Electronic Notes in Theoretical Computer Science 119 (2005) 11–31 

[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

Synthesis of Optimal Strategies Using HyTech

Patricia Bouyer [1](#_bookmark2)*,*[2](#_bookmark2)*,*[3](#_bookmark2)

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

Franck Cassez [1](#_bookmark2)*,*[2](#_bookmark2)*,*[4](#_bookmark2)

*IRCCyN, UMR 6597, CNRS, France*

# Emmanuel Fleury [5](#_bookmark2)

*Computer Science Department, BRICS, Aalborg University, Denmark*

# Kim G. Larsen [6](#_bookmark2)

*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](#_bookmark28)]. 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.

*Keywords:* optimal control, timed systems, strategy synthesis

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](mailto:bouyer@lsv.ens-cachan.fr)

4 Email: [cassez@irccyn.ec-nantes.fr](mailto:cassez@irccyn.ec-nantes.fr)

5 Email: [fleury@cs.auc.dk](mailto:fleury@cs.auc.dk)

6 Email: [kgl@cs.auc.dk](mailto:kgl@cs.auc.dk)

1571-0661 © 2005 Published by Elsevier B.V. Open access under [CC BY-NC-ND license](http://creativecommons.org/licenses/by-nc-nd/3.0/).

doi:10.1016/j.entcs.2004.07.006

# Introduction

In recent years the application of model-checking techniques to scheduling problems has become an established line of research. Static scheduling prob- lems with timing constraints may often be formulated as reachability problems 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](#_bookmark25),[7](#_bookmark29),[9](#_bookmark33),[12](#_bookmark36),[14](#_bookmark38),[17](#_bookmark41)].

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](#_bookmark30),[8](#_bookmark31),[16](#_bookmark40)].

A few years ago, the ability to consider quite general performance mea- sures has been given. Priced extensions of timed automata have been intro- duced [[5](#_bookmark32),[3](#_bookmark26)] 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](#_bookmark28)], 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 algo- rithm using the tool HyTech [[11](#_bookmark35)]. The outline of the paper is as follows: in section [2](#_bookmark0) we recall the definition of Priced Timed Game Automata and present an example; in section [3](#_bookmark5) we unveil an optimal cost computation method; in section [4](#_bookmark15) we detail the algorithm to synthesize the optimal strategies and we

give some conclusions in section [6](#_bookmark23).

The HyTech files given in Fig. [5](#_bookmark17) and Fig. [7](#_bookmark21) 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 expla- nations can be found in [[6](#_bookmark28)].

# Priced Timed Games

* 1. *Preliminaries*

Let *X* be a finite set of real-valued variables called clocks. We denote B(*X*) the set of constraints *ϕ* generated by the grammar: *ϕ* ::= *x* ∼ *k* | *ϕ* ∧ *ϕ* where *k* ∈ Z, *x, y* ∈ *X* and ∼∈ {*<,* ≤*,* =*, >,* ≥}. A *valuation* of the variables in *X*

is a mapping from *X* to R≥0 (thus an element of R*X* ). For a valuation *v* and a set *R* ⊆ *X* we denote *v*[*R*] the valuation that agrees with *v* on *X* \ *R* and is zero on *R*. We denote *v* + *δ* for *δ* ∈ R≥0 the valuation s.t. for all *x* ∈ *X*, (*v* + *δ*)(*x*)= *v*(*x*)+ *δ*.

≥0

* 1. *The (R)PTGA Model*

Definition 2.1 [RPTGA] A *Priced Timed Game Automaton* (PTGA) *J* is a tuple (*L, l*0*,* Act*, X, E,* inv*,* cost) where: *L* is a finite set of *locations*; *l*0 ∈ *L* is the *initial* location; Act = Act*c* ∪ Act*u* is the set of *actions* (partitioned into controllable and uncontrollable actions); *X* is a finite set of *real-valued clocks*; *E* ⊆ *L* × B(*X*) × Act × 2*X* × *L* is a finite set of *transitions* ; inv : *L* −→ B(*X*) associates to each location its *invariant*; cost : *L* ∪ *E* −→ 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 Goal ⊆ *L*.

* 1. *Runs, Costs of Runs*

Let *J* = (*L, l*0*,* Act*, X, E,* inv*,* cost) be a RPTGA. A *configuration* of *J* is a

pair (*l, v*) in *L* × R*X* . A *run ρ* = (*l ,v* ) −−*δ*0→ (*l*' *, v*' ) −−*e*0→ (*l ,v* )

−−*δ*1→

≥0 0 0 0 0 1 1

(*l*' *, v*' ) −−*e*1→ ··· (*l ,v* ) −−*δn*→ (*l*' *, v*' ) −−*en*→ (*l*

1 1 *n n n n*

*n*+1

*, vn*+1) *...* in *J* is a finite or

infinite sequence of alternating time (*δi* ∈ R≥0) and discrete (*ei* ∈ Act) steps

such that for every *i* ≥ 0: (*li, vi*) and (*l*'*, v*') are configurations of *J*; for each

*i i*

*ei* ∈ Act there exists a transition (*l*'*, g, ei, Y, li*+1) ∈ *E* such that *v*' |= *g* and

*i*

*vi*+1 = *v*'[*Y* ]; for each *δi* ∈ R≥0 *li* = *l*'

and *v*'

*i*

= *vi* + *δi*. The *cost* of a

*i i* *i*

discrete or time step *t* = (*l, v*) −−*α*→ (*l*'*, v*') is given by Cost(*t*) = *α.*cost(*l*) if *α* ∈ R≥0 and Cost(*t*)= cost((*l, g, α, Y, l*)) if *α* ∈ Act. A run *ρ* of *J* is *winning* if at least one of the states along *ρ* is in the set Goal. We note Runs(*J*) (resp. WinRuns(*J*)) the set of (resp. winning) runs in *J* and Runs((*l, v*)*, J*) (resp. WinRuns((*l, v*)*, J*)) the set of (resp. winning) runs in *J* starting in configuration (*l, v*). If *ρ* is a *finite* run with *n* = 2*k* steps we note *last* (*ρ*) =

(*lk, vk*) and the *cost* of the run *ρ* is defined by: Cost(*ρ*)= Σ0≤*i*≤*n*−1 Cost(*ti*).

Example 2.2 Consider the RPTGA in Fig. [1](#_bookmark1). Plain arrows represent control- lable actions (Act*c* = {*c*1*, c*2}) whereas dashed arrows represent uncontrollable actions (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.



*x* ≥ 2; *c*2; cost = 1

*u*

*l*2

*l*0

*x* ≤ 2; *c*1 ; *y* := 0

*l*1

cost(*l*2) = 10

*u*

Goal

cost(*l*0) = 5

[*y* = 0]

*l*3

*x* ≥ 2; *c*2 ; cost = 7

cost(*l*3) = 1

Fig. 1. A Reachability Priced Time Game Automaton *A*

* 1. *Strategies, Costs of Strategies*

Definition 2.3 [Strategy] Let *J* be a (R)PTGA. A *strategy f* over *J* is a partial function from Runs(*J*) to Act*c* ∪ {*λ*}.

Definition 2.4 [Outcome] Let *J* = (*L, l*0*,* Act*, X, E,* inv*,* cost) be a (R)PTGA and *f* a strategy over *J*. The *outcome* Outcome((*l, v*)*,f* ) of *f* from configu- ration (*l, v*) in *J* is the subset of Runs((*l, v*)*, J*) defined inductively by:

* + - (*l, v*) ∈ Outcome((*l, v*)*,f* ),
    - if *ρ* ∈ Outcome((*l, v*)*,f* ) then *ρ*' = *ρ*

−*e*→ (*l*'*, v*') ∈ Outcome((*l, v*)*,f* ) if

*ρ*' ∈ Runs((*l, v*)*, J*) and one of the following three conditions hold:

1. *e* ∈ Act*u*,
2. *e* ∈ Act*c* and *e* = *f* (*ρ*),
3. *e* ∈ R

and ∀0 ≤ *e*' *< e,* ∃(*l*''*, v*'') ∈ (*L* × R*X* ) s.t. *last* (*ρ*) *e*'

(*l*''*, v*'') ∧

≥0

*f* (*ρ e*'

( '' ''

≥0 −−→

−−→

*l ,v* )) = *λ*.

* an infinite run *ρ* is in ∈ Outcome((*l, v*)*,f* ) if all the finite prefixes of *ρ* are in Outcome((*l, v*)*,f* ).

A strategy *f* over a RPTGA *J* is *winning* from (*l, v*) whenever all max- imal [1](#_bookmark3) runs in Outcome((*l, v*)*,f* ) are winning. We denote WinStrat((*l, v*)*, J*) the set of winning strategies from (*l, v*) in *J*. Let *f* be a winning strategy from configuration (*l, v*). The *cost* of *f* from (*l, v*) is defined by:

Cost((*l, v*)*,f* )= sup{Cost(*ρ*) | *ρ* ∈ Outcome((*l, v*)*,f* )}

* 1. *Optimal Control Problems*

Let (*l*0*,* 0) denote the initial configuration of a RPTGA *J*. The three main problems we address in this paper are:

1 Roughly speaking a run is *maximal* if it can not be extended in the future by a controllable action (see [[6](#_bookmark28)] page 6, section 2.2); this point is discussed in the sequel in section [3.2](#_bookmark10).

*Optimal Cost Computation Problem:* we want to compute the optimal cost one can expect in a RPTGA *J* from (*l*0*,* 0), *i.e.* to compute

OptCost((*l*0*,* 0)*, J*)= inf{Cost((*l*0*,* 0)*,f* ) | *f* ∈ WinStrat((*l*0*,* 0)*, J*)}

*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* ∈ WinStrat((*l*0*,* 0)*, J*) such that:

Cost((*l*0*,* 0)*,f* )= OptCost((*l*0*,* 0)*, J*)

*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 strate- gies. In this case there is a family of strategies *fε* such that

|Cost((*l*0*,* 0)*, fε*) − OptCost((*l*0*,* 0)*, J*)| *< ε*

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

Example 2.5 [No optimal strategy] For the PTGA of Fig. [2](#_bookmark4) there is no op- timal strategy. *c* is a controllable action. Nevertheless we can define a family of strategies *fε* with 0 *< ε* ≤ 1 by: *f* (*l*0*,x <* 1 − *ε*)= *λ*, *f* (*l*0*,x* =1 − *ε*)= *c* and *f* (*l*1*,x* ≤ 1) = *c*. The cost of such a strategy is 1+ *ε*. So we can get as close as we want to 1 but there is no optimal winning strategy.

cost(*l*0)= 1 cost(*l*1)= 2

*l*0

*x <* 1; *c*

*l*1

*x* = 1; *c*

Goal

[*x <* 1] [*x* ≤ 1]

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

Example 2.6 We consider again Fig. [1](#_bookmark1). 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 5*t* + 10(2 − *t*)+1 is the minimal cost through *l*2 and

5*t* + (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* ≤ 2

such that max(21 − 5*t,* 9+ 4*t*) is minimum, which is *t* = 4 giving a minimal

3

cost of 141. In Fig. [3](#_bookmark6) 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](#_bookmark13).

3

*y y*

*Æ*0 → *Æ*1

4

3



*Æi* → Goal

Delay

2 2

4

3

0 2 *x* 0 2 *x*

*l*0 *li*, *i* ∈ {2*,* 3}

Fig. 3. *Optimal strategy for the RPTGA of Fig.* [*1*](#_bookmark1)*. Optimal cost is* 14 1 *.*

3

# Optimal Cost Computation

In this section we show that computing the optimal cost for a RPTGA amounts to solving a simple [2](#_bookmark8) control problem on a linear hybrid game automaton [[19](#_bookmark43)]. As a consequence well-known algorithms [[19](#_bookmark43),[8](#_bookmark31)] 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.

* 1. *From Priced Timed Games to Linear Hybrid Games*

Assume we want to compute the optimal cost of the RPTGA A given in Fig. [1](#_bookmark1). We translate this automaton into a linear hybrid game automaton (LHGA for short) H (see Fig. [4](#_bookmark9)) where the cost function is encoded into a variable Cost of the LHGA. In H the variable Cost decreases with rate *k* in a location *l* (*i.e.*

*d*Cost = −*k* in *l*) if Cost(*l*) = *k* in A. As for discrete transitions the variable

*dt*

Cost is updated by Cost'

in A is *k*.

= Cost − *k* in H if the corresponding transition’s cost

Let CompWin be a semi-algorithm (e.g. [[19](#_bookmark43),[8](#_bookmark31)]) that computes the largest set of winning states for a reachability hybrid game. Using CompWin we can compute the largest set of winning states for H with the goal states given by

2 Without cost.

*x* ≥ 2; *c*2



*u*

*l*

2

Cost′ = Cost − 1

*l*0

*x* ≤ 2; *c*1 ; *y* := 0

*l*1

*d*Cost = −10

*dt*

*u*

Goal

*d*Cost = −5

*dt*

[*y* = 0]

*l*3

*x* ≥ 2; *c*2

*d*Cost = −1 Cost′ = Cost − 7

Fig. 4. The Linear Hybrid Game Automaton *H*.

*dt*

Goal∧Cost ≥ 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 CompWin(H*,* Goal ∧ Cost ≥ 0). The meaning of the set *W* = CompWin(H*,* Goal ∧ Cost ≥ 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](#_bookmark28)], Theorem 5 and Lemma 6) that the (largest) set of winning states *W* is a union of zones of the form (*l, R* ∧ Cost > *h*) where *l* is a location, *R* ⊆ R*X* , *h* is a piece-wise affine function on *R* and >∈ {*>,* ≥}. 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 Cost > *k* with *k* ∈ Q≥0 and >∈ {*>,* ≥}. 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 > is equal to *>* there is no optimal strategy and if > is ≥ there is one.

≥0

Thus computing CompWin(H*,* Goal ∧ Cost ≥ 0) is a semi-algorithm for computing the optimal cost of the RPTGA A. Moreover we can decide (if CompWin terminates) whether there exists an optimal strategy or not: in case the initial winning cost set is of the form Cost *> k* there is no optimal strategy but a family of strategies *fε* (for *ε >* 0) with cost lower than *k* + *ε*. When this set if of the form Cost ≥ *k* (and assuming CompWin terminates) we can compute an optimal strategy (this point is dealt with in the next section [4](#_bookmark15)). The formal definitions and proofs of this reduction are given in [[6](#_bookmark28)] (Defi-

nition 12, Lemma 5, Theorems 4 and 5, Corollaries 1 and 2).

* 1. *The π Operator*

The computation of the winning states (with CompWin) is based on the defini- tion of a *controllable predecessor* operator [[16](#_bookmark40),[8](#_bookmark31)]. Let *J* = (*L, l*0*,* Act*, X, E,* inv*,*

cost) be a RPTGA and *Q* its set of configurations. For a set *X* ⊆ *Q* and

*a* ∈ Act we define Pred*a*(*X*) = {*q* ∈ *Q* | *q* −−*a*→ *q*'*, q*' ∈ *X*}. The control-

lable and uncontrollable discrete predecessors of *X* are defined by cPred(*X*)=

*c*∈Act*c*

Pred*c*(*X*), respectively uPred(*X*)=

*u*∈Act*u*

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 Pred*t*(*X, Y* ) if from *q* we can reach *q*' ∈ *X* by time elapsing and along the path from *q* to *q*' we avoid *Y* . Formally this is defined by:

*t* ≥0 [0*,δ*]

Pred (*X, Y* )= {*q* ∈ *Q* | ∃*δ* ∈ R s.t. *q* −*δ*→ *q*'*, q*' ∈ *X* ∧ Post (*q*) ⊆ *Y* } (1)

where Post

[0*,δ*]

(*q*) = {*q*' ∈ *Q* | ∃*t* ∈ [0*, δ*] | *q* −→*t*

*q*'}. We are then able to define

a *controllable predecessor* operator *π* as follows:

*π*(*X*)= Pred*t* *X* ∪ cPred(*X*)*,* uPred(*X*) (2)

This definition of *π* captures the choice that uncontrollable actions cannot be used to win (this choice is made in [[13](#_bookmark37)] and in [[6](#_bookmark28)]). As a matter of fact there is no way to win in the RPTGA of Fig. [1](#_bookmark1) 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 *J* where time cannot elapse is denoted *STOP* . Then a new definition of *π* where uncontrollable actions can be used to win is given by:

*π*'(*X*)= Pred*t* *X* ∪ cPred(*X*) ∪ (uPred(*X*) ∩ *STOP* )*,* uPred(*X*) (3)

Note that this choice does not change the results presented in [[6](#_bookmark28)]. In the example of Fig. [1](#_bookmark1), 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 (*l*1*,x* ≥ 0 ∧ *y* = 0) to the set of winning states.

The semi-algorithm CompWin computes the least fixed point *W* of the functional *λX.*{*X*0}∪ *π*'(*X*) as the limit of an increasing sequence of sets of states *Wi* (starting from set *W*0 = *X*0) where *Wi*+1 = *π*'(*Wi*)). If *J* is a RPTGA, the result of the computation of CompWin on the associated LHGA *H* starting from Goal ∧ Cost ≥ 0 is *W* = *µX.*{Goal ∧ Cost ≥ 0}∪ *π*'(*X*). This

result is also denoted CompWin(*H,* Goal ∧ Cost ≥ 0) and gives the largest set of winning states.

* 1. *Termination Issues*

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

Let *J* be a RPTGA satisfying:

* + - *J* is bounded, *i.e.* all clocks in *J* are bounded [3](#_bookmark14) ;
    - the cost function of *J* is *strictly non-zeno*, *i.e.* there exists some *κ >* 0 such that the accumulated cost of every cycle in the region automaton associated with *J* is at least *κ*. Note that this condition can be checked. For more complete explanations, see [[6](#_bookmark28)].

Then the semi-algorithm CompWin(*H,* Goal ∧ Cost ≥ 0) terminates (*H* is the hybrid game defined from *J* in the previous section). The formal statement and proof of this claim is given by Theorem 6 in [[6](#_bookmark28)]. We thus get:

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

* 1. *Implementation of* CompWin *in* HyTech

HyTech [[10](#_bookmark34),[11](#_bookmark35)] 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 H of Fig. [4](#_bookmark9) is given in Fig. [5](#_bookmark17), lines [7](#_bookmark17)–[20](#_bookmark17). 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 H we add two boolean variables *u* and *c* (Fig. [5](#_bookmark17), line [4](#_bookmark17)) that are negated on each discrete uncontrol- lable (resp. controllable) transitions (Fig. [5](#_bookmark17), lines [10](#_bookmark17)–[19](#_bookmark17)). 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*

3 This hypothesis is not a restriction, see [[15](#_bookmark39)].

and over a variable *t* that has rate [4](#_bookmark16) −1. We can express the cPred (and uPred) operator with existential quantifiers and two variables *t* and *c* as follows:

cPred(*X*)= {*q* | ∃*c* ∈ Act s.t. *q* →−*c*

*c*

*q*'*, q*' ∈ *X*}

= {*q* | ∃*t* s.t. ∃*c* s.t. *t* =0 ∧ *c* =0 ∧

(*q, t, c*) ∈ pre(*X* ∧ *t* =0 ∧ *c* = 1)}

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](#_bookmark17), line [39](#_bookmark17)). For uncontrollable predecessors we replace *c* by *u* (Fig. [5](#_bookmark17), line [41](#_bookmark17)). Note that the computation of *STOP* states (Fig. [5](#_bookmark17), line [32](#_bookmark17)) can also be done using our extra variables *t, c, u*.

Safe Time Predecessors. The other operator Pred*t*(*Z, Y* ) is a bit more compli- cated. 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 determin- istic automata as in this case 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](#_bookmark43)] for a detailed explanation.) From equation ([1](#_bookmark11)) we get:

Pred (*Z, Y* )= {*q* | ∃*t* ≥ 0 s.t. *q* →−*t*

*t*

*q*'*, q*' ∈ *Z and*

∀0 ≤ *t* ≤ *t, q* →−*t*

1

*q*'' =⇒ *q*'' /∈ *Y* }

= {*q* | ∃*t* ≥ 0 s.t. *q* →−*t*

*q*'*, q*' ∈ *Z and*

¬(∃0 ≤ *t* ≤ *t, q* →−*t*

1

*q*'' ∧ *q*'' ∈ *Y* )}

The latter formula can be encoded in HyTech using the hide operator (Fig. [5](#_bookmark17), lines [47](#_bookmark17)–[55](#_bookmark17)) and two auxiliary variables *t* and *t*1 that evolves at rate −1 (note that those variables are not part of the model but only used in existentially quantified formulas and they do not constrain the behavior of H.) Finally if the computation of *π*' terminates (Fig. [5](#_bookmark17), lines [35](#_bookmark17)–[59](#_bookmark17)) 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](#_bookmark17), line [61](#_bookmark17)).

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

# 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 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*

5: t,t1: analog; −− *used for existential quantiﬁcation*

automaton H

synclabs: ;

initially l0 & x=0 & y=0;

10: 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;

15: 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}

20: 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 ;

25:

−− *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*

30: 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*'

35: X := iterate X from winning using {

−− *uncontrollable predecessors of X:* uPred(*X*)

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 ;

40: −− *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)) ;

45: −− *time predecessors of Z from which we can reach Z*

−− *and avoid* uPred(*X*) *all along; X* := Pred*t*(*Z,* uPred(*X*))

X := hide t in

(hide c,u in t>=0 & c=0 & u=0 & pre(Z & t=0 & c=0 & u=0)

endhide) &

50: ~(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)

55: endhide;

−− *add the newly computed regions to the set of already*

−− *computed region*

fix := fix | X ;

} ;

60: −− *print the result*

print omit all locations hide x,y in fix & init\_reg endhide;

Fig. 5. Computation of the Optimal Cost.

* 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 ∀*ρ, ρ*' ∈ Runs(*J*)*, last*(*ρ*)= *last*(*ρ*') implies that *f* (*ρ*)= *f* (*ρ*'). State-based strategies are also called *memoryless* strategies in game theory [[18](#_bookmark42),[8](#_bookmark31)];
    - *polyhedral* if for all *a* ∈ Act*c* ∪ {*λ*}, *f* −1(*a*) is a finite union of convex poly- hedra for each location of the game;
    - *realizable*, whenever the following holds: for all *ρ* ∈ Outcome(*q, f* ) s.t. *f*

is defined on *ρ* and *f* (*ρ*) = *λ*, there exists some *δ >* 0 such that for all

0 ≤ *t < δ*, there exists *q*' with *ρ* −→*t*

*q*' ∈ Outcome(*q, f* ) and *f* (*ρ* −→*t*

*q*')= *λ*.

Strategies which are not realizable are not interesting because they generate empty sets of outcomes. Nevertheless it is not clear from [[16](#_bookmark40),[4](#_bookmark30)] how to extract strategies for RPTGA and ensure their realizability as shown by the following example.

Fig. 6. A timed game automaton

*x >* 1; *c*

*l*0 Goal

Example 4.1 Consider the PTGA of Fig. [6](#_bookmark18) 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* (*l*0*,x* ≤ 1) = *λ* and *f* (*l*0*,x >* 1) = *c*. This strategy is not realizable. In the sequel, we build a strategy which is *f* (*l*0*,x <* 2) = *λ* and *f* (*l*0*,x* ≥ 2) = *c*. Now assume the constraint on the transition is 1 *< x* ≤ 2. In this case we start with the following strategy (not realizable): *f* (*l*0*,x* ≤ 1) = *λ* and *f* (*l*0*,* 1 *< x* ≤ 2) = *c*. To make it realizable we will take the first half of 1 *< x* ≤ 2 and have a delay action on it i.e. *f* (*l*0*,x <* 3 ) = *λ* and

2

*f* (*l*0*,* 3 ≤ *x* ≤ 2) = *c*. In the following, we will restrict our attention to

2

realizable strategies and simply refer to them as strategies.

A secondary result is provided in [[6](#_bookmark28)] 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](#_bookmark28)]) *Let J be a RPTGA. If the semi-algorithm* CompWin *terminates for the hybrid game associated with J*

*(see section* [*3.1*](#_bookmark7)*), then we can compute a winning strategy which is: polyhedral, realizable and stated-based.*

Let *H* be the LHGA associated to the RPTGA *J* = (*L, l*0*,* Act*, X, E,* inv*,* cost).

A state of *H* is a triple (*l, v, c*) where *l* ∈ *L, v* ∈ R*X* and *c* ≥ 0 (*c* is the value of

≥0

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](#_bookmark7)) we can synthetize realizable state-based winning strategies for *J* (see [[6](#_bookmark28)], 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 *J* (this means we want to build a state- based strategy for the original RPTGA *J*.) To this extent we introduce the notion of *cost-independent* strategies.

Let *W* = CompWin(*H,* Goal ∧ Cost ≥ 0) be the set of winning states of

*H*. A state-based strategy *f* for *H* is *cost-independent* if (*l, v, c*) ∈ *W* and (*l, v, c*') ∈ *W* implies *f* (*l, v, c*)= *f* (*l, v, c*'). Cost-independent strategies in *H* will then be used for having state-based strategies in *J*. Theorem 7 of [[6](#_bookmark28)] gives then sufficient conditions for the existence of a state-based, optimal, realizable strategy in *J* and, when back to the automaton reads as follows:

Theorem 4.3 (Adapted from Theorem 7 of [[6](#_bookmark28)]) *Let J be a RPTGA. H is the associated LHGA. If* CompWin *terminates for H and the set of winning states is W* = CompWin(*H,* Goal ∧ Cost ≥ 0) *is a union of sets of the form*

(*l, R* ∧ Cost ≥ *h*) *where l is a location, R* ⊆ R*X and h is a piece-wise affine*

≥0

*function on R, then there exists a winning realizable state-based strategy f defined over WG* = ∃Cost*.W s.t. for each q* ∈ *WG, f* ∈ WinStrat(*q, WG*) *and* Cost(*q, f* )= OptCost(*q*)*.*

Note that under the previous conditions we build a strategy *f* which is *uniformly optimal i.e.* optimal for all states of *WG*. A syntactical criterion to enforce the condition of theorem [4.3](#_bookmark19) 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 *J*.

The example of Fig. [1](#_bookmark1) satisfies the assumptions of Theorem [4.3](#_bookmark19) 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](#_bookmark6).

* 1. *Synthesis of Optimal Strategies*

The set of winning states of *H* is computed iteratively using the functional *π*' defined by equation ([3](#_bookmark12)). 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 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.*

NonStop(*X*)= {*q* ∈ *X* | ∃*t >* 0 | *q* + *t* ∈ *X* ∧ ∀0 ≤ *t*' ≤ *t, q* + *t*' ∈ *X*} (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](#_bookmark28)]). The tags will contain information about how a new set of winning states *Wi*+1 = *π*'(*Wi*) has been obtained.

We start with *W* + = ∅ and *W*0 = Goal ∧ Cost ≥ 0. Assuming *Wi* and *W* +

0

are the sets obtained after *i* iterations of *π*' we define *W* +

*i*+1

*i*

as follows:

1. let *Y* = *Wi*+1 \ *Wi* where *Wi*+1 = *π*'(*Wi*);
2. for each *c* ∈ Act , we define the tagged set *Y* ∩ cPred*c*(*W* ) [*c*] with the

*c*

*i*

intended meaning: “ *Y* ∩ cPred*c*(*Wi*) has been added to the set of winning states by a Pred*c* and doing a *c* from this set leads to *Wi*”;

1. define another tagged set NonStop(*Y* ) [*λ*] with the intended meaning: “ 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 *Wi* or cPred(*Wi*) or uPred(*Wi*) ∩ *STOP* ”;

1. define *W* +

*i*+1

+

*W*

*i*+1

by :

= *W* + ∪ NonStop(*Y* ) [*λ*] ∪

*i*

*c*∈Act*c*

*Y* ∩ cPred*c*(*W* ) [*c*]

Computation of an Optimal Strategy. If CompWin terminates in *j* iterations we end up with *W* + = *W* +. Note that by construction a state *q* of *W* may

*i*

*j*

belong to several tagged sets *X*[*λ*], *X*[*c*1], ..., *X*[*cn*] (where *c* ∈ Act

for each

0 1 *n i c*

*i* ∈ [1*, n*]) of *W* +. If the assumptions of theorem [4.3](#_bookmark19) are satisfied all the *Xi*’s

are of the form *X*' ∧ Cost ≥ *hi* where *X*' ⊆ {*li*}× R*X* for some location *li*

*i i* ≥0

and *hi* : *X*' → R≥0 is a piecewise affine function. Thus the infimum of *hi* over

*i*

*X*' is reachable and equal to the minimum of *hi*.

*i*

Theorem 7 of [[6](#_bookmark28)] states that in this case an optimal state-based strategy

*f* ∗ for *q* a winning state of *J* (no cost) will be obtained by taking the local

optimal choice: let *m* = min*i*∈[0*,n*] *hi*(*q*); then defining *f* ∗(*q*)= *ci* if *hi*(*q*)= *m*

gives an optimal strategy.

As it can be the case that *hi*(*q*) = *hj*(*q*) = *m* with *i* /= *j*, we impose a total order и on the set of events in Act*c* and define *f* ∗(*q*) = *ci* where *i* = max{*j* | *hj*(*q*) = *m*}. To avoid realizability problems (see proof of Lemma 6 in [[6](#_bookmark28)]) on the boundary of a set *Xi* if *h*0(*q*)= *m* (which means that the optimal cost can be achieved by time elapsing) and *hi*(*q*) = *m* for some *i* ∈ [1*, n*] we define *f* ∗(*q*)= *ci*. This can be easily defined in our setting by extending и to Act*c* ∪ {*λ*} and making *λ* the smallest element.

After these algorithmics explanations, we can summarize how we can syn- thesize an optimal, cost-independent strategy. We denote *W* + the set defined

[*c*]

by:

+

*W* =

[*c*]

*Si* (5)

For each *c* ∈ Act*c* ∪ {*λ*}, *W* +

[*c*]

*S*[*c*]∈*W* +

is a set of the form *Xc* ∧ Cost ≥ *hc* where

*i*

*hc* is a piecewise affine function on *Xc* (*Xc* is a union of convex polyhedra). Note that the constraint Cost ≥ *hc* 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 *J*, 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 ∈ Act*c* ∪ {*λ*}. We denote [*c*1 ≤ *c*2] the set of winning states of *J* where it is better to do action *c*1 than action *c*2 (*hc*1 (*q*) ≤ *hc*2 (*q*)). This set is defined by:

[*c*1 ≤ *c*2]= {*q* ∈ *Xc*1 | ∃*α*1 | (*q, α*1) ∈ *W* + *and*

[*c*1]

(6)

∀*α*2 | (*q, α*2) ∈ *W* + *, α*1 ≤ *α*2}

[*c*2]

= *q* ∈ *Xc*1 | ∃*α*1 | (*q, α*1) ∈ *W* + ∧

[*c*1]

¬ ∃*α* | (*q, α* ) ∈ *W* + *and α*

2

2

[*c*2]

2

1

*< α* } (7)

Each set [*c*1 ≤ *c*2] is a polyhedral set. For each *c* ∈ Act*c* ∪ {*λ*} define

Opt(*c*)= [*c* ≤ *c*'] (8)

*c*'/=*c*

Opt(*c*) is the set of states for which *c* is an action that gives the optimal cost. *W* ∗ = *c*∈Act ∪{*λ*} Opt(*c*) is thus equal to the set of states on which we need to define the optimal strategy. Given the total order и on Act*c* ∪ {*λ*} with *λ* и *c*1 и ··· и *cn*, we can define an optimal strategy *f* ∗ as follows: for

*c*

*i* ∈ [0*,n* − 1], let *Bi* = *W* ∗ \ (∪*k>iBk*) ∩ Opt(*ci*) and *Bn* = *W* ∗ ∩ Opt(*cn*); define then *f* ∗(*q*)= *ci* if *q* ∈ *Bi*. *f* ∗ is an optimal strategy that is (winning), state-based, realizable and polyhedral.

* 1. *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](#_bookmark17) and use it in the guards of controllable transitions: controllable action *ck* of Fig. [4](#_bookmark9) corresponds to the guard *a* = *k* in the HyTech model. The HyTech model of Fig. [5](#_bookmark17) is enriched as follows: we add the guard *a* =1 to line [11](#_bookmark17), *a* = 2 to lines [16](#_bookmark17) and [18](#_bookmark17). In this way we achieve the tagging of controllable predecessors as now the computation of cPred (line [39](#_bookmark17) of Fig. [7](#_bookmark21)) will compute a tagged region that will be a union of polyhedra with some *a* = *k* constraints. Note that we also modify line [44](#_bookmark17) of Fig. [5](#_bookmark17) and replace it by line [20](#_bookmark21) in Fig. [7](#_bookmark21) 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 NonStop(*Y* ) we use again our extra variables *t, c, u* and add the tag *a* =0 to the result set. Lines [35](#_bookmark21)–[36](#_bookmark21) of Fig. [7](#_bookmark21) 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](#_bookmark21) in Fig. [7](#_bookmark21).

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 *hci* by the means of a constraint between the Cost variable and the rest of the state variables. To compute *hci* we need to split the state space according to each action *ci*: this is achieved by lines [44](#_bookmark21)–[46](#_bookmark21) where the state space that corresponds to *hci* is stored in ri.

It remains to compute for each pair of actions (*c*1*, c*2) (*ci* can be *λ*), the set [*c*1 ≤ *c*2] states. The encoding in HyTech of the formula given by equation ([7](#_bookmark20)) 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](#_bookmark21)–[63](#_bookmark21).

# 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

var

−− *same set of variables here as lines* [*22*](#_bookmark17)*–*[*24*](#_bookmark17) *in Fig.* [*5*](#_bookmark17) *plus some new vars:*

a: discrete; cost0,cost1,cost2: analog;

5: 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;

10: 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

15:

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;

20: 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

25: (hide c,u in t1>=0 & t1<=t & c=0 & u=0 &

pre(uPrebarX & t1=0 & c=0 & u=0)

endhide) endhide)

endhide;

30:

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

fix := fix | X ;

−− *computation of* NonStop(*Y* )

35: 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 ;

} ;

40: −− *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 hci*

r0 := hide a,cost in cost0=cost & fix\_strat & a=0 endhide ; 45: 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 hci* (*q*) ≤ *hcj* (*q*)

inf\_0\_1 := hide cost0 in r0 & ~(hide cost1 in r1 & cost1<cost0 endhide) endhide ;

50: ...

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" ;

55: 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 ;

60: 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.

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 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* = (low*x,* high*y,* Y) is the sum of the rates of Antenna 1 and Antenna 2 (both are working) *i.e.* 1+ 20 = 21 and Cost(*s*)= 21 in our model. Once the connection with the base station is established (either *x* ≥ 10 or *y* ≥ 10) the message is delivered with an energy consumption depending on the antenna (Cost = 7 for Antenna 1 and 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).

jam*x*?; *x* := 0 jam*y* ?; *y* := 0

low*x*

cost =1

*x* ≥ 5; *y* := 0

high*x*

cost = 10

*x* ≥ 10;

cost =7

jam*x*?;

*x* := 0

Goal*x* Antenna 1

low*y*

cost =2

*y* ≥ 2; *x* := 0

high*y*

cost = 20

*y* ≥ 10;

cost =1

jam*y* ?;

*y* := 0

Goal*y* Antenna 2

*x >* 6;

jam*x*!

*x >* 6; jam*y* !

*y >* 6; jam*x*!

*X*

*Y*

Jammer

*y >* 6;

jam*y* !

Fig. 8. Mobile Phone Example.

This system can be represented by a network of PTGAs (see Fig. [8](#_bookmark22) 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 synchroniza- tion: 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 control- lable 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](#_bookmark22), jam*x*? (resp. jam*y*?) synchronizes with jam*x*! (resp. jam*y*!). The HyTech code of this example can be found in [[6](#_bookmark28)] and on the web page <http://www.lsv.ens-cachan.fr/aci-cortos/ptga/>.

Results of our Experiments. We got that the optimal cost (lowest energy con- sumption) that can be ensured is 109. The optimal strategy is graphically represented on Fig. [9](#_bookmark24). 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.

# 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](#_bookmark28)] where we proved this problem was decidable (under some hypotheses we recall in this paper). Morever, we also provide an implemen- tation of our algorithm in HyTech and demonstrate it on small case-studies. In a recent paper [[2](#_bookmark27)] 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).

*y y*

10 10

0 10 *x*

low*x*.low*y*.X

*y y*

14

3

0 10 *x*

low*x*.low*y*.Y

10 10

2

 Delay

 high*x* →low*x * high*y* →low*y * low*x* →Goal*x*

0 10 *x*

high*x*.low*y*.X

0 10 *x*

low*x*.high*y*.Y

low*y*

→Goal*y*

*y*



(10*,* 20)

(10*,* 12)

*y y*



*y* =2

10 10 10

6

5

2

0 4 *x*

high*x*.high*y*.X

0 5 10 *x*

high*x*.low*y*.Y

0 5 10 *x*

high*x*.high*y*.Y

# References

Fig. 9. Optimal Strategy for the Mobile Phone Example

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/.](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.

1. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.