mathcal{A}$ coincides with optimal cost in $\mathcal{A}_{\mathsf{cp}}$.

We will use this corner-point abstraction to characterize cases where the cost functions behave safely in the forward exploration of the weighted timed automaton.

We reuse notations of the previous proof.

**Proposition D.1.** *Let $(\ell, \mathcal{Z}_i)_{i\geq 0}$ be an infinite sequence of symbolic states computed during the symbolic exploration of $\mathcal{A}$, which are lower-bounded, and such that there is $Y \subseteq X$ with $\pi_Y((Z_{i+1})_Y) \subseteq \pi_Y((Z_i)_Y)$, but if*

$$\kappa_i = \Big( \min_{\substack{v \in \overline{(Z_i)_Y} \\ \pi_Y(v) = u}} \zeta_i(v) \Big)_{\substack{u \in \prod_{y\in Y}[0, M(y)] \\ u \text{ has integral coord.}}}$$

*then $(\kappa_i)_{i\geq 0}$ is an infinite antichain. Then there is a reachable cycle in $\mathcal{A}_{\mathsf{cp}}$ with negative cost, which is not purely composed of delay transitions.*

*Proof.* We consider the corner-point abstraction $\mathcal{A}_{\mathsf{cp}}$.

There is $u \in \prod_{y\in Y}[0, M(y)]$ such that $u$ has integral coordinates, and such that the sequence $(\kappa_i(u))_{i\geq 0}$ is decreasing. Let $v_i \equiv u$ be such that $v_i \in \overline{Z_i}$ and $\zeta_i(v_i)$ yields the minimum for $\kappa_i(u)$. Let $\rho_i$ be a run (in the closure of $\mathcal{A}$) that leads to $(\ell, v_i)$ with cost $\zeta_i(v_i)$ (notice that we can assume it goes through points with integral coordinates). If we project $\rho_i$ onto $\pi_i$ in the corner-point abstraction, state $(\ell, v_i)$ coincides with $u$ as a corner of some region $R_i$. Gathering all this paths into a tree, we get that this tree is infinite, it has finite branching, therefore, by Koenig's lemma, there is an infinite branch in the tree, and along that branch, there is some state $(\ell, (R, u))$ which is visited infinitely often. The value of the

cost at each occurrence of this state is decreasing, hence there is a reachable cycle with negative cost. Assume that this cycle is purely made of delay moves: this means that $R$ is the maximal region (all clocks are above their maximal constants), and that the transitions correspond to delaying in that maximal regions, with a negative cost. This means that we would then compute a non lower-bounded priced zone; which is not possible by assumption. So this cycle is not purely made of delay transitions. □

The previous proposition expresses the fact that the cost becomes arbitrarily small, hence decreases to $-\infty$. If the mentioned cycle is co-reachable, then this means that the foreseen optimal cost is $-\infty$; on the other hand, if the cycle is not co-reachable, then the optimal cost might be finite, but this branch of the exploration will make the algorithm fail to terminate.

Conversely, priced zones generated along such a reachable cycle will form an antichain, but this does not mean that Algorithm 1 will not terminate, since this antichain can be dominated by some priced zone previously computed in which the cost function is not lower-bounded.

## References


[ARF14] Omar Al-Bataineh, Mark Reynolds, and Tim French. Finding best and worst case execution times of systems using Difference-Bound Matrices. In *Proc. 12th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'14)*, volume 8711 of *Lecture Notes in Computer Science*, pages 38–52. Springer, 2014.

[BBL08] Patricia Bouyer, Ed Brinksma, and Kim G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):2–23, 2008.

[BBLP06] Gerd Behrmann, Patricia Bouyer, Kim G. Larsen, and Radek Pelànek. Zone based abstractions for timed automata exploiting lower and upper bounds. *International Journal on Software Tools for Technology Transfer*, 8(3):204–215, 2006.

[Hig52] Graham Higman. Ordering by divisibility in abstract algebras. In *Proc. London Math. Soc.*, volume 2, pages 326–336, 1952.

[HSW12] Frédéric Herbreteau, B. Srivathsan, and Igor Walukiewicz. Better abstractions for timed automata. In *Proc. 27th Annual Symposium on Logic in Computer Science (LICS'12)*, pages 375–384. IEEE Computer Society Press, 2012.

[LBB+01] Kim G. Larsen, Gerd Behrmann, Ed Brinksma, Angskar Fehnker, Thomas Hune, Paul Pettersson, and Judi Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer, 2001.