Available online at www.sciencedirect.com

ScienceDirect

Theoretical Computer Science 390 (2008) 197–213

Theoretical Computer Science

www.elsevier.com/locate/tcs

Optimal reachability for multi-priced timed automata

Kim Guldstrand Larsen*, Jacob Illum Rasmussen

Aalborg University, Denmark

Abstract

In this paper, we prove the decidability of the minimal and maximal reachability problems for multi-priced timed automata, an extension of timed automata with multiple cost variables evolving according to given rates for each location. More precisely, we consider the problems of synthesizing the minimal and maximal costs of reaching a given target location. These problems generalize conditional optimal reachability, i.e., the problem of minimizing one primary cost under individual upper bound constraints on the remaining, secondary, costs, and the problem of maximizing the primary cost under individual lower bound constraints on the secondary costs. Furthermore, under the liveness constraint that all traces eventually reach the goal location, we can synthesize all costs combinations that can reach the goal.

The decidability of the minimal reachability problem is proven by constructing a zone-based algorithm that always terminates while synthesizing the optimal cost tuples. For the corresponding maximization problem, we construct two zone-based algorithms, one with and one without the above liveness constraint. All algorithms are presented in the setting of two cost variables and then lifted to an arbitrary number of cost variables.

1. Introduction

Recently, research has been focused on extending the framework of timed automata (TA), [1], towards linear hybrid automata (LHA), [2], by allowing continuous variables with non-uniform rates and maintaining a decidable reachability problem.

One such class of models is that of priced (or weighted) timed automata (PTA), [3,4], which are timed automata augmented with a single cost variable. For this class of timed automata, the minimum-cost reachability problem, i.e., finding the minimum cost of reaching some goal location, is decidable. The restriction with respect to linear hybrid automata is that the cost variable can be tested in neither guards nor invariants, cannot be reset,[1] and grows monotonically.

Ignoring the variable c_2, Fig. 1 depicts a PTA for which the rate of c_1 is 1 and 2 in locations l_1 and l_2, respectively. The type of reachability question we can ask for this model is: What is the cheapest way of reaching the "happy" location? In this case the answer is 3, and is achieved by delaying for 1 time unit in l_1, taking the transition to l_2 and delaying for 1 time unit before proceeding to l_3.

* Corresponding author.
E-mail addresses: kgl@cs.aau.dk (K.G. Larsen), illum@cs.aau.dk (J.I. Rasmussen).

[1] In the literature variables with these two properties are sometimes referred to as observers.

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2007.09.021

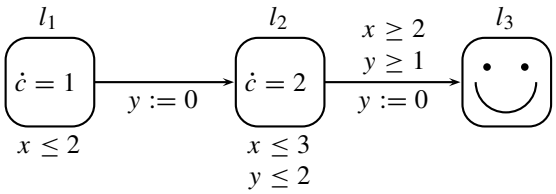

Fig. 1. Example priced timed automaton.

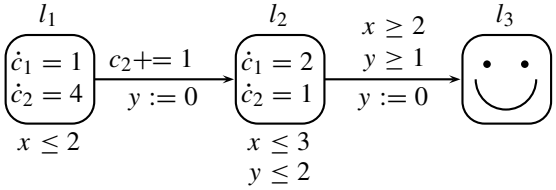

Fig. 2. Example dual-priced timed automaton.

A natural extension of the PTA is to allow a secondary cost variable, thus arriving at dual-priced timed automata (DPTA). Now, finding a single optimum is meaningless, as the costs of two traces, say (c_1, c_2) and (c'_1, c'_2), can be incomparable when e.g. $c_1 < c'_1$ and $c'_2 < c_2$. Thus, minimal cost reachability corresponds to finding the set of incomparable, minimal cost pairs for reaching the goal location. Similarly, we can pose the corresponding maximization problem of determining the set of incomparable, maximal cost of reaching a target location. That implies that if there is a maximal[2] path avoiding the goal, the maximal cost is the single tuple (∞, ∞). We refer to the minimization and maximization problems as synthesizing the minimal and maximal reachability costs. The minimization and maximization problems correspond to the costs generated by optimal strategies in the two extrema of a game where the player, respectively the adversary, has full control. That is, the costs synthesized in the minimization problem are the smallest cost the player can guarantee when he/she decides all the moves, while the costs of the maximization problem are the largest costs the adversary can enforce under complete control.

A specialized problem of the minimal and maximal reachability costs is *optimal conditional reachability*. For minimization, optimal conditional reachability is determining the cheapest primary cost of reaching the "happy" location under some upper bound constraint on the secondary cost. Fig. 2 depicts a model with two cost variables for which we can pose questions of the type: What is the minimum cost for c_1 of reaching the "happy" location while respecting $c_2 \leq 4$. The answer to this question is $\frac{11}{3}$, and is obtained by delaying for $\frac{1}{3}$ time units in l_2, then proceeding to l_2 and waiting $\frac{5}{3}$ time units before proceeding to l_3. This example illustrates that unlike minimum-cost reachability for PTA, optimal conditional reachability with two cost variables may have non-integral solutions.[3]

If we generalize DPTA to allow any finite number of cost variables, we arrive at *multi-priced timed automata* (MPTA), as depicted in Fig. 2. Minimal and maximal reachability for MPTA is a direct extension with the purpose of synthesizing k-tuples of cost. The main contribution of this paper is the decidability of synthesizing minimal and maximal reachability costs for MPTA.

Relevant work on MPTA include the model checking problem of MPTA with respect to weighted CTL, which has been studied by Brihaye et al., [5], and proven to be undecidable, even with discrete time.

The discrete version of minimal and maximal reachability is called multi-constrained routing, and is well-known to be NP-complete, [6]. Recently, the problem has been reconsidered by Puri and Tripakis in [7] where several algorithms are proposed for solving the minimization problem, both exactly and approximately.

For simplicity of proof, we prove decidability of minimal and maximal reachability for MPTA by proving decidability for the simpler DPTA model. To show that the result can be extended from DPTA to MPTA, throughout the paper we provide descriptions of how important aspects are extended from pairs of costs to k-tuples of costs.

[2] A path is maximal if it is deadlocked, infinite, or in a location that allows an infinite delay.

[3] The simple model in Fig. 2 is acyclic, so optimal conditional reachability could be reduced to linear programming.

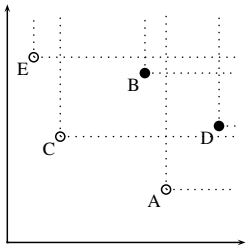

Fig. 3. Domination of points in $\mathbb{R}^2_+$.

This paper extends the authors' results from [8] which only considers minimal-cost conditional optimal reachability for DPTA. In the present paper, the results generalize to optimal reachability, for which conditional reachability is a subset. Furthermore, we also consider the problem of maximal-cost reachability.

The rest of this paper is organized as follows. In Section 2, we give an abstract framework for symbolic reachability in terms of dual-priced transition systems. Section 3 provides generic algorithms for minimal and maximal reachability. In Section 4, we introduce dual-priced timed automata as a syntactic model for dual-priced transition systems. In Section 5, we introduce dual-priced zones as the main construct for dual-priced symbolic states. In Section 6, we define a successor operator on the constructs of the previous section. In Section 7, we discuss termination of our algorithms. Finally, we conclude the paper in Section 8 and point out directions for future research.

2. Minimal and maximal reachability costs

The notation defined in this section aims at being consistent with that of [7].

The partial order, $\preceq$, over $\mathbb{R}^2_+$ (note $(\infty,\infty)\in\mathbb{R}^2_+$), defined such that $(a,b)\preceq(c,d)$ iff $a\leq c$ and $b\leq d$, is called a *domination order*. We will use the notation $(a,b)\prec(c,d)$ iff $(a,b)\preceq(c,d)$, and either $a<c$ or $b<d$. Given a set of points $A\subseteq\mathbb{R}^2_+$, an element $(c,d)\in A$ is said to be *redundant* if there exists another element $(a,b)\in A$ such that $(a,b)\preceq(c,d)$. When extending domination to sets $A,B\subseteq\mathbb{R}^2_+$, we have two constituents of the Egli-Milner ordering as defined below:

$$A\preceq_{\inf}B\iff\forall(c,d)\in B\ :\ \exists(a,b)\in A\ :\ (a,b)\preceq(c,d) \tag{1}$$

$$A\preceq_{\sup}B\iff\forall(a,b)\in A\ :\ \exists(c,d)\in B\ :\ (a,b)\preceq(c,d). \tag{2}$$

Fig. 3 depicts a set of points with black and white bullets denoting, respectively, redundant and non-redundant points. Furthermore, if A is the set of white nodes and B is the set of black nodes, we have $A\preceq_{\inf}B$ but not $A\preceq_{\sup}B$ as point E does not dominate any black point. Note that if point E were included in both A and B, both relationships would hold.

Furthermore, we define the infimum and supremum on sets in $\mathbb{R}^2_+$ in the natural way. That is, for $A\subseteq\mathbb{R}^2_+$:

– $\inf A\subseteq\mathbb{R}^2_+$ contains (a,b) iff

$$\begin{aligned}&\exists(c,d)\in A\ :\ (a,b)\preceq(c,d),\\&\forall(c,d)\in A\ :\ (c,d)\not\prec(a,b),\end{aligned} \tag{3}$$

and for all (c,d) satisfying requirements (3), we have $(a,b)\not\prec(c,d)$.

– $\sup A\subseteq\mathbb{R}^2_+$ contains (a,b) iff

$$\begin{aligned}&\exists(c,d)\in A\ :\ (c,d)\preceq(a,b),\\&\forall(c,d)\in A\ :\ (a,b)\not\prec(c,d),\end{aligned} \tag{4}$$

and for all (c,d) satisfying requirements (4), we have $(c,d)\not\prec(a,b)$.

A dual-priced transition system (DPTS) is a structure $\mathcal{T}=(S,s_0,\Sigma,\rightarrow)$ where S is a, possibly infinite, set of states, s_0 is the initial state, Σ is an alphabet of labels, and $\rightarrow$ is partial function with signature $S\times\Sigma\times S\hookrightarrow\mathbb{R}^2_+$. Without loss of generality, we require all states to have at least one outgoing transition, i.e., $\forall s\in S\ :\ \exists s'\in S\ :\rightarrow(s,a,s')=(c_1,c_2)$ for some a,c_1,c_2. In other words, all states are deadlock free.

For brevity, we use $s \xrightarrow[c_1,c_2]{a} s'$ whenever $\rightarrow(s, a, s') = (c_1, c_2)$. Furthermore, we use the same notation without some of the parameters to denote the existence of these parameters. For example, $s \rightarrow s'$ denotes the existence of c_1, c_2, a such that $\rightarrow(s, a, s') = (c_1, c_2)$.

An execution ε of $\mathcal{T}$ is an infinite sequence

$$\varepsilon = s_0 \xrightarrow[c_1^1,c_2^1]{a_1} s_1 \xrightarrow[c_1^2,c_2^2]{a_2} \cdots \xrightarrow[c_1^n,c_2^n]{a_n} s_n \xrightarrow[c_1^{n+1},c_2^{n+1}]{a_{n+1}} \cdots .$$

In the rest of this paper, we restrict our attention to DPTSs where the sum of each of the costs diverges on all infinite paths.[4] We denote by $\mathcal{E}_{\mathcal{T}}$ the set of all executions in a DPTS, $\mathcal{T}$, and simply $\mathcal{E}$ when the underlying DPTS is understood.

Let ε be the execution above; then the cost of ε with respect some set of target or goal states, $G \subseteq S$, is defined as:

$$\mathsf{Cost}_G(\varepsilon) = \begin{cases} (\infty, \infty) & \text{if } \forall\, i \geq 0 \ : \ s_i \notin G \\ \sum_{i=1}^{n} (c_1^i, c_2^i) & \text{if } \exists\, n \geq 0 \ : \ s_n \in G \wedge \forall\, 0 \leq i < n \ : \ s_i \notin G \end{cases} \tag{5}$$

where the summation is component-wise. Intuitively, the cost of an execution is the least price of reaching a state in G along the execution and (∞, ∞) if the entire execution avoids states in G.

In this paper, we focus on algorithms for synthesizing the maximal and minimal reachability costs of achieving some goal set. Formally, for a set of goals $G \subseteq S$ we want to determine

$$\inf \left\{ \mathsf{Cost}_G(\varepsilon) \ : \ \varepsilon \in \mathcal{E} \right\}, \quad \text{and} \tag{6}$$

$$\sup \left\{ \mathsf{Cost}_G(\varepsilon) \ : \ \varepsilon \in \mathcal{E} \right\}. \tag{7}$$

The intuition behind these synthesis problems is best described in terms of two extrema of the game where the player is trying to minimize the cost of reaching a goal, while the adversary tries to maximize the cost of reaching the goal or prevent it being reached. At each state only the player or the adversary can choose the successor state. Eq. (6) describes the minimal costs that the player can guarantee in the game where the adversary controls no states. Eq. (7) describes the other extreme, i.e., the maximal costs the adversary can guarantee in the game where the player controls no states. Thus, in the second case, the adversary guarantees (∞, ∞) whenever there is an infinite path avoiding the goal.

In order to effectively analyze dual-priced transition systems, we suggest dual-priced symbolic states (or simply symbolic states) of the form (A, π) where $A \subseteq S$ and $\pi : A \rightarrow 2^{\mathbb{R}^2_+}$. Intuitively, the reachability of the symbolic state (A, π) has the interpretation that all s of A are reachable with all costs in $\pi(s)$. To express successors of symbolic states, we use the Post-operator $\mathsf{Post}_a(A, \pi) = (B, \eta)$ where:

$$B \quad = \{s' \mid \exists s \in A : s \xrightarrow{a} s'\}, \quad \text{and} \tag{8}$$

$$\eta(s) \ = \{(c_1 + c, c_2 + c') \mid \exists s' \in A : s' \xrightarrow[c_1,c_2]{a} s \text{ and } \pi(s') = (c, c')\}. \tag{9}$$

A symbolic execution ξ of a dual-priced transition system is a sequence $\xi = (A_0, \pi_0), \ldots, (A_n, \pi_n)$, where $A_0 = \{s_0\}$, $\pi_0(s_0) = \{(0, 0)\}$ and for $1 \leq i \leq n$ we have $(A_i, \pi_i) = \mathsf{Post}_a(A_{i-1}, \pi_{i-1})$ for some $a \in \Sigma$. The correspondence between executions and symbolic executions is captured below:

- For each execution ε of $\mathcal{T}$ ending in s, there is a symbolic execution ξ ending in (A, π) such that $s \in A$ and $\mathsf{Cost}(\varepsilon) \in \pi(s)$.
- Let ξ be a symbolic execution of $\mathcal{T}$ ending in (A, π); then for each s and $(c, c') \in \pi(s)$, there is an execution ε ending in s such that $\mathsf{Cost}(\varepsilon) = (c, c')$.

The above states that symbolic states accurately capture the costs of reaching all states in the state space. Thus, we can use symbolic states to determine (6) and (7). We will use the notation $\mathsf{Cost}(A, \pi)$ to denote the set of all costs in A.

In the following, we will provide two algorithms for determining the minimal and maximal reachability costs for reaching a set of goal states. The reason we provide different algorithms for the two problems is that guaranteeing

[4] The requirement for cost divergence on all infinite paths is used for simplicity, but our results also extend to the case where costs converge in a finite number of steps on infinite paths.

termination for the maximality problem involves cycle detection whereas the termination of the minimum problem involves providing a well-quasi ordering over symbolic states. Since symbolic states contain possibly uncountably many states with corresponding cost information, we require symbolic states to exhibit certain properties in order to be algorithmically manageable.

Property 1. *Symbolic states should be a well-defined collection such that:*

A. Any symbolic state (A, π) *can be represented effectively.*
B. The initial state $(\{s_0\}, \pi_0)$ *is a symbolic state where* $\pi_0 = \lambda x.\{(0, 0)\}$.
C. Symbolic states are closed under the Post*-operator.*
D. Sets of costs, e.g. $\mathsf{Cost}(A, \pi)$*, are effectively computable and representable. Furthermore, we can compute* inf *and* sup *on costs sets and retain an effective representation.*

Furthermore, the algorithms for computing the maximal reachability costs require that:

I. The set of symbolic states without cost information is finite.
II. Symbolic states are closed under intersection and subtraction with goal states.

3. Symbolic reachability algorithms

3.1. Minimal reachability costs

We define the relation $\sqsubseteq_{\inf}$ on symbolic states such that $(B, \eta) \sqsubseteq_{\inf} (A, \pi)$ iff $A \subseteq B$ and $\eta(s) \preceq_{\inf} \pi(s)$ for all $s \in A$. In other words, B is bigger than A and for each state $s \in A$, all $\pi(s)$ are dominated by $\eta(s)$.

Algorithm 1 computes the minimal reachability costs.

Algorithm 1. General algorithm for computing the minimal reachability costs of reaching G.

proc Minimal $\equiv$
  COST $\leftarrow \{(\infty, \infty)\}$
  PASSED $\leftarrow \emptyset$
  WAITING $\leftarrow \{(\{s_0\}, \pi_0)\}$
  while WAITING $\neq \emptyset$ **do**
    $(A, \pi) \leftarrow$ takeAny(WAITING)
    COST $\leftarrow \inf\big($ COST $\cup\, \mathsf{Cost}(A \cap G, \pi)\big)$
    if $\forall\, (B, \eta) \in$ PASSED $: (B, \eta) \not\sqsubseteq_{\inf} (A, \pi)$ **then**
      WAITING $\leftarrow$ WAITING $\cup \bigcup_{a \in \Sigma} \mathsf{Post}_a(A, \pi)$
      PASSED $\leftarrow$ PASSED $\cup \{(A, \pi)\}$
    end if
  end while
  return COST

The algorithm maintains two lists, PASSED and WAITING, that hold the states already explored and the states waiting to be explored, respectively. Initially, the PASSED list is empty and the WAITING list contains only the initial state. The algorithm iterates as long as the WAITING list is non-empty.

At each iteration, the algorithm selects a state (A, π) from the WAITING list. The set of states is checked for intersection with the set of goal states. If the intersection is non-empty, the costs of all goal states in A are added to COST and we apply infimum on the result.

Whether A intersects with the goal states or not, we go through the PASSED list and check whether it contains any (B, η) such that $(B, \eta) \sqsubseteq_{\inf} (A, \pi)$. If it does, (A, π) is discarded as it is dominated by (B, η) Otherwise, we add all successors of (A, π) to the WAITING list and add it to the PASSED list.

The algorithm terminates when the WAITING list is empty and at this point, COST holds $\inf\{\mathsf{Cost}_G(\varepsilon) \;:\; \varepsilon \in \mathcal{E}\}$. Termination of the algorithm is guaranteed if $\sqsubseteq_{\inf}$ is a well-quasi ordering on symbolic states.

For optimization of the algorithm, further pruning of elements in the WAITING list can be performed simultaneously with the inclusion check, e.g., keeping only elements where the set of states have costs that not redundant in COST. This is correct since both primary and secondary costs increase monotonically in any trace. Furthermore, for any encountered pair (A, π) with $s \in A$, we could prune $\pi(s)$ for redundant elements.

3.2. Maximal reachability costs

We provide two algorithms for synthesizing the maximal reachability costs, a general algorithm and an algorithm under the liveness assumption that all paths will eventually reach a goal location. We present the latter algorithm first.

Algorithm 2 synthesizes the maximal reachability costs for reaching a set of goals under the assumption that the liveness property holds, i.e., any path will eventually reach the goal. The idea is that this property can be independently checked in a more inexpensive manner without involving the cost information.

Algorithm 2. Algorithm for synthesizing the maximal reachability costs for reaching some goal under the assumption that all paths eventually reach the goal.

proc MaximalReach ≡
Require: All paths eventually reach a goal
  COST ← {(0, 0)}
  WAITING ← {({s_0}, π_0)}
  while WAITING ≠ ∅ **do**
    (A, π) ← takeAny(WAITING)
    COST ← sup(COST ∪ $\mathsf{Cost}(A, \pi)$)
    WAITING ← WAITING ∪ $\bigcup_{a \in \Sigma} \mathsf{Post}_a(A \backslash G, \pi)$
  end while
  return COST

Like Algorithm 1, the algorithm maintains a WAITING list of states waiting to be explored, initially containing only the initial state. Emptiness WAITING terminates the main loop of the algorithm. The set of costs, COST, initially contains only the element (0, 0). In each iteration of the algorithm, a symbolic state (A, π) is removed from the WAITING list. COST is updated by applying sup to the union of itself and the costs of all states in A, and not just for states intersecting the goal as in Algorithm 1. Updating only according to encountered goal states would be correct; however, we choose the above approach due to a subtlety described in Section 7. For now, it suffices to understand that updating according to all states is correct because we know that any state in A will eventually reach the goal with a cost that dominates the current costs and COST is updated accordingly.

Since we only compute the successors of non-goal states, termination is guaranteed by the liveness assumption as the set we apply the Post operator to, eventually, is empty on any path. Correctness follows from the fact that all reachable goal states get added to WAITING and COST is updated appropriately every time a goal state is removed from WAITING.

Algorithm 2 above assumes that all paths eventually reach the goal. Obviously, this assumption does not hold in general, and in the following we provide an algorithm for synthesizing the maximal reachability cost for reaching some goal and $\{(\infty, \infty)\}$ if there is an infinite path avoiding goal states. That is, the algorithm checks the liveness assumption from Algorithm 2 while synthesizing the maximal reachability costs along the way. The algorithm is inspired by a time bounded liveness synthesis algorithm for timed automata, [9].

Algorithm 3 is built around an algorithm for finding infinite paths avoiding goal states. This is done in a depth-first manner by maintaining a STACK of visited states without cost information. Initially, STACK is empty and COST contains only (0, 0) and calling Search on the initial symbolic state. Ignoring lines 1–3 and 5, we see that Search is a classical recursive depth-first search algorithm restricted to only non-goal states where STACK holds the path from the initial state to the current state in the search tree. Lines 1–3 apply a procedure onStack to check where the current set of states is already on the stack. In such case, the algorithm terminates, returning $\{(\infty, \infty)\}$ since we have arrived at a state set already on STACK, thus proving the existence of an infinite path avoiding the goal.

Like the previous algorithm, COST is updated for every explored state in A, but unlike Algorithm 2, we are not guaranteed that all states eventually reach the goal. However, if a state does not reach the goal, it belongs to an infinite

Algorithm 3. General algorithm for computing the maximal reachability costs for reaching a goal. STACK and COST are global variables. Note that **exit** terminates the entire recursion stack and not just the current level.

```
proc MaximalLive ≡
  COST ← {(0, 0)}
  STACK ← ∅
  Search({s_0}, π_0)
  return COST
proc Search(A, π) ≡
 1: if onStack(STACK,A) then
 2:    COST ← {(∞, ∞)}; exit
 3: end if
 4: push(STACK,A \ G)
 5: COST ← sup { COST ∪ Cost(A, π) }
 6: for all a ∈ Σ do
 7:    Search(Post_a(A\G, π));
 8: end for
 9: pop(STACK)
```

path avoiding the goal, in which case the algorithm returns $\{(\infty, \infty)\}$ when this is detected. Thus, the potentially erroneous updating of COST is temporary and unimportant.

Termination is guaranteed as the set of symbolic states without cost information is finite; thus any symbolic path is either infinite and will contain a loop of symbolic states detected by onStack, or it is finite. Correctness is guaranteed by the fact that all reachable goal states are reached while updating COST appropriately, or the algorithm terminates early with $\{(\infty, \infty)\}$ due to the existence of a infinite path avoiding the goal state(s).

The algorithm can be easily optimized by using a PASSED list of explored states known not to contain any infinite loop. Any symbolic state (A, π) can be pruned before computing successors if there is a $(B, \eta) \in$PASSED such that $A \subseteq B$ and $\pi(s) \preceq_{\text{sup}} \eta(s)$ for all $s \in A$. States are added to PASSED when they are popped from STACK.

3.3. Related optimality problems

Using the above algorithms for finding minimal and maximal reachability costs, we can compute the related problems of conditional optimal reachability.

The minimal (resp. maximal) primary cost of reaching a set of goal states, $G \subseteq S$, under an upper (resp. lower) bound, p, on the secondary cost is termed the conditional minimal (resp. maximal) cost and given as:

$$\mathsf{mincost}_{\leq p}(G) = \inf\{c_1 \mid \exists\, \varepsilon \in \mathcal{E} \;:\; (c_1, c_2) \in \mathsf{Cost}_G(\varepsilon) \wedge c_2 \leq p\}$$
$$\mathsf{maxcost}_{\geq p}(G) = \sup\{c_1 \mid \exists\, \varepsilon \in \mathcal{E} \;:\; (c_1, c_2) \in \mathsf{Cost}_G(\varepsilon) \wedge c_2 \geq p\}.$$

The above conditional optimal reachability problems are computable using either the algorithm for minimal or maximal reachability costs.

Furthermore, note as Algorithms 2 and 3 has no pruning based on costs, we could choose to store all cost pairs for all encountered goal states. Thus, we are able to compute all combinations of costs for reaching the goal, unless there is an infinite loop avoiding the goal, in which case the infinity tuple is returned. In other words, the algorithm would compute:

$\{(\infty, \infty)\}$ if there is an infinite path avoiding G, and
$\{\mathsf{Cost}_G(\varepsilon) \mid \varepsilon \in \mathcal{E}\}$ otherwise.

3.4. Framework instantiation

Every aspect in these sections about transition systems, including the generic algorithms, can be directly extended to multi-priced transition systems with k-tuples of costs.

The above framework may be instantiated by providing a concrete syntax for dual-priced transition systems and data structures for symbolic states that exhibit the properties listed in Property 1. In the following sections, we provide such an instantiation of the above framework.

4. Dual-priced timed automata

In this section, we define dual-priced timed automata which constitute a proper subset of linear hybrid automata, [2], and a proper superset of priced timed automata, [3], or weighted timed automata, [4], and in turn timed automata, [1]. DPTA will serve as a concrete syntax for dual-priced transition systems. First, however, we recall some basic notation from the theory of timed automata.

We work with a finite set $\mathcal{X}$ of positive, real-valued variables called clocks. $\mathcal{B}(\mathcal{X})$ is the set of formulae obtained as conjunctions of atomic constraints of the form $x \bowtie n$, where $x \in \mathcal{X}$, $n \in \mathbb{N}$, and $\bowtie \in \{\leq, =, \geq\}$.[5] We refer to the elements of $\mathcal{B}(\mathcal{X})$ as clock constraints. $\mathcal{U}(\mathcal{X})$ is the set of clock constraints involving only upper bounds.

Clock values are represented as functions from $\mathcal{X}$ to the set of non-negative reals $\mathbb{R}_+$ called clock valuations and ranged over by u, u' etc.

For a clock valuation $u \in (\mathcal{X} \to \mathbb{R}_+)$ and a clock constraint $g \in \mathcal{B}(\mathcal{X})$, we write $u \in g$ when u satisfies all the constraints of g. For $t \in \mathbb{R}_+$, we define the operation $u + t$ to be the clock valuation that assigns $u(x) + t$ to all clocks, and for $R \subseteq \mathcal{X}$ the operation $u[R \to 0]$ to be the clock valuation that agrees with u for all clocks in $\mathcal{X} \setminus R$ and assigns zero to all clocks in R. $u[x \to 0]$ is shorthand for $u[\{x\} \to 0]$. Furthermore, u_0 is defined to be the clock valuation that assigns zero to all clocks.

Definition 1 (*Dual-Priced Timed Automata*). A dual-priced timed automaton is a 6-tuple $\mathcal{A} = (L, l_0, \mathcal{X}, E, I, \vec{P})$ where $\vec{P} = \{\mathcal{P}_1, \mathcal{P}_2\}$,[6] L is a finite set of locations, l_0 is the initial location, $\mathcal{X}$ is a finite set of clocks, $E \subseteq L \times \mathcal{B}(\mathcal{X}) \times 2^{\mathcal{X}} \times (\mathbb{N} \times \mathbb{N}) \times L$ is the set of edges, $I : L \to \mathcal{U}(\mathcal{X})$ assigns invariants to locations, and $\mathcal{P}_i : L \to \mathbb{N}$ assigns prices to locations, $i \in \{1, 2\}$.

The concrete state semantics of a DPTA $\mathcal{A} = (L, l_0, \mathcal{X}, E, I, \vec{P})$ are given in terms of a dual-priced transition system with state set $L \times (\mathcal{X} \to \mathbb{R}_+)$, initial state (l_0, u_0), alphabet $\Sigma = E \cup \{\delta\}$, and the transition relation $\to$ defined as:

- $(l, u) \xrightarrow[t \cdot \mathcal{P}_1(l), t \cdot \mathcal{P}_2(l)]{\delta} (l, u + t)$ if $\forall\, 0 \leq t' \leq t : u + t' \in I(l)$ and
- $(l, u) \xrightarrow[c, c']{e} (l', u')$ if $e = (l, g, R, (c, c'), l') \in E, u \in g, u' = u[R \to 0]$.

We will often write concrete states as (l, u, c_1, c_2) to denote the assumption of some underlying execution ε ending in (l, u) with $\mathsf{Cost}(\varepsilon) = (c_1, c_2)$. We deal with the fact that the DPTS defined above is not deadlock-free after we have defined dual-priced symbolic states in the following section.

A concrete dual-priced state (l, u, c_1, c_2) is said to dominate another state (l', u', c_1', c_2') iff $l = l'$, $u = u'$, and $(c_1, c_2) \preceq (c_1', c_2')$. In such cases, we write $(l, v, c_1, c_2) \preceq (l', v', c_1', c_2')$.

For convenience reasons, we assume some restrictions on the structure of the DPTA in the rest of the paper. First, any DPTA should be bounded, i.e., all locations have upper bound invariants on all clocks. Second, at least one clock is reset on every transition. Note that neither restriction compromises the generality of our result, as it is well-known that any TA can be transformed into a semantically equivalent bounded TA, and that result extends directly to DPTA. Furthermore, the reset assumption can be guaranteed by introducing an extra clock which is reset on every transition. Finally, for simplicity we assume that the goal of any reachability problem is a single location without requirements on clock values for the location. In Section 7, we describe how to lift this restriction.

4.1. Relation to linear hybrid automata

Any DPTA is an LHA where the value of the rate of each clock variable is one in every location, and the rates in location l of the primary and secondary costs are $\mathcal{P}_1(l)$ and $\mathcal{P}_2(l)$, respectively.

[5] For simplification, we do not include strict inequalities; note, however, that everything covered in this paper extends directly to strict inequalities, which is why we compute infimum costs as opposed to minimum costs.

[6] If we let $\vec{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_k\}$, we have an MPTA with analogous semantics.

Tools such as HYTECH, [10], can perform forward symbolic reachability analysis on LHA over a set of variables $\vec{x}$ using symbolic state structures (l, A, b) where l is a location and $A \cdot \vec{x} \leq b$ defines a convex polyhedra of valid variable assignments. One of the main properties of this kind of reachability analysis is that the Post operators defined for LHA maintains convexity of the state set. However, the reachability problem for LHA is, in general, undecidable, so termination of the reachability algorithm is not guaranteed. However, a consequence of our result is that for the class of DPTAs, HYTECH will terminate when performing conditional reachability.

5. Dual-priced zones

We propose dual-priced zones as a syntactic construct for providing a symbolic semantics for the dual-priced transition system induced by DPTA.

The constructs of our proposal for dual-priced symbolic states are *zones* and *cost functions*. Zones are well known from the analysis of timed systems, and efficient implementations of zones as difference bound matrices are used in real-time verification tools such as KRONOS, [11], and UPPAAL, [12]. Briefly, zones are convex collections of clock valuations that can be described solely using difference constraints of the form $x_i - x_j \leq m$ where $m \in \mathbb{Z}$ and $x_i, x_j \in \mathcal{X} \cup \{x_0\}$. x_0 is a special clock whose value is fixed to zero. That way, constraints of the form $x_i \geq n$ can be written as $x_0 - x_i \leq -n$, and similarly for other constraints involving a single variable. Zones are ranged over by $Z, Z_1, Z', \ldots$. When a clock valuation u satisfies the difference constraints of a zone Z, we write $u \in Z$.

The second construct is a cost function, which is an affine function over $\mathcal{X}$, i.e., a cost function d is a function with signature $(\mathcal{X} \to \mathbb{R}_+) \to \mathbb{R}_+$ that can be written syntactically as $a_1 \cdot x_1 + \cdots + a_n \cdot x_n + b$ where $x_i \in \mathcal{X}$, $1 \leq i \leq n$ and $a_i, b \in \mathbb{Z}$. The cost of a clock valuation u in a cost function d is given by $d(u) = a_1 \cdot u(x_1) + \cdots + a_n \cdot u(x_n) + b$. We range over cost functions by d, e, d_1, e_1, d', e' etc. For ease of notation, we define a number of operations on cost functions. Let $m \in \mathbb{Z}$, $p \in \mathbb{N}$ and $x_i, x_j \in \mathcal{X}$; then the substitution operation $d[x_i/\varphi]$ for $\varphi \in \{m, x_j + m\}$ is defined as $d[x_i/\varphi] = a_1 \cdot x_1 + \cdots + a_i \cdot \varphi + \cdots + a_n \cdot x_n$. The delay operation $d^{\uparrow p, x_i}$ is defined as $d^{\uparrow p, x_i} = a_1 \cdot x_1 + \cdots + (p - \sum_{j \neq i} a_j) \cdot x_i + \cdots + a_n \cdot x_n$, meaning we want the sum of the coefficients to match p by assigning the correct coefficient to x_i.

Let C be a set of pairs of cost functions, i.e. $C = \{(e_1, d_1), \ldots, (e_k, d_k)\}$ and u a clock valuation, then $C(u) = \{(e_1(u), d_1(u)), \ldots (e_k(u), d_k(u))\}$ is a set of points in $\mathbb{R}^2_+$. We denote by $\lambda(C(u))$ the set of all convex combinations of $C(u)$, i.e. the convex hull.

For the construction of dual-priced symbolic states, we propose dual-priced zones as given in Definition 2 below.

Definition 2 (*Dual-Priced Zone*). A dual-priced zone is a pair (Z, C) where Z is a zone and C is a finite set of pairs of cost functions $\{(e_1, d_1), \ldots, (e_k, d_k)\}$.

We construct dual-priced symbolic states as structures (l, Z, C) where l is a location and (Z, C) is a dual-priced zone. A dual-priced symbolic state (l, Z, C) comprises all concrete states (l', u, c_1, c_2) where $l' = l$, $u \in Z$, and $(c_1, c_2) \in \lambda(C(u))$. The dual-priced states defined above satisfy Property 1A that all dual-priced symbolic states are effectively representable, since the set of cost functions is finite and cost functions can be represented as the a vector of coefficients to the clocks. Furthermore, the initial state $(l_0, v_0, \{(\vec{0}, \vec{0})\})$ where $\vec{0}$ is the zero vector is a dual-priced symbolic state, thus satisfying Property 1B. Property 1I states that for the maximization algorithms, the set of symbolic states without cost information has to be finite. In Section 4, the DPTAs were restricted to be fully bounded which is well known to guarantee a finite number of zones, thus satisfying the property. Finally, the reachability goal has been restricted to locations, thus trivially satisfying Property 1II that symbolic states are closed under intersection and subtraction with goal states.

Recall that the DPTS defined in the previous section is not deadlock-free. We could deal with this by defining a new kind of transition and add a self-loop to each deadlocked state. For dual-priced symbolic states, that would correspond to an identical successor with consistently higher cost. But since dual-priced symbolic states involve only a single location, we can detect whether any given dual-priced symbolic state is deadlocked and return (∞, ∞) in the maximization algorithms. In the rest of the paper, we assume that this problem is treated in either way and we do not consider it further.

Note that dual-priced zones extend directly to multi-priced zones with k-tuples of cost functions and, in turn, multi-priced symbolic states.

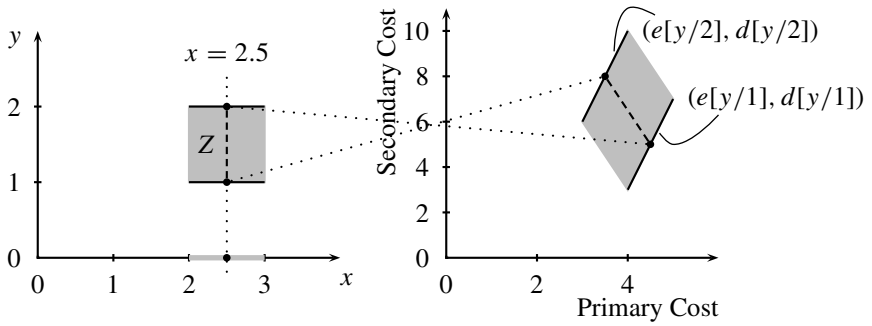

Fig. 4. The relationship between the zone, Z, defined by the constraints $2 \leq x \leq 3$ and $1 \leq y \leq 2$ with cost functions (e, d) with $e = x + y$ and $d = 4x - 3y + 1$.

In [3], efficient data structures for symbolic minimum-cost reachability for priced timed automata (PTA) are provided. These are so-called priced zones which effectively are zones Z with an associated cost function e. For representing cost in the discrete case described in [7], subsets of $\mathbb{N} \times \mathbb{N}$ are used for representing reachability costs.

The immediate combination of the two suggest the use of zones together with sets of pairs of cost functions. The following example illustrates why we also need to consider convex combinations of the cost functions.

Consider the zone of Fig. 4 described by the constraints $2 \leq x \leq 3$ and $1 \leq y \leq 2$ with the pair of cost functions (e, d) where $e = x + y$ and $d = 4x - 3y + 1$. Now, if we need to compute the projection of the zone onto the first axis due to a reset of y, what should the set of pairs of cost functions be to represent or dominate the possible cost values? The suggestion following the lines of reasoning from [3] would be to use the two pairs of cost functions $(e[y/2], d[y/2])$ and $(e[y/1], d[y/1])$. This choice, however, has a loss of information if we do not allow convex combinations. The point $(x = 2.5, y = 0)$ is obtained from Z by projection from any point satisfying $(x = 2.5, 1 \leq y \leq 2)$ corresponding to costs given by any convex combination between $(3.5, 8)$ and $(4.5, 5)$. However, maintaining only the these two points is incorrect, as neither of the points dominate any point in their convex combination.

6. Post operator

The projection operation in the previous section serves as a first step toward a Post operator. Consider the zone in Fig. 4, and assume it is associated with two pairs of cost functions (e_1, d_1) and (e_2, d_2), between which we allow arbitrary convex combinations. Now, if we perform a projection onto the first axis, we split each pair of cost functions in two, i.e. (e_i^L, d_i^L) and (e_i^U, d_i^U), $i \in \{1, 2\}$, corresponding to the lines $L : y = 1$ and $U : y = 2$, respectively, giving four cost functions. Originally, for any clock valuation u in the zone and $0 \leq \alpha \leq 1$, the convex combination between $(e_1(u), d_1(u))$ and $(e_2(u), d_2(u))$ wrt. α is a valid cost pair. However, when we split the cost functions, the cost corresponding to e.g. $(e_1(u), d_1(u))$ is given by some convex combination of (e_1^L, d_1^L) and (e_1^U, d_1^U) for the clock valuation $u[y \to 0]$, and similarly for $(e_2(u), d_2(u))$ using the same convex combination. Contrary to the definition of dual-priced zones, this suggests not allowing arbitrary convex combinations between (e_1^L, d_1^L) and (e_1^U, d_1^U), (e_2^L, d_2^L) and (e_2^U, d_2^U), but rather "binary tree" convex combinations of the form: Choose the same convex combination between (e_1^L, d_1^L), (e_1^U, d_1^U) and (e_2^L, d_2^L), (e_2^U, d_2^U) and take any convex combination of the resulting pairs. However, the following key lemma states that if this set is convex, it is identical to the set of arbitrary convex combinations between the four.

Lemma 1. *Assume a set of pairs of points in* $\mathbb{R}_+^2$

$$\{(a_1, b_1), \ldots, (a_n, b_n)\}, \quad a_i \in \mathbb{R}_+^2, b_i \in \mathbb{R}_+^2, 1 \leq i \leq n.$$

For $0 \leq \alpha \leq 1$, *let:*

$$A_\alpha = \{\alpha \cdot a_i + (1 - \alpha) \cdot b_i | 1 \leq i \leq n\} \quad \text{and}$$
$$B = \{a_i, b_i | 1 \leq i \leq n\}.$$

Now, if $\bigcup_\alpha \lambda(A_\alpha)$ *is convex (i.e.* $\bigcup_\alpha \lambda(A_\alpha) = \lambda(\bigcup_\alpha \lambda(A_\alpha))$*) then* $\bigcup_\alpha \lambda(A_\alpha) = \lambda(B)$.

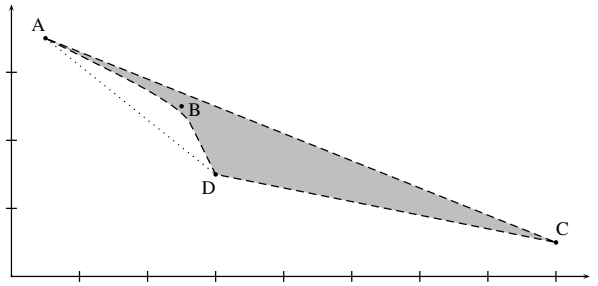

Fig. 5. Counter example.

Proof. We prove the lemma in two steps. First, we show that $\bigcup_\alpha \lambda(A_\alpha) \subseteq \lambda(B)$ and, secondly, that $\lambda(B) \subseteq \bigcup_\alpha \lambda(A_\alpha)$.
1. Let c be a convex combination of A_α for any $0 \leq \alpha \leq 1$; that is,

$$c = \lambda_1(\alpha a_1 + (1-\alpha)b_1) + \cdots + \lambda_n(\alpha a_n + (1-\alpha)b_n) \tag{10}$$
$$= \lambda_1\alpha a_1 + \lambda_1(1-\alpha)b_1 + \cdots + \lambda_n\alpha a_n + \lambda_n(1-\alpha)b_n, \tag{11}$$

where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$. Now, (11) is a convex combination of B; thus, $c \in \lambda(B)$, and in turn $\bigcup_\alpha \lambda(A_\alpha) \subseteq \lambda(B)$.
2. Each point a_i can be given as a convex combination of A_α where $\alpha = 1$ using $\lambda_i = 1$ and $\lambda_j = 0$ for $j \neq i$. Similarly for b_i with $\alpha = 0$. Now, since all a_i, b_i are included in the convex set $\bigcup_\alpha \lambda(A_\alpha)$, we know that $\lambda(B) \subseteq \lambda(\bigcup_\alpha \lambda(A_\alpha)) = \bigcup_\alpha \lambda(A_\alpha)$. □

Note that the proof makes no mention of $\mathbb{R}^2_+$; thus, the Lemma 1 is directly extendable to pairs of points in $\mathbb{R}^k_+$.

At first glance, $\bigcup_\alpha \lambda(A_\alpha)$ in Lemma 1 might seem universally convex; however, Fig. 5 depicts the contrary, where Lemma 1 does not hold. Let $P = \{(A, B), (C, D)\}$, now, $\bigcup_\alpha \lambda(P_\alpha)$ (the gray area with the dashed line) is not convex and not equal to $\lambda(\{A, B, C, D\})$, particularly, all points on the line from A to D are not included in the former.

Before defining the Post operator on dual-priced states of the form (l, Z, C), we need to introduce a number of definitions and operations. Let Z be a zone, the delay operation $Z^\uparrow$ and the reset $\{x\}Z$ with respect to a clock $x \in \mathcal{X}$ are defined as $Z^\uparrow = \{u + t | u \in Z \text{ and } t \geq 0\}$ and $\{x\}Z = \{u[x \to 0] | u \in Z\}$. It is well known from timed automata that both $Z^\uparrow$ and $\{x\}Z$ are representable as zones.

Given a zone Z, if $x_i - x_j \leq m$ is a constraint in Z, then $(Z \wedge (x_i - x_j = m))$ is a *facet* of Z, a *lower relative facet* of x_j, and an *upper relative facet* of x_i. The set of lower (resp. upper) relative facets of a clock x_i in a zone Z is denoted $LF_{x_i}(Z)$ (resp. $UF_{x_i}(Z)$).

The following lemma for facets is proven in [3].

Lemma 2. *Let Z be a zone over a clock set, $\mathcal{X}$, with $x \in \mathcal{X}$; then:*

(1) $Z^\uparrow = \bigcup_{F \in UF_{x_0}(Z)} F^\uparrow = Z \cup \bigcup_{F \in LF_{x_0}(Z)} F^\uparrow$ *and*
(2) $\{x\}Z = \bigcup_{F \in LF_x(Z)} \{x\}F = \bigcup_{F \in UF_x(Z)} \{x\}F$.

Lemma 2(1) is most intuitively understood knowing that x_0 is fixed to zero; that way UF_{x_0} is the set of all lower bound constraints on clocks in $\mathcal{X}$ (i.e. $x \geq n$) and LF_{x_0} is the set of all upper bound constraints on clocks in $\mathcal{X}$ (i.e. $x \leq n$).

Definition 3. Given a zone Z and a clock x, $LUF_x(Z)$ is the unique smallest collection of pairs $\{(L_1, U_1), \ldots, (L_n, U_n)\}$ such that for all $1 \leq i, j \leq n, i \neq j$ we have (i) $L_i \cap L_j = U_i \cap U_j = \emptyset$, (ii) $\{x\}L_i = \{x\}U_i$, and (iii) $L_i \subseteq F$, $U_i \subseteq F'$ for some $F \in LF_x(Z)$ and $F' \in UF_x(Z)$.

We call the elements of $LUF_x(Z)$ partial relative facets with regard to x. Fig. 6 illustrates the concept of partial relative facets.

Let d be a cost function and let F be a relative facet of a zone in the sense that $x_i - x_j = m$ (or $x_i = m$) is a constraint in F; then we use the shorthand notation d^F for $d[x_i/x_j + m]$ (or $d[x_i/m]$).

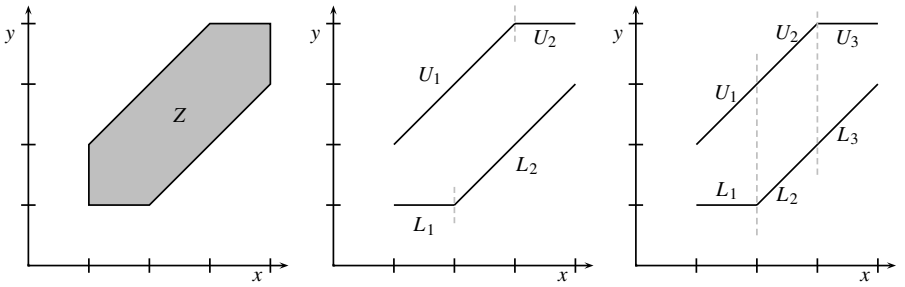

Fig. 6. From left to right (i): a zone, Z, (ii): $LF_y(Z) = \{L_1, L_2\}$ and $UF_y(Z) = \{U_1, U_2\}$ (iii): $LUF_y = \{(L_1, U_1), (L_2, U_2), (L_3, U_3)\}$.

Definition 4 (Post *Operator*). Let $\mathcal{A} = (L, l_0, \mathcal{X}, E, I, \vec{P})$ be a DPTA with $l \in L$ and $e = (l, g, \{x\}, (c, c'), l') \in E$,[7] let Z be zone, let Z' be a zone where $x \in \mathcal{X}$ is fixed at zero, and let $C = \{(e_1, d_1), \ldots, (e_k, d_k)\}$ be a set of pairs of cost functions; then

$$\mathsf{Post}_\delta(l, Z', C) = \left\{(l, (Z' \wedge I(l))^\uparrow \wedge I(l), \{(e_i^{\uparrow \mathcal{P}_1(l),x}, d_i^{\uparrow \mathcal{P}_2(l),x}) | 1 \le i \le k\})\right\}$$

$$\mathsf{Post}_e(l, Z, C) = \bigcup_{(L,U) \in LUF_x(Z \wedge g)} \left\{(l', \{x\}(U), C')\right\}$$

where $C' = \{(e_i^L + c, d_i^L + c'), (e_i^U + c, d_i^U + c') | 1 \le i \le k\}$.

The simplification of the Post_δ operator is no restriction given the reset assumption we made in Section 4: we simply just allow Post_δ after a Post_e, which is, actually, how symbolic reachability is performed in tools such as UPPAAL and KRONOS. The Post operator as given above extends directly to multi-priced zones and the binary split in Post_e remains binary.

As shorthand notation, we write $(l, u, c_1, c_2) \in \mathsf{Post}_e(l, Z, C)$ to indicate that $(l, u, c_1, c_2) \in (l', Z', C')$ for some $(l', Z', C') \in \mathsf{Post}_e(l, Z, C)$.

Before we prove the soundness and completeness of the Post operator, we illustrate in Fig. 7 its behavior on the running example of Fig. 2.

Lemma 3. *Given dual-priced symbolic state (l, Z, C) where $C = \{(e_1, d_1), \ldots, (e_k, d_k)\}$ and $a \in \{e, \delta\}$ where $e = (l, g, \{x\}, (c, c'), l')$, we have*

$$(l', u', c_1', c_2') \in \mathsf{Post}_a(l, Z, C) \iff$$
$$\exists (l, u, c_1, c_2) \in (l, Z, C) : (l, u, c_1, c_2) \xrightarrow{a} (l', u', c_1', c_2').$$

Proof. We choose only to prove the lemma for Post_e, as the analogous proof for Post_δ is straightforward, since each concrete successor has a unique concrete predecessor, given the requirement that Post_δ is always applied after a clock reset. We prove each direction of the bi-implication separately.

$\Longleftarrow$ - *Completeness*: Let $(l, u, c_1, c_2) \in (l, Z, C)$. The costs (c_1, c_2) are given as a convex combination of $C(u)$, i.e. there are $0 \le \lambda_i \le 1$ and $\sum_i \lambda_i = 1$ for $1 \le i \le k$ such that:

$$(c_1, c_2) = \sum_i \lambda_i \cdot (e_i(u), d_i(u)). \tag{12}$$

The discrete successor of (l, u, c_1, c_2) with respect to e is given as $(l', u[x \to 0], c_1 + c, c_2 + c')$, which we will now prove is contained in $\mathsf{Post}_e(l, Z, C)$.

[7] For the general case with multiple resets, we consecutively split the pairs of cost functions for each clock that is reset.

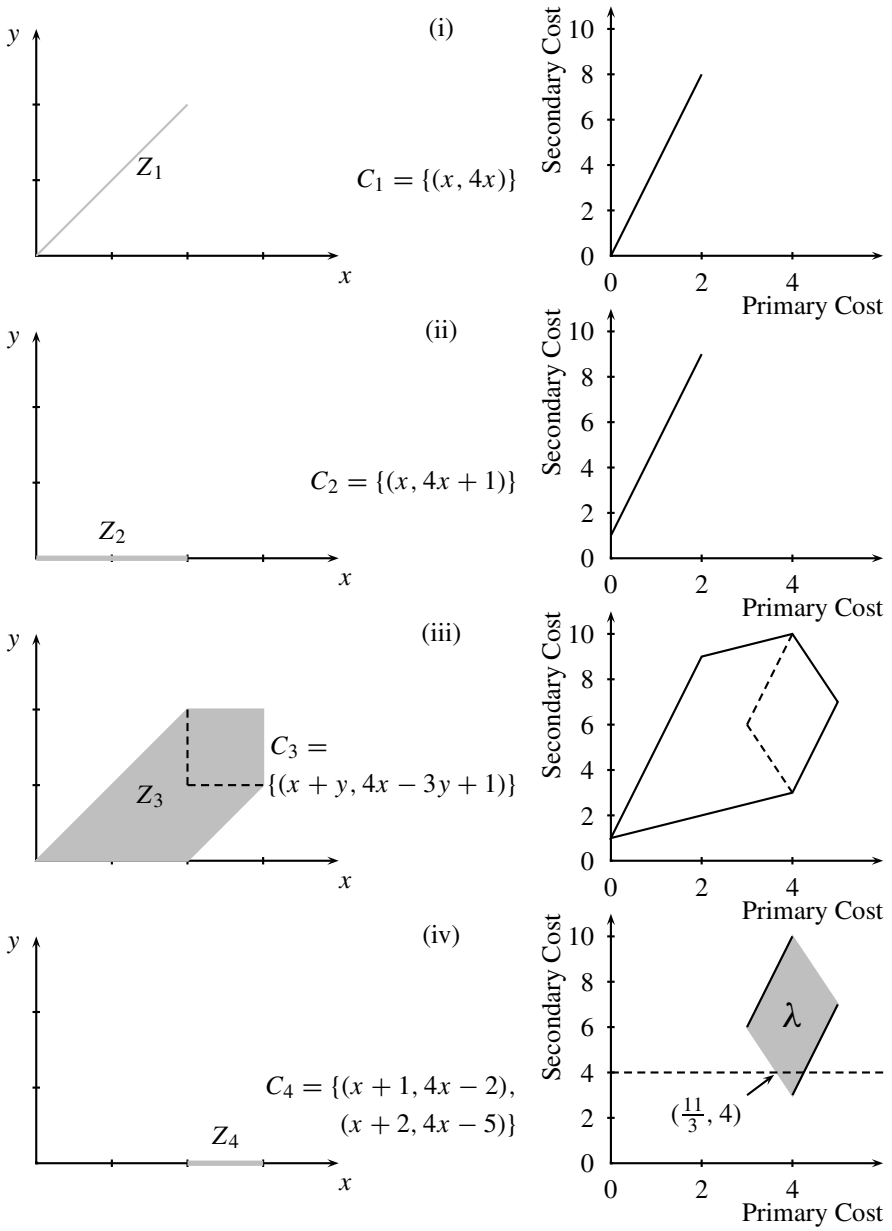

Fig. 7. Reachability analysis for $\mathsf{mincost}_{\leq 4}(\{l_3\})$ on the DPTA in Fig. 2 starting from the initial state (l_1, Z_0, C_0). Areas enclosed by black lines in the cost part indicate all cost pairs computable from the cost functions. (i) $(l_1, Z_1, C_1) = \mathsf{Post}_\delta(l_1, Z_0, C_0)$ (ii) $(l_2, Z_2, C_2) = \mathsf{Post}_e(l_1, Z_1, C_1)$ where $e = (l_1, -, \{y\}, (0, 1), l_2)$ (iii) $(l_2, Z_3, C_3) = \mathsf{Post}_\delta(l_2, Z_2, C_2)$. The dashed area indicates the subset of the zone satisfying the guard of $e' = (l_2, x \geq 2 \wedge y \geq 1, \{y\}, (0, 0), l_3)$ (iv) $(l_3, Z_4, C_4) = \mathsf{Post}_{e'}(l_2, Z_3, C_3)$. The gray area in the cost part indicates the convex combinations between the lines describing the two cost functions. The cost pairs below the dashed line are the ones satisfying the constraint on the secondary cost. Note that $\mathsf{mincost}_{\leq 4}(\{l_3\}) = \frac{11}{3}$.

Let $(L, U) \in LUF_x(Z)$ such that $u[x \to 0] \in \{x\}L$. Given the convexity of zones, there exist a unique $v \in L$ and $w \in U$ where $v(x) \leq u(x) \leq w(x)$ and $u(y) = v(y) = w(y)$ for $y \neq x$, i.e., $u(x) = \alpha \cdot v(x) + (1 - \alpha) \cdot w(x)$ for some $0 \leq \alpha \leq 1$. Furthermore, the affinity of cost functions provide us with

$$(e_i(u), d_i(u)) = \alpha \cdot (e_i(v), d_i(v)) + (1 - \alpha) \cdot (e_i(w), d_i(w)), \tag{13}$$

for all $1 \leq i \leq k$ and the same α as above.

Now, choose $(l', u', c'_1, c'_2) \in \mathsf{Post}_e(l, Z, C)$ where $u' = u[x \to 0]$ and (c'_1, c'_2) is given by (14), which we can rewrite as:

$$\sum_i \lambda_i \cdot (\alpha \cdot (e_i^L(u') + c, d_i^L(u') + c') + (1-\alpha) \cdot (e_i^U(u') + c, d_i^U(u') + c')) \tag{14}$$

$$= (c, c') + \sum_i \lambda_i \cdot (\alpha \cdot (e_i^L(u'), d_i^L(u')) + (1-\alpha) \cdot (e_i^U(u'), d_i^U(u'))) \tag{15}$$

$$= (c, c') + \sum_i \lambda_i \cdot (\alpha \cdot (e_i(v), d_i(v)) + (1-\alpha) \cdot (e_i(w), d_i(w))) \tag{16}$$

$$= (c, c') + \sum_i \lambda_i \cdot (e_i(u), d_i(u)) = (c_1 + c, c_2 + c'). \tag{17}$$

The step from (15) to (16) follows from the definition of e_i^L, d_i^L, e_i^U, and d_i^U, and the step from (16) to (17) uses (13). Thus, the discrete successor of each concrete state in (l, Z, C) is contained in $\mathsf{Post}_e(l, Z, C)$.

$\Longrightarrow$ - *Soundness*: Let $(l', u', c'_1, c'_2) \in \mathsf{Post}_e(l, Z, C)$ such that $u' \in \{x\}L$ for some $(L, U) \in LUF_x(Z)$. Assume that:

$$(c'_1, c'_2) = \sum_i \lambda_i \cdot (\alpha \cdot (e_i^L(u') + c, d_i^L(u') + c') + (1-\alpha) \cdot (e_i^U(u') + c, d_i^U(u') + c')) \tag{18}$$

for some $0 \leq \alpha, \lambda_i \leq 1$ and $\sum_i \lambda = 1$.

Let $v \in L$ and $w \in U$ be the unique clock valuations in Z where $u'(y) = v(y) = w(y)$ for $y \neq x$. $u \in Z$ is then the unique clock valuation with $u(y) = \alpha \cdot v(y) + (1-\alpha) \cdot w(y)$ for all y with the same α as above. Choose the cost pair $(c_1, c_2) = \sum_i \lambda_i \cdot (e_i(u), d_i(u))$. Now, $(l, u, c_1, c_2) \in (l, Z, C)$ and the proof of completeness gives us that $(l, u, c_1, c_2) \xrightarrow{e} (l', u', c'_1, c'_2)$.

Now, we have that all e-successors and only e-successors of concrete states in (l, Z, C) are in the subset of $\mathsf{Post}_e(l, Z, C)$ with costs that can be written according to (18). Since DPTA are a subset of linear hybrid automata, we know that e-successors maintain convexity. Since (l, Z, C) is, by definition, convex, we know that the set of concrete states $(l', u', c'_1, c'_2) \in \mathsf{Post}_e(l, Z, C)$ with costs according to (18) is convex. Lemma 1 now states that this set is identical to all concrete states in $\mathsf{Post}_e(l, Z, C)$. □

When allowing k-tuples of costs as opposed to pairs, the proof of Lemma 3 is analogous: Whenever we choose concrete states using α and $(1-\alpha)$, we instead use $\alpha_1, \ldots \alpha_k$ with $\sum_i \alpha_i = 1$.

Lemma 3 states that the properties of our proposed Post operator correspond to the requirements of Post defined in Section 2, thus satisfying Property 1C stating that symbolic states are closed under Post.

7. Termination

In this section, we first define the ordering $\sqsubseteq_{\text{inf}}$ on the structure of locations with dual-priced zones and then prove that it is a well-quasi ordering.

Note that given a zone Z with m corner points, any cost function e associated with Z can be represented as an element of $\mathbb{N}^m$ giving the cost at each of the corner points, since cost functions are affine and each corner point of a zone has integral values. Thus, we can view the set of cost function pairs C of a dual-priced symbolic state (l, Z, C) as a subset of $2^{\mathbb{N}^m \times \mathbb{N}^m}$, if Z has m corner points, and whenever we refer to this representation, we write C_Z. Given a pair $(\bar{e}, \bar{d})$ of m-vectors in C_Z, we write $\bar{e} \leq \bar{d}$ if $\bar{e}$ is component-wise less than or equal to $\bar{d}$.

Definition 5 ($\sqsubseteq_{\text{inf}}$). Given two dual-priced symbolic states (l, Z, C) and (l', Z', C'), we write $(l, Z, C) \sqsubseteq_{\text{inf}} (l', Z', C')$ iff: (i) $l = l'$; (ii) $Z' \subseteq Z$; and (iii) for all $(\bar{e}', \bar{d}') \in C'_Z$, there exists a $(\bar{e}, \bar{d}) \in C_{Z \wedge Z'}$ such that $\bar{e} \leq \bar{e}'$ and $\bar{d} \leq \bar{d}'$.

The order $\sqsubseteq_{\text{inf}}$ on k-tuples of costs are defined analogously. Note that $(l, Z, C) \sqsubseteq_{\text{inf}} (l', Z', C')$ implies that for all $u \in Z'$, $\lambda(C(u)) \preceq \lambda(C'(u))$, but not the reverse, i.e., our $\sqsubseteq_{\text{inf}}$ is stronger than domination; however, the above definition suffices to guarantee termination.

Lemma 4. $\sqsubseteq_{\text{inf}}$ *is a well-quasi ordering.*

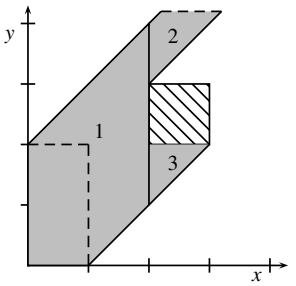

Fig. 8. Delay of the zone $(0 \leq x \leq 1 \wedge 0 \leq y \leq 2)$ under the restriction of $(2 < x < 3 \wedge 2 < y < 3)$ results in the three zones marked 1, 2, and 3.

The proof of Lemma 4 follows directly from the fact that $(\mathbb{N}, \leq)$ satisfies the property of being a so-called better-quasi ordering in the sense of [13]. We will not give the (somewhat involved) formal definition of this notion here, but refer the interested reader to [13], where better-quasi orderings are developed for proving the termination of infinite state systems. We merely recall that better-quasi orderings are closed under Cartesian products and power sets, and better-quasi orderings imply well-quasi orderings. For k-tuples of cost, the proof is identical as we consider k Cartesian products on $\mathbb{N}^m$ instead of pairs.

The representation of cost functions above can be use to prove that the final Property 1D is satisfied. The collection of costs given by the C_Z representation of cost functions defines the convex space of individual costs, i.e., $\lambda(C_Z)$, implying that $\mathsf{Cost}(A, \pi)$ is computable and representable. So, we need to show that we can compute inf and sup on cost sets in order to satisfy Property 1D. The following argument is provided for inf only as the argument for sup is analogous. Computing $\inf C_Z$ is straightforward, as C_Z is finite; however, $\inf \lambda(C_Z)$ is equal to $\lambda(\inf C_Z)$ only when the cardinality of $\inf C_Z$ is two or less. Rather, $\inf \lambda(C_Z)$ can be represented as individual incomparable line segments between pairs from $\inf C_Z$. Since there are finitely many such line segments to compare, $\inf \lambda(C_Z)$ is both computable and representable from $\inf C_Z$, and thus satisfies the property. For the extension to k-tuples of costs, we need to consider sets of k points making up the border of costs space as k-cornered hyperplanes; however, the computability and representation results are maintained.

In Section 4, we restricted reachability to location-based goals only; however, timed automata usually allow reachability goals to include several locations and restrictions on clocks. This poses no problem for Algorithm 1, as we are only concerned with minimal costs. However, for a synthesis of maximal reachability costs, we can potentially delay into a state satisfying the goal requirements computing a cost greater than the first encountered goal state. To illustrate this point clearly, consider the concrete state with one clock $(l, x = 0)$, where l has no invariants and cost rate 1 and the goal is location l with $x \geq 1$. Now, both delay transitions $(l, x = 0) \xrightarrow[1,1]{\delta} (l, x = 1)$ and $(l, x = 0) \xrightarrow[2,2]{\delta} (l, x = 2)$ achieve a goal state, where the latter has larger cost than the former. However, this is undesirable, as the latter intuitively delays past the former, making that the first encountered goal state according to definition of trace costs in Section 2. The solution to this problem is to redefine the Post_δ-operator to restrict delays to states that are both non-goal and are not reached by delaying through the goal. Note that this successor set is not necessarily representable by a single zone, but rather by a finite collection of zones which are directly extensible dual-priced zones. Fig. 8 depicts the restricted delay operations.

Using the restricted Post_δ operator instead in Algorithms 2 and 3 means that goal states are never visited if they are reached through delays. However, as Post_δ will compute states with costs arbitrarily close to the first encountered goal state, and COST is updated for every visited state, the algorithm maintains correctness.

Now, we have fully instantiated the framework defined in Section 2 with syntax, data structures, a Post operator, and a well-quasi order. Based on this, we can conclude that with this instantiation, Algorithm 1 computes minimal reachability costs and Algorithms 2 and 3 compute maximal costs for DPTA. These results are summarized in the following theorem.

Theorem 1. *The synthesis of minimal and maximal reachability costs for DPTA is decidable.*

Along with the definitions of the framework of dual-priced transitions systems, DPTA, data structure for dual-priced symbolic states, the Post operator, and $\sqsubseteq_{\inf}$ we have discussed the straightforward extension to k-tuples of cost, and thus MPTA. Based on this, we state the following corollary of Theorem 1.

Corollary 1. *The synthesis of minimal and maximal reachability costs for MPTA is decidable.*

Given the related reachability problems outlined in Section 3.3, we have:

Corollary 2. *Optimal conditional reachability for MPTA is decidable.*

And finally:

Corollary 3. *Under the liveness constraint that all paths eventually reach the goal, the synthesis of all reachable cost pairs is decidable.*

8. Conclusion & Future Work

We have proved the decidability of synthesizing minimal and maximal reachability costs for multi-priced timed automata. The results are obtained from zone-based algorithms for computing conditional reachability which, in turn, might lead to an efficient implementation. These results generalize decidability for optimal conditional reachability, and the example of Fig. 2 illustrates that integral solutions for this problem are not guaranteed; thus the immediate discrete time semantics for MPTA will not, in general, give correct results. However, discrete analysis of MPTA can be applied, but a correct time granularity must be chosen beforehand. In the case of Fig. 2, a valid time granularity is $\frac{1}{3}$. However, a valid choice of granularity is non-trivial and remains a study for future attention.

Under the liveness constraint that all paths eventually reach the goal, we can synthesize all possible costs of reaching the goal, which implies that the problem of computing optimal conditional reachability under combinations of upper and lower bound constraints on the secondary costs is decidable.

Except implementation of optimal reachability in the tool UPPAAL CORA, future research includes considering approximations along the lines of the ones proposed by Puri and Tripakis in [7]. Also, the complexity and efficiency of Algorithms 1–3 should be analyzed.

Acknowledgement

The authors would like to thank Stavros Tripakis for introducing them to multi-constrained optimization problems.

References

[1] R. Alur, D. Dill, Automata for modelling real-time systems, in: Proc. of Int. Colloquium on Algorithms, Languages and Programming, in: Lecture Notes in Computer Science, vol. 443, Springer-Verlag, 1990, pp. 322–335.
[2] R. Alur, C. Courcoubetis, T.A. Henzinger, P.-H. Ho, Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems, in: Hybrid Systems, in: Lecture Notes in Computer Science, vol. 736, Springer-Verlag, 1993, pp. 209–229.
[3] K. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, J. Romijn, As cheap as possible: Efficient cost-optimal reachability for priced timed automata, in: Proc. of Computer Aided Verification, in: Lecture Notes in Computer Science, vol. 2102, 2001, 493+ pp. URL: citeseer.nj.nec.com/article/larsen01as.html.
[4] R. Alur, S. La Torre, G.J. Pappas, Optimal paths in weighted timed automata, in: Proc. of Hybrid Systems: Computation and Control, in: Lecture Notes in Computer Science, vol. 2034, Springer-Verlag, 2001, pp. 49–62. URL: citeseer.nj.nec.com/alur01optimal.html.
[5] T. Brihaye, V. Bruyère, J.-F. Raskin, Model-checking weighted timed automata, in: Proc. of Formal Modelling and Analysis of Timed Systems, in: Lecture Notes in Computer Science, vol. 3253, Springer-Verlag, 2004, pp. 277–292.
[6] M.R. Garey, D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., 1990.
[7] A. Puri, S. Tripakis, Algorithms for the multi-constrained routing problem, in: Proc. of Scandinavian Workshop on Algorithm Theory, vol. 2368, Springer-Verlag, 2002, pp. 338–347.
[8] K.G. Larsen, J.I. Rasmussen, Optimal conditional reachability for multi-priced timed automata, in: Proc. of Foundations of Software Science and Computational Structures, in: Lecture Notes in Computer Science, vol. 3441, Springer-Verlag, 2005, pp. 234–249.
[9] G. Behrmann, K.G. Larsen, J.I. Rasmussen, Beyond liveness — efficient parameter synthesis for time bounded liveness, in: Proc. of Formal Modelling and Analysis of Timed Systems, in: Lecture Notes in Computer Science, vol. 3829, Springer-Verlag, 2005, pp. 81–94.
[10] T.A. Henzinger, P.-H. Ho, H. Wong-Toi, HYTECH: A model checker for hybrid systems, International Journal on Software Tools for Technology Transfer 1 (1–2) (1997) 110–122. URL: citeseer.nj.nec.com/henzinger97hytech.html.

[11] M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, S. Yovine, Kronos: A model-checking tool for real-time systems, in: Proc. of Computer Aided Verification, in: Lecture Notes in Computer Science, vol. 1427, Springer-Verlag, 1998, pp. 546–550. URL: citeseer.nj.nec.com/bozga98kronos.html.

[12] K.G. Larsen, P. Pettersson, W. Yi, UPPAAL in a nutshell, International Journal on Software Tools for Technology Transfer 1 (1–2) (1997) 134–152. URL: citeseer.nj.nec.com/larsen97uppaal.html.

[13] P.A. Abdulla, A. Nylén, Better is better than well: On efficient verification of infinite-state systems, in: Proc. of the 15th Annual IEEE Symposium on Logic in Computer Science, IEEE Computer Society, 2000, pp. 132–140.