Symbolic Optimal Reachability in Weighted Timed Automata

Patricia Bouyer(✉), Maximilien Colange, and Nicolas Markey

LSV – CNRS, ENS Cachan, Université Paris Saclay, Cachan, France
{bouyer,colange,markey}@lsv.fr

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 1990s 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 2000s as an intermediary model for modelling resource-consumption

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

S. Chaudhuri and A. Farzan (Eds.): CAV 2016, Part I, LNCS 9779, pp. 513–530, 2016.
DOI: 10.1007/978-3-319-41528-4_28

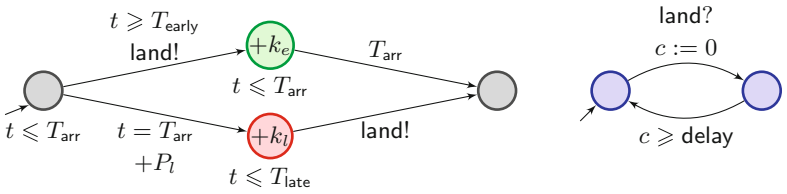


Fig. 1. A (simplified) model of the Aircraft Landing System [LBB+01]: aircrafts (left) have an optimal landing time T_{arr} within a possible landing interval $[T_{\text{early}}, T_{\text{late}}]$. The aircraft can speed up (which incurs some extra cost, modelled by k_e) to land earlier than T_{arr}, 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).

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 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 Sect. 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 ϱ 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 ϱ ends in some $s = (\ell, v)$ with $\ell \in \mathsf{Goal}$, we say that ϱ 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 ℓ represents a cost rate, and delaying t time units in a location ℓ 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* ϱ of a weighted timed automaton is a run of the underlying timed automaton. The cost of ϱ, denoted $\mathsf{cost}(\varrho)$, is the sum of the costs of all the simple moves along ϱ.

Example 1. We consider the weighted timed automaton $\mathcal{A}$ depicted in Fig. 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:

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

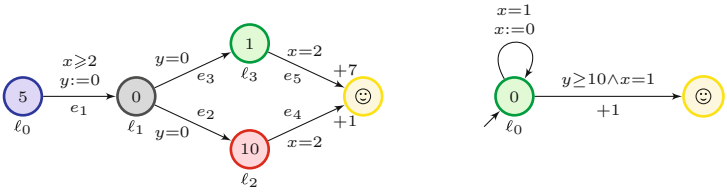


Fig. 2. Examples of weighted timed automata

The cost of ϱ is $\mathsf{cost}(\varrho) = 5 \cdot 0.1 + 1 \cdot 1.9 + 7 = 9.4$ (the cost per time unit is 5 in ℓ_0, 1 in ℓ_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 *ϵ-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 $\epsilon > 0$, ϵ-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 ζ is meant to represent the optimal cost so far (that is, $\zeta(v)$ is the optimal cost so far for reaching configuration (ℓ, 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 ζ_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 (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

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
```

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 ℓ 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 ϱ from the initial state $(\ell_0, \mathbf{0}_X)$ to (ℓ, v), such that $\zeta(v) \leqslant \mathsf{cost}(\varrho) \leq \zeta(v) + \epsilon$. For a location ℓ, a priced zone $\mathcal{Z} = (Z, \zeta)$ and a run ϱ starting in a configuration s, we say that ϱ *ends in* $(\ell, \mathcal{Z})$ if ϱ leads from s to a configuration (ℓ, v) with $v \in Z$ and $\mathsf{cost}(\varrho) \geqslant \zeta(v)$. The post operation 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 Post^* denotes the iteration of the Post operator;
- for every run ϱ from a configuration s to a configuration s', and every mixed move τ from s', if ϱ ends in $(\ell, \mathcal{Z})$, then $\varrho\tau$ ends in an element of $\mathsf{Post}(\ell, \mathcal{Z})$.
- for every run ϱ from $(\ell_0, \mathbf{0}_X)$, there exists $(\ell, \mathcal{Z}) \in \mathsf{Post}^*(\ell_0, \mathcal{Z}_0)$ such that ϱ 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 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 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

[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).

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 Fig. 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].

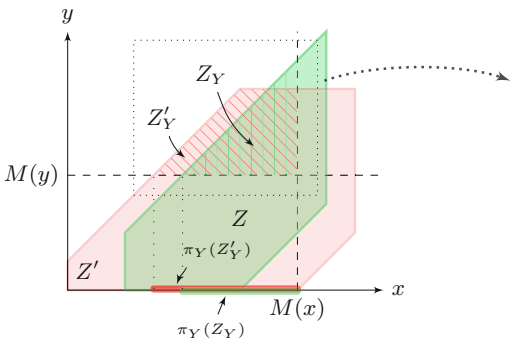


Fig. 3. Two-dimensional zones Z and Z', and sub-zones Z_Y and Z'_Y for $Y = \{x\}$.

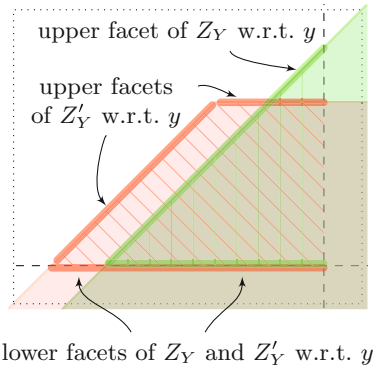


Fig. 4. Simple facets of Z_Y and Z'_Y w.r.t. clock y.

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 Sect. 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, ζ). We define the natural projection $\pi_Y : \mathbb{R}^X_{\geq 0} \to \mathbb{R}^Y_{\geq 0}$, 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 ζ and ζ' 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 ζ_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:

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

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 ζ 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 ζ'. 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 ζ_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 ζ_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 ζ and ζ' are lower-bounded on Z_Y and Z'_Y respectively. Then the following holds:*

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

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 ζ is not lower-bounded on Z but ζ' 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 ζ' is not lower-bounded on Z'_Y, then $\mathcal{Z}_Y \sqsubseteq \mathcal{Z}'_Y$.*

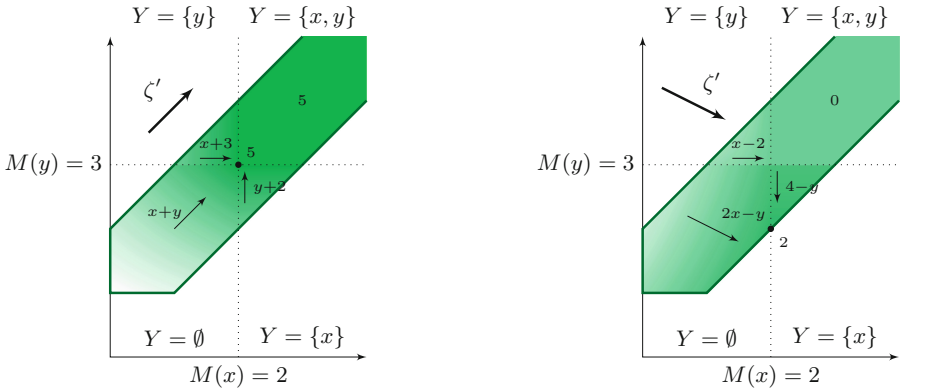
Corollary 11. *Let $\mathcal{Z} = (Z, \zeta)$ and $\mathcal{Z}' = (Z', \zeta')$ be two priced zones. Then we can effectively decide whether $\mathcal{Z} \sqsubseteq \mathcal{Z}'$.*

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 ζ. In Fig. 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 Fig. 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 μ.*

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 ω-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_ℓ be the minimum (resp. maximum) delay that can be delayed in ℓ if location ℓ 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 Fig. 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 Fig. 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/.

Table 1. Experimental results

			# Waiting	# Passed	# stored	# tests	# succ. tests	time (s.)
ASL	+P	$\sqsubseteq$	11,820	4,785	9,324	3.7×10^{05}	13,676	0.3
		$\Subset$	32,322	13,036	26,555	2.9×10^{06}	32,263	0.7
	-P	$\sqsubseteq$	1.7×10^{06}	1.5×10^{06}	6.9×10^{05}	8.1×10^{08}	1.2×10^{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	$\sqsubseteq$	6.0×10^{05}	4.8×10^{05}	5.6×10^{05}	6.2×10^{06}	1.7×10^{05}	11.3
		$\Subset$	1.5×10^{06}	1.3×10^{06}	1.4×10^{06}	9.1×10^{07}	7.0×10^{05}	27.5
	-P	$\sqsubseteq$	1.3×10^{06}	1.3×10^{06}	1.3×10^{06}	2.5×10^{07}	7.0×10^{05}	23.9
		$\Subset$	5.8×10^{06}	5.8×10^{06}	5.4×10^{06}	1.1×10^{09}	1.9×10^{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

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 Fig. 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 min. 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 Fig. 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] Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.-H.: Hybrid automata: an algorithmic approach to specification and verification of hybrid systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) HS 1993. LNCS, vol. 736, pp. 209–229. Springer, Heidelberg (1993)

[AD90] Alur, R., Dill, D.L.: Automata for modeling real-time systems. In: Paterson, M.S. (ed.) ICALP 1990. LNCS, vol. 443, pp. 322–335. Springer, Heidelberg (1990)

[AD94] Alur, R., Dill, D.L.: A theory of timed automata. Theoret. Comput. Sci. **126**(2), 183–235 (1994)

[AFM+03] Amnell, T., Fersman, E., Mokrushin, L., Pettersson, P., Yi, W.: TIMES: a tool for schedulability analysis and code generation of real-time systems. In: Larsen, Kim Guldstrand, Niebert, Peter (eds.) FORMATS 2003. LNCS, vol. 2791, pp. 60–72. Springer, Heidelberg (2004)

[ALP01] Alur, R., La Torre, S., Pappas, G.J.: Optimal paths in weighted timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 49–62. Springer, Heidelberg (2001)

[ARF14] Al-Bataineh, O., Reynolds, M., French, T.: Finding best and worst case execution times of systems using difference-bound matrices. In: Legay, A., Bozga, M. (eds.) FORMATS 2014. LNCS, vol. 8711, pp. 38–52. Springer, Heidelberg (2014)

[BBBR07] Bouyer, P., Brihaye, T., Bruyère, V., Raskin, J.-F.: On the optimal reachability problem. Form. Methods Syst. Des. **31**(2), 135–175 (2007)

[BBFL03] Behrmann, G., Bouyer, P., Fleury, E., Larsen, K.G.: Static guard analysis in timed automata verification. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 254–270. Springer, Heidelberg (2003)

[BBL08] Patricia, B., Brinksma, E., Larsen, K.G.: Optimal infinite scheduling for multi-priced timed automata. Form. Methods Syst. Des. **32**(1), 2–23 (2008)

[BBLP06] Behrmann, G., Bouyer, P., Larsen, K.G., Pelànek, R.: Zone based abstractions for timed automata exploiting lower and upper bounds. Int. J. Softw. Tools Technol. Transf. **8**(3), 204–215 (2006)

[BCM16] Bouyer, P., Colange, M., Markey, N.: Symbolic optimal reachability in weighted timed automata. Technical report abs/1602.00481, CoRR (2016). http://arxiv.org/abs/1602.00481

[BFH+01] Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J.M.T., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) HSCC 2001. LNCS, vol. 2034, pp. 147–161. Springer, Heidelberg (2001)

[BFLM11] Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N.: Quantitative analysis of real-time systems using priced timed automata. Commun. ACM **54**(9), 78–87 (2011)

[BLR05] Behrmann, G., Larsen, K.G., Rasmussen, J.I.: Optimal scheduling using priced timed automata. ACM Sigmetrics Perform. Eval. Rev. **32**(4), 34–40 (2005)

[Bou04] Bouyer, P.: Forward analysis of updatable timed automata. Form. Methods Syst. Des. **24**(3), 281–320 (2004)

[BV04] Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge (2004)

[BY03] Bengtsson, J.E., Yi, W.: On clock difference constraints and termination in reachability analysis of timed automata. In: Dong, J.S., Woodcock, J. (eds.) ICFEM 2003. LNCS, vol. 2885, pp. 491–503. Springer, Heidelberg (2003)

[DHS+14] Deshpande, A., Herbreteau, F., Srivathsan, B., Tran, T.-T., Walukiewicz, I.: Fast detection of cycles in timed automata. Technical report abs/1410.4509, CoRR (2014). http://arxiv.org/abs/1410.4509

[DOT+10] Dalsgaard, A.E., Chr, M., Olesen, M.T., Hansen, R.R., Larsen, K.G.: METAMOC: modular execution time analysis using model checking. In: Proceedings of 10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010). OpenAccess Series in Informatics (OASIcs), vol. 15, pp. 113–123. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2010)

[GELP10] Gustavsson, A., Ermedahl, A., Lisper, B., Pettersson, P.: Towards WCET analysis of multicore architectures using UPPAAL. In: Proceedings of 10th International Workshop on Worst-Case Execution Time Analysis (WCET 2010). OpenAccess Series in Informatics (OASIcs), vol. 15, pp. 101–112. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2010)

[HKPV98] Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? J. Comput. Syst. Sci. **57**(1), 94–124 (1998)

[HKSW11] Herbreteau, F., Kini, D., Srivathsan, B., Walukiewicz, I.: Using non-convex approximations for efficient analysis of timed automata. In: Proceedings of 30th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011). LIPIcs, vol. 13, pp. 78–89. Leibniz-Zentrum für Informatik (2011)

[HSW12] Frédéric Herbreteau, B., Srivathsan, I.W.: Better abstractions for timed automata. In: Proceedings of 27th Annual Symposium on Logic in Computer Science (LICS 2012), pp. 375–384. IEEE Computer Society Press (2012)

[HT15] Herbreteau, F., Tran, T.-T.: Improving search order for reachability testing in timed automata. In: Sankaranarayanan, S., Vicario, E. (eds.) FORMATS 2015. LNCS, vol. 9268, pp. 124–139. Springer, Heidelberg (2015)

[JR11] Jaubert, R., Reynier, P.-A.: Quantitative robustness analysis of flat timed automata. In: Hofmann, M. (ed.) FOSSACS 2011. LNCS, vol. 6604, pp. 229–244. Springer, Heidelberg (2011)

[LBB+01] Larsen, K.G., Behrmann, G., Brinksma, E., Fehnker, A., Hune, T., Pettersson, P., Romijn, J.M.T.: As cheap as possible: efficient cost-optimal reachability for priced timed automata. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 493–505. Springer, Heidelberg (2001)

[RLS06] Rasmussen, J.I., Larsen, K.G., Subramani, K.: On using priced timed automata to achieve optimal scheduling. Form. Methods Syst. Des. **29**(1), 97–114 (2006)

[RS01] Rückmann, J.-J., Stein, O.: On linear and linearized generalized semi-infinite optimization problem. Ann. Oper. Res. **101**(1–4), 191–208 (2001)