OPTIMAL-REACHABILITY AND CONTROL FOR ACYCLIC WEIGHTED TIMED AUTOMATA*

Salvatore La Torre
University of Pennsylvania &
Università degli Studi di Salerno
sallat@dia.unisa.it

Supratik Mukhopadhyay
University of Pennsylvania
supratik@saul.cis.upenn.edu

Aniello Murano
Università degli Studi di Salerno
murano@dia.unisa.it

Keywords: Timed Automata, Control Synthesis, Optimization.

Abstract Weighted timed automata extend timed automata with costs on both locations and transitions. In this framework we study the optimal reachability and the optimal control synthesis problems for the automata with acyclic control graphs. This class of automata is relevant for some practical problems such as some static scheduling problems or air-traffic control problems. We give a nondeterministic polynomial time algorithm to solve the decision version of the considered optimal reachability problem. This algorithm matches the known lower bound on the reachability for acyclic timed automata, and thus the problem is NP-complete. We also solve in doubly exponential time the corresponding control synthesis problem.

*The first and the second authors were supported in part by the NSF award CCR99-70925, by the SRC award 99-TJ-688, and by the DARPA ITO Mobies award F33615-00-C-1707. The first and the third authors were supported in part by the MURST in the framework of the project "Metodi Formali per la Sicurezza" (MEFISTO). The first author was also supported by the NSF ITR award.

Introduction

Timed automata were introduced by Alur and Dill [2] to model real-time systems, that is, systems interacting with physical processes and whose correct behavior crucially depends upon real-time considerations. A timed automaton is a finite automaton augmented with a finite set of real-valued *clocks*. Transitions are enabled according the current *location* and the current clock values. In a transition, clocks can be instantaneously reset, and the value of a clock is exactly the time elapsed since the last time it was reset. A basic decision problem on timed automata is *reachability*. Given an automaton A, the reachability problem can be stated as the problem of deciding if there exists a run of A from a location s to a location t. Reachability for timed automata is known to be PSPACE-complete [2], and can be used in many contexts, such as the verification of *safety properties* ("nothing bad will eventually happen") or the *emptiness problem* ("is the language accepted by a given automaton empty?").

Designing reactive systems often requires each component to be viewed as an open system, that is, a system which interacts with its environment and whose behavior depends on its current state as well as the behavior of the environment. In open systems, a central problem is *controller synthesis*: "given a specification φ and a *plant* P, we want to determine a *controller* for P such that φ is satisfied". For example, a typical specification in an air-traffic management problem is to require that a controller must try to prevent aircraft collisions. A plant P is described by all its possible behaviors and a controller for P is then a system that can observe the state of P and issue a *control action* to influence its behavior. In an air-traffic management problem, control actions might consist of establishing priorities among aircrafts, or forcing an aircraft to modify its trajectory. Specifications in control synthesis problems can be given using different formalisms. In a reachability control synthesis, the specification is usually expressed by a set of states with the meaning that the desired controller must drive the plant to these states despite of the behavior of the environment. Reachability control synthesis problems on timed automata are considered in [5, 7, 15, 17]. LTL specifications for rectangular hybrid systems are studied in [14]. LTL and CTL specifications for timed automata are investigated using an automata-theoretic approach in [11]. Also specifications given as timed automata [10] and TCTL formulas [12] have been recently addressed.

Associating with a run of a timed automaton a performance measure, it is possible to compare runs, and thus, search for an optimal run connecting two locations. The most natural performance measure for timed automata is clearly time. Time-optimal reachability was first considered in [9], where the problem of computing lower and upper bounds on time delays in timed automata was solved. Minimum-time reachability is also considered in [18] and is shown to be NP-complete acyclic timed automata in [3]. The related problem for open systems is the synthesis of a time-optimal controller which is shown to be decidable in [5]. We recall that, it is known that the related decision problem is EXPTIME-complete [14, 15]. Besides time, other cost functions have been considered. Given a timed automaton, a weight w can be associated with each

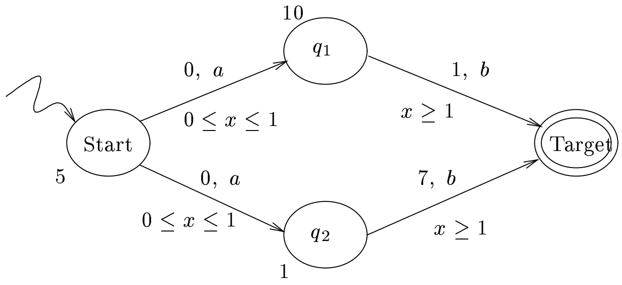

Figure 1. A weighted timed automaton.

location q such that w gives the cost of a unit of time spent in q. On such a model, given a cost interval I and two states s and t, the decision problem "is t reachable from s at a cost $c \in I$?" (*duration-bounded reachability*) is addressed in [1]. A *weighted timed automaton* extends a timed automaton with costs on both locations and transitions[1]. On this model the optimal-reachability problem was independently addressed in [4] and [8].

The techniques, used in [4, 8] to solve the optimal-reachability problem, do not extend to solving the corresponding optimal-synthesis problem. In fact, the linearity of the cost function implies that optimal runs can be found among those having only transitions taken in one of the following cases: as soon as the corresponding guard holds true, right before the corresponding guard holds false, or when at least a clock variable has an integer value. In the optimal-synthesis problem we consider this property does not hold. To see this consider the weighted timed automaton in Figure 1. In the automaton, nondeterminism is used to model the choices of the environment and the symbols labeling the edges are the actions that can be taken by the controller. Denoting by t the time spent in location *Start*, the minimum cost to reach *Target* through location q_1 is given by the infimum value of $(11-5t)$ for $t \in [0,1]$ and the minimum cost to reach *Target* through q_2 is the infimum of $(4t+8)$ for $t \in [0,1]$. In this example, the controller is only allowed to select the time at which a transition from *Start* is taken (since both transitions are labeled by a) and the environment selects instead the transition. Thus the best choice for the controller is t^* such that $\max 11 - 5t, 4t + 8$ is minimum, that is, $t^* = \frac{1}{3}$. In case we consider the optimal-reachability problem on the same automaton (i.e., the controller is allowed to select the actual transition in presence of nondeterminism), an optimal run goes through location q_1 with the first transition taken at time 1.

In this paper we are concerned with both the optimal reachability and the optimal control synthesis for acyclic weighted timed automata. There are several interesting problems that can be modeled by acyclic automata such as some static scheduling problems and air-traffic control problems. Recently the *job-shop scheduling problem* [6] has been modeled as minimum-time reachability on acyclic timed automata. Considering different costs for each task to schedule is realistic and can be captured by a weighted timed automaton. We show that the optimal reachability problem for acyclic weighted timed automata is in FP^{NP}. We recall that in the general case, the known upper bound is EXPTIME[4]. We give an algorithm that combines binary search along with a solution to the corresponding decision problem: "is there a run r from s to t whose cost is not larger than c?" (*cost-bounded reachability*). We prove that the cost-bounded reachability can be solved in nondeterministic polynomial time, and since reachability is NP-hard for acyclic timed automata [3], we get that this problem is NP-complete. We also solve the optimal synthesis problem for acyclic weighted timed automata in doubly exponential time. Our solution consists of reducing this problem to deciding some particular kind of first-order logic formulas of the theory of reals with addition and order. The translation causes an exponential blow-up and the satisfiability of such formulas can be decided in exponential time. We recall that in general the first-order theory of reals with addition and order is decidable in doubly exponential time [13].

The rest of the paper is organized as follows. In Section 1 we recall the definition of weighted timed automata and introduce our notation. Our solution to the optimal reachability problem for acyclic weighted timed automata is given in Section 2. In Section 3 the optimal control synthesis problem for acyclic weighted timed automata is solved. Finally, in Section 4 we give our conclusions.

1. Weighted timed automata

A timed automaton models a real-time system. We assume that there is a central (real-valued) clock, and the model can use a finite set of *clock variables* (also said simply *clocks*) along with timing constraints to check the satisfaction of timing requirements. Each clock can be seen as a chronograph synchronized with the central clock, thus it can be read or set to zero (reset): after a reset, a clock restarts automatically. In each automaton, timing constraints are expressed by clock constraints. Let C be a set of clocks, the set of clock constraints $\Xi(C)$ contains:

- $x \leq y+c$, $x \geq y+c$, $x \leq c$ and $x \geq c$ $\forall x, y \in C$ and for a rational number c; we call such constraints *atomic* clock constraints;
- $\neg\delta$ and $\delta_1 \wedge \delta_2$ where $\delta, \delta_1, \delta_2 \in \Xi(C)$.

Furthermore, a *clock interpretation* is a mapping $\nu : C \longrightarrow \mathbb{R}_+$. We denote by $\vec{0}$ the clock interpretation mapping each clock to 0. If ν is a clock interpretation, λ is a set of clocks and d is a real number, we denote with $\nu(\lambda, d)$ the clock interpretation that for each clock $x \in \lambda$ gives 0 and for each clock $x \notin \lambda$ gives the value $\nu(x) + d$.

Definition 1 *A* timed automaton A *is a tuple* $(\Sigma, \mathcal{Q}, \mathcal{Q}_0, C, \Delta, inv)$ *where:*

- Σ *is a finite set of symbols (the alphabet);*
- $\mathcal{Q}$ *is a finite set of locations;*

- $\mathcal{Q}_0 \subseteq \mathcal{Q}$ *is the set of initial locations;*
- C *is a finite set of* n *clock variables;*
- Δ *is a finite subset of* $\mathcal{Q} \times \Sigma \times \Xi(C) \times 2^C \times \mathcal{Q}$ *(edges);*
- $inv : \mathcal{Q} \longrightarrow \Xi(C)$ *maps each location* q *to its invariant* $inv(q)$.

A *state* of a timed automaton A is a pair $\langle q, \nu \rangle$ where $q \in \mathcal{Q}$ and $\nu \in \mathbb{R}^n_+$. An *initial state* is a pair $\langle q_0, \vec{0} \rangle$ where $q_0 \in \mathcal{Q}_0$ is an initial location. The semantics of a timed automaton is given by a transition system over the set of states. The transitions of this system are divided into *discrete steps* and *time steps.* A discrete step is $\langle q, \nu \rangle \xrightarrow{\sigma} \langle q', \nu' \rangle$ where $(q, \sigma, \delta, \lambda, q') \in \Delta$, ν satisfies δ, $\nu' = \nu(\lambda, 0)$, and ν' satisfies $inv(q')$. A time step is $\langle q, \nu \rangle \xrightarrow{d} \langle q, \nu' \rangle$ where $\nu' = \nu + d$, $d \geq 0$, and $\nu + d'$ satisfies $inv(q)$ for all $0 \leq d' \leq d$. A *step* is $\langle q, \nu \rangle \xrightarrow{\sigma, d} \langle q', \nu' \rangle$ where $\langle q, \nu \rangle \xrightarrow{d} \langle q, \nu'' \rangle$ and $\langle q, \nu'' \rangle \xrightarrow{\sigma} \langle q', \nu' \rangle$, for some $\nu'' \in \mathbb{R}^n$. A *timed sequence* (σ, τ) over the alphabet Σ is such that $\sigma \in \Sigma^*$, $\tau \in \mathbb{R}^*_+$, and $|\sigma| = |\tau|$. The sequence τ is called a *time sequence.* In a timed sequence, each symbol σ_i at input is associated with a positive real number τ_i, which expresses (except for the first symbol σ_1) the time which has elapsed since the symbol σ_{i-1} was at input. Time τ_1 represents instead the time at which the symbol σ_1 appears at input assuming that time is 0 when the computation starts. A run r of a timed automaton A on a timed sequence (σ, τ), where $\sigma = \sigma_1 \ldots \sigma_k$ and $\tau = \tau_1 \ldots \tau_k$, is a finite sequence $\langle q_0, \nu_0 \rangle \xrightarrow{\sigma_1, \tau_1} \langle q_1, \nu_1 \rangle \xrightarrow{\sigma_2, \tau_2} \ldots \xrightarrow{\sigma_k, \tau_k} \langle q_k, \nu_k \rangle$. We say that r starts at q_0 and ends at q_k. We denote the set of A runs by $\mathrm{Run}(A)$. A timed automaton is *acyclic* if its control graph is acyclic.

A *weighted timed automaton* is a timed automaton A with cost functions:

- $J_s : \Delta \longrightarrow \mathbb{N}$ (*switch cost*), and
- $J_d : \mathcal{Q} \longrightarrow \mathbb{N}$ (*duration cost*).

Given a run r of A, let $e_1, \ldots, e_k$ be the sequence of transitions taken in r and $q_0, \ldots, q_k$ be the sequence of visited locations. We associate to r the following costs:

- $J_s(r) = \sum_{i=1}^{k} J_s(e_i)$, and
- $J_d(r) = \sum_{i=0}^{k-1} t_i \cdot J_d(q_i)$.

The total cost associated with a run r is then $J(r) = J_s(r) + J_d(r)$. As an example of a weighted timed automaton consider the one modeling a simple air traffic control system given in Figure 2. A different air traffic control system modeled by an acyclic weighted timed automaton is described in [16].

Example 1 Consider the timed transition system in Figure 2. It models a scenario in which two aircrafts send a landing request to a control tower, and

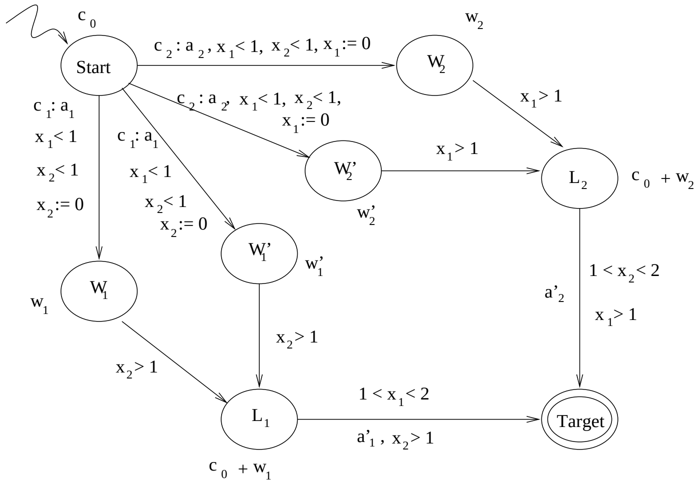

Figure 2. An air-traffic control problem.

our goal is to allow both the aircrafts to land safely and at minimum cost. Safety requires that only one aircraft at a time must be acknowledged for landing, thus there are two possible choices: aircraft 1 waits for aircraft 2 to land (action a_1), or vice-versa (action a_2). There are costs c_1 and c_2 relative to actions a_1 and a_2 respectively, that correspond to the cost to pay for forcing respectively aircraft 1 and aircraft 2 to wait. Moreover, the cost of waiting may vary according to different environmental situations. We consider two possible scenarios for each taken action. If action a_1 is taken, depending on the environment the cost of waiting is either w_1 or w'_1 per time unit. Analogously, we have costs w_2 and w'_2 relatively to action a_2. Since it is realistic to reduce the amount of time a runway stays unused, we penalize this event by a cost c_0 per time unit. Finally, we assume that the landing of each aircraft takes at least time 1 since the related acknowledgment was issued by the control tower. Actions a'_1 and a'_2 correspond to the landing of aircrafts 1 and 2, respectively, after they are forced to wait.

2. Optimal reachability

Given an automaton A, the *reachability problem* can be stated as the problem of deciding if there exists a run of A from a location s to a location t. Here we study a related optimization problem on acyclic weighted timed automata. Given a weighted timed automaton A, a source location src and a target location trg, the *optimal cost* J^* to reach trg from src is the infimum of $J(r)$ over all the runs r from $\langle src, \vec{0}\rangle$ to a state $\langle trg, \nu\rangle$, for some clock interpretation ν. If location trg is reachable from $\langle s, \vec{0}\rangle$, the optimal cost is well defined while a run matching this cost may not exist. Thus the *optimal reachability problem* can be stated as the problem of determining a run of optimal cost, if there exists one, and an infinite family of runs parameterized on ξ and containing for any $\xi > 0$ a run r such that $J(r) < J^* + \xi$, otherwise. In this section we give an algorithm to determine the optimal cost J^* given an acyclic weighted timed automaton, and two locations *src* and *trg*. Clearly a solution to the optimal reachability problem can be constructed by tracking back the transitions leading to the optimal cost.

In relation to the optimal reachability problem we consider the following *cost-bounded reachability problem*: "given a positive real c and a weighted timed automaton A, is there a run r from the state $\langle src, \vec{0}\rangle$ to a location *trg* such that $J(r) \leq c$?" We prove that this problem is NP-complete for acyclic weighted timed automata.

In the rest of the section we assume that the constants in the clock constraints of the considered automata are natural numbers. All the obtained results extend to the general case in the following way. Let u be the minimum common denominator among all the constants used in a timed automaton. We use $\frac{1}{u}$ instead of 1 as time unit and thus all the results rescale according to this.

Denote by $c_{\max}$ the largest constant used in the clock constraints incremented by 1, by $J_d^{\max}$ the largest duration cost, by $J_s^{\max}$ the largest switch cost, and by M the longest path in the control graph of a timed automaton. We have the following result.

Lemma 1 *Given two locations src and trg of a acyclic weighted timed automaton A, if trg is reachable from $\langle src, \vec{0}\rangle$ then there exists a run r such that $J(r) \leq M \cdot (J_s^{\max} + J_d^{\max} \cdot c_{\max})$.*

Proof. We observe that on any run from $\langle src, \vec{0}\rangle$ to a state in location *trg* there are at most M transitions. Moreover, any transition guard either is satisfied within time $c_{\max}$ or stays unsatisfied forever. Thus the lemma holds. ■

The above result gives a simple polynomial-time reduction from the reachability problem for acyclic timed automata.

Theorem 1 *The cost-bounded reachability problem for acyclic weighted timed automata is NP-complete.*

Proof. NP-hardness is consequence of the NP-hardness of the reachability problem for acyclic timed automata [3] and Lemma 1. To prove NP-membership we use an algorithm that nondeterministically guesses a linear program of linear size and solves it. Since all the variables involved in the guessed linear program are real-valued, this algorithm runs in nondeterministic polynomial time. We sketch the algorithm in the following. For the sake of simplicity we assume that the control graph of the automaton we consider does not have multiple edges connecting two given locations. We denote by $q_0, \ldots, q_m$ the locations of the weighted timed automaton, where q_0 is the source location and q_m is the target location, and by $e_{i,j}$ the transition from a location q_i to a location q_j. Observe first that given a boolean formula φ and a formula φ' which is a conjunction of literals containing exactly a literal for each variable in φ, there exists a linear-time algorithm to check if φ' implies φ (this is equivalent to evaluate φ on a given variable assignment). Denote by $\delta_{i,j}$ and $\lambda_{i,j}$ respectively the clock constraint and the reset associated to a transition $e_{i,j}$. Let $\varphi_1, \ldots, \varphi_k$ be the atomic clock constraints in $\delta_{i,j}$. Given a sequence of bits $\beta = b_1 \ldots b_k$, let φ'_h be φ_h, if $b_h = 1$, and $\neg\varphi_h$, otherwise. We define $\Delta^\beta_{i,j}$ be $\bigwedge_h \varphi'_h$, if $\bigwedge_h \varphi'_h$ implies $\delta_{i,j}$, and the constant FALSE, otherwise. Moreover, we define a variable $a_{i,j}$ which evaluates 1, if transition $e_{i,j}$ is taken, and 0 otherwise. We use a variable $\nu_{i,j}$ to store the clock interpretation after that transition $e_{i,j}$ is taken, and a variable $t_{i,j}$ to store the time spent in location q_i before taking $e_{i,j}$. For a given location q_i we denote by $succ(i)$ the set $\{j \mid e_{i,j}$ is a transition$\}$ and by $pre(i)$ the set $\{j \mid e_{j,i}$ is a transition$\}$. Denote by $l_i = \sum_{j \in succ(i)} a_{i,j}$ for $i = 0, \ldots, m-1$ and $l_m = \sum_{j \in pre(m)} a_{j,m}$. The objective of the program is to minimize the function $\sum_{i,j} a_{i,j}\, J_s(e_{i,j}) + \sum_{i,j} l_i\, J_d(q_i)\, t_{i,j}$ according to the following constraints:

1 $\sum_{j \in succ(0)} a_{0,j} = 1$ (exactly a transition must be taken from q_0);

2 $\sum_{j \in pre(i)} a_{j,i} = \sum_{h \in succ(i)} a_{i,h}$ (conservation property: if q_i is reached, just a transition can be taken);

3 $\sum_{j \in pre(m)} a_{j,m} = 1$ (the target location must be reached);

4 $\nu_{0,j} = \vec{0}(\lambda_{0,j}, t_{0,j})$, and for $i > 0$, $\nu_{i,j} = \sum_{k \in pre(i)} \nu_{k,i}(\lambda_{i,j}, t_{i,j})\, a_{k,i}$ (resets);

5 $a_{0,j} = 0 \vee \Delta^\beta_{0,j}(\vec{0} + t_{0,j})$, and for $i > 0$, $a_{i,j} = 0 \vee \Delta^\beta_{i,j}(\sum_{k \in pre(i)}(\nu_{k,i}\, a_{k,i}) + t_{i,j})$ (time constraints).

We claim that the above program gives a nondeterministic polynomial-time algorithm to solve the cost-bounded reachability problem. To see this, observe that a valuation of all $a_{i,j}$'s fulfilling the above constraints 1, 2, and 3 selects a path from q_0 to q_m. Moreover, providing a valuation for any sequence β, for any transition $e_{i,j}$, we have either an empty region (if some $e_{i,j}$ has been selected and sequence chosen for it does not correspond to a conjunctive formula implying $\delta_{i,j}$) or a linear program P such that the optimal solution corresponds to the optimal cost of a run corresponding to the selected path. Since any of such P contains only real-valued variables, the optimal solution can be computed in polynomial time. Moreover, the size of any P is linear in the size of the given automaton. Thus, we have proved NP membership for the cost-bounded reachability problem. ■

The nondeterministic polynomial-time algorithm solving the cost-bounded reachability problem can be used to obtain a nondeterministic polynomial-time algorithm to solve the optimal reachability problem.

Theorem 2 *The optimal reachability problem for acyclic weighted timed automata is in FP^{NP}.*

Proof. Since the cost function is linear in the time, we have that, under the assumption that all constants in the automaton are integers, the optimal cost is also an integer. The optimal cost can be computed simply by combining a binary search in the interval from 0 to $M \cdot (J_s^{\max} + J_d^{\max} \cdot c_{\max})$, with an algorithm solving the cost-bounded reachability problem. By Theorem 1 this last step can be done in nondeterministic polynomial-time. Since the algorithm for computing the optimum cost, as well as the optimum run, calls a polynomial number of times the nondeterministic polynomial time procedure, we get that the total algorithm is in FP^{NP}. ∎

3. Optimal Control

We model the controller synthesis problem for timed automata as a *timed game* of a *controller* against the *environment*, and then the synthesis of a controller corresponds to determine a winning strategy. In our settings, the game is modeled as a nondeterministic timed automaton, where actions represent the choices of the controller and the nondeterminism is used to model the possible choices of the *environment* for a given action taken by the controller[2]. We use a special action denoted by ε to capture the case that the controller is *idle* and the environment is moving. When both players are idle, a time step is taken. In the following we will denote by Σ^ε the set of actions including the idle action ε. A *play* of a timed game is constructed in the following way. At each time, a player declares how long it will wait idling until its next choice. At the time one of the players or both move, both the players are allowed to redeclare their next move and the time they will issue it. That is, if a player moves before the other, the latter is allowed to change its former decision. A play is represented by a run of the automaton. Formally, a *timed game* is a tuple (A, src, trg) where

- $A = (\Sigma^\varepsilon, \mathcal{Q}, \mathcal{Q}_0, C, \Delta, inv)$ is a timed automaton,
- $src \in \mathcal{Q}$ is the *source location* and $trg \in \mathcal{Q}$ is the *target location*.

A *weighted timed game* is defined in the same way by considering a weighted timed automaton instead of a timed automaton. In the following we can assume that A has no outgoing transition from the target location trg. The goal of our computations is to reach trg. Given a run $r = \langle q_0, \nu_0\rangle \xrightarrow{\tau_1,\sigma_1} \langle q_1, \nu_1\rangle \xrightarrow{\tau_2,\sigma_2} \ldots \xrightarrow{\tau_k,\sigma_k} \langle q_k, \nu_k\rangle$, we denote by r_i the run $\langle q_0, \nu_0\rangle \xrightarrow{\tau_1,\sigma_1} \langle q_1, \nu_1\rangle \xrightarrow{\tau_2,\sigma_2} \ldots \xrightarrow{\tau_i,\sigma_i} \langle q_i, \nu_i\rangle$. A *strategy* is a function $\mathcal{F} : Plays(\mathcal{F}) \longrightarrow \mathbb{R} \times \Sigma$, where $Plays(\mathcal{F}) \subseteq \mathrm{Run}(A)$, $\langle q_0, \nu_0\rangle = \langle src, \vec{0}\rangle \in Plays(\mathcal{F})$ and for any $r = \langle q_0, \nu_0\rangle \xrightarrow{\tau_1,\sigma_1} \langle q_1, \nu_1\rangle \xrightarrow{\tau_2,\sigma_2} \ldots \xrightarrow{\tau_k,\sigma_k} \langle q_k, \nu_k\rangle$ belonging to $Plays(\mathcal{F})$, it holds that for $i = 1, \ldots, k-1$, either $\mathcal{F}(r_i) = (\tau_{i+1}, \sigma_{i+1})$ or $\mathcal{F}(r_i) = (d, \sigma)$, $\tau_{i+1} < d$ and $\sigma_{i+1} = \varepsilon$. In other words a strategy gives the moves of the controller on each play which is "consistent" with the strategy and the case $\sigma_{i+1} = \varepsilon$ corresponds to a move of the environment taken before the next declared move of the controller according to a strategy $\mathcal{F}$. A run is *maximal* if it ends in a state from which it will be possible only to have time steps. A strategy $\mathcal{F}$, is *winning* if all maximal runs $r \in Plays(\mathcal{F})$ end in a state of the target location. Given a winning strategy $\mathcal{F}$, the cost of $\mathcal{F}$ is defined as $J(\mathcal{F}) = \sup_{r \in Plays(\mathcal{F})} J(r)$. The optimal cost for a winning strategy is defined as

$$J^* = \inf_{\mathcal{F}} J(\mathcal{F})$$

and there exists an optimal winning strategy if and only if there exists $\mathcal{F}^*$ such that

$$J(\mathcal{F}^*) = J^*.$$

If there exists a winning strategy, the optimal cost is always well defined while an optimal winning strategy may not exist. When this is the case, it is possible to determine a family of strategies parameterized on ξ such that for any $\xi > 0$ there exists a strategy in this family such that its cost J is such that $J \leq J^* + \xi$. In the following we refer as an optimal solution of a weighted timed game to either a winning strategy, if one exists, or to such a family of winning strategies, otherwise. We focus on the following optimization problem: "given a weighted timed game, determine if there exists a winning strategy, and in this case, the cost of an optimal winning strategy".

In the sequel, we will refer to the controller as the protagonist and to the environment as the adversary. Let $\mathcal{C}(q, \nu)$ be the cost of an optimal winning strategy for the protagonist when the game starts at the state $\langle q, \nu \rangle$. For the target location trg, $\mathcal{C}(trg, \nu) = 0$ for all $\nu \in \mathbb{R}^n$. The cost $\mathcal{C}(q, \nu)$ of an optimal strategy for any state with location q, different from trg, is computed as described below.

Given a $t \geq 0$, the cost incurred by the protagonist taking an action σ after time t is the maximum cost over all the transitions on σ enabled at time t and all the transitions on ε enabled at any $0 \leq t' < t$. While the protagonist aims to minimize the cost of a strategy, the adversary will chose the transitions that will maximize it. Let $\mathcal{E}_\sigma$ be the set of σ transitions and E be the set of ε transitions (adversary transitions) that are enabled at time t. Then we have the following recurrence:

$$\mathcal{C}(q, \nu) = \inf_{t \geq 0} \max \begin{cases} \min_{\sigma \in \Sigma} \max_{e \in \mathcal{E}_\sigma} (J_d(q) \cdot t + J_s(e) + \mathcal{C}(q', \nu(\lambda, t))) \\ \sup_{0 \leq t' \leq t} \max_{e \in E} (J_d(q) \cdot t' + J_s(e) + \mathcal{C}(q'', \nu(\lambda', t'))) \end{cases} \tag{1}$$

In order to compute the cost of an optimal strategy for a game starting at state $\langle q, \nu \rangle$, we need to solve the above equation. In the rest of this section, we outline an algorithm for solving it. Notice that Equation 1 is defined inductively and involves max, min, sup, and inf operators. Since these operators are definable in first order logic and the graph is acyclic, it is possible to describe Equation 1 in terms of a formula in the first order theory of reals with addition and order. Moreover, this theory is decidable and admits quantifier elimination.

More precisely, let $p(Q, \vec{X}, C)$ be an $(n+2)$-ary predicate defined over the set of locations, the set of clock values and the set of costs corresponding to strategies adopted by the protagonist, with $p(q, \nu, c)$ standing for the fact that the cost of an optimal strategy starting from the state $\langle q, \nu \rangle$ is c. In the rest of this section we will use upper case letters to denote variables and lower case letters to denote constants; also, whenever we omit quantifiers on variables, we will assume that they are universally quantified. Observe that for the target location trg, $p(trg,\nu,0)$ is true for all clock values $\nu \in \mathbb{R}^n$. For any other location q, different from trg, we briefly sketch how to define a first order logic formula that describes Equation 1, and leave the details to the full version of this paper. The first order logic formula corresponding to q is given by[3]:

$$p(q,\vec{X},C) \longleftarrow \exists T \geq 0.\ \varphi(q,\vec{X},T,C),$$

where the first order logic formula φ corresponds to $\mathcal{C}(q,\nu)$. From Equation 1:

$$\begin{aligned}\varphi(q,\vec{X},T,C) \quad \equiv \quad & \varphi_{lb}(q,\vec{X},T,C) \\ & \wedge \forall C'.(\varphi_{lb}(q,\vec{X},T,C') \longrightarrow C \geq C'),\end{aligned}$$

where the formula φ_{lb} holds true for any C expressing a lower bound on the costs of winning strategies from $\langle q,\vec{X}\rangle$. We define φ_{lb} as:

$$\varphi_{lb}(q,\vec{X},T,C) \quad \equiv \quad \forall C'.\forall T'.((T' \geq 0 \wedge \psi_1(q,\vec{X},T',C')) \longrightarrow C' \geq C).$$

In the above formula, the first order formula ψ_1 corresponds to $g(t)$ such that $\mathcal{C}(q,\nu) = \inf_{t\geq 0} g(t)$ corresponds to Equation 1. We define ψ_1 as:

$$\begin{aligned}\psi_1(q,\vec{X},T,C) \quad \equiv \quad & (\psi_2(q,\vec{X},T,C) \vee \psi_3(q,\vec{X},T,C)) \\ & \wedge (\forall C'.\psi_2(q,\vec{X},T,C') \longrightarrow C \geq C') \\ & \wedge (\forall C'.\psi_3(q,\vec{X},T,C') \longrightarrow C \geq C'),\end{aligned}$$

where the first order formulas ψ_2 and ψ_3 correspond respectively to terms $\min_{\sigma\in\Sigma}\max_{e\in\mathcal{E}_\sigma}(J_d(q)\cdot t + J_s(e) + \mathcal{C}(q',\nu(\lambda,t)))$ and $\sup_{0\leq t'\leq t}\max_{e\in E}(J_d(q)\cdot t' + J_s(e) + \mathcal{C}(q'',\nu(\lambda',t')))$ in Equation 1. We give below the definition of ψ_3, and the first order formula ψ_2 can be defined similarly:

$$\begin{aligned}\psi_3(q,\vec{X},T,C) \quad \equiv \quad & \exists T''.0 \leq T'' \leq T \wedge \psi_{ub}(q,\vec{X},T'',C) \\ & \wedge \forall C'.(\psi_{ub}(q,\vec{X},T'',C') \longrightarrow C' \geq C),\end{aligned}$$

where the first order formula ψ_{ub} holds true for any upper bound C on the costs of winning strategies from $\langle q,\vec{X}\rangle$. We define this last formula as:

$$\psi_{ub}(q,\vec{X},T,C) \quad \equiv \quad \forall T'.\forall C'.((0 \leq T' \leq T \wedge \psi_4(q,\vec{X},T',C')) \longrightarrow C \geq C'),$$

where the first order formula ψ_4 corresponds to $\max_{e\in E}(J_d(q)\cdot t' + J_s(e) + \mathcal{C}(q'',\nu(\lambda',t')))$ in Equation 1. We define ψ_4 as:

$$\begin{aligned}\psi_4(q,\vec{X},T,C) \quad \equiv \quad & \textstyle\bigvee_{e\in E}\alpha(q,\vec{X},e,T,C) \\ & \wedge \textstyle\bigwedge_{e\in E}(\forall C'.\alpha(q,\vec{X},e,T,C') \longrightarrow C \geq C'),\end{aligned}$$

where E is the set of ε moves enabled at location q. The first order subformula α above corresponds to the expression $(J_d(q)\cdot t' + J_s(e) + \mathcal{C}(q'',\nu(\lambda',t')))$ in Equation 1 and is defined as follows:

$$\begin{aligned}\alpha(q,\vec{X},e,T,C) \quad \equiv \quad & \exists C'.\ p(q',\vec{X}(\lambda,T),C') \wedge C = C' + J_d(q)\cdot T + J_s(e) \\ & \wedge \delta[\vec{X} \leftarrow \vec{X} + T],\end{aligned}$$

where $e = (q,\varepsilon,\delta,\lambda,q')$, and $\delta[\vec{X} \leftarrow \vec{X} + T]$ is the formula obtained by replacing all free occurrences of $\vec{X}$ in δ by $\vec{X} + T$.

Expanding the definition of φ, for each location q, different from t, we obtain a formula $p(q, \vec{X}, C) \longleftarrow \varphi$ such that q does not occur in φ. Since the control graph of the timed automaton is acyclic, a forward chaining starting from the fact $p(trg, \vec{X}, 0)$ for the target node trg will terminate. The length of the chaining is bounded by the size of the control graph of the automaton. At each step in the chaining, we eliminate the quantifiers occurring in the formula on the right-hand-side of the implication. Notice that at each step of the deduction the quantifier depth is constant. Hence at each step the quantifier elimination can be done in exponential time in the size of the formula (the Ferrante-Rackoff [13] algorithm runs in exponential time for such formulas). Since in the worst case the size of each formula is exponential in the size of the automaton, the complexity of the deduction is doubly exponential. Thus the following theorem holds.

Theorem 3 *The optimal control problem for weighted acyclic timed automata can be solved in doubly exponential time.*

4. Conclusions

In this paper we have solved the optimal reachability and the optimal control synthesis for acyclic weighted timed automata. We have proved that the considered optimal reachability problem is FP^{NP}. Moreover, we give a doubly exponential upper bound for the optimal control synthesis problem for acyclic automata. Proving a better upper bound or a lower bound matching our solution is an open problem. It is also an open problem to find a solution to this problem for general weighted timed automata. Our solution strongly exploits the acyclicity of the control graph and thus does not allow a direct extension to the general case.

Notes

1. In the literature, this model is also known as *linearly priced timed automaton.*
2. Games are usually defined in a symmetric way with respect to each of the players. In optimal control synthesis, the interest is focused on the winning strategies of the controller.
3. We write a formula φ as $\varphi(\beta)$ to denote that the expression β possibly occurs in φ.

References

[1] R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing accumulated delays in real-time system. In *Proc. of the Fifth International Conference on Computer-Aided Verification, CAV'93*, LNCS 697, pages 181 – 193. Springer, 1993.

[2] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183 – 235, 1994.

[3] R. Alur, S. La Torre and S. Mukhopadhyay. *Subclasses of Timed Automata with NP-complete Reachability Problem.* CIS Department, University of Pennsylvania, 2001. URL:"`http://www.cis.upenn.edu/~latorre/Papers/ata.ps.gz`".

[4] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *Proc. of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01*, LNCS 2034, pages 49 – 62. Springer, 2001.

[5] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Proc. of the 2nd International Workshop on Hybrid Systems: Computation and Control*, LNCS 1569, pages 19 – 30. Springer, 1999.

[6] Y. Abdedaim and O. Maler. Job-shop scheduling using timed automata. In *Proc. of the 13th Intern. Conference on Computer Aided Verification, CAV'01*, LNCS 2102, pages 478 –492, 2001.

[7] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Proc. of the 2nd International Workshop on Hybrid Systems*, LNCS 999, pages 1 – 20. Springer, 1995.

[8] G. Behrman, T. Hune, A. Fehnker, K. Larsen, P. Pettersson, R. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. of the 4th International Workshop on Hybrid Systems: Computation and Control, HSCC'01*, LNCS 2034, pages 147–161. Springer, 2001.

[9] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proc. of the 3rd International Conference on Computer Aided Verification*, LNCS 575, pages 399 – 409. Springer, 1991.

[10] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. of the 19th Annual Symposium on Theoretical Aspects of Computer Science, STACS'02*, LNCS 2285, pages 571 – 582. Springer, 2002.

[11] M. Faella, S. La Torre, and A. Murano. Automata-theoretic decision of timed games. In *Proc. of the 3rd Intern. Workshop on Verification, Model Checking, and Abstract Interpretation, VMCAI'02*, LNCS 2294, pages 94 –108. Springer, 2002.

[12] M. Faella, S. La Torre, and A. Murano. Dense Real-time Games. In *Proc. of the 17th Annual IEEE Symposium on Logic in Computer Science, LICS'02*, IEEE Computer Society Press, 2002.

[13] J. Ferrante and C. Rackoff. A decision procedure for the first order theory on real addition with order. *SIAM Journal of Computing*, 4(1):69 – 76, 1975.

[14] T. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Proc. of the 10th International Conference on Concurrency Theory, CONCUR'99*, LNCS 1664, pages 320 – 335, 1999.

[15] T. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221(1–2):369–392, 1999.

[16] K. G. Larsen, G. Behrman, E. Brinksma, A. Fehnker, T. Hune, P. Petersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. of the 13th International Conference on Computer Aided Verification, CAV'01*, LNCS, pages 493–505. Springer, 2001.

[17] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. of the 12th Annual Symposium on Theoretical Aspects of Computer Science, STACS'95*, LNCS 900, pages 229 – 242. Springer, 1995.

[18] P. Niebert, S. Tripakis, and S. Yovine. Minimum-time reachability for timed automata. In *Proc. of the 8-th IEEE Mediterranean Conference on Control and Automation*, 2000.