Backward Symbolic Optimal Reachability in Weighted Timed Automata

Rémi Parrot, Didier Lime

Backward Symbolic Optimal Reachability in Weighted Timed Automata

Rémi Parrot[1] and Didier Lime[1][0000−0001−9429−7586]

École Centrale de Nantes, LS2N, UMR CNRS 6004, Nantes, France

Abstract. We address the problem of computing the infimum accumulated weight for the reachability of some goal location in weighted timed automata. While there already exist efficient techniques to solve this problem, we propose here a *backwards* symbolic algorithm computing the accumulated weight to the goal, instead of the accumulated weight from the initial state. Going backwards has in itself a few advantages: most notably it does not require any extrapolation operation to ensure termination. Also it may be more efficient than going forward if the set of co-reachable states is smaller than the set of reachable states. Backwards algorithms are also instrumental in several problems beyond reachability, like control problems for instance. We obtain our backward algorithm by proposing extensions of the classical action and time predecessor operations on zones to account for weights. We have implemented the approach and report on its performance.

1 Introduction

The design of timed systems, including for instance critical real-time embedded systems, is a challenging issue, with high stakes. Safety is of course of particular interest but given the limited resources (e.g., memory or energy) such systems usually have, so is optimisation.

This has lead to the development of dedicated formalisms, like timed automata [1] and extensions of those, namely *weighted* (or priced) time automata [2,3], to account for resource consumption.

The success of those formalisms relies on the availability of efficient algorithms and data structures (in particular *Difference Bound Matrices*, DBM) for their analysis, and of state-of-the-art tools, like Uppaal [19], implementing them. Those successful techniques have been extended to the setting of weighted timed automata [17] and then further refined [22,23,11], retaining much of their efficiency, and are available in tools like Uppaal-CORA [1] and TiAMo[2].

Much of the efficiency of those tools for solving infimum weight reachability in weighted timed automata comes from the extension of zones, representing in a symbolic manner the values of the clocks, to weighted zones including an expression of the weight from the initial state. This extension of course comes with extensions of the algorithms to efficiently handle zones represented as DBMs. Based

[1] `http://people.cs.aau.dk/~adavid/cora/index.html`

[2] `https://git.lsv.fr/colange/tiamo`

on that data structure the exploration of the state-space classically proceeds in a forward manner, by iteratively computing successors, rather than backwards, by computing predecessors. This is mostly due to the fact that state-of-the-art tools extend timed automata with finite-range integer variables for modelling convenience, and that there are many predecessors by a transition with an integer variable assignment like $i \leftarrow 2$.

The backwards approach has some advantages of its own however: most notably it does not require any extrapolation operation to ensure termination [12,16], while its treatment in the weighted case is not trivial [11] (and arguably it is already the case for plain timed automata [9]). Also it may be more efficient than going forward if the set of co-reachable states is smaller than the set of reachable states. Backwards algorithms are also instrumental in several problems beyond reachability, like control problems for instance [21,13]. A hybrid forward / backward approach like that of [13], generalised in [14], also allows to circumvent the "predecessor of assignment" problem mentioned above.

We therefore propose here an extension of the classical time and action predecessors for timed automata, lifting them to the weighted case in the spirit of [17], by encoding in zones the accumulated weight to the goal, instead of the accumulated weight from the start. This allows us to easily adapt the classical exploration algorithm of [23] so that it works in a backward manner. We have implemented this algorithm and report on its performances.

Backwards algorithms for weighted timed automata have already been studied in more general contexts: for probabilistic timed automata in [6,7], where the property studied is cost-bounded reachability with only non-negative weights, and for optimal timed control in [10] with an additional assumption on weight cycles. The main difference with our work is that in both cases, they are not interested in the efficient representation and computation of the symbolic states, which is the crux of this article.

The paper is organized as follows: Section 2 recalls the basics of weighted timed automata, Section 3 introduces the operators needed to perform backward infimum weight reachability and the corresponding algorithm. We proceed with a small experimental evaluation in Section 4 and, finally, we conclude in Section 5.

2 Weighted Timed Automata

We denote by $\mathbb{R}$ the set of real numbers, by $\mathbb{Q}$ the set of rational numbers, by $\mathbb{Z}$ the set of integers, and by $\mathbb{N}$ the set of natural numbers (including 0). The subset of non-negative real numbers is denoted by $\mathbb{R}_{\geq 0}$.

For any sets A and B we denote by B^A the set of mappings from A to B.

A *clock constraint* on finite set X is a *conjunction* of expressions of the form $x \sim k$, with $x \in X$, $k \in \mathbb{N}$, and $\sim \in \{<, \leq, =, \geq, >\}$. We denote by $\mathcal{C}(X)$ the set of clock constraints, and by $\mathcal{C}'(X)$ its subset where $\sim \in \{<, \leq\}$.

Definition 1 (Weighted Timed Automaton). *A weighted timed automaton (WTA) is a tuple* $\mathcal{A} = (L, l_0, X, E, \mathsf{Inv}, \mathsf{weight})$ *where:*

- L *is a finite set of* locations*;*
- $l_0 \in L$ *is an* initial *location;*
- X *is a finite set of* clocks*;*
- $E \subseteq L \times \mathcal{C}(X) \times 2^X \times L$ *is a finite set of* edges*. Let* $(l, g, R, l') \in E$*. This corresponds to an edge in the automaton with source location* l*,* guard g*, set of clocks to reset to zero* R*, and target location* l'*;*
- $\mathsf{Inv} \in (\mathcal{C}'(X))^L$ *is a mapping giving for each location an* invariant*;*
- $\mathsf{weight} : \mathbb{Z}^{L \cup E}$ *is a* weight *mapping giving for each edge a* discrete weight *and for each location a* weight rate*.*

A *clock valuation* is a mapping from X to $\mathbb{R}_{\geq 0}$. We denote by $\vec{0}$ the null valuation such that $\forall x, \vec{0}(x) = 0$. Given a clock x and $d \in \mathbb{R}_{\geq 0}$, $v + d$ is the valuation such that $\forall x, (v+d)(x) = v(x)+d$. Valuation $v-d$ is defined similarly. Given a set of clocks to reset to zero, R, $v[R]$ is the valuation such that $v[R](x) = 0$ if $x \in R$ and $v(x)$ otherwise.

Given a clock constraint g, we say that valuation v satisfies g, denoted by $v \models g$ if substituting each clock x with $v(x)$ in g gives a boolean expression that evaluates to true. In the sequel, we often slightly abuse the notations and denote also by g the set of valuations that satisfy g.

A state of a WTA is a tuple $(l, v, w) \in L \times \mathbb{R}^X_{\geq 0} \times \mathbb{R}$. The semantics of a WTA is a timed transition system [18]:

Definition 2 (Semantics of a WTA). *The semantics of WTA* $\mathcal{A} = (L, l_0, X, E, \mathsf{Inv}, \mathsf{weight})$ *is the timed transition system* $(\mathcal{Q}, q_0, \rightarrow)$ *where:*

- $\mathcal{Q}$ *is the subset of* $L \times \mathbb{R}^X_{\geq 0} \times \mathbb{R}$ *such that for all* $(l, v) \in \mathcal{Q}, v \models \mathsf{Inv}(l)$*;*
- $q_0 = (l_0, \vec{0}, 0)$*;*
- $\rightarrow$ *is a subset of* $\mathcal{Q} \times (E \cup \mathbb{R}_{\geq 0}) \times \mathcal{Q}$*. We write* $q \xrightarrow{\alpha} q'$ *to denote that* $(q, \alpha, q') \in \rightarrow$*. Relation* $\rightarrow$ *is decomposed as:*
  - *Discrete transitions: for* $e \in E$*,* $(l, v, w) \xrightarrow{e} (l', v', w')$ *iff* $e = (l, g, R, l') \in E$*, with* $v \models g$*,* $v' = v[R]$*, and* $w' = w + \mathsf{weight}(e)$*;*
  - *Time elapsing: for* $d \in \mathbb{R}_{\geq 0}$*,* $(l, v, w) \xrightarrow{d} (l, v', w')$ *iff* $v' = v + d$*, and* $w' = w + \mathsf{weight}(l) \cdot d$*.*

A *run* in a WTA is a possibly infinite sequence $\rho = q_1 \alpha_1 q_2 \alpha_2 \cdots$ such that for all i, $q_i \xrightarrow{\alpha_i} q_{i+1}$. We denote by $\mathsf{init}(\rho)$ the first state of run ρ. Similarly, $\mathsf{last}(\rho)$ denotes the last state of finite run ρ. We denote by $\mathsf{Runs}(q, \mathcal{A})$ the set of runs starting in state q and by $\mathsf{Runs}(\mathcal{A})$ the set $\mathsf{Runs}(q_0, \mathcal{A})$.

A state q is *reachable* if there exists a finite run ρ such that $\mathsf{init}(\rho) = q_0$ and $\mathsf{last}(\rho) = q$. A location l is reachable if (l, v, w) is reachable for some v and some w.

The weight of a finite run ρ is $\mathsf{weight}(\rho) = w' - w$ where (l, v, w) is the first state of ρ for some value of l and v, and (l', v', w') is the last state of ρ for some value of l' and v'.

Let $\mathsf{lastl}(\rho)$ denote the *location* of the last state in finite run ρ. The infimum weight to reach some location l is defined as $\mathsf{infweight}(l) = +\infty$ if l is not reachable, and $\mathsf{infweight}(l) = \inf_{\substack{\rho \in \mathsf{Runs}(\mathcal{A}) \\ \mathsf{lastl}(\rho) = l}} \mathsf{weight}(\rho)$ otherwise.

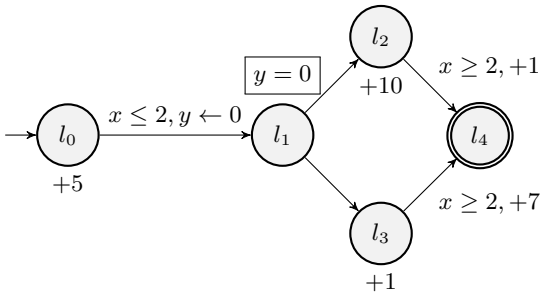

Fig. 1. A weighted timed automaton.

Example 1. Figure 1 presents a classical example of weighted timed automaton from [10]. Actually, it is a weighted timed game in that article but we treat it here as a WTA. It has two clocks x and y, constrained by the guards written on transitions (e.g. $x \leq 2$ from l_0 to l_1) and invariants (e.g. $\boxed{y = 0}$ on l_1), and possibly reset to zero (e.g. $y \leftarrow 0$ from l_0 to l_1). The initial location is l_0. Weight is updated discretely on transitions (on both transitions to l_4, $+1$ and $+7$) and with time with a derivative written below the location (e.g. $+5$ for l_0). When no such indication is present, we assume the weight is not updated. The minimum weight to reach l_4 by going through l_2 (and leaving it as soon as possible) is $5t + 10(2 - t) + 1 = 21 - 5t$, with $0 \leq t \leq 2$ the time spent in l_0. When going through l_3, it is $5t + (2 - t) + 7 = 9 + 4t$. Hence the minimum weight to reach l_4 is 9, obtained for $t = 0$ and by going through l_3.

In Section 3, we propose a backwards symbolic zone-based algorithm to compute $\mathsf{infweight}(l)$. Given a set of goal locations Goal, we assume that the weight of all runs to a location in Goal is uniformly lower-bounded: there exists a constant M such that $\forall l \in \mathsf{Goal}, \mathsf{infweight}(l) \geq M$. This condition in particular prevents negative weight cycles, the detection of which would make the algorithm more complex. For futher informations see [11].

3 Weighted Symbolic Predecessor

There is in general an infinite number of states in a WTA. In order to provide an algorithm to compute infimum weights we group them into a finite number of *symbolic states*. Such a symbolic state consists of all the states that can be reached by taking a given sequence of edges (whatever the delays in between). They are therefore defined by the common location l of all those states and the union $\mathcal{D}$ of all their valuations.

Assuming an arbitrary order on clocks, clock valuations can be seen as vectors of $\mathbb{R}_{\geq 0}^{|X|}$, where $|X|$ is the (finite) number of clocks. Set $\mathcal{D}$ is a convex polyhedron of $\mathbb{R}_{\geq 0}^{|X|}$ [10]. Its projection on clocks is defined as a conjunction of constraints of

the form $x_i \sim k_{i0}$, $-x_i \sim k_{0i}$, or $x_i - x_j \sim k_{ij}$, with for all i, j, $x_i, x_j \in X, k_{ij} \in \mathbb{Z}$ and $\sim \in \{<, \leq\}$ [18]. Such a polyhedron is called a *zone*.

In [23], the authors prove that inequalities relating weight and clock variables can be computed and handled separately from the projection of Z on clock variables, and that polyhedron $\mathcal{D}$ can be represented by a finite union of *weighted zones*.

Definition 3 (Weighted zone). *A weighted zone is a tuple $\mathcal{Z} = (Z, w, r)$ where:*

- *Z is a zone;*
- *w is the weight of the point in Z with infimum coordinates, called the offset, and noted Δ_Z (it exists and is unique due to the forms of the constraints shaping Z);*
- *$r \in \mathbb{Z}^X$ gives for each clock its contribution to the evolution of the weight for points in Z.*

Then for a given valuation $v \in \mathbb{R}^X_{\geq 0}$, the weight of the valuation in the weighted zone $\mathcal{Z}$ is:

$$\mathsf{Weight}(v, \mathcal{Z}) = w + \sum_{x \in X} r(x)(v(x) - \Delta_Z(x))$$

Now weighted symbolic states are represented by pairs of the form $(l, \mathcal{Z})$, and are subsets of the set of states $\mathcal{Q}$.

In the backward computation context, the weight of weighted zones will represent the (opposite of the) infimum remaining weight from a given valuation, to reach the goal. Thus, the weight of the null valuation in the weighted zone obtained in the initial location is exactly the opposite of the infimum weight to reach the goal from the initial state.

Definition 4. *Let $\mathcal{A} = (L, l_0, X, E, \mathsf{Inv}, \mathsf{weight})$ be a WTA. Let $e = (l, g, R, l')$ be an edge, and let $\mathcal{D}$ be a set of states of (the semantics of) $\mathcal{A}$. Then:*

$$\mathsf{Pred}_e(\mathcal{D}) = \{(l, v, w) \mid \exists (l', v', w') \in \mathcal{D} \text{ s. t. } (l, v, w) \xrightarrow{e} (l', v', w')\}$$

$$\mathsf{Pred}_\delta(\mathcal{D}) = \{(l, v, w) \mid \exists d \geq 0, (l', v', w') \in \mathcal{D}, \text{ s. t. } (l, v, w) \xrightarrow{d} (l', v', w') \\ \text{and } w = \sup\{w' - t \cdot \mathsf{weight}(l) \mid t \geq 0, (l, v + t, w') \in \mathcal{D}\}\}$$

Given a set of states D of an (unweighted) timed automaton, any point in the past of D is in general the time predecessor of an infinity of points in D (corresponding to infinitly many delays), Pred_δ computes one with an optimal weight among all of them.

Let $\mathsf{cl}(Z)$ denote the topological closure of zone Z. Note that if $\mathcal{D}$ is actually defined by a weighted zone $\mathcal{Z} = (Z, w, r)$, we can equivalently write the sup in Pred_δ as $\max\{\mathsf{Weight}(v+t, \mathcal{Z}) - t \cdot \mathsf{weight}(l) \mid t \geq 0, (l, v+t) \in \mathsf{cl}(Z)\}$. Moreover, if $\mathcal{Z}$ is unbounded then the supremum might not be finite and then the result of Pred_δ is empty.

In the following we will sometimes need to shrink a weighted zone by intersecting it with a non-weighted zone. This implies a change of offset.

Definition 5. *Let $\mathcal{Z} = (Z, w, r)$ be a weighted zone, and let Z' be a zone. Then $\mathcal{Z} \cap Z'$ is the weighted zone (Z'', w'', r), where $Z'' = Z \cap Z'$, and $w'' = \mathsf{Weight}(\Delta_{Z''}, \mathcal{Z})$. In particular, for a guard g we have $\mathcal{Z} \cap g = (Z'', w'', r)$, where $Z'' = Z \wedge g$, and $w'' = \mathsf{Weight}(\Delta_{Z''}, \mathcal{Z})$.*

Definition 6 (Facet). *The facets of the zone Z are the derived zones $\mathsf{cl}(Z) \wedge (x = n)$, for each constraint $x \sim n$ defining the zone. The facets can be grouped as follows:*

- *The facets defined by* lower *bounds on* individual *clocks, $x \geq n$, are called* lower facets*, and we denote $\mathsf{LF}(Z)$ the set of lower facets of Z;*
- *Similarly, the facets defined by* upper *bounds on* individual *clocks, $x \leq n$, are called* upper facets*, and we denote $\mathsf{UF}(Z)$ the set of upper facets of Z.*

In the following, for a weighted zone $\mathcal{Z} = (Z, w, r)$ we denote by $\mathsf{UF}(\mathcal{Z})$ and $\mathsf{LF}(\mathcal{Z})$ the set of weighted zones defined naturally by $\mathcal{F} \in \mathsf{UF}(\mathcal{Z})$ (resp. $\mathsf{LF}(\mathcal{Z})$) iff there exists $F \in \mathsf{UF}(Z)$ (resp. $\mathsf{LF}(Z)$) s.t. $\mathcal{F} = (F, w_F, r)$, with $w_F = \mathsf{Weight}(\Delta_F, \mathcal{Z})$.

In [17], the authors also define *relative* facets, corresponding to diagonal constraints, but we will not need them here.

Some more operations on weighted zones are required to compute the predecessors of weighted symbolic states.

First we need a relaxation operator to account for all clock valuations that might be predecessors of some valuations by an edge with a reset. This leads to an "inverse reset" operator.

Definition 7. *Let $\mathcal{Z} = (Z, w, r)$ be a weighted zone, and $R \subseteq X$ a subset of the clock set.*

We denote by $\mathsf{relax}_R(Z)$ the zone Z from which all constraints (except non-negativity) on every clock in R are removed. That is: $\mathsf{relax}_R(v) = \{v' \in \mathbb{R}^X_{\geq 0} | \forall x \notin R,\ v'(x) = v(x)\}$ and $\mathsf{relax}_R(Z) = \bigcup_{v \in Z} \mathsf{relax}_R(v)$.

Then the zone obtained by taking backward a reset of the clocks in R is $Z[R]^{-1} = \mathsf{relax}_R(Z \wedge (R = 0))$, where $R = 0$ is a shorthand for $\bigwedge_{y \in R}(y = 0)$.

We further define $\mathcal{Z}[R]^{-1} = (Z', w', r')$ such that: $Z' = Z[R]^{-1}$, $w' = w$, and $r'(x) = 0$ if $x \in R$ and $r(x)$ otherwise.

Second, we need to account for the past of clock valuations, that is valuations from which we can reach a given valuation (or a set of them) by some delay.

Definition 8. *Let $\mathcal{Z} = (Z, w, r)$ be a weighted zone, where F is a lower or upper facet of Z, derived from a constraint $y \sim n$, and $\mathcal{F} = \mathcal{Z} \cap (y \sim n) = (F, w_F, r)$ the related weighted zone. Let $p \in \mathbb{Z}$ be a weight-rate.*

Then, the past of Z is $Z^{\downarrow} = \{v \mid \exists d \in \mathbb{R}_{\geq 0},\ v + d \in Z\}$.

And similarly, we define the past of $\mathcal{F}$ with rate p as $\mathcal{F}^{\downarrow p} = (Z', w', r')$ such that $Z' = F^{\downarrow}$, $r'(y) = -(\sum_{x \neq y} r(x) - p)$, $r'(x) = r(x)$ for every $x \neq y$, and $w' = w_F + \sum_{x \in X} r'(x) \cdot (\Delta_{Z'}(x) - \Delta_F(x))$.

The intuition behind w' is that we compute the weight of the new offset $\Delta_{Z'}$ relatively to the offset of the facet. Notice that in the case where F is a lower facet, we have $\Delta_F = \Delta_Z$ and $w_F = w$.

Finally, we define a notation for the subtraction of weight, on weighted zones.

Definition 9. *Let $\mathcal{Z} = (Z, w, r)$ be a weighted zone, and let $n \in \mathbb{Z}$. Then $\mathcal{Z} - n$ is the weighted zone $(Z, w - n, r)$.*

We now have all the tools to compute the Pred-operations on weighted symbolic states. We start with the action predecessor, i.e., predecessor by an edge.

Theorem 1 (Action predecessor). *Let $\mathcal{A} = (L, l_0, X, E, \mathsf{Inv}, \mathsf{weight})$ be a WTA. Let $e = (l, g, R, l') \in E$, and let $\mathcal{Z}' = (Z', w', r')$ be a weighted zone. Then:*

$$\mathsf{Pred}_e((l', \mathcal{Z}')) = (l, (\mathcal{Z}'[R]^{-1} - \mathsf{weight}(e)) \cap g \cap \mathsf{Inv}(l))$$

Proof. If we forget about the weights, it is a classical result that symbolic states are closed under action predecessor operator. We assume that it is also the case for weighted symbolic states, and we prove that it is sufficient to have only one weighted symbolic state to describe the action predecessor of one weighted symbolic state (unlike for action Post-operator [17]).

We note $\mathsf{Pred}_e((l', \mathcal{Z}')) = (l, \mathcal{Z}_1)$ with $\mathcal{Z}_1 = (Z_1, w_1, r_1)$ and $(\mathcal{Z}'[R]^{-1} - \mathsf{weight}(e)) \cap g \cap \mathsf{Inv}(l) = \mathcal{Z}_2$ with $\mathcal{Z}_2 = (Z_2, w_2, r_2)$.

We want to prove that $\mathcal{Z}_1 = \mathcal{Z}_2$, which is equivalent to $Z_1 = Z_2$ and for every $v \in Z_1$, $\mathsf{Weight}(v, \mathcal{Z}_1) = \mathsf{Weight}(v, \mathcal{Z}_2)$. Equality $Z_1 = Z_2$ directly follows from the literature on timed automata (see, e.g., [12]), so we focus on weights. Note that with respect to that, the invariant plays no role since it cannot modify the zone offset due to its particular form.

Equality of weight offsets We now prove that $w_1 = w_2$. Let $q = \mathsf{weight}(e)$.

Let us define Δ'_{Z_1} such that $(l, \Delta_{Z_1}) \xrightarrow{e} (l', \Delta'_{Z_1})$. By definition of Pred_e, we have: $w_1 = \mathsf{Weight}(\Delta_{Z_1}, \mathcal{Z}_1) = \mathsf{Weight}(\Delta'_{Z_1}, \mathcal{Z}') - q$. Thus, $w_1 = w' + \sum_{x \in X} r'(x)(\Delta'_{Z_1}(x) - \Delta_{Z'}(x)) - q$. But, $\Delta'_{Z_1} = \Delta_{Z_1}[R]$, so $\Delta'_{Z_1}(x) = 0$ if $x \in R$ and $\Delta_{Z_1}(x)$ if $x \notin R$. And $Z' \wedge (R = 0) \neq \emptyset$ (otherwise the transition e would not have been taken), therefore $\forall x \in R$, $\Delta_{Z'}(x) = 0$. So we obtain: $w_1 = w' + \sum_{x \notin R} r'(x)(\Delta_{Z_1}(x) - \Delta_{Z'}(x)) - q$.

Besides, we have, by definition of the operations $\mathcal{Z} \cap g$, and $\mathcal{Z} - q$, $w_2 = \mathsf{Weight}(\Delta_{Z_2}, \mathcal{Z}'[R]^{-1}) - q$. We note $\mathcal{Z}_3 = \mathcal{Z}'[R]^{-1} = (Z_3, w_3, r_3)$. We have, by definition of the operation $[R]^{-1}$: $Z_3 = Z'[R]^{-1}$ (thus $\Delta_{Z_3} = \Delta_{Z'}$), and $w_3 = w'$, and $r_3(x) = 0$ if $x \in R$ and $r'(x)$ if $x \notin R$. So, it gives $\forall v \in Z_3$, $\mathsf{Weight}(v, \mathcal{Z}_3) = w' + \sum_{x \notin R} r'(x)(v(x) - \Delta_{Z'}(x))$. Finally, we obtain: $w_2 = w' + \sum_{x \notin R} r'(x)(\Delta_{Z_2}(x) - \Delta_{Z'}(x)) - q$

As $Z_1 = Z_2$, we have $\Delta_{Z_1} = \Delta_{Z_2}$, so we can conclude that $w_1 = w_2$.

Equality of weight We finally prove that for any $v \in Z_1$, $\mathsf{Weight}(v, \mathcal{Z}_1) = \mathsf{Weight}(v, \mathcal{Z}_2)$.

Let $v \in Z_1$, then $\exists v' \in Z'$ such that $(l, v) \xrightarrow{e} (l', v')$ and $\mathsf{Weight}(v, \mathcal{Z}_1) = \mathsf{Weight}(v', \mathcal{Z}') - q$. Thus we have: $w_1 + \sum_{x \in X} r_1(x)(v(x) - \Delta_{Z_1}(x)) = w' + \sum_{x \in X} r'(x)(v'(x) - \Delta_{Z'}(x)) - q$. Yet $w_1 = w' + \sum_{x \notin R} r'(x)(\Delta_{Z_1}(x) - \Delta_{Z'}(x)) - q$, so by injecting the value of w_1 we obtain:
$\sum_{x \in X} r_1(x)(v(x) - \Delta_{Z_1}(x)) = \sum_{x \in R} r'(x)(v'(x) - \Delta_{Z'}(x)) + \sum_{x \notin R} r'(x)(v'(x) - \Delta_{Z'}(x) - (\Delta_{Z_1}(x) - \Delta_{Z'}(x)))$. Moreover, $\forall x \in R, v'(x) = \Delta_{Z'}(x) = 0$ and $\forall x \notin R, v'(x) = v(x)$, therefore we get: $\sum_{x \in X} r_1(x)(v(x) - \Delta_{Z_1}(x)) = \sum_{x \notin R} r'(x)(v(x) - \Delta_{Z_1}(x))$ and, by adding w_1 to both sides, this gives: $\mathsf{Weight}(v, \mathcal{Z}_1) = w_1 + \sum_{x \notin R} r'(x)(v(x) - \Delta_{Z_1}(x))$.

Moreover, $r_2(x) = 0$ if $x \in R$ and $r'(x)$ otherwise, by definition of $\mathcal{Z}_2$. Thus, the weight in $\mathcal{Z}_2$ is: $\mathsf{Weight}(v, \mathcal{Z}_2) = w_2 + \sum_{x \notin R} r'(x)(v(x) - \Delta_{Z_2}(x))$. Since $w_1 = w_2$ and $\Delta_{Z_1} = \Delta_{Z_2}$, this finally gives $\mathsf{Weight}(v, \mathcal{Z}_1) = \mathsf{Weight}(v, \mathcal{Z}_2)$. □

As for the time Post-operator [17], weighted symbolic states are not directly closed under Pred_δ operator: a split of the weighted zone is needed. For example, let us consider the weighted zone (Z, w, r) depicted in Figure 2, with $w = -3$, $r(x) = 2$ and $r(y) = -1$. If we want to compute the time predecessor of this weighted zone in l_1 with a weight rate of 3, we will have to split $Z^{\downarrow}$ in three subzones: Z, ${F_1}^{\downarrow} = (Z^{\downarrow} \setminus Z) \wedge (x - y \leq 1)$ and ${F_2}^{\downarrow} = (Z^{\downarrow} \setminus Z) \wedge (x - y \geq 1)$.

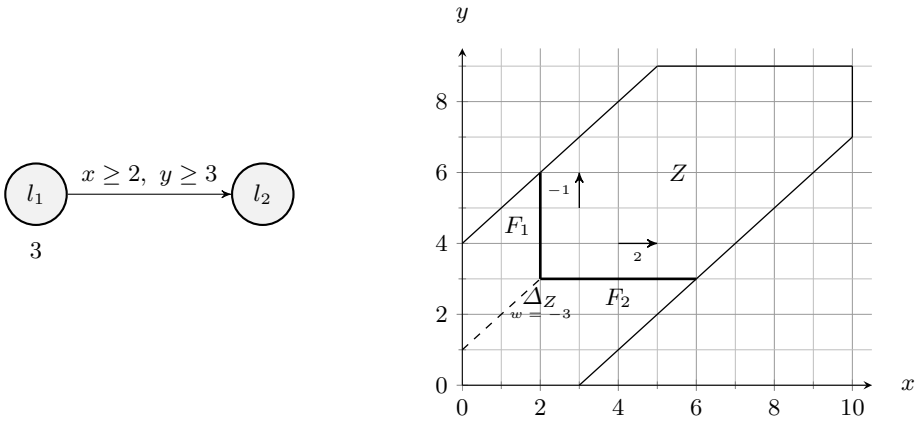

Fig. 2. Example of time predecessor of a weighted zone

The following theorem formalizes this intuition and gives an expression to compute the Pred_δ operator.

Theorem 2 (Time predecessor). *Let $\mathcal{A} = (L, l_0, X, E, \mathsf{Inv}, \mathsf{weight})$ be a WTA. Let $l \in L$, with $\mathsf{weight}(l) = p$, $\mathsf{Inv}(l) = J$, and let $\mathcal{Z}' = (Z', w', r')$ be a weighted*

zone. Then:

$$\mathsf{Pred}_\delta((l,\mathcal{Z}')) = \begin{cases} (l,\mathcal{Z}') \cup \bigcup_{\mathcal{F}\in\mathsf{LF}(\mathcal{Z}')}(l,\mathcal{F}^{\downarrow p}\cap J), & \text{if } p \geq \sum_{x\in X} r'(x) \\ \bigcup_{\mathcal{F}\in\mathsf{UF}(\mathcal{Z}')}(l,\mathcal{F}^{\downarrow p}\cap J), & \text{if } p < \sum_{x\in X} r'(x) \end{cases}$$

To prove Theorem 2, we need three technical lemmas.

Lemma 1. *Let Z be a zone. Then the following holds:*

1. *if $\mathsf{UF}(Z) \neq \emptyset$, then $Z^\downarrow \subseteq \bigcup_{F\in\mathsf{UF}(Z)} F^\downarrow$*
2. *$Z^\downarrow \subseteq Z \cup \bigcup_{F\in\mathsf{LF}(Z)} F^\downarrow$*

Proof. 1. Assume $\mathsf{UF}(Z) \neq \emptyset$. So there exists some facets $F_i = (x_i = n_i) \wedge Z$ for $x_i \in X$, $n_i \in \mathbb{N}$, and $x_i \leq n_i$ the upper constraints of Z.
Consider $v \in Z^\downarrow$. Then there exists $d \geq 0$ such that $v + d \in Z$. In particular, the latter satisfies the constraint of the upper bound constraints. For the corresponding to F_i: $(v+d)(x_i) \leq n_i$. Let $d_m = \min_i(n_i - v(x_i))$. By construction $d_m \geq d \geq 0$. We prove that $v + d_m$ belongs to some upper facet. By definition of d_m, there exists x_j such that $v(x_j) + d_m = n_j$ and $\forall x_i \neq x_j,\ v(x_i) + d_m \leq n_i$, so $v + d_m$ satisfies the upper bound constraints of $\mathsf{cl}(Z)$.
Moreover, for every $x_i \in X$, and every lower bound constraint $(x_i \geq m_i)$ of $\mathsf{cl}(Z)$, $v + d \models (x_i \geq m_i)$, and since $d_m \geq d$, $v(x_i) + d_m \geq m_i$. Finally, diagonal constraints are trivially verified since $(v+d_m)(x_j) - (v+d_m)(x_k) = v(x_j) - v(x_k) = (v+d)(x_j) - (v+d)(x_k)$.
We conclude that $v + d_m \in \mathsf{cl}(Z)$ and $v + d_m \models (x_j = n_j)$, which means $v + d_m \in F_j$ and therefore $v \in {F_j}^\downarrow$. And, finally, $Z^\downarrow \subseteq \bigcup_{F\in\mathsf{UF}(Z)} F^\downarrow$.

2. In the sequel, we write $\prec$ for an element of $\{<, \leq\}$ and $\succ$ for an element of $\{>, \geq\}$. Let $v \in Z^\downarrow$, then $\exists d \in \mathbb{R}_{\geq 0}$ such that $v + d \in Z$. Then for every $x_i \in X$, if $(x_i \succ m_i)$ is the corresponding lower bound constraint of Z, then $v + d \models (x_i \succ m_i)$. Let $d_M = \max_{x_i\in X}(m_i - v(x_i))$.
If $d_M \leq 0$ then $\forall x_i,\ m_i - v(x_i) \leq d_M \leq 0$ so $v(x_i) \geq m_i$. Yet $\forall x_i$, if $(x_i \prec m_i)$ is the corresponding upper bound constraint of Z, then $v+d \models (x_i \prec n_i)$ and then $v(x_i) \prec n_i$. Similarly any diagonal constraint $(x_i - x_j \prec p_{ij})$ of Z is also satisfied by $v + d$, and thus by v since $v(x_i) + d - (v(x_j) + d) = v(x_i) - v(x_j)$. So $v \in Z$.
Otherwise, $d_M > 0$. By definition of d_m, there exists x_j such that $v(x_j) + d_M = m_j$ and $\forall x_i \neq x_j,\ v(x_i) + d_M \geq m_i$. Yet $\forall x_i \in X,\ d \geq m_i - v(x_i)$, in particular $d \geq m_j - v(x_j) = d_M$. So for every x_i, $v(x_i) + d_M \leq v(x_i) + d \leq n_i$. As before, $v + d_M$ also trivially satisfies the diagonal constraints of $\mathsf{cl}(Z)$ and therefore, $v + d_M \in \mathsf{cl}(Z)$ and $v + d_M \models (x_j = m_j)$. So $v + d_M \in F_j$ with $F_j = \mathsf{cl}(Z) \wedge (x_j = m_j) \in \mathsf{LF}(Z)$. Therefore $v \in {F_j}^\downarrow$, and finally $v \in \bigcup_{F\in\mathsf{LF}(Z)} F^\downarrow$. □

Lemma 2. *Let Z be a zone, and F be a facet of Z derived from a constraint on a single clock $x \sim n$. Let $v \in F^\downarrow$. Then there exists d_F such that $v + d_F \in F$. And the following holds:*

1. *if* $F \in \mathsf{LF}(Z)$, *then* $d_F = \min_{\substack{d \geq 0 \\ v+d \in \mathsf{cl}(Z)}} (d)$
2. *if* $F \in \mathsf{UF}(Z)$, *then* $d_F = \max_{\substack{d \geq 0 \\ v+d \in \mathsf{cl}(Z)}} (d)$

Proof. The facet is defined by $F = \mathsf{cl}(Z) \wedge (x = n)$. Thus, $v + d_F \in F$ gives $v(x) + d_F = n$.

1. $F \in \mathsf{LF}(Z)$: Assume there exists $d \in \mathbb{R}_{\geq 0}$ such that $v+d \in \mathsf{cl}(Z)$ and $d < d_F$. We have in particular $v(x) + d < v(x) + d_F = n$, therefore $v + d$ does not satisfy the guard $(x \geq n)$, hence $v + d \notin \mathsf{cl}(Z)$. This is a contradiction, and the result follows.
2. $F \in \mathsf{UF}(Z)$: Assume there exists $d \in \mathbb{R}_{\geq 0}$ such that $v+d \in \mathsf{cl}(Z)$ and $d > d_F$. We have in particular $v(x) + d > v(x) + d_F = n$, therefore $v + d$ does not satisfy the guard $(x \leq n)$, hence $v + d \notin \mathsf{cl}(Z)$. This is a contradiction, and the result follows. □

Lemma 3. *Let $\mathcal{Z} = (Z, w, r)$ be a weighted zone, F be a lower or upper facet of Z, derived from a constraint $y \sim n$, with $\sim \in \{<, \leq, \geq, >\}$, and $\mathcal{F} = (F, w, r)$ be the corresponding weighted zone. Let $p \in \mathbb{N}$ be a weight-rate.*

Then, $\mathsf{Weight}(v, \mathcal{F}^{\downarrow p}) = \mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p$, *with* $d_F = n - v(y)$.

Proof. Writing $m = \sum_{x \in X} r(x) - p$, we have:

$$\begin{aligned}
\mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p &= w + \sum_{x \in X} r(x)(v(x) + d_F - \Delta_Z(x)) - d_F \cdot p \\
&= w + \sum_{x \in X} r(x)(v(x) - \Delta_Z(x)) + d_F \cdot m \\
&= w + \sum_{x \in X} r(x)(v(x) - \Delta_Z(x)) - m \cdot (v(y) - n)
\end{aligned}$$

Moreover, F is derived from the constraint $y \sim n$, we have $\Delta_F(y) = n$. Then, $\mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p$ can be rewritten as:

$$\begin{aligned}
&\mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p = w + \sum_{x \in X} r(x)(v(x) - \Delta_Z(x)) - m \cdot (v(y) - \Delta_F(y)) \\
&= w + \sum_{x \in X} r(x)(\Delta_F(x) - \Delta_Z(x)) + \sum_{x \in X} r(x)(v(x) - \Delta_F(x)) - m \cdot (v(y) - \Delta_F(y)) \\
&= w_F + \sum_{x \neq y} r(x)(v(x) - \Delta_F(x)) - (\sum_{x \neq y} r(x) - p) \cdot (v(y) - \Delta_F(y))
\end{aligned}$$

Let us denote by (Z', w', r') the weighted zone $\mathcal{F}^{\downarrow p}$. Then by definition $r'(y) = -(\sum_{x \neq y} r(x) - p)$ and $\forall x \neq y,\ r'(x) = r(x)$:

$$\mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p = w_F + \sum_{x \in X} r'(x)(v(x) - \Delta_F(x))$$

Recall that the weight w' of the offset $\Delta_{Z'}$ of $\mathcal{F}^{\downarrow p}$ is defined as:

$$w' = w_F + \sum_{x \in X} r'(x)(\Delta_{Z'}(x) - \Delta_F(x))$$

So finally:

$$\mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p = w' + \sum_{x \in X} r'(x)(v(x) - \Delta_{Z'}(x)) = \mathsf{Weight}(v, \mathcal{F}^{\downarrow p})$$

□

Proof (Theorem 2). In order to prove this theorem we proceed by double inclusion.

$\boxed{\subseteq}$ Let $(l, v, w) \in \mathsf{Pred}_\delta((l, \mathcal{Z}'))$.Then there exists $d \in \mathbb{R}_{\geq 0}$ such that $v + d \in Z'$, by definition of Pred_δ, and therefore $v \in Z'^{\downarrow}$. Also, by definition of Pred_δ, we have:

$$w = \max\{\mathsf{Weight}(v + t, \mathcal{Z}') - t \cdot \mathsf{weight}(l) \mid t \geq 0, v + t \in \mathsf{cl}(Z')\}$$

For every $t \in \mathbb{R}_{\geq 0}$, writing $m = \sum_{x \in X} r'(x) - p$, we have:

$$\begin{aligned} \mathsf{Weight}(v + t, \mathcal{Z}') - t \cdot \mathsf{weight}(l) &= w' + \sum_{x \in X} r'(x)(v(x) + t - \Delta_{Z'}(x)) - t \cdot p \\ &= w' + \sum_{x \in X} r'(x)(v(x) - \Delta_{Z'}(x)) + t \cdot m \\ &= \mathsf{Weight}(v, \mathcal{Z}') + t \cdot m \end{aligned}$$

In order to maximize this, we consider the derivative $m = \sum_{x \in X} r'(x) - p$.

- **If** $p \geq \sum_{x \in X} r'(x)$ **:** Then we need to minimize t. Lemma 1 gives $v \in Z' \cup \bigcup_{F \in \mathsf{LF}(Z')} F^{\downarrow}$.
  - **If** $v \in Z'$ **:** Then, we can take $t = 0$ to minimize the weight, and we obtain $w = \mathsf{Weight}(v, \mathcal{Z}')$. Thus $(l, v, w) \in (l, \mathcal{Z}')$.
  - **Else** $v \in \bigcup_{F \in \mathsf{LF}(Z')} F^{\downarrow}$ **:** Then, there exists $F \in \mathsf{LF}(Z')$ such that $v \in F^{\downarrow}$. Facet F is defined by $F = \mathsf{cl}(Z') \wedge (y = n)$, with $y \in X$ and $n \in \mathbb{N}$. Then, $v \in F^{\downarrow}$ gives that there exists $d_F \in \mathbb{R}_{\geq 0}$ s.t. $v + d_F \in F$. Thus, $v(y) + d_F = n$. Lemma 2 gives that d_F is the minimal value of d such that $d \geq 0$ and $v + d \in \mathsf{cl}(Z')$.
    Then, if we note $\mathcal{F} = (F, w', r')$, we have $v \in F^{\downarrow}$. Moreover $w = \mathsf{Weight}(v + d_F, \mathcal{Z}') - d_F \cdot p$, thus Lemma 3 gives $w = \mathsf{Weight}(v, \mathcal{F}^{\downarrow p})$. Finally, $v \models J$ by definition of the time predecessor, thus $(l, v, w) \in (l, \mathcal{F}^{\downarrow p} \cap J)$.
- **Else** $p < \sum_{x \in X} r'(x)$ **:** Then we need to maximize t.
  Let us suppose that $\mathsf{UF}(Z') = \emptyset$. Then $\bigcup_{\mathcal{F} \in \mathsf{UF}(\mathcal{Z}')}(l, \mathcal{F}^{\downarrow p} \cap J) = \emptyset$. Let t_M the value of t that maximizes the weight, that is to say $w = \mathsf{Weight}(v +$

$t_M, \mathcal{Z}') - t_M \cdot \mathsf{weight}(l)$ and $v + t_M \in \mathsf{cl}(Z')$. Let $\epsilon > 0$, then $v + t_M + \epsilon \in \mathsf{cl}(Z')$ because Z' has no upper facet. Moreover, because $t_M + \epsilon > t_M$ and because $\sum_{x \in X} r'(x) - p > 0$, we have $\mathsf{Weight}(v + t_M + \epsilon, \mathcal{Z}') - (t_M + \epsilon) \cdot \mathsf{weight}(l) > \mathsf{Weight}(v + t_M, \mathcal{Z}') - t_M \cdot \mathsf{weight}(l)$. Which means that t_M does not maximize the weight: $\mathsf{Weight}(v, \mathcal{Z}) < \mathsf{Weight}(v + t_M, \mathcal{Z}') + t_M \cdot \mathsf{weight}(l)$. So the supremum in the expression of Pred_δ is infinite and $\mathsf{Pred}_\delta((l, \mathcal{Z}')) = \emptyset$ also (and we actually could not take a point from it).
Assume now that $\mathsf{UF}(Z') \neq \emptyset$. Lemma 1 gives $v \in \bigcup_{F \in \mathsf{UF}(Z')} F^{\downarrow}$. Then, there exists $F \in \mathsf{UF}(Z')$ such that $v \in F^{\downarrow}$. Facet F is defined by $F = \mathsf{cl}(Z') \wedge (y = n)$, with $y \in X$ and $n \in \mathbb{N}$. Then, $v \in F^{\downarrow}$ gives that there exists $d_F \in \mathbb{R}_{\geq 0}$ s.t. $v + d_F \in F = \mathsf{cl}(Z') \wedge (y = n)$. Thus, $v(y) + d_F = n$. Lemma 2 gives that d_F is the maximal value of d such that $d \geq 0$ and $v + d \in \mathsf{cl}(Z')$.
Thus, if we note $\mathcal{F} = (F, w', r')$, we have $w = \mathsf{Weight}(v + d_F, \mathcal{Z}') + d_F \cdot p$, thus Lemma 3 gives $w = \mathsf{Weight}(v, \mathcal{F}^{\downarrow p})$ and $(v, w) \in \mathcal{F}^{\downarrow p}$. Moreover, $v \models J$ by definition of the time predecessor, thus $v \in \mathcal{F}^{\downarrow p} \cap J$. Therefore, $(l, v, w) \in (l, \mathcal{F}^{\downarrow p} \cap J)$.

This concludes the proof for the left-to-right inclusion.
$\boxed{\supseteq}$ Consider now the right-to-left inclusion:

- **If** $p \geq \sum_{x \in X} r'(x)$ **:** Let $(l, v, w) \in (l, \mathcal{Z}' \cap J) \cup \bigcup_{\mathcal{F} \in \mathsf{LF}(\mathcal{Z}')} (l, \mathcal{F}^{\downarrow p} \cap J)$. We still have $\mathsf{Weight}(v + t, \mathcal{Z}') - t \cdot \mathsf{weight}(l) = \mathsf{Weight}(v, \mathcal{Z}') + t \cdot (\sum_{x \in X} r'(x) - p)$, for every $t \in \mathbb{R}_{\geq 0}$.
  - **If** $(l, v, w) \in (l, \mathcal{Z}')$ **:** Then, we have $v \in \mathcal{Z}'$, so there exists $d \in \mathbb{R}_{\geq 0}$ s.t. $v + d \in \mathcal{Z}'$ $(d = 0)$. Also, since $\sum_{x \in X} r'(x) - p \leq 0$, the maximum of $\mathsf{Weight}(v, \mathcal{Z}') + t \cdot (\sum_{x \in X} r'(x) - p)$ is obtained for $t = 0$, thus $w = \mathsf{Weight}(v, \mathcal{Z}')$. We thus have $(l, v, w) \in \mathsf{Pred}_\delta((l, \mathcal{Z}'))$.
  - **Else** $\exists \mathcal{F} \in \mathsf{LF}(\mathcal{Z}')$ **s.t.** $(l, v, w) \in (l, \mathcal{F}^{\downarrow p} \cap J)$ **:** We note $\mathcal{F} = (F, w', r')$, with $F = \mathsf{cl}(Z') \wedge (y = n)$. Then $v \in F^{\downarrow}$, thus $\exists d \in \mathbb{R}_{\geq 0}$ s.t. $v + d \in F \subseteq \mathsf{cl}(Z')$. Moreover, $v \models J$ because $v \in F^{\downarrow} \wedge J$. Then, Lemma 3 gives $w = \mathsf{Weight}(v, \mathcal{F}^{\downarrow p}) = \mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p$, with $d_F = n - v(y)$. Also, we have $v + d_F \in F \subseteq \mathsf{cl}(Z')$ (because $v(y) + d_F = n$). Furthermore d_F is the minimal $d \geq 0$ such that $v + d \in \mathsf{cl}(\mathcal{Z}')$ according to lemma 2. Then $w = \max\{\mathsf{Weight}(v + t, \mathcal{Z}') - t \cdot \mathsf{weight}(l) \mid t \geq 0, v + t \in \mathsf{cl}(Z')\}$, and finally $(l, v, w) \in \mathsf{Pred}_\delta((l, \mathcal{Z}'))$
- **Else** $p < \sum_{x \in X} r'(x)$ **:** Then $\exists \mathcal{F} \in \mathsf{UF}(\mathcal{Z}')$ s.t. $(l, v, w) \in (l, \mathcal{F}^{\downarrow p} \cap J)$, we note $\mathcal{F} = (F, w', r')$, with $F = \mathsf{cl}(Z') \wedge (y = n)$. We still have $\mathsf{Weight}(v + t, \mathcal{Z}') - t \cdot \mathsf{weight}(l) = \mathsf{Weight}(v, \mathcal{Z}') + t \cdot (\sum_{x \in X} r'(x) - p)$, for every $t \in \mathbb{R}_{\geq 0}$. First $v \in F^{\downarrow}$, thus $\exists d \in \mathbb{R}_{\geq 0}$ s.t. $v + d \in F \subseteq \mathsf{cl}(Z')$. Second, $v \models J$ because $v \in F^{\downarrow} \wedge J$. Then, Lemma 3 gives $w = \mathsf{Weight}(v, \mathcal{F}^{\downarrow p}) = \mathsf{Weight}(v + d_F, \mathcal{Z}) - d_F \cdot p$, with $d_F = n - v(y)$. Also, we have $v + d_F \in F \subseteq \mathsf{cl}(Z')$ (because $v(y) + d_F = n$). Furthermore d_F is the maximal $d \geq 0$ such that $v + d \in \mathcal{Z}'$ according to Lemma 2. Then as $\sum_{x \in X} r'(x) - p > 0$, we indeed have $w = \max\{\mathsf{Weight}(v + t, \mathcal{Z}') - t \cdot \mathsf{weight}(l) \mid t \geq 0, v + t \in \mathsf{cl}(Z')\}$, and finally $(l, v, w) \in \mathsf{Pred}_\delta((l, \mathcal{Z}'))$. □

Using the Pred_δ and Pred_e operators, we can straightforwardly adapt the algorithm of [23] to work backwards, which gives Algorithm 1.

Starting from a set of goal locations Goal, we build initial set of symbolic states by combining with each of these locations the universal zone $\mathbb{R}^X_{\geq 0}$ (defined by all clocks should be non-negative) on set of clocks X, with a weight uniformly equal to 0.

The algorithm works as usual with a passed list PASSED and a waiting list WAITING. At each iteration, we pick a waiting symbolic state and if it contains the initial state of the automaton, which is then necessarily the offset of the zone, we check if the corresponding weight (the opposite of the weight of the offset since we start from 0 at the goal and subtract the weights as we go backwards) is better than the current value of WEIGHT. If so we update WEIGHT.

Then we add all predecessors of the current symbolic state to the waiting list, unless some bigger and cheaper symbolic state has already been visited.

To capture this last notion of bigger and cheaper we use the classical subsumption operator $\preccurlyeq$ defined as:

Definition 10. *Let $(l, \mathcal{Z})$, with $\mathcal{Z} = (Z, w, r)$, and $(l', \mathcal{Z}')$, with $\mathcal{Z}' = (Z', w', r')$ be two symbolic states. We say that $(l, \mathcal{Z})$ is subsumed by $(l', \mathcal{Z}')$, and we write $(l, \mathcal{Z}) \preccurlyeq (l', \mathcal{Z}')$, if: (1) $l = l'$, (2) $Z \subseteq Z'$ and (3) for all $v \in Z$, $\mathsf{Weight}(v, \mathcal{Z}) \leq \mathsf{Weight}(v, \mathcal{Z}')$.*

In the usual definition the weight in $\mathcal{Z}$ would be higher than in $\mathcal{Z}'$ but remember that our weight is the *opposite* of the remaining weight to the goal.

Algorithm 1 Symbolic algorithm for optimal weight

```
1: WEIGHT ← +∞
2: PASSED ← ∅
3: WAITING ← {(l, (ℝ^X_{≥0}, 0, 0⃗)) | l ∈ Goal}
4: while WAITING ≠ ∅ do
5:    select and remove S = (l, (Z, w, r)) from WAITING
6:    if l = l_0 and 0⃗ ∈ Z and −w < WEIGHT then
7:       WEIGHT ← −w
8:    end if
9:    if for all S' ∈ PASSED, S ⋠ S' then
10:      add S to PASSED
11:      for all e = (l, g, R, l') ∈ E, for all S' ∈ Pred_δ(Pred_e(S)), add S' to WAITING
12:   end if
13: end while
14: return WEIGHT
```

Algorithm 1 has the classical advantage of exploring only co-reachable states (but may of course explore non-reachable states). Also in contrast to the discrete successor operator for weighted symbolic states, the Pred_e operator never splits zones. Finally, zone abstraction/normalization is not necessary to ensure termi-

nation when computing backwards [12,16], while it should be handled carefully when working forward [11].

4 Implementation and Experiments

We have implemented the technique in Roméo [20]. The implementation and all the benchmarks presented here are freely available[3]. Note that Roméo is a tool designed for time Petri nets, a model close to timed automata, but with some expressiveness differences [5]. The zone graph techniques are however perfectly usable for time Petri nets [15]. The forward technique of [17] is not implemented in Roméo, so we instead compare with the similar forward technique presented in [8]. Also, since Roméo deals with Petri nets, where markings can be seen as the values of a finite set of integer variables, a purely backward method would be impractical as explained in the introduction, so we have implemented a mixed forward backward approach in which we first precompute the reachable state-space and then compute backward on this. Therefore, for the comparison to be fair, we look at examples with negative costs (but no negative cycles), for which the whole state-space would have to be explored anyway.

First we look at the aircraft landing problem described in [4]. The modelling with a Petri-net like model (even using also additional integer variables) is fairly different from the original one: in particular we cannot test a global clock without resetting it with time Petri nets. In accordance with the above comments, we have also made it so that planes that land early actually get some bonus (negative cost). For all these reasons, we had to limit to a small subset of the planes in the original model to get some reasonable performances.

Second we look at the scheduling example of [8], in which we need to execute some periodic task set, on two processors, possibly using renewable energy (which counts as a negative cost), the availability of which depends on meteorological conditions. We add an additional constraints that instances of tasks should not overlap, which reduces the state-space quite a bit.

Both approaches give the same results on all examples, which is a good point.

The results are presented in Table 1.

		Landing				Scheduling			
Aircrafts / Tasks		3	4	5	6	2	3	4	5
Forward	Time (s)	**1**	**4**	**14**	**50**	1	17	196	1044
	Mem. (MB)	**8**	**48**	**205**	**756**	17	177	1501	5826
Backward	Time (s)	< 1	6	46	322	$<$ **1**	**6**	**50**	**251**
	Mem. (MB)	17	112	504	1804	**13**	**61**	**209**	**504**

Table 1. Results on an Intel Core i7-7700 CPU @ 3.60GHz with 32GB of RAM.

[3] `http://romeo.rts-software.org/releases/FORMATS2020.tgz`

We see that for the aircraft landing problem, the forward approach performs clearly better, though both techniques scale exponentially with the number of aircrafts (and hence of clocks, as expected). In the scheduling problem, for the original problem of [8], with 4 tasks, we get 60s (785MB) forward and 52s (402MB) backward. If we increase the execution time of task 2 from 4 to 16 (reaching a utilization factor of 1 for processor 1 if it were alone), we get the numbers in Table 1, where the backward approach is now clearly better.

We conjecture the performance is heavily impacted by the size of the co-reachable state-space. For the aircraft problem, using internal statistics, we estimate the number of co-reachable states to represent more than 80% of the reachable state-space, while we estimate it to less than 50% in the scheduling problem. It is even less (around 35%) for the original version but most of the time (around 65%) is used for the state-space precomputation, which is much bigger than with the modified task 2 (where the precomputation only takes 15% of the total time).

5 Conclusion

We have proposed extensions of the classical backwards operators for timed automata so that they can compute the remaining weight to some goal location in a weighted setting. This allows us to devise a backwards optimal cost reachability algorithm.

On the practical side, we have implemented the algorithm in the tool Roméo, and we have reported on its performance on two (slightly modified) case-studies from the literature. This experimental evaluation shows that the algorithm may outperform the classical forward approach, in particular, as could be expected, when the set of co-reachable states is significantly smaller than the set reachable states.

While this algorithm has advantages on its own, it is also a step towards symbolic and efficient verification and optimization for more expressive properties and we now want to investigate timed computation tree logic and controllability.

References

1. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *HSCC'01*, volume 2034 of *LNCS*, pages 49–62, Rome, Italy, 2001. Springer.
3. G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC'01*, volume 2034 of *LNCS*, pages 147–161, Rome, Italy, 2001. Springer.
4. G. Behrmann, K. G. Larsen, and J. I. Rasmussen. Optimal scheduling using priced timed automata. *SIGMETRICS Performance Evaluation Review*, 32(4):34–40, 2005.

5. B. Bérard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux. The expressive power of time Petri nets. *Theoretical Computer Science*, 474:1–20, 2013.
6. J. Berendsen, D. N. Jansen, and J. Katoen. Probably on time and within budget: On reachability in priced probabilistic timed automata. In *3rd International Conference on the Quantitative Evaluation of Systems (QEST 2006)*, pages 311–322, Riverside, California, USA, Sept. 2006. IEEE Computer Society.
7. J. Berendsen, D. N. Jansen, and F. W. Vaandrager. Fortuna: Model checking priced probabilistic timed automata. In *7th International Conference on the Quantitative Evaluation of Systems (QEST 2010)*, pages 273–281, Williamsburg, Virginia, USA, Sept. 2010. IEEE Computer Society.
8. H. Boucheneb, D. Lime, B. Parquier, O. H. Roux, and C. Seidner. Optimal reachability in cost time petri nets. In U. Nestmann and K. Wolter, editors, *15th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS 2017)*, volume 10419 of *Lecture Notes in Computer Science*, pages 58–73, Berlin, Germany, Sept. 2017. Springer.
9. P. Bouyer. Untameable timed automata! In *STACS'03*, volume 2607 of *LNCS*, pages 620–631. Springer, 2003.
10. P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal strategies in priced timed game automata. In *FSTTCS'04*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.
11. P. Bouyer, M. Colange, and N. Markey. Symbolic optimal reachability in weighted timed automata. In *CAV'16*, volume 9779 of *LNCS*, pages 513–530, Toronto, Canada, July 2016. Springer.
12. P. Bouyer and F. Laroussinie. Model checking timed automata. In *Modeling and Verification of Real-time Systems*, pages 111–140. ISTE – John Wiley & Sons, 2008.
13. F. Cassez, A. David, E. Fleury, K. G. Larsen, and D. Lime. Efficient on-the-fly algorithms for the analysis of timed games. In *CONCUR'05*, volume 3653 of *LNCS*, pages 66–80, San Fransisco, CA, USA, Aug. 2005. Springer.
14. S. Enevoldsen, K. G. Larsen, and J. Srba. Abstract dependency graphs and their application to model checking. In *TACAS'19*, volume 11427 of *LNCS*, pages 316–333. Springer, 2019.
15. G. Gardey, O. H. Roux, and O. F. Roux. State space computation and analysis of time Petri nets. *Theory and Practice of Logic Programming (TPLP). Special Issue on Specification Analysis and Verification of Reactive Systems*, 6(3):301–320, 2006.
16. F. Herbreteau, B. Srivathsan, and I. Walukiewicz. Better abstractions for timed automata. *Information & Computation*, 251:67–90, 2016.
17. K. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *CAV'01*, volume 2102 of *LNCS*, pages 493–505, 2001.
18. K. G. Larsen, P. Pettersson, and W. Yi. Model-checking for real-time systems. In *Fundamentals of Computation Theory*, pages 62–88, 1995.
19. K. G. Larsen, P. Pettersson, and W. Yi. UPPAAL in a Nutshell. *Journal of Software Tools for Technology Transfer (STTT)*, 1(1-2):134–152, 1997.
20. D. Lime, O. H. Roux, C. Seidner, and L.-M. Traonouez. Romeo: A parametric model-checker for Petri nets with stopwatches. In S. Kowalewski and A. Philippou, editors, *15th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2009)*, volume 5505 of *LNCS*, pages 54–57, York, United Kingdom, Mar. 2009. Springer.
21. O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *STACS'95*, volume 900 of *LNCS*, pages 229–242. Springer, 1995.

22. J. I. Rasmussen, K. G. Larsen, and K. Subramani. Resource-optimal scheduling using priced timed automata. In *TACAS'04*, volume 2988 of *LNCS*, pages 220–235. Springer, 2004.
23. J. I. Rasmussen, K. G. Larsen, and K. Subramani. On using priced timed automata to achieve optimal scheduling. *Formal Methods in System Design*, 29(1):97–114, 2006.