Synthesis of Optimal Strategies Using HyTech

Patricia Bouyer [1,2,3]

LSV, UMR 8643, CNRS & ENS de Cachan, France

Franck Cassez [1,2,4]

IRCCyN, UMR 6597, CNRS, France

Emmanuel Fleury [5]

Computer Science Department, BRICS, Aalborg University, Denmark

Kim G. Larsen [6]

Computer Science Department, BRICS, Aalborg University, Denmark

Abstract

Priced timed (game) automata extend timed (game) automata with costs on both locations and transitions. The problem of synthesizing an optimal winning strategy for a priced timed game under some hypotheses has been shown decidable in [6]. In this paper, we present an algorithm for computing the optimal cost and for synthesizing an optimal strategy in case there exists one. We also describe the implementation of this algorithm with the tool HyTech and present an example.

Key words: optimal control, timed systems, strategy synthesis

1 Introduction

In recent years the application of model-checking techniques to scheduling problems has become an established line of research. Static scheduling problems with timing constraints may often be formulated as reachability problems

[1] Work supported by the ACI Cortos, a program of the French government.
[2] Visits to Aalborg supported by CISS, Aalborg University, Denmark.
[3] Email: bouyer@lsv.ens-cachan.fr
[4] Email: cassez@irccyn.ec-nantes.fr
[5] Email: fleury@cs.auc.dk
[6] Email: kgl@cs.auc.dk

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

on timed automata, viz. as the possibility of reaching a given goal state. Real-time model checking tools such as Kronos and Uppaal have been applied on a number of industrial and benchmark scheduling problems [1,7,9,12,14,17].

Often the scheduling strategy needs to take into account uncertainty with respect to the behavior of an environmental context. In such situations the scheduling problem becomes a dynamic (timed) game between the controller and the environment, where the objective for the controller is to find a *dynamic* strategy that will guarantee the game to end in a goal state [4,8,16].

A few years ago, the ability to consider quite general performance measures has been given. Priced extensions of timed automata have been introduced [5,3] where a cost c is associated with each location ℓ giving the cost of a unit of time spent in ℓ. Within this framework, it is possible to measure performance of runs and to give optimality criteria for reaching a given set of states.

In [6], we have combined the notions of games and prices and we have proved that, under some hypotheses, the optimal cost in priced timed game automata is computable and that optimal strategies can then be synthetized.

In this paper, we present an algorithm for extracting optimal strategies in priced timed game automata. We also provide an implementation of the algorithm using the tool HyTech [11]. The outline of the paper is as follows: in section 2 we recall the definition of Priced Timed Game Automata and present an example; in section 3 we unveil an optimal cost computation method; in section 4 we detail the algorithm to synthesize the optimal strategies and we give some conclusions in section 6.

The HyTech files given in Fig. 5 and Fig. 7 are available on the web page `http://www.lsv.ens-cachan.fr/aci-cortos/ptga/`. The detailed proofs of the theorems we refer to, as well as complementary definitions and explanations can be found in [6].

2 Priced Timed Games

2.1 Preliminaries

Let X be a finite set of real-valued variables called clocks. We denote $\mathcal{B}(X)$ the set of constraints φ generated by the grammar: $\varphi ::= x \sim k \mid \varphi \wedge \varphi$ where $k \in \mathbb{Z}$, $x, y \in X$ and $\sim \in \{<, \leq, =, >, \geq\}$. A *valuation* of the variables in X is a mapping from X to $\mathbb{R}_{\geq 0}$ (thus an element of $\mathbb{R}_{\geq 0}^X$). For a valuation v and a set $R \subseteq X$ we denote $v[R]$ the valuation that agrees with v on $X \setminus R$ and is zero on R. We denote $v + \delta$ for $\delta \in \mathbb{R}_{\geq 0}$ the valuation s.t. for all $x \in X$, $(v + \delta)(x) = v(x) + \delta$.

2.2 The (R)PTGA Model

Definition 2.1 [RPTGA] A *Priced Timed Game Automaton* (PTGA) G is a tuple $(L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, \mathsf{cost})$ where: L is a finite set of *locations*; $\ell_0 \in L$ is the *initial* location; $\mathsf{Act} = \mathsf{Act}_c \cup \mathsf{Act}_u$ is the set of *actions* (partitioned into controllable and uncontrollable actions); X is a finite set of *real-valued clocks*; $E \subseteq L \times \mathcal{B}(X) \times \mathsf{Act} \times 2^X \times L$ is a finite set of *transitions*; $\mathsf{inv} : L \longrightarrow \mathcal{B}(X)$ associates to each location its *invariant*; $\mathsf{cost} : L \cup E \longrightarrow \mathbb{N}$ associates to each location a *cost rate* and to each discrete transition a *cost* value. We assume that PTGA are deterministic w.r.t. controllable actions (renaming). A *reachability* PTGA (RPTGA) is a PTGA with a distinguished set of locations $\mathsf{Goal} \subseteq L$.

2.3 Runs, Costs of Runs

Let $G = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, \mathsf{cost})$ be a RPTGA. A *configuration* of G is a pair (ℓ, v) in $L \times \mathbb{R}^X_{\geq 0}$. A *run* $\rho = (\ell_0, v_0) \xrightarrow{\delta_0} (\ell'_0, v'_0) \xrightarrow{e_0} (\ell_1, v_1) \xrightarrow{\delta_1} (\ell'_1, v'_1) \xrightarrow{e_1} \cdots (\ell_n, v_n) \xrightarrow{\delta_n} (\ell'_n, v'_n) \xrightarrow{e_n} (\ell_{n+1}, v_{n+1}) \ldots$ in G is a finite or infinite sequence of alternating time ($\delta_i \in \mathbb{R}_{\geq 0}$) and discrete ($e_i \in \mathsf{Act}$) steps such that for every $i \geq 0$: (ℓ_i, v_i) and (ℓ'_i, v'_i) are configurations of G; for each $e_i \in \mathsf{Act}$ there exists a transition $(\ell'_i, g, e_i, Y, \ell_{i+1}) \in E$ such that $v'_i \models g$ and $v_{i+1} = v'_i[Y]$; for each $\delta_i \in \mathbb{R}_{\geq 0}$ $\ell_i = \ell'_i$ and $v'_i = v_i + \delta_i$. The *cost* of a discrete or time step $t = (\ell, v) \xrightarrow{\alpha} (\ell', v')$ is given by $\mathsf{Cost}(t) = \alpha.\mathsf{cost}(\ell)$ if $\alpha \in \mathbb{R}_{\geq 0}$ and $\mathsf{Cost}(t) = \mathsf{cost}((\ell, g, \alpha, Y, \ell))$ if $\alpha \in \mathsf{Act}$. A run ρ of G is *winning* if at least one of the states along ρ is in the set Goal. We note $\mathsf{Runs}(G)$ (resp. $\mathsf{WinRuns}(G)$) the set of (resp. winning) runs in G and $\mathsf{Runs}((\ell, v), G)$ (resp. $\mathsf{WinRuns}((\ell, v), G)$) the set of (resp. winning) runs in G starting in configuration (ℓ, v). If ρ is a *finite* run with $n = 2k$ steps we note $\mathit{last}(\rho) = (\ell_k, v_k)$ and the *cost* of the run ρ is defined by: $\mathsf{Cost}(\rho) = \sum_{0 \leq i \leq n-1} \mathsf{Cost}(t_i)$.

Example 2.2 Consider the RPTGA in Fig. 1. Plain arrows represent controllable actions ($\mathsf{Act}_c = \{c_1, c_2\}$) whereas dashed arrows represent uncontrollable actions ($\mathsf{Act}_u = \{u\}$). Cost rates in locations ℓ_0, ℓ_2 and ℓ_3 are 5, 10 and 1 respectively. In ℓ_1 the environment may choose to move to either ℓ_2 or ℓ_3. However, due to the invariant $y = 0$ this choice must be made instantaneously.

2.4 Strategies, Costs of Strategies

Definition 2.3 [Strategy] Let G be a (R)PTGA. A *strategy* f over G is a partial function from $\mathsf{Runs}(G)$ to $\mathsf{Act}_c \cup \{\lambda\}$.

Definition 2.4 [Outcome] Let $G = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, \mathsf{cost})$ be a (R)PTGA and f a strategy over G. The *outcome* $\mathsf{Outcome}((\ell, v), f)$ of f from configuration (ℓ, v) in G is the subset of $\mathsf{Runs}((\ell, v), G)$ defined inductively by:

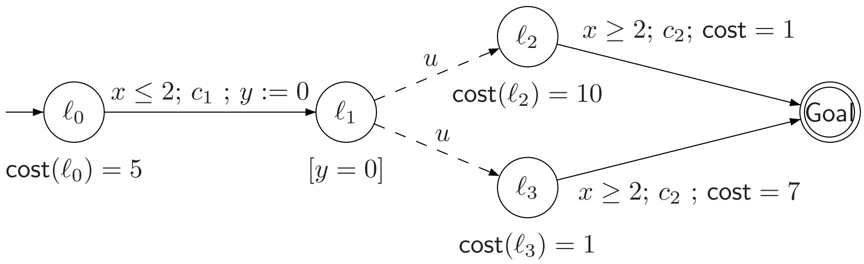

Fig. 1. A Reachability Priced Time Game Automaton $\mathcal{A}$

- $(\ell, v) \in \mathsf{Outcome}((\ell, v), f)$,
- if $\rho \in \mathsf{Outcome}((\ell, v), f)$ then $\rho' = \rho \xrightarrow{e} (\ell', v') \in \mathsf{Outcome}((\ell, v), f)$ if $\rho' \in \mathsf{Runs}((\ell, v), G)$ and one of the following three conditions hold:
  (i) $e \in \mathsf{Act}_u$,
  (ii) $e \in \mathsf{Act}_c$ and $e = f(\rho)$,
  (iii) $e \in \mathbb{R}_{\ge 0}$ and $\forall 0 \le e' < e, \exists (\ell'', v'') \in (L \times \mathbb{R}^X_{\ge 0})$ s.t. $\mathit{last}(\rho) \xrightarrow{e'} (\ell'', v'') \wedge f(\rho \xrightarrow{e'} (\ell'', v'')) = \lambda$.
- an infinite run ρ is in $\in \mathsf{Outcome}((\ell, v), f)$ if all the finite prefixes of ρ are in $\mathsf{Outcome}((\ell, v), f)$.

A strategy f over a RPTGA G is *winning* from (ℓ, v) whenever all maximal[1] runs in $\mathsf{Outcome}((\ell, v), f)$ are winning. We denote $\mathsf{WinStrat}((\ell, v), G)$ the set of winning strategies from (ℓ, v) in G. Let f be a winning strategy from configuration (ℓ, v). The *cost* of f from (ℓ, v) is defined by:

$$\mathsf{Cost}((\ell, v), f) = \sup\{\mathsf{Cost}(\rho) \mid \rho \in \mathsf{Outcome}((\ell, v), f)\}$$

2.5 Optimal Control Problems

Let $(\ell_0, \mathbf{0})$ denote the initial configuration of a RPTGA G. The three main problems we address in this paper are:

Optimal Cost Computation Problem: we want to compute the optimal cost one can expect in a RPTGA G from $(\ell_0, \mathbf{0})$, *i.e.* to compute

$$\mathsf{OptCost}((\ell_0, \mathbf{0}), G) = \inf\{\mathsf{Cost}((\ell_0, \mathbf{0}), f) \mid f \in \mathsf{WinStrat}((\ell_0, \mathbf{0}), G)\}$$

Optimal Strategy Existence Problem: we want to determine whether the optimal cost can actually be reached *i.e.* if there is an optimal strategy $f \in \mathsf{WinStrat}((\ell_0, \mathbf{0}), G)$ such that:

$$\mathsf{Cost}((\ell_0, \mathbf{0}), f) = \mathsf{OptCost}((\ell_0, \mathbf{0}), G)$$

[1] Roughly speaking a run is *maximal* if it can not be extended in the future by a controllable action (see [6] page 6, section 2.2); this point is discussed in the sequel in section 3.2.

Optimal Strategy Synthesis Problem: in case an optimal strategy exists we want to compute a witness.

As the example below shows there are PTGA with no optimal winning strategies. In this case there is a family of strategies f_ε such that

$$|\mathsf{Cost}((\ell_0, \mathbf{0}), f_\varepsilon) - \mathsf{OptCost}((\ell_0, \mathbf{0}), G)| < \varepsilon$$

Thus another problem is, given ε, to compute such an f_ε strategy. This latter synthesis problem is not dealt with in this paper.

Example 2.5 [No optimal strategy] For the PTGA of Fig. 2 there is no optimal strategy. c is a controllable action. Nevertheless we can define a family of strategies f_ε with $0 < \varepsilon \leq 1$ by: $f(\ell_0, x < 1 - \varepsilon) = \lambda$, $f(\ell_0, x = 1 - \varepsilon) = c$ and $f(\ell_1, x \leq 1) = c$. The cost of such a strategy is $1 + \varepsilon$. So we can get as close as we want to 1 but there is no optimal winning strategy.

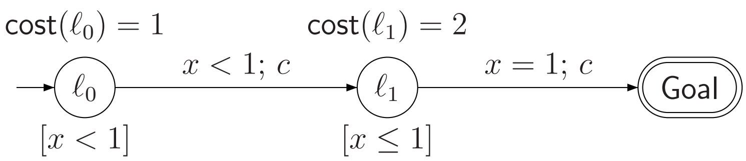

Fig. 2. A PTGA with no reachable optimal cost.

Example 2.6 We consider again Fig. 1. We want to compute an optimal strategy for the controller from the initial configuration. Obviously, once ℓ_2 or ℓ_3 has been reached the optimal strategy for the controller is to move to Goal asap (taking a c_2 action). The crucial (and only remaining) question is how long the controller should wait in ℓ_0 before taking the transition to ℓ_1 (doing c_1). Obviously, in order for the controller to win this duration must be no more than two time units. However, what is the optimal choice for the duration in the sense that the overall cost of reaching Goal is minimal? Denote by t the chosen delay in ℓ_0. Then $5t + 10(2 - t) + 1$ is the minimal cost through ℓ_2 and $5t + (2 - t) + 7$ is the minimal cost through ℓ_3. As the environment chooses between these two paths the best choice for the controller is to delay $t \leq 2$ such that $\max(21 - 5t, 9 + 4t)$ is minimum, which is $t = \frac{4}{3}$ giving a minimal cost of $14\frac{1}{3}$. In Fig. 3 we illustrate the optimal strategy for all states reachable from the initial state provided by our HyTech implementation that will be described in section 3.4.

3 Optimal Cost Computation

In this section we show that computing the optimal cost for a RPTGA amounts to solving a simple[2] control problem on a linear hybrid game automaton [19].

[2] Without cost.

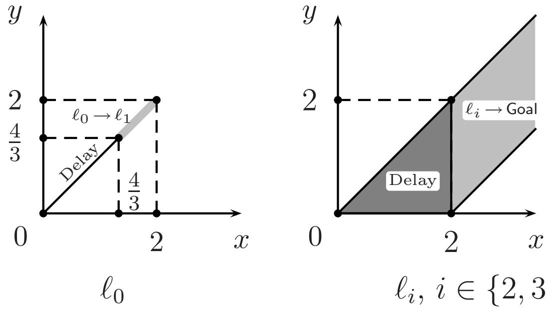

Fig. 3. *Optimal strategy for the RPTGA of Fig. 1. Optimal cost is* $14\frac{1}{3}$.

As a consequence well-known algorithms [19,8] for computing winning states of reachability hybrid games enable us to compute the optimal cost of a RPTGA. We then show how to use the HyTech tool to implement the computation of the optimal cost for RPTGA.

3.1 From Priced Timed Games to Linear Hybrid Games

Assume we want to compute the optimal cost of the RPTGA $\mathcal{A}$ given in Fig. 1. We translate this automaton into a linear hybrid game automaton (LHGA for short) $\mathcal{H}$ (see Fig. 4) where the cost function is encoded into a variable Cost of the LHGA. In $\mathcal{H}$ the variable Cost decreases with rate k in a location ℓ (*i.e.* $\frac{d\mathsf{Cost}}{dt} = -k$ in ℓ) if $\mathsf{Cost}(\ell) = k$ in $\mathcal{A}$. As for discrete transitions the variable Cost is updated by $\mathsf{Cost}' = \mathsf{Cost} - k$ in $\mathcal{H}$ if the corresponding transition's cost in $\mathcal{A}$ is k.

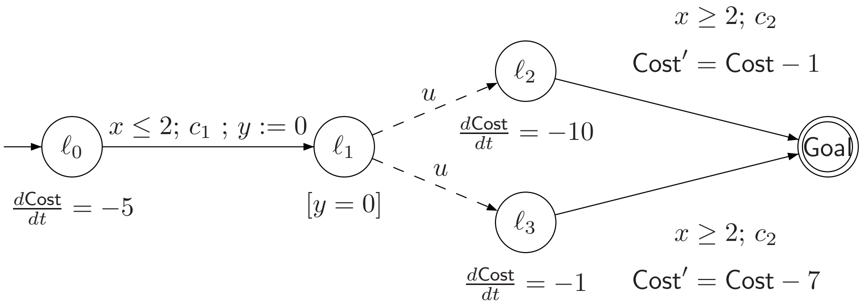

Fig. 4. The Linear Hybrid Game Automaton $\mathcal{H}$.

Let $\mathsf{CompWin}$ be a semi-algorithm (e.g. [19,8]) that computes the largest set of winning states for a reachability hybrid game. Using $\mathsf{CompWin}$ we can compute the largest set of winning states for $\mathcal{H}$ with the goal states given by $\mathsf{Goal} \wedge \mathsf{Cost} \geq 0$. The meaning of this new reachability game is that we want to win without having spent all the resources (Cost) we started with. Assume the corresponding (largest) set of winning states is denoted $\mathsf{CompWin}(\mathcal{H}, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$. The meaning of the set $W = \mathsf{CompWin}(\mathcal{H}, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$ is that in order to win one has to start in the region given by W and if one starts outside W the opponent has a strategy to win *i.e.* we lose. We can prove (see [6], Theorem 5 and Lemma 6) that the (largest) set of winning states W is a union of zones of the form $(\ell, R \wedge \mathsf{Cost} \succ h)$ where ℓ is a location, $R \subseteq \mathbb{R}_{\geq 0}^X$, h is a piece-wise affine function on R and $\succ \in \{>, \geq\}$. Hence we have the answer to the optimal reachability game: we intersect the set of initial states with the set of winning states W, and in case it is not empty, the projection on the Cost axis yields a constraint on the cost like $\mathsf{Cost} \succ k$ with $k \in \mathbb{Q}_{\geq 0}$ and $\succ \in \{>, \geq\}$. By definition of the winning set of states in reachability games, this is the largest set from which we can win, no cost lower than or equal to k is winning and we can deduce that k is the optimal cost. Also we can decide whether there is an optimal strategy or not: if $\succ$ is equal to $>$ there is no optimal strategy and if $\succ$ is $\geq$ there is one.

Thus computing $\mathsf{CompWin}(\mathcal{H}, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$ is a semi-algorithm for computing the optimal cost of the RPTGA $\mathcal{A}$. Moreover we can decide (if $\mathsf{CompWin}$ terminates) whether there exists an optimal strategy or not: in case the initial winning cost set is of the form $\mathsf{Cost} > k$ there is no optimal strategy but a family of strategies f_ε (for $\varepsilon > 0$) with cost lower than $k + \varepsilon$. When this set if of the form $\mathsf{Cost} \geq k$ (and assuming $\mathsf{CompWin}$ terminates) we can compute an optimal strategy (this point is dealt with in the next section 4).

The formal definitions and proofs of this reduction are given in [6] (Definition 12, Lemma 5, Theorems 4 and 5, Corollaries 1 and 2).

3.2 The π Operator

The computation of the winning states (with $\mathsf{CompWin}$) is based on the definition of a *controllable predecessor* operator [16,8]. Let $G = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, \mathsf{cost})$ be a RPTGA and Q its set of configurations. For a set $X \subseteq Q$ and $a \in \mathsf{Act}$ we define $\mathsf{Pred}^a(X) = \{q \in Q \mid q \xrightarrow{a} q', q' \in X\}$. The controllable and uncontrollable discrete predecessors of X are defined by $\mathsf{cPred}(X) = \bigcup_{c \in \mathsf{Act}_c} \mathsf{Pred}^c(X)$, respectively $\mathsf{uPred}(X) = \bigcup_{u \in \mathsf{Act}_u} \mathsf{Pred}^u(X)$. We also need a notion of *safe* timed predecessors of a set X w.r.t. a set Y. Intuitively a state q is in $\mathsf{Pred}_t(X, Y)$ if from q we can reach $q' \in X$ by time elapsing and along the path from q to q' we avoid Y. Formally this is defined by:

$$\mathsf{Pred}_t(X, Y) = \{q \in Q \mid \exists \delta \in \mathbb{R}_{\geq 0} \text{ s.t. } q \xrightarrow{\delta} q',\ q' \in X \wedge \mathsf{Post}_{[0,\delta]}(q) \subseteq \overline{Y}\} \quad (1)$$

where $\mathsf{Post}_{[0,\delta]}(q) = \{q' \in Q \mid \exists t \in [0, \delta] \mid q \xrightarrow{t} q'\}$. We are then able to define a *controllable predecessor* operator π as follows:

$$\pi(X) = \mathsf{Pred}_t\left(X \cup \mathsf{cPred}(X), \mathsf{uPred}(\overline{X})\right) \quad (2)$$

This definition of π captures the choice that uncontrollable actions cannot be used to win (this choice is made in [13] and in [6]). As a matter of fact there is no way to win in the RPTGA of Fig. 1 with this definition of π: ℓ_1 cannot be a winning state if we start iterating the computation of π from Goal as π only adds predecessors that can reach a winning state by a controllable transition.

Another choice is possible: uncontrollable actions may be used to win if they are forced to happen. This second choice is rather involved when one wants to give a new definition of π in the general case. We adopt a position which is half-way between the previous two extremes: if an uncontrollable action is enabled from a state q where time cannot elapse and leads to a winning state q', and no uncontrollable transitions enabled at q can lead to a non-winning state, we declare q as winning. Assume the set of configurations of G where time cannot elapse is denoted $STOP$. Then a new definition of π where uncontrollable actions can be used to win is given by:

$$\pi'(X) = \mathsf{Pred}_t\left(X \cup \mathsf{cPred}(X) \cup (\mathsf{uPred}(X) \cap STOP), \mathsf{uPred}(\overline{X})\right) \tag{3}$$

Note that this choice does not change the results presented in [6]. In the example of Fig. 1, from location ℓ_1 only uncontrollable transitions are enabled, but they are bound to happen within a bounded amount of time (in this case as soon as we reach ℓ_1 because of the invariant $y = 0$). π' will add configuration $(\ell_1, x \geq 0 \wedge y = 0)$ to the set of winning states.

The semi-algorithm $\mathsf{CompWin}$ computes the least fixed point W of the functional $\lambda X.\{X_0\} \cup \pi'(X)$ as the limit of an increasing sequence of sets of states W_i (starting from set $W_0 = X_0$) where $W_{i+1} = \pi'(W_i)$). If G is a RPTGA, the result of the computation of $\mathsf{CompWin}$ on the associated LHGA H starting from $\mathsf{Goal} \wedge \mathsf{Cost} \geq 0$ is $W = \mu X.\{\mathsf{Goal} \wedge \mathsf{Cost} \geq 0\} \cup \pi'(X)$. This result is also denoted $\mathsf{CompWin}(H, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$ and gives the largest set of winning states.

3.3 Termination Issues

An important issue about the previous semi-algorithm $\mathsf{CompWin}$ is whether it terminates or not. We have identified a class of RPTGA for which $\mathsf{CompWin}$ terminates on the associated hybrid game.

Let G be a RPTGA satisfying:

- G is bounded, *i.e.* all clocks in G are bounded [3];
- the cost function of G is *strictly non-zeno*, *i.e.* there exists some $\kappa > 0$ such that the accumulated cost of every cycle in the region automaton associated with G is at least κ. Note that this condition can be checked. For more complete explanations, see [6].

Then the semi-algorithm $\mathsf{CompWin}(H, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$ terminates (H is the hybrid game defined from G in the previous section). The formal statement and proof of this claim is given by Theorem 6 in [6]. We thus get:

Theorem 3.1 *Let G be a RPTGA satisfying the above-mentionned hypotheses (boundedness and strict non-zenoness of the cost). Then the optimal cost is computable for G.*

[3] This hypothesis is not a restriction, see [15].

3.4 Implementation of CompWin *in* HyTech

HyTech [10,11] is a tool that implements "pre" and "post" operators for linear hybrid automata. Moreover it is possible to write programs that use these operators (and many others) on polyhedra in order to compute sets of states. The specification in HyTech of our LHGA $\mathcal{H}$ of Fig. 4 is given in Fig. 5, lines 7–20. We detail this specification in the sequel.

Controllable and Uncontrollable Predecessors. HyTech provides the **pre** operator that computes at once the time predecessors and the discrete predecessors of a set of states. As we need to distinguish between time predecessors, discrete controllable predecessors and discrete uncontrollable predecessors, we use the following trick: in the HyTech source code of the LHA $\mathcal{H}$ we add two boolean variables u and c (Fig. 5, line 4) that are negated on each discrete uncontrollable (resp. controllable) transitions (Fig. 5, lines 10–19). In HyTech one can existentially quantify over a variable t by using the **hide** operator. Then the controllable predecessors can be computed by existentially quantifying over c and over a variable t that has rate[4] -1. We can express the cPred (and uPred) operator with existential quantifiers and two variables t and c as follows:

$$\begin{aligned}\mathsf{cPred}(X) &= \{q \mid \ \exists c \in \mathsf{Act}_c \text{ s.t. } q \xrightarrow{c} q',\, q' \in X\} \\ &= \{q \mid \exists t \text{ s.t. } \exists c \text{ s.t. } t = 0 \wedge c = 0 \wedge \\ &\qquad\qquad (q,t,c) \in \mathbf{pre}(X \wedge t = 0 \wedge c = 1)\}\end{aligned}$$

where **pre** is the predecessor operator of HyTech.

We impose that the value of t stays unchanged to ensure that we just take discrete predecessors (Fig. 5, line 39). For uncontrollable predecessors we replace c by u (Fig. 5, line 41). Note that the computation of $STOP$ states (Fig. 5, line 32) can also be done using our extra variables t, c, u.

Safe Time Predecessors. The other operator $\mathsf{Pred}_t(Z,Y)$ is a bit more complicated. We just need to express it with existential quantification so that it is easy to compute it with HyTech. Also we assume we have time deterministic automata as in this case $\mathsf{Pred}_t(Z,Y)$ is rather simple (if we do not have time determinism a more complicated encoding must be done and we refer the reader to [19] for a detailed explanation.) From equation (1) we get:

$$\begin{aligned}\mathsf{Pred}_t(Z,Y) &= \{q \mid \exists t \geq 0 \text{ s.t. } q \xrightarrow{t} q',\, q' \in Z \ \ \textit{and} \\ &\qquad\qquad \forall 0 \leq t_1 \leq t,\, q \xrightarrow{t} q'' \implies q'' \notin Y\} \\ &= \{q \mid \exists t \geq 0 \text{ s.t. } q \xrightarrow{t} q',\, q' \in Z \ \ \textit{and} \\ &\qquad\qquad \neg(\exists 0 \leq t_1 \leq t,\, q \xrightarrow{t} q'' \wedge q'' \in Y)\}\end{aligned}$$

The latter formula can be encoded in HyTech using the **hide** operator (Fig. 5, lines 47–55) and two auxiliary variables t and t_1 that evolves at rate -1 (note

[4] any rate different from 0 would also do but we need another variable t with rate -1 later on and use this one.

```
var
  x,y: clock;
  cost: analog; -- the cost variable
  c,u: discrete;  -- used to indicate controllable and uncontrollable transitions
  t,t1: analog;  -- used for existential quantification

automaton H
synclabs: ;
 initially  l0 & x=0 & y=0;
loc l0: while x>=0 & y>=0 wait {dcost=-5,dt=-1,dt1=-1}
        when x>=0 & x<=2 do {u'=u,c'=1-c,y'=0} goto l1;
loc l1: while y=0 wait {dcost=0,dt=-1,dt1=-1}
        when True do {u'=1-u,c'=c} goto l2;
        when True do {u'=1-u,c'=c} goto l3;
loc l3: while x>=0 & y>=0 wait {dcost=-10,dt=-1,dt1=-1}
        when x>=2 do {c'=1-c,u'=u,cost'=cost-1} goto Win;
loc l4: while x>=0 & y>=0 wait {dcost=-1,dt=-1,dt1=-1}
        when x>=2 do {c'=1-c,u'=u,cost'=cost-7} goto Win;
loc Win: while True wait {dcost=0,dt=-1,dt1=-1}
end

var init_reg,winning,fix,  -- sets of states
    STOP, -- set of STOP states from which time cannot elapse
    uPreX,uPrebarX,cPreX,X,Y,Z : region ;

-- first define the initial and winning regions
init_reg := loc[H]=l0 & x=0 ;
winning := loc[H]=Win & cost>=0;
-- fix is the fixpoint we want to compute i.e. the set of winning states W
fix := winning;
-- stopped states
STOP := ~(hide t,c,u in t>0 & c=0 & u=0 & pre(True & t=0 & c=0 & u=0) endhide) ;

-- compute the fixpoint of π'
X := iterate X  from winning using {
    -- uncontrollable predecessors of X̄: uPred(X̄)
    uPrebarX := hide t,u in t=0 & u=0 & pre(~X & u=1 & t=0) endhide;
    -- controllable predecessors of X: cPred(X)
    cPreX := hide t,c in t=0 & c=0 & pre(X & t=0 & c=1) endhide ;
    -- uncontrollable predecessors leading to winning states: uPred(X)
    uPreX := hide t,u in t=0 & u=0 & pre(X & u=1 & t=0) endhide;
    -- Z is the the first argument of π' in the paper;
    -- Z = X ∪ cPred(X) ∪ (uPred(X) ∩ STOP)
    Z := (X | cPreX | (uPreX & STOP)) ;
    -- time predecessors of Z from which we can reach Z
    -- and avoid uPred(X̄) all along; X := Pred_t(Z, uPred(X̄))
    X := hide t in
            (hide c,u in t>=0 & c=0 & u=0 & pre(Z & t=0 & c=0 & u=0)
             endhide) &
            ~(hide t1 in
                 (hide c,u in t1>=0 & t1<=t & c=0 & u=0 &
                        pre(uPrebarX & t1=0 & c=0 & u=0)
                  endhide)
              endhide)
         endhide;
    -- add the newly computed regions to the set of already
    -- computed region
    fix := fix | X ;
} ;
-- print the result
print omit all locations hide x,y in fix & init_reg endhide;
```

Fig. 5. Computation of the Optimal Cost.

that those variables are not part of the model but only used in existentially quantified formulas and they do not constrain the behavior of $\mathcal{H}$.) Finally if the computation of π' terminates (Fig. 5, lines 35–59) the set `fix` contains all the winning states. It then suffices to compute the projection on `cost` in the initial state to obtain the optimal cost (Fig. 5, line 61).

Doing this we have solved the first two problems: computing the optimal cost and deciding whether there exists an optimal strategy.

4 Optimal Strategies Computation

In this section we show how to compute an optimal strategy when one exists. Then we give the HyTech implementation of this computation and discuss some properties of those strategies.

4.1 Strategy Synthesis For RPTGA

First we recall some basic properties of strategies for (unpriced) Timed Game Automata (TGA).

A strategy f is

- *state-based* whenever $\forall \rho, \rho' \in \mathsf{Runs}(G)$, $\mathit{last}(\rho) = \mathit{last}(\rho')$ implies that $f(\rho) = f(\rho')$. State-based strategies are also called *memoryless* strategies in game theory [18,8];
- *polyhedral* if for all $a \in \mathsf{Act}_c \cup \{\lambda\}$, $f^{-1}(a)$ is a finite union of convex polyhedra for each location of the game;
- *realizable*, whenever the following holds: for all $\rho \in \mathsf{Outcome}(q, f)$ s.t. f is defined on ρ and $f(\rho) = \lambda$, there exists some $\delta > 0$ such that for all $0 \leq t < \delta$, there exists q' with $\rho \xrightarrow{t} q' \in \mathsf{Outcome}(q, f)$ and $f(\rho \xrightarrow{t} q') = \lambda$.

Strategies which are not realizable are not interesting because they generate empty sets of outcomes. Nevertheless it is not clear from [16,4] how to extract strategies for RPTGA and ensure their realizability as shown by the following example.

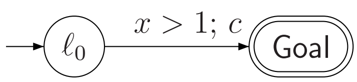

Fig. 6. A timed game automaton

Example 4.1 Consider the PTGA of Fig. 6 where c is a controllable action. The game is to enforce state Goal. The most natural strategy f would be to do a c when $x > 1$ and to wait until x reaches a value greater than 1. Formally this yields $f(\ell_0, x \leq 1) = \lambda$ and $f(\ell_0, x > 1) = c$. This strategy is not realizable. In the sequel, we build a strategy which is $f(\ell_0, x < 2) = \lambda$ and $f(\ell_0, x \geq 2) = c$. Now assume the constraint on the transition is $1 < x \leq 2$. In this case we start with the following strategy (not realizable): $f(\ell_0, x \leq 1) = \lambda$ and $f(\ell_0, 1 < x \leq 2) = c$. To make it realizable we will take the first half of $1 < x \leq 2$ and have a delay action on it i.e. $f(\ell_0, x < \frac{3}{2}) = \lambda$ and $f(\ell_0, \frac{3}{2} \leq x \leq 2) = c$. In the following, we will restrict our attention to realizable strategies and simply refer to them as strategies.

A secondary result is provided in [6] for Linear Hybrid Games (Theorem 2 page 7) that can be rephrased in the context of RPTGA as:

Theorem 4.2 (Adapted from Theorem 2 of [6]) *Let G be a RPTGA. If the semi-algorithm* CompWin *terminates for the hybrid game associated with G (see section 3.1), then we can compute a winning strategy which is: polyhedral, realizable and stated-based.*

Let H be the LHGA associated to the RPTGA $G = (L, \ell_0, \mathsf{Act}, X, E, \mathsf{inv}, \mathsf{cost})$. A state of H is a triple (ℓ, v, c) where $\ell \in L, v \in \mathbb{R}^X_{\geq 0}$ and $c \geq 0$ (c is the value of the variable Cost of H). Thus if we synthetize a realizable winning state-based strategy f for H, we obtain a strategy that depends on the cost value. In case there is a winning strategy for H (see section 3.1) we can synthetize realizable state-based winning strategies for G (see [6], Corollary 2). This result is already satisfying but we would like to build strategies that are independent of the cost value *i.e.* in which there is no need for extra information to play the strategy on the original RPTGA G (this means we want to build a state-based strategy for the original RPTGA G.) To this extent we introduce the notion of *cost-independent* strategies.

Let $W = \mathsf{CompWin}(H, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$ be the set of winning states of H. A state-based strategy f for H is *cost-independent* if $(\ell, v, c) \in W$ and $(\ell, v, c') \in W$ implies $f(\ell, v, c) = f(\ell, v, c')$. Cost-independent strategies in H will then be used for having state-based strategies in G. Theorem 7 of [6] gives then sufficient conditions for the existence of a state-based, optimal, realizable strategy in G and, when back to the automaton reads as follows:

Theorem 4.3 (Adapted from Theorem 7 of [6]) *Let G be a RPTGA. H is the associated LHGA. If* CompWin *terminates for H and the set of winning states is $W = \mathsf{CompWin}(H, \mathsf{Goal} \wedge \mathsf{Cost} \geq 0)$ is a union of sets of the form $(\ell, R \wedge \mathsf{Cost} \geq h)$ where ℓ is a location, $R \subseteq \mathbb{R}^X_{\geq 0}$ and h is a piece-wise affine function on R, then there exists a winning realizable state-based strategy f defined over $W_G = \exists \mathsf{Cost}.W$ s.t. for each $q \in W_G$, $f \in \mathsf{WinStrat}(q, W_G)$ and $\mathsf{Cost}(q, f) = \mathsf{OptCost}(q)$.*

Note that under the previous conditions we build a strategy f which is *uniformly optimal i.e.* optimal for all states of W_G. A syntactical criterion to enforce the condition of theorem 4.3 is that the constraints (guards) on controllable actions are non-strict and constraints on uncontrollable actions are strict. We now give an algorithm to extract such an optimal, state-based, realizable and winning strategy for a RPTGA G.

The example of Fig. 1 satisfies the assumptions of Theorem 4.3 and thus we can compute an optimal strategy for this model. Moreover, the strategy we obtain using HyTech is precisely the one we described in Fig. 3.

4.2 Synthesis of Optimal Strategies

The set of winning states of H is computed iteratively using the functional π' defined by equation (3). In the sequel we need to compute the states that can let a strict positive delay elapse to define the strategy for the delay action. For a set X we denote $\mathsf{NonStop}(X)$ the set of states in X from which a strict positive delay can elapse and all the intermediary states lie in X *i.e.*

$$\mathsf{NonStop}(X) = \{q \in X \,|\, \exists t > 0 \,|\, q + t \in X \wedge \forall 0 \leq t' \leq t,\, q + t' \in X\} \tag{4}$$

Tagged Sets. To synthetize strategies we compute iteratively a set of extended "tagged" states W^+ during the course of the computation of W (this follows from Theorem 2 and Lemma 6 of [6]). The tags will contain information about how a new set of winning states $W_{i+1} = \pi'(W_i)$ has been obtained.

We start with $W_0^+ = \emptyset$ and $W_0 = \mathsf{Goal} \wedge \mathsf{Cost} \geq 0$. Assuming W_i and W_i^+ are the sets obtained after i iterations of π' we define W_{i+1}^+ as follows:

(i) let $Y = W_{i+1} \setminus W_i$ where $W_{i+1} = \pi'(W_i)$;

(ii) for each $c \in \mathsf{Act}_c$, we define the tagged set $\Big(Y \cap \mathsf{cPred}^c(W_i)\Big)^{[c]}$ with the intended meaning: "$Y \cap \mathsf{cPred}^c(W_i)$ has been added to the set of winning states by a Pred^c and doing a c from this set leads to W_i";

(iii) define another tagged set $\Big(\mathsf{NonStop}(Y)\Big)^{[\lambda]}$ with the intended meaning: "$\mathsf{NonStop}(Y)$ has been added to the set of winning states by the (strictly positive) time predecessors operator and letting time elapse will lead to either W_i or $\mathsf{cPred}(W_i)$ or $\mathsf{uPred}(W_i) \cap \mathit{STOP}$";

(iv) define W_{i+1}^+ by :

$$W_{i+1}^+ = W_i^+ \cup \Big(\mathsf{NonStop}(Y)\Big)^{[\lambda]} \cup \bigcup_{c \in \mathsf{Act}_c} \Big(Y \cap \mathsf{cPred}^c(W_i)\Big)^{[c]}$$

Computation of an Optimal Strategy. If $\mathsf{CompWin}$ terminates in j iterations we end up with $W^+ = W_j^+$. Note that by construction a state q of W may belong to several tagged sets $X_0^{[\lambda]}$, $X_1^{[c_1]}$, $\ldots$, $X_n^{[c_n]}$ (where $c_i \in \mathsf{Act}_c$ for each $i \in [1, n]$) of W^+. If the assumptions of theorem 4.3 are satisfied all the X_i's are of the form $X_i' \wedge \mathsf{Cost} \geq h_i$ where $X_i' \subseteq \{\ell_i\} \times \mathbb{R}_{\geq 0}^X$ for some location ℓ_i and $h_i : X_i' \to \mathbb{R}_{\geq 0}$ is a piecewise affine function. Thus the infimum of h_i over X_i' is reachable and equal to the minimum of h_i.

Theorem 7 of [6] states that in this case an optimal state-based strategy f^* for q a winning state of G (no cost) will be obtained by taking the local optimal choice: let $m = \min_{i\in[0,n]} h_i(q)$; then defining $f^*(q) = c_i$ if $h_i(q) = m$ gives an optimal strategy.

As it can be the case that $h_i(q) = h_j(q) = m$ with $i \neq j$, we impose a total order $\sqsubset$ on the set of events in Act_c and define $f^*(q) = c_i$ where $i = \max\{j \mid h_j(q) = m\}$. To avoid realizability problems (see proof of Lemma 6 in [6]) on the boundary of a set X_i if $h_0(q) = m$ (which means that the optimal cost can be achieved by time elapsing) and $h_i(q) = m$ for some $i \in [1, n]$ we define $f^*(q) = c_i$. This can be easily defined in our setting by extending $\sqsubset$ to $\mathsf{Act}_c \cup \{\lambda\}$ and making λ the smallest element.

After these algorithmics explanations, we can summarize how we can synthesize an optimal, cost-independent strategy. We denote $W^+_{[c]}$ the set defined by:

$$W^+_{[c]} = \bigcup_{S_i^{[c]}\in W^+} S_i \tag{5}$$

For each $c \in \mathsf{Act}_c \cup \{\lambda\}$, $W^+_{[c]}$ is a set of the form $X^c \wedge \mathsf{Cost} \geq h^c$ where h^c is a piecewise affine function on X^c (X^c is a union of convex polyhedra). Note that the constraint $\mathsf{Cost} \geq h^c$ is a polyhedron which constrains the Cost variable and the clocks. In what follows, a pair (q, α) will represent a state of H (α is the value of the Cost variable). For each winning state q of G, we want to compute the minimal cost for winning and which action we should do if we want to win with the optimal cost. Let us consider two actions $c_1,\ c_2 \in \mathsf{Act}_c \cup \{\lambda\}$. We denote $[c_1 \leq c_2]$ the set of winning states of G where it is better to do action c_1 than action c_2 ($h^{c_1}(q) \leq h^{c_2}(q)$). This set is defined by:

$$\begin{aligned}
[c_1 \leq c_2] &= \{q \in X^{c_1} \ \mid \exists \alpha_1 \,|\, (q, \alpha_1) \in W^+_{[c_1]} \ \textit{and} \\
&\qquad\qquad\qquad\quad \forall \alpha_2 \,|\, (q, \alpha_2) \in W^+_{[c_2]},\ \alpha_1 \leq \alpha_2\} && (6)\\
&= \big\{q \in X^{c_1} \ \mid \exists \alpha_1 \,|\, (q, \alpha_1) \in W^+_{[c_1]} \wedge \\
&\qquad\qquad\qquad\quad \neg\big(\exists \alpha_2 \,|\, (q, \alpha_2) \in W^+_{[c_2]} \ \textit{and}\ \alpha_2 < \alpha_1\big)\big\} && (7)
\end{aligned}$$

Each set $[c_1 \leq c_2]$ is a polyhedral set. For each $c \in \mathsf{Act}_c \cup \{\lambda\}$ define

$$\mathsf{Opt}(c) = \bigcap_{c'\neq c} [c \leq c'] \tag{8}$$

$\mathsf{Opt}(c)$ is the set of states for which c is an action that gives the optimal cost. $W^* = \bigcup_{c\in\mathsf{Act}_c\cup\{\lambda\}} \mathsf{Opt}(c)$ is thus equal to the set of states on which we need to define the optimal strategy. Given the total order $\sqsubset$ on $\mathsf{Act}_c \cup \{\lambda\}$ with $\lambda \sqsubset c_1 \sqsubset \cdots \sqsubset c_n$, we can define an optimal strategy f^* as follows: for $i \in [0, n-1]$, let $B_i = \big(W^* \setminus (\cup_{k>i} B_k)\big) \cap \mathsf{Opt}(c_i)$ and $B_n = W^* \cap \mathsf{Opt}(c_n)$; define then $f^*(q) = c_i$ if $q \in B_i$. f^* is an optimal strategy that is (winning), state-based, realizable and polyhedral.

4.3 Implementation in HyTech

Controllable Tagged Sets. We first show how to compute tagged sets of states. Our HyTech encoding consists in adding a discrete variable a to the HyTech model of Fig. 5 and use it in the guards of controllable transitions: controllable action c_k of Fig. 4 corresponds to the guard $a = k$ in the HyTech model. The HyTech model of Fig. 5 is enriched as follows: we add the guard $a = 1$ to line 11, $a = 2$ to lines 16 and 18. In this way we achieve the tagging of controllable predecessors as now the computation of cPred (line 39 of Fig. 7) will compute a tagged region that will be a union of polyhedra with some $a = k$ constraints. Note that we also modify line 44 of Fig. 5 and replace it by line 20 in Fig. 7 where a is hidden from the new `cPreX` as a is not needed to compute the winning set of states.

New NonStop States. To compute $\mathsf{NonStop}(Y)$ we use again our extra variables t, c, u and add the tag $a = 0$ to the result set. Lines 35–36 of Fig. 7 achieves this.

W^+ is stored in the region `fix_strat` in the HyTech code. To compute W^+_{i+1} we update `fix_strat` as described by line 38 in Fig. 7.

Computation of the Optimal Strategy. To compare the costs for each action and determine the optimal one we use the trick described in the previous subsection. Each tagged set gives the function h^{c_i} by the means of a constraint between the Cost variable and the rest of the state variables. To compute h^{c_i} we need to split the state space according to each action c_i: this is achieved by lines 44–46 where the state space that corresponds to h^{c_i} is stored in `ri`.

It remains to compute for each pair of actions (c_1, c_2) (c_i can be λ), the set $[c_1 \leq c_2]$ states. The encoding in HyTech of the formula given by equation (7) is quite straightforward using the **hide** operator that corresponds to existential quantification. The strategy is then computed as described at the end of the previous subsection by lines 54–63.

5 Experiments

Using a HyTech-code as described in this paper, we have done some more experiments. The most important example we have treated is a model of a mobile phone with two antennas trying to connect to a base station with an environment which can possibly jam some transmissions.

Description of the Mobile Phone Example. We consider a mobile phone with two antennas emitting on different channels. Making the initial connection with the base station takes 10 time units whatever antenna is in use. Statistically, a jam of the transmission (*e.g.* collision with another phone) may appear every 6 time units in the worst case. When a collision is observed, the antenna tries to transmit with a higher level of energy for a while (at least 5

```
var
  -- same set of variables here as lines 22–24 in Fig. 5 plus some new vars:
  a: discrete;
  cost0,cost1,cost2: analog;
  fix_strat,nonstop,Y,
  r0,r1,r2,
  B0,B1,B2,
  inf_0_1,inf_0_2,inf_1_0,inf_1_2,inf_2_0,inf_2_1: region;

init_reg := loc[H]=l0 & x=0 ;
winning := loc[H]=Win & cost>=0;
fix := winning;
STOP := ~(hide t,c,u in t>0 & c=0 & u=0 & pre(True & t=0 & c=0 & u=0) endhide) ;
fix_strat := False; -- this is new and corresponds to W_0^+ = ∅

X := iterate X from winning using {
    uPrebarX := hide t,u in t=0 & u=0 & pre(~X & u=1 & t=0) endhide;
    cPreX := hide t,c in t=0 & c=0 & pre(X & t=0 & c=1) endhide ;
    uPreX := hide t,u in t=0 & u=0 & pre(X & u=1 & t=0) endhide;
    Z := (X | (hide a in cPreX endhide) | (uPreX & ~uPrebarX & STOP)) ;
    X := hide t in
            (hide c,u in t>=0 & c=0 & u=0 & pre(Z & t=0 & c=0 & u=0)
             endhide) &
            ~(hide t1 in
                 (hide c,u in t1>=0 & t1<=t & c=0 & u=0 &
                        pre(uPrebarX & t1=0 & c=0 & u=0)
                  endhide)
              endhide)
         endhide;

    Y := X & ~fix ; -- store the real new states in Y

    fix := fix | X ;
    -- computation of NonStop(Y)
    nonstop := a=0 & Y &
        hide t,c,u in t>0 & c=0 & u=0 & pre(Y & t=0 & u=0 & c=0) endhide;
    -- computation of fix_strat
    fix_strat := fix_strat | (Y & cPreX) | nonstop ;
} ;
-- print the result as before
print omit all locations hide x,y in fix & init_reg endhide;

-- rename the cost fonction; then ri corresponds to h^ci
r0 := hide a,cost in cost0=cost & fix_strat & a=0 endhide ;
r1 := hide a,cost in cost1=cost & fix_strat & a=1 endhide ;
r2 := hide a,cost in cost2=cost & fix_strat & a=2 endhide ;

-- compute the state space inf_i_j where h^ci(q) <= h^cj(q)
inf_0_1 := hide cost0 in r0 & ~(hide cost1 in r1 & cost1<cost0 endhide) endhide ;
...
inf_2_1 := hide cost2 in r2 & ~(hide cost1 in r1 & cost1<cost2 endhide) endhide ;

-- Output the result taking the best move according to the total order (Act_c ∪ {λ}, ⊏)
prints "Optimal Winning Strategy" ;
prints "do control from l3 or l4 to Win (a=2) on";
B2 := inf_2_0 & inf_2_1 ;
print B2 ;
prints "do control from l0 to l1 (a=1) on";
B1 := inf_1_0 & inf_1_2 & ~B2 ;
print B1 ;
prints "do wait (a=0) on";
B0 := inf_0_1 & inf_0_2 & ~B1 & ~B2 ;
print B0;
```

Fig. 7. Synthesis of Optimal Strategies.

time units for Antenna 1 and at least 2 time units for Antenna 2) and then can switch back to the lower consumption mode. Unfortunately, switching back to the low consumption mode requires more resources and forces to interrupt the other transmission (Antenna 1 resets variable y of Antenna 2 and vice-versa). The overall cost rate (consumption per time unit) for the mobile phone in a product state $s = (\text{low}_x, \text{high}_y, \text{Y})$ is the sum of the rates of Antenna 1 and Antenna 2 (both are working) *i.e.* $1 + 20 = 21$ and $\mathsf{Cost}(s) = 21$ in our model. Once the connection with the base station is established (either $x \geq 10$ or $y \geq 10$) the message is delivered with an energy consumption depending on the antenna ($\mathsf{Cost} = 7$ for **Antenna 1** and $\mathsf{Cost} = 1$ for **Antenna 2**). The aim is to connect the mobile phone with an energy consumption (cost) as low as possible whatever happens in the network (jam).

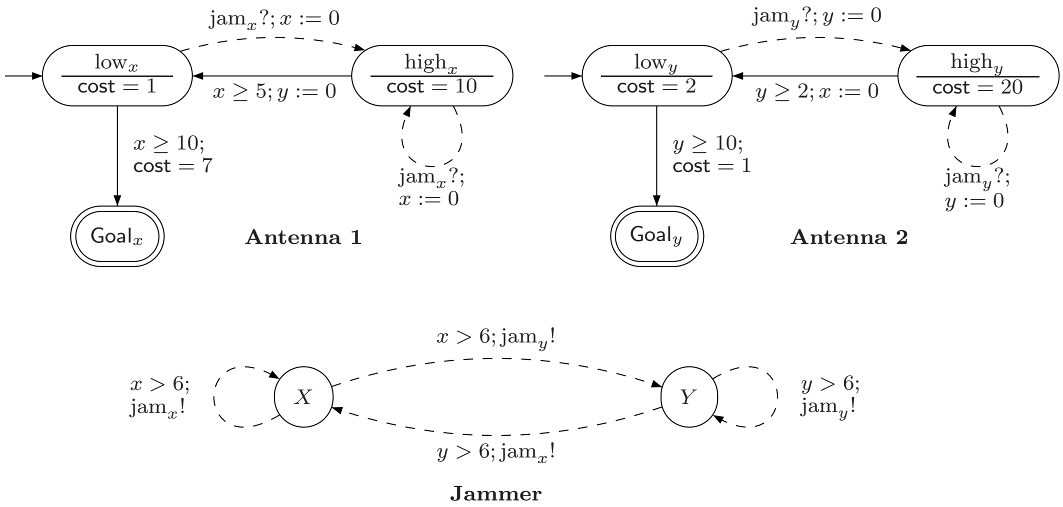

Fig. 8. Mobile Phone Example.

This system can be represented by a network of PTGAs (see Fig. 8 where plain arrows represent controllable actions whereas dsahed arrows represent uncontrollable actions) and the problem reduces to finding an optimal strategy for reaching one of the goal states Goal_x or Goal_y. Note that our original model is a single PTGA and not a network of PTGAs, but networks of PTGA can be used as well because it does not add expressive power and it is simple to define the composed PTGA: in a global location (being a tuple of locations of simple PTGAs), the cost is simply the sum of the costs of all single locations composing it. *Idem* for a composed transition resulting from a synchronization: the cost of the synchronized transition is the sum of the costs of the two initial transitions. Of course, one has to pay attention that no controllable action can synchronize with an uncontrollable action in order that we can define properly the nature, controllable or not, of the synchronization. In this example, see Fig. 8, $\text{jam}_x?$ (resp. $\text{jam}_y?$) synchronizes with $\text{jam}_x!$ (resp.

jam_y!). The HyTech code of this example can be found in [6] and on the web page http://www.lsv.ens-cachan.fr/aci-cortos/ptga/.

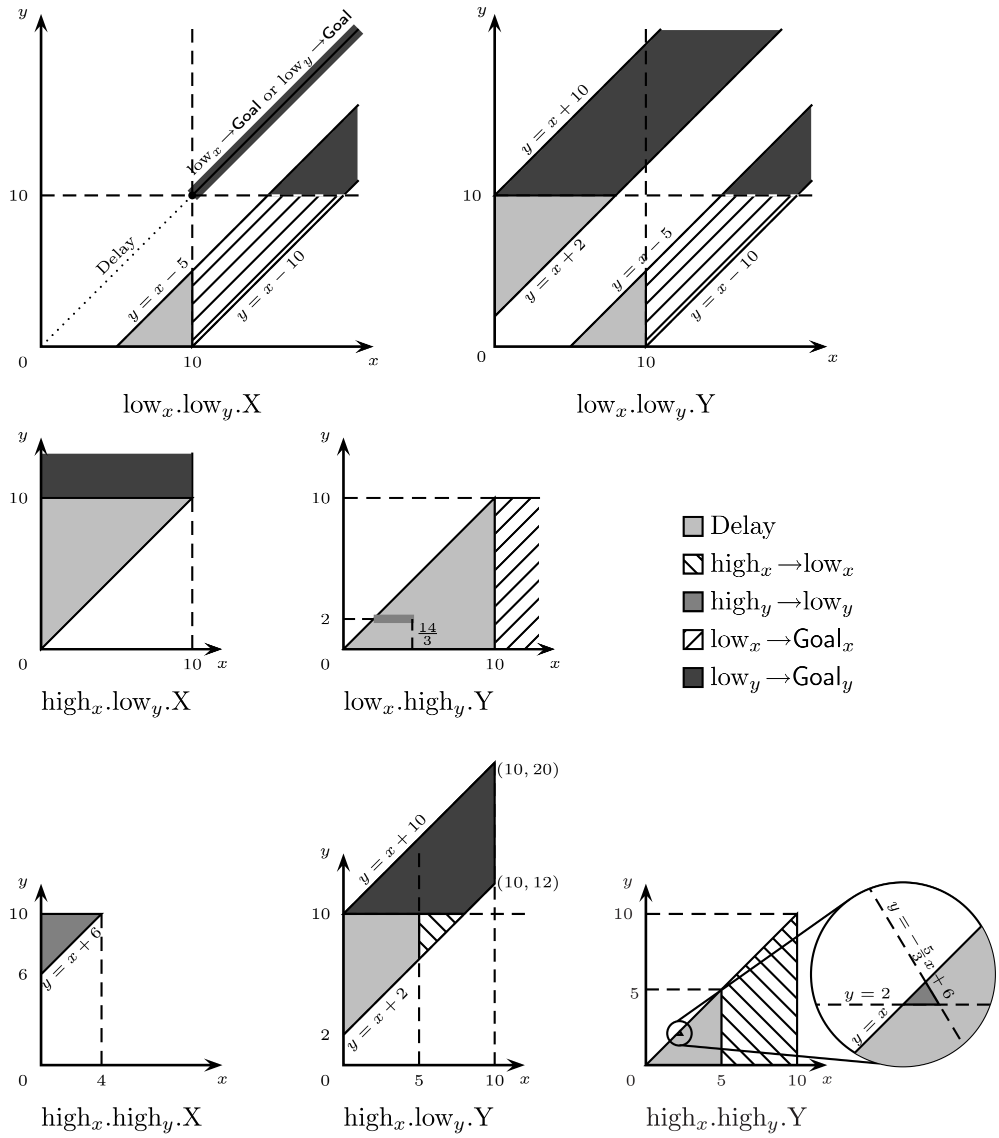

Fig. 9. Optimal Strategy for the Mobile Phone Example

Results of our Experiments. We got that the optimal cost (lowest energy consumption) that can be ensured is 109. The optimal strategy is graphically represented on Fig. 9. The strategy is non-trivial and the actions to take depend on a complex partitioning of the clock space. The computation took 828s on a 12" PowerBook G4 running Mac OS X.

6 Conclusion

In this paper we have described an algorithm to synthesize optimal strategies for a sub-class of priced timed game automata. The algorithm is based on the work described in [6] where we proved this problem was decidable (under some hypotheses we recall in this paper). Morever, we also provide an implementation of our algorithm in HYTECH and demonstrate it on small case-studies. In a recent paper [2] Alur *et al.* addressed a related problem *i.e.* "compute the optimal cost within k steps". They give a complexity bound for this restricted "bounded" problem and prove that the splitting incurred by the computation of the optimal cost within k steps only yields an exponential number (in k and the size of the automaton) of subregions. They do not consider the problem of strategy synthesis.

Our future work consists in extending the class of systems for which the algorithm we provided terminates. The synthesis of sub-optimal strategies (when no optimal strategy exists) is currently being investigated. We would also like to extend this work to more general winning conditions (like safety conditions) and with other performance criteria (as for example the price per unit of time along infinite schedules).

References

[1] Y. Abdeddaim. *Modélisation et résolution de problèmes d'ordonnancement à l'aide d'automates temporisés.* PhD thesis, Institut National Polytechnique de Grenoble, Grenoble, France, 2002.

[2] R. Alur, M, Bernadsky, and P. Madhusudan. Optimal reachability in weighted timed games. In *Proc. 31st International Colloquium on Automata, Languages and Programming (ICALP'04)*, Lecture Notes in Computer Science. Springer, 2004. To appear.

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

[4] E. Asarin, O. Maler, A, Pnueli, and J. Sifakis. Controller synthesis for timed automata. In *Proc. IFAC Symposium on System Structure and Control*, pages 469–474. Elsevier Science, 1998.

[5] G. Behrmann, A. Fehnker, T. Hune, K.G. Larsen, P. Pettersson, Judi Romijn, and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In *Proc. 4th International Workshop on Hybrid Systems: Computation and Control (HSCC'01)*, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer, 2001.

[6] P. Bouyer, F. Cassez, E. Fleury, and K.G. Larsen. Optimal strategies in priced timed game automata. Research Report BRICS RS-04-4, Denmark, Feb. 2004. Available at `http://www.brics.dk/RS/04/4/`.

[7] E. Brinksma, A. Mader, and A. Fehnker. Verification and optimization of a PLC control schedule. *Journal of Software Tools for Technology Transfer (STTT)*, 4(1):21–33, 2002.

[8] L. de Alfaro, T.A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *Proc. 12th International Conference on Concurrency Theory (CONCUR'01)*, volume 2154 of *Lecture Notes in Computer Science*, pages 536–550. Springer, 2001.

[9] A. Fehnker. Scheduling a steel plant with timed automata. In *Proc. 6th International Conference on Real-Time Computing Systems and Applications (RTCSA'99)*, pages 280–286. IEEE Computer Society Press, 1999.

[10] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. A user guide to HYTECH. In *Proc. 1st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'95)*, volume 1019 of *Lecture Notes in Computer Science*, pages 41–71. Springer, 1995.

[11] T.A. Henzinger, P.-H. Ho, and H. Wong-Toi. HyTech: A model-checker for hybrid systems. *Journal on Software Tools for Technology Transfer (STTT)*, 1(1–2):110–122, 1997.

[12] T. Hune, K.G. Larsen, and P. Pettersson. Guided synthesis of control programs using UPPAAL. In *Proc. IEEE ICDS International Workshop on Distributes Systems Verification and Validation*, pages E15–E22. IEEE Computer Society Press, 2000.

[13] S. La Torre, S. Mukhopadhyay, and A. Murano. Optimal-reachability and control for acyclic weighted timed automata. In *Proc. 2nd IFIP International Conference on Theoretical Computer Science (TCS 2002)*, volume 223 of *IFIP Conference Proceedings*, pages 485–497. Kluwer, 2002.

[14] K.G. Larsen. Resource-efficient scheduling for real time systems. In *Proc. 3rd International Conference on Embedded Software (EMSOFT'03)*, volume 2855 of *Lecture Notes in Computer Science*, pages 16–19. Springer, 2003. Invited presentation.

[15] K.G. Larsen, G. Behrmann, E. Brinksma, A. Fehnker, T. Hune, P. Pettersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. 13th International Conference on Computer Aided Verification (CAV'01)*, volume 2102 of *Lecture Notes in Computer Science*, pages 493–505. Springer, 2001.

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

[17] P. Niebert and S. Yovine. Computing efficient operations schemes for chemical plants in multi-batch mode. *European Journal of Control*, 7(4):440–453, 2001.

[18] W. Thomas. On the synthesis of strategies in infinite games. In *Proc. 12th Annual Symposium on Theoretical Aspects of Computer Science (STACS'95)*, volume 900, pages 1–13. Springer, 1995. Invited talk.

[19] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. 36th IEEE Conference on Decision and Control*, pages 4607–4612. IEEE Computer Society Press, 1997.