As Soon as Possible: Time Optimal Control for Timed Automata*

Eugene Asarin[1] and Oded Maler[2]

[1] Institute for Information Transmission Problems, 19
Bol. Karetnyi per. 101447 Moscow, Russia
asarin@iitp.ru
[2] VERIMAG, 2, av. de Vignate, 38610 Gières, France
Oded.Maler@imag.fr

Abstract. In this work we tackle the following problem: given a timed automaton, and a target set F of configurations, restrict its transition relation in a systematic way so that from every state, the remaining behaviors reach F *as soon as possible*. This consists in extending the controller synthesis problem for timed automata, solved in [MPS95,AMPS98], to deal with *quantitative* properties of behaviors. The problem is formulated using the notion of a *timed game automaton*, and an optimal strategy is constructed as a fixed-point of an operator on the space of value functions defined on state-clock configurations.

1 Introduction

Most of the research on verification and synthesis of discrete systems is based on the following approach: the set L of all possible behaviors of the system is partitioned into L_φ and $L_{\neg\varphi}$, the former consisting of behaviors satisfying a certain property φ. Verification is the task of checking whether $L_{\neg\varphi}$ is empty while synthesis is the restriction of the transition relation of the system in order to achieve this fact. A typical example would be the property that all behaviors of the system eventually reach a subset F of the state-space.

In many situations, however, this all-or-nothing classification of behaviors as good or bad is too crude, and we would like to distinguish, for example, between behaviors which reach F within one or million steps. This suggests a richer model where *quantitative* performance measures are associated with behaviors based on length, cumulative cost of states and transitions, probabilities, etc. Verification is then transformed into finding the worst-case performance measure, while synthesis is rephrased as the search for an optimal controller, minimizing the above quantity. Timed automata (TA) [AD94] are system models where quantitative timing constraints are added to discrete transition systems. So far most of the

* This work was partially supported by the European Community Esprit-LTR Project 26270 VHS (Verification of Hybrid systems), the French-Israeli collaboration project 970MAEFUT5 (Hybrid Models of Industrial Plants) and by Research Grants 97-01-00692 and 96-15-96048 of Russian Foundation of Basic Research.

F.W. Vaandrager and J.H. van Schuppen (Eds.): HSCC'99, LNCS 1569, pp. 19–30, 1999.

work on verification and synthesis for TA concentrated on qualitative all-or-nothing properties[1] and the only (important) role of the timing information was to constrain the set of possible behaviors. In this work we associate a most natural performance measure to behaviors of TA, namely the *time* which elapses until they reach a certain set F of configurations.

In order to treat the synthesis problem we use our previously-introduced [MPS95,AMPS98] model of *Timed Game Automaton* (TGA) which is nothing but the usual timed automaton of [AD94] where the actions are partitioned into those of the controller and those of the uncontrolled environment. On this model various controller synthesis problems for *qualitative* properties can be formulated and solved. One example is the eventuality problem: find the set of "winning" states from which the controller can enforce the TGA into a set F, and compute the "strategy" for these states. In this work we solve the following quantitative generalization: find for each state the *minimal* time in which the controller can enforce the automaton into F, regardless of uncontrolled events, and construct a controller which achieves this optimum for each state. Previous techniques could only tell whether or not this minimal time is finite.

As in [AMPS98] the solution is obtained by a backward fixed-point calculation on the state-space of the TGA. The main difference is that the iteration is performed on a function from the TGA state-space into $\mathbb{R}$ which denotes roughly the min-max of the temporal distance from the state to F (this is similar to value/policy iterations used in dynamic programming, e.g. [Ber95]). Since the state-space of a TGA is non-countable, these functions cannot be tabulated, as is done in shortest-path algorithms on finite graphs. Fortunately we prove that as in the restricted case of functions from clocks to $\{0, 1\}$, all the value functions encountered in the iteration of our synthesis algorithm belong to a special well-founded class of functions admitting a simple linear-algebraic representation. This guarantees the termination of our algorithm.

The rest of the paper is organized as follows: In section 2 we re-introduce the *Timed Game Automaton* model. In section 3 we formulate the controller synthesis problem and describe our synthesis algorithm. In section 4 we prove that the algorithm is effective and correct (partial correctness and termination).

2 Games on Timed Systems

2.1 The Context

Timed games are extensions of discrete games (or equivalently of plant models used in the theory of supervisory control for discrete event systems [RW87]) where the players' actions may take place anywhere on the physical time axis, subject to certain timing constraints. For such games we take the model of *timed*

[1] To be more precise, quantitative information can be expressed in various real-time logics, but not in its full richness: one can verify whether all behaviors of a system reach the target within a given fixed bound, but not determine the minimal constant for which such a property is true.

automata [AD94], in which automata are equipped with auxiliary continuous variables called *clocks* which grow uniformly when the automaton is in some state. The clocks interact with the transitions by participating in pre-conditions (guards) for certain transitions and they are possibly reset when some transitions are taken.

In this continuous-time setting, a player might choose at a given moment to wait some time t and *then* take a transition. In this case, it should consider not only what the adversary can do *after* this action but also the possibility that the latter *will not wait for* t *time*, and perform an action at some $t' < t$.

For a more elaborate description of the game-theoretic approach to untimed and timed synthesis, the reader is encouraged to look at the lengthy introductions of [AMP95,AMPS98] as well as [TLS98].

2.2 Timed Game Automata

Let Q be a finite set of *states* and let $X = \mathbb{R}_+^d$ for some integer d be the *clock space*. We denote elements of X as $\boldsymbol{x} = (x_1, \ldots, x_d)$ and use $\boldsymbol{x} + t$ for $\boldsymbol{x} + (t, t, \ldots, t)$. Elements of $Q \times X$ are called *configurations*. A subset of X is called a *k-zone* if it can be obtained as a boolean combination of inequalities of the form $x_i \leq c$, $x_i < c$, $x_i - x_j \leq c$, where $c \in \{0, 1, \ldots, k\}$. The set of zones is denoted by $\mathcal{Z}(X)$. A zone Z is *right-open* if for any $\boldsymbol{x} \in Z$ there is a $\tau > 0$ such that the interval $(\boldsymbol{x}, \boldsymbol{x} + \tau)$ is also included in Z. These properties of subsets of X extend naturally to subsets of $Q \times X$, e.g. we say that $P \subseteq Q \times X$ is a zone if it can be written as finite union of sets of the form $\{q_i\} \times P_i$ such that every P_i is a zone. A function $\rho : X \to X$ is a *reset* function if it sets some of the coordinates of its argument to 0 and leaves the others intact. The set of all such functions is denoted by $\mathcal{J}(X)$. We assume two finite sets of actions A and B, a special empty action ε and let $A^\varepsilon = A \cup \{\varepsilon\}$ and $B^\varepsilon = B \cup \{\varepsilon\}$. The set A represents our actions (i.e. possible actions of the controller), while B stands for uncontrollable actions of the environment. The action ε stands for "wait and see".

Definition 1 (Timed Game Automaton).
A Timed game automaton (TGA) is a tuple $\mathcal{A} = (Z, A, B, T^A, T^B, \delta, \rho)$ *where* $Z \subseteq Q \times X$ *is a zone,* Q *and* X *are the state and clock spaces,* A *and* B *are two distinct action alphabets,* $T^A \subseteq Q \times X \times A^\varepsilon$ *and* $T^B \subseteq Q \times X \times B^\varepsilon$ *are timing constraints for the two types of actions, the functions* $\delta : Q \times A^\varepsilon \times B^\varepsilon \to Q$ *and* $\rho : Q \times A^\varepsilon \times B^\varepsilon \to \mathcal{J}(X)$ *indicate which state is reached when performing a (possibly joint) action and which clocks are reset in that occasion.*

Further requirements are the following: for every state q and action $a \in A^\varepsilon$, the set $T^A(q, a) = \{\boldsymbol{x} : (q, \boldsymbol{x}, a) \in T^A\}$ is a k-zone. We assume k to be fixed throughout the paper — it is the largest constant in the definition of the TGA. Similar requirements hold for T^B. We assume that $\delta(q, \varepsilon, \varepsilon) = q$ and that $\rho(q, \varepsilon, \varepsilon)$ is the identity function (if both sides refrain from action nothing happens). We require that the automaton is *strongly non-Zeno*, that is, in every

cycle in the transition graph of the automaton (induced by δ), there is at least one transition which resets a clock variable x_i to zero, and at least one transition which can be taken only if $x_i \geq 1$. This is a very important condition as it prevents the controller and the environment to achieve their goals using unrealistic tricks that stop time.

The last requirement is that the zone $T^A(q, \varepsilon)$ is *right-open* for any q. This means that if player A is allowed to wait in a configuration $(q, \boldsymbol{x})$, then it can really wait for a small additional positive amount of time. This requirement is important for preventing infeasible zero-time idling in A's strategy

Intuitively, when the automaton is at a configuration $(q, \boldsymbol{x})$, time can progress as long as both players agree, that is, $(q, \boldsymbol{x}, \varepsilon) \in T^B \cap T^A$. As soon as one of them can take an action, i.e. $(q, \boldsymbol{x}, a) \in T^A$ for some $a \in A$ or $(q, \boldsymbol{x}, b) \in T^B$ for some $b \in B$, or both, a transition can be taken. This can be formalized as follows:

Definition 2 (Steps and Runs). *A* joint step *of a TGA $\mathcal{A}$ is $(q, \boldsymbol{x}) \longrightarrow (q', \boldsymbol{x}')$ which is either*

1. *a time step (of duration t):*

$$(q, \boldsymbol{x}) \xrightarrow{t} (q, \boldsymbol{x} + t)$$

   such that $t > 0$, and for every $t' < t$,
   $(q, \boldsymbol{x} + t', \varepsilon) \in T^A \cap T^B$.
2. *a discrete step*

$$(q, \boldsymbol{x}) \xrightarrow{(a,b)} (q', \boldsymbol{x}')$$

   such that $(a, b) \neq (\varepsilon, \varepsilon)$, $(q, \boldsymbol{x}, a) \in T^A$, $(q, \boldsymbol{x}, b) \in T^B$, $q' = \delta(q, a, b)$, and $\boldsymbol{x}' = \rho(q, a, b)(\boldsymbol{x})$.

A run *of a TGA $\mathcal{A}$ starting from $(q_0, \boldsymbol{x}_0)$ is a sequence of joint steps*

$$\xi : (q_0, \boldsymbol{x}_0) \longrightarrow (q_1, \boldsymbol{x}_1) \longrightarrow \ldots$$

Note that $(q, \boldsymbol{x}, \varepsilon) \in T^A \cap T^B$ means that both A and B agree to let time progress by a *positive* amount. On the other hand it is possible to reach a state where ε is not permitted by one or more of the two players: in such a situation the only thing that can happen is a discrete step.

For ease of notation we introduce the total transition function $\bar{\delta} : Q \times X \times A^\varepsilon \times B^\varepsilon \to Q \times X \cup \{\top, \bot\}$. Put

$$\bar{\delta}(q, \boldsymbol{x}, a, b) = \begin{cases} (q, \boldsymbol{x}) \text{ when } a = b = \varepsilon, (q, \boldsymbol{x}, \varepsilon) \in T^A \cap T^B \\ \bot \text{ when } (q, \boldsymbol{x}, a) \notin T^A \\ \top \text{ when } (q, \boldsymbol{x}, a) \in T^A, (q, \boldsymbol{x}, b) \notin T^B \\ (\delta(q, a, b), \rho(q, a, b)(\boldsymbol{x})) \quad \text{otherwise.} \end{cases}$$

Notice that the last line corresponds to the "normal" case when $(a, b) \neq (\varepsilon, \varepsilon)$, $(q, \boldsymbol{x}, a) \in T^A$, and $(q, \boldsymbol{x}, b) \in T^B$.

Clearly, replacing T^A by any subset of it, will restrict the range of action that the player A can chose at certain configurations, and hence decrease the set of behaviors of the TGA. We formulate various controller synthesis problems as finding appropriate restrictions of T^A, that we call *strategies*. These strategies are not necessary deterministic as they might allow A more than one action at a given configuration.

3 The Problem and the Algorithm

Definition 3 (Brachystochronic Problem). *Given a TGA $\mathcal{A}$ and a set $F \subset Q \times X$ find a strategy $T^A_* \subseteq T^A$ for player A which allows him or her to reach the target set F as fast as possible whatever player B does.*

The following algorithm is a generalization of the synthesis method for eventuality games on timed automata. The algorithm iterates over the value function which gives an upper estimate of arrival times to the set F. When we have a value function $f : Q \times X \cup \{\top, \bot\} \to \mathbb{R}_+ \cup \{0, \infty\}$ at any step of the iteration it means that we have a controller which allows to reach F from $(q, \boldsymbol{x})$ in no more than $f(q, \boldsymbol{x})$ time. The algorithm for finding the value function has the following form:

Algorithm 1 (Value Iteration for TGA).

{Initialization}

$$f_0(q, \boldsymbol{x}) = \begin{cases} 0 \text{ when } (q, \boldsymbol{x}) \in F \cup \{\top\} \\ \infty \qquad \text{otherwise;} \end{cases}$$

$n := 0$;

{Iteration}

repeat

$n := n + 1$;

$f_n := \pi(f_{n-1})$;

until $f_n = f_{n-1}$;

{Strategy extraction}

$f_* := f(n)$;

$T^A_* = \alpha(f_*)$.

The operators used in the algorithm are as follows:

$$\pi(f) = \min\{f, \pi_{\mathrm{Act}}(f), \pi_{\mathrm{Idle}}(f)\}, \tag{1}$$

where

$$\pi_{\mathrm{Act}}(f)(q, \boldsymbol{x}) = \min_{a \in A} \max_{b \in B^\varepsilon} f(\bar{\delta}(q, \boldsymbol{x}, a, b)) \tag{2}$$

and

$$\pi_{\text{Idle}}(f)(q,\boldsymbol{x}) = \inf_{t\in\mathbb{R}_+} v(q,\boldsymbol{x},t) \tag{3}$$

where

$$v(q,\boldsymbol{x},t) = \max(\sup_{\tau<t} g(q,\boldsymbol{x},\tau), t + f(q,\boldsymbol{x}+t)) \tag{4}$$

and

$$g(q,\boldsymbol{x},\tau) = \max_{b\in B}(\tau + f(\bar{\delta}(q,\boldsymbol{x}+\tau,\varepsilon,b))) \tag{5}$$

Intuitively $g(q,\boldsymbol{x},\tau)$ specifies the worst (largest) f of the configuration in which the game might be after B has performed an action at τ while A was idling. Similarly $v(q,\boldsymbol{x},t)$ is the worst thing that can happen to A after deciding to idle for t time at $(q,\boldsymbol{x})$: it includes all the possible outcomes of B's actions at $\tau < t$. The best waiting time for A is the t which minimizes $v(q,\boldsymbol{x},t)$ and its outcome π_{Idle} is compared with π_{Act} which denotes the best outcome of an immediate action.

The strategy extraction operator is given by the formula

$$\alpha(f)(q,\boldsymbol{x}) = \begin{cases} \alpha_{Act}(f)(q,\boldsymbol{x}) \text{ when } \pi_{\text{Act}}(f)(q,\boldsymbol{x}) < \pi_{\text{Idle}}(f)(q,\boldsymbol{x}) \\ \{\varepsilon\} \text{ when } \pi_{\text{Idle}}(f)(q,\boldsymbol{x}) < \pi_{\text{Act}}(f)(q,\boldsymbol{x}) \\ \alpha_{Act}(f)(q,\boldsymbol{x}) \cup \{\varepsilon\} \text{ when } \pi_{\text{Act}}(f)(q,\boldsymbol{x}) = \pi_{\text{Idle}}(f)(q,\boldsymbol{x}) < \infty \\ \emptyset \text{ when } \pi_{\text{Act}}(f)(q,\boldsymbol{x}) = \pi_{\text{Idle}}(f)(q,\boldsymbol{x}) = \infty \end{cases}$$

where

$$\alpha_{Act}(f)(q,\boldsymbol{x}) = \{a \in T^A(q,\boldsymbol{x}) | \max_{b\in B^\varepsilon} f(\bar{\delta}(q,\boldsymbol{x},a,b) = \pi_{\text{Act}}(f(q,\boldsymbol{x}))\}.$$

Unfortunately, the subtleties of real-valued time complicate the situation a bit. The problem is that the strategy extracted by α might violate the right-openness requirement. This can happen in a situation where player A has to take a transition the sooner the better but *after* time t_0. This is reflected in the strategy $\alpha(f)$ as follows: player A has action ε enabled until t_0 *including* t_0 and a "good" discrete transition enabled *after* t_0. But formally speaking, this strategy is blocking at time t_0.

To overcome this problem we introduce non-blocking approximations $\alpha_\varsigma(f)$ for small $\varsigma > 0$, which satisfy the right-openness conditions (but use non-integers in the guards). As we will show in the next section, taking this strategy for ς small enough, player A can reach F in time arbitrarily close to $f_*(q,\boldsymbol{x})$.

The ς-relaxed strategy is obtained from $\alpha(f)$ by enabling the ε action on a narrow stripe: $\alpha_\varsigma(f) = \alpha(f) \cup \{(q,\boldsymbol{x},\varepsilon) : (q,x) \in S_\varsigma(f)\}$, where

$$S_\varsigma(f) = \text{interior}\{(q,\boldsymbol{x}) : \pi_{\text{Idle}}(f)(q,\boldsymbol{x}) < \pi_{\text{Act}}(f)(q,\boldsymbol{x}) + \varsigma\}.$$

4 Correctness Proof

4.1 Partial correctness

Our first aim is to prove that if the algorithm converges then it gives the optimal least-restrictive solution of the problem.

Lemma 1. *If player A uses the strategy $T^A_{n,\varsigma} = \alpha_\varsigma(f_{n-1}))$ from an initial position in $(q, \boldsymbol{x})$ with $v = f_n(q, \boldsymbol{x}) < \infty$, then the TGA reaches the target set F in no more then $v+n\varsigma$ time with no more then n transitions whatever the adversary does.*

Proof. Straightforward induction over n. □

Let $\lambda > 0$ be any given (small) positive number.

Corollary 1. *Suppose that algorithm 1 converges in N iterations. Let $\varsigma = \lambda/N$ If player A uses the strategy $T^A_{*,\varsigma} = \alpha_\varsigma(f_*)$ from an initial position $(q, \boldsymbol{x})$ such that $v = f_*(q, \boldsymbol{x}) < \infty$ then the TGA reaches the target set F in no more then $v + \lambda$ time whatever the adversary does.*

Lemma 2. *For every $(q, \boldsymbol{x})$ and n, player B can prevent the TGA from reaching F during at least n steps or $f_n(q, \boldsymbol{x}) - \lambda$ time, whatever player A does.*

Proof. Straightforward induction over n. □

Corollary 2. *Whatever player A does from an initial position $(q, \boldsymbol{x})$ player B can prevent the TGA from reaching the target set F during at least $f_*(q, \boldsymbol{x}) - \lambda$ time.*

Corollary 3. *If from an initial position $(q, \boldsymbol{x})$ player A makes a transition not in T^A_*, then the adversary B can prevent the TGA from reaching the target set F during strictly more than $f_*(q, \boldsymbol{x})$ time.*

These corollaries imply the partial correctness of the algorithm.

Claim. Suppose the algorithm 1 converges. Then

1. The function $f_*(q, \boldsymbol{x})$ gives the infimal time necessary for A to drive the TGA from $(q, \boldsymbol{x})$ to F whatever B does;
2. The strategy $T^A_{*,\varsigma} = \alpha_\varsigma(f_*)$ guarantees arrival to F in time $f_*(q, \boldsymbol{x}) + \lambda$;
3. It is least restrictive: any strategy which guarantees arrival to F in time $f_*(q, \boldsymbol{x})$ is a sub-strategy of T^A_*.

4.2 Simple functions

Now we need to prove two properties of Algorithm 1. The first one is effectiveness: the algorithm manipulates functions from $Q \times \mathbb{R}^n$ to $\mathbb{R}$ and we need to find a finite representation of these functions in order to implement the algorithm. The second issue is to prove that the algorithm converges. Both effectiveness and finite convergence follow from the fact that all the functions used in the algorithm belong to the following class which is closed under the operator π.

Definition 4. *A function $f : X \to \mathbb{R}_+ \cup \{0, \infty\}$ is referred to as a* k-simple *one iff it can be represented as*

$$f(\boldsymbol{x}) = \begin{cases} c_i \text{ when } \boldsymbol{x} \in D_i \\ d_j - x_{l_j} \text{ when } \boldsymbol{x} \in E_j \end{cases}$$

where $D_i, i = 1, \ldots, M$ and $E_j, j = 1, \ldots, N$ are k-zones, $E_j \subseteq \{\boldsymbol{x} | x_{l_j} \leq k\}$ and $c_i, d_j \in \mathbb{N} \cup \{\infty\}$. A function $f : Q \times X \to \mathbb{R}_+ \cup \{0, \infty\}$ is k-simple if $f(q, \cdot)$ is k-simple for any fixed $q \in Q$

The additional boundedness condition on the E_j's means that the value of x_{l_j} can influence f only when $x_{l_j} < k$, the largest constant in the definition of the TGA. Clearly, any k-simple function admits a finite representation as a list of zones D_i and E_j and of constants c_i and d_j. It is easy to see that equality of k-simple functions is decidable, and that they satisfy some closure properties:

Lemma 3. *If $f(\boldsymbol{x})$, $g(\boldsymbol{x})$ are k-simple and $\rho(\boldsymbol{x})$ a reset function, then* $\max(f, g)$, $\min(f, g)$ *and $f(\rho(\boldsymbol{x}))$ are k-simple as well. The set $\{\boldsymbol{x} | f(\boldsymbol{x}) < g(\boldsymbol{x})\}$ is a k-zone.*

Notice that all closure results mentioned in this subsection are effective in the sense that it is possible to transform algorithmically representations of the original functions into representations of the resulting functions.

Another property of k-simple functions will be crucial for the proof of convergence of the algorithm.

Lemma 4 (Well-foundedness). *Any decreasing sequence of k-simple functions is finite.*

4.3 Effectiveness and convergence

The key step in the proof of convergence and effectiveness of Algorithm 1 is the closure of simple function under the operators used in the algorithm. Let $\Delta(\boldsymbol{x}, Z)$ be the largest backward distance from $\boldsymbol{x}$ to a zone Z, i.e. $\Delta(\boldsymbol{x}, Z) = \sup\{t \geq 0 | \boldsymbol{x} + t \in Z\}$. It is easy to see that Δ is simple in the first argument.

Lemma 5 (Closure of k-simple functions under π). *If $f(q, \boldsymbol{x})$ is k-simple and the TGA is k-bounded, then $\pi_{\text{Act}}(f)$, $\pi_{\text{Idle}}(f)$ and $\pi(f)$ are k-simple. Their representations can be found effectively.*

Proof. For $\pi_{\mathrm{Act}}(f)$ it follows immediately from lemma 3 and the fact that for any fixed a, b and q the function $\bar{\delta}(q, \cdot, a, b)$ is a reset function (restricted to a zone).

The case of $\pi_{\mathrm{Idle}}(f)$ is more difficult. Looking at equations (3), (4), and (5) one can see that $\pi_{\mathrm{Idle}}(f)$ is obtained by elimination (via inf and sup) of the t variable from functions whose domain includes state-space and time. For example $g(q, \boldsymbol{x}, \tau)$ of (5) is defined as

$$g(q, \boldsymbol{x}, \tau) = \max_{b \in B}\{\tau + f(\bar{\delta}(q, \boldsymbol{x} + \tau, \varepsilon, b))\} = \tau + \max_{b \in B}\{\bar{\delta}(q, \boldsymbol{x} + \tau, \varepsilon, b)\}$$

and from the closure of $k-$simple function under reset, we can conclude that $g(q, \boldsymbol{x}, t)$ is of the form $t + f'(q, \boldsymbol{x} + t)$ for a k-simple function f'. This motivates the definition of the following class of functions:

Definition 5 (Bizones and Nice Functions). *A* k-bizone *is a union of sets of the form* $\{(\boldsymbol{x}, t) | \boldsymbol{x} \in C \wedge \boldsymbol{x} + t \in D\}$ *where* C *and* D *are* k*-zones. A function* $f : X \times \mathbb{R}_+ \to \mathbb{R}_+ \cup \{0, \infty\}$ *is referred to as* k-nice *iff it can be represented as*

$$g(\boldsymbol{x}, t) = \begin{cases} c_i + t \text{ when } (\boldsymbol{x}, t) \in D_i \\ d_j - x_{l_j} \text{ when } (\boldsymbol{x}, t) \in E_j, \end{cases}$$

where $D_i, i = 1, \ldots, M$ *and* $E_j, j = 1, \ldots, N$ *are* k*-bizones,* $E_j \subseteq \{\boldsymbol{x} | x_{l_j} \leq k\}$ *and* $c_i, d_j \in \mathbb{N} \cup \{\infty\}$.

Sublemma 6 (Properties of k-Nice Functions).

1. *If* $f(\boldsymbol{x})$ *is* k*-simple, then* $t + f(\boldsymbol{x} + t)$ *is* k*-nice.*
2. *If* g *and* h *are* k*-nice so are* $\min\{g, h\}$ *and* $\max\{g, h\}$.
3. *If* $g(\boldsymbol{x}, \tau)$ *is* k*-nice, then* $h(\boldsymbol{x}, t) = \sup_{\tau < t} g(\boldsymbol{x}, \tau)$ *is* k*-nice.*
4. *If* $g(\boldsymbol{x}, t)$ *is* k*-nice, then* $\inf_{t \in \mathbb{R}_+} g(\boldsymbol{x}, t)$ *is* k*-simple.*

Proof.

1. Immediate from definitions.
2. Let
$$g(\boldsymbol{x}, t) = \begin{cases} c_i + t \text{ when } (\boldsymbol{x}, t) \in D_i \\ d_i - x_{l_i} \text{ when } (\boldsymbol{x}, t) \in E_i, \end{cases}$$
and
$$h(\boldsymbol{x}, t) = \begin{cases} r_m + t \text{ when } (\boldsymbol{x}, t) \in P_m \\ s_m - x_{l_n} \text{ when } (\boldsymbol{x}, t) \in Q_m, \end{cases}$$
and let
$$u(\boldsymbol{x}, t) = \max\{g(\boldsymbol{x}, t), h(\boldsymbol{x}, t)\} = \begin{cases} g(\boldsymbol{x}, t) \text{ when } g(\boldsymbol{x}, t) > h(\boldsymbol{x}, t) \\ h(\boldsymbol{x}, t) \text{ when } g(\boldsymbol{x}, t) \leq h(\boldsymbol{x}, t), \end{cases}$$
To obtain a representation for $u(\boldsymbol{x}, t)$ we combine each line of the formula for g with each line of the formula for h and verify that the following types of sets, which appear when comparing lines of these formulae, are bizones.

$\{(\boldsymbol{x},t)|r+t \le c+t\}$**:** In fact it is either $\mathbb{R}^{n+1}_+$ or $\emptyset$, hence a bizone.
$\{(\boldsymbol{x},t)|s-x_i \le c+t\}$**:** It can be written as $\{(\boldsymbol{x},t)|x_i+t \ge s-c\}$, which is a bizone.
$\{(\boldsymbol{x},t)|s-x_i \le d-x_i\}$**:** It is either $\mathbb{R}^{n+1}_+$ or $\emptyset$, hence a bizone.
$\{(\boldsymbol{x},t)|s-x_i \le d-x_j\}$**:** It can be written as $\{(\boldsymbol{x},t)|x_i-x_j \ge s-d\}$, which is a bizone.

When intersected with $D_i \cap P_m$, $D_i \cap Q_m$, $E_i \cap P_m$, and $E_i \cap Q_m$ these sets produce bizones participating in the definition of the function $u(\boldsymbol{x},t)$. The boundedness condition is inherited from the bizones E_i and Q_m. The reasoning for min is identical.

3. Let
$$g(\boldsymbol{x},\tau) = \begin{cases} c_i+\tau \text{ when } (\boldsymbol{x},\tau) \in D_i \\ d_j - x_{l_j} \text{ when } (\boldsymbol{x},\tau) \in E_j, \end{cases}$$
and $h(\boldsymbol{x},t) = \sup_{\tau<t} g(\boldsymbol{x},\tau)$.
Observe that the set $[\boldsymbol{x},\boldsymbol{x}+t] = \{\boldsymbol{x}+\tau | t \in [0,t]\}$ intersects finitely many D_i and E_j sets from the definition of g. We associate with them the following (partial) functions:
$$u_j(\boldsymbol{x},t) = d_j - x_{l_j} \qquad \text{when } [\boldsymbol{x},\boldsymbol{x}+t] \cap E_j \neq \emptyset$$
$$v_i(\boldsymbol{x},t) = c_i - t \qquad \text{when } \boldsymbol{x}+t \in D_i$$
$$w_i(\boldsymbol{x},t) = c_i + \Delta(\boldsymbol{x},D_i) \text{ when } [\boldsymbol{x},\boldsymbol{x}+t] \cap D_i \neq \emptyset \ \wedge \ \boldsymbol{x}+t \notin D_i$$
All the defining conditions are bizones, the functions are k-nice in $\boldsymbol{x}$ and t and so is h which is their max.
4. Similar to statement 3.

This concludes the proof of the sublemma. □

It follows immediately from the sublemma that $\pi_{\text{Idle}}(f)$ is k-simple. The case of $\pi(f)$ is immediate from two previous cases and Lemma 3. □

Corollary 4. *Algorithm 1 is effective.*

Corollary 5. *Algorithm 1 always converges.*

Proof. The sequence f_n is a decreasing sequence of k-simple functions. By virtue of Lemma 4 it stabilizes. □

Together with Claim 4.1, this concludes the correctness proof of the algorithm.

Theorem 1 (Main Result). *Algorithm 1 always converges and produces the least-restrictive optimal strategy for the brachystochronic problem for Timed Automata.*

5 Example

Consider the 1-clock TGA of figure 1 where the adversary B is trivial and the target set F is $\{q_4\}$. From q_0 player A can choose between waiting, going to q_1 (a losing sink), to q_2 (while resetting the clock) or to q_3. Intuitively, for smaller values of x it might be better to do b and go to q_2 because A does not lose much time by resetting the clock and can benefit from the smaller transition guard. The value function and strategy obtained by our algorithm are depicted below.

q	x	f	T_*^A
q_4	$[0,\infty)$	0	$\{\varepsilon\}$
q_1	$[0,\infty)$	∞	$\emptyset$
q_2	$[0,2)$	$2-x$	$\{\varepsilon\}$
	$[2,\infty)$	0	$\{a\}$
q_3	$[0,5)$	$5-x$	$\{\varepsilon\}$
	$[5,\infty)$	0	$\{a\}$
q_0	$[0,3)$	2	$\{b\}$
	$[3,3]$	2	$\{\varepsilon, b\}$
	$(3,4)$	$5-x$	$\{\varepsilon\}$
	$[4,5]$	$5-x$	$\{\varepsilon, c\}$
	$(5,\infty)$	0	$\{c\}$

As one can see, q_4 is a winning state right from the start while from q_1 you can never reach q_4. The clock spaces of q_2 and q_3 are partitioned into two intervals: one after the clock values reach their respective guards, where the value of f is 0, and the interval before that, where f measures the time until the satisfaction of the guards. Finally $f(q_0, x)$ is obtained as $\min\{f(q_1, x), f(q_2, 0), f(q_3, x)\}$ and one can observe that $x = 3$ is the breakpoint after which it is better to take the c transition to q_3. In the absence of an adversary, not all the complexities of the algorithm are demonstrated in this example.

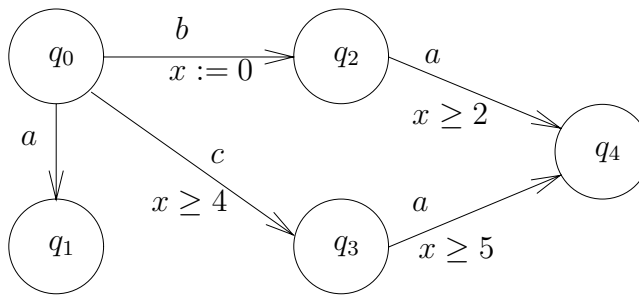

Fig. 1. A TGA with a trivial adversary. Missing guards and invariants are *true*.

6 Concluding Remarks

This work constitutes yet another step in the process of lifting classical results from automata to timed automata, and the main result can be rephrased as finding (min-max) shortest paths in non-countable (but well-behaving) graphs. The algorithm can be easily extended to models where integer costs are associated with transitions. However, assigning different costs to the passage of time at different states will transform the model into a more general hybrid system (such as the ones whose synthesis problem is treated in [W97]), and will probably make the problem harder, if not undecidable.

From the application point of view, many problems related to digital circuit design or scheduling problems in manufacturing and telecommunication, can be formulated as optimal control problems in the framework we have introduced. We believe that unifying the qualitative "property-based" approach, which is dominant in verification, with optimization-oriented approaches used elsewhere[2] is important for the development of hybrid systems research.

References

AD94. Alur, R., Dill, D.L.: A Theory of Timed Automata. *Theoretical Computer Science* **126** (1994) 183–235

AMP95. Asarin, E., Maler, O., Pnueli, A.: Symbolic Controller Synthesis for Discrete and Timed Systems. In: Antsaklis, P., Kohn, W., Nerode, A., Sastry, S. (eds.): *Hybrid Systems II.* Lecture Notes in Computer Science, Vol. 999. Springer (1995) 1–20

AMPS98. Asarin, E., Maler, O., Pnueli, A., Sifakis, J.: Controller Synthesis for Timed Automata. In: *Proc. IFAC Symposium on System Structure and Control.* Elsevier (1998) 469–474

Ber95. Bertsekas, D.P.: *Dynamic Programming and Optimal Control.* Athena Scientific (1995)

MPS95. Maler, O., Pnueli, A., Sifakis, J.: On the Synthesis of Discrete Controllers for Timed Systems. In: Mayr, E.W., Puech, C. (eds.): *Proc. STACS'95.* Lecture Notes in Computer Science, Vol. 900. Springer (1995) 229–242

PA89. Passino, K.M., Antsaklis, P.J.: On the Optimal Control of Discrete Event Systems. In: *Proc. CDC'89.* IEEE (1989) 2713–2718

RW87. Ramadge, P.J., Wonham, W.M.: Supervisory Control of a Class of Discrete Event Processes. *SIAM J. of Control and Optimization* **25** (1987) 206–230

TLS98. Tomlin, C., Lygeros, J., Sastry, S.: Synthesizing Controllers for Nonlinear Hybrid Systems, in Henzinger, T.A., Sastry, S. (eds.): *Hybrid Systems: Computation and Control.* Lecture Notes in Computer Science, Vol. 1386. Springer (1998) 360–373

W97. Wong-Toi, H.: The Synthesis of Controllers for Linear Hybrid Automata. In: *Proc. CDC'97.* IEEE (1997) 4607–4612

[2] In [PA89], for example, transition costs are added to the discrete event models and an optimal controller synthesis problem is formulated and solved.