Optimal Reachability for Weighted Timed Games ⋆

Rajeev Alur Mikhail Bernadsky P. Madhusudan

University of Pennsylvania

Abstract

Weighted timed automata are timed automata annotated with costs on locations and transitions. The optimal game-reachability problem for these automata is to find the best-cost strategy of supplying inputs so as to ensure reachability of a target set within a specified number of iterations. The only known complexity bound for this problem is a doubly-exponential upper bound. We establish a singly-exponential upper bound and show that there exist automata with exponentially many states in a single region with pair-wise distinct optimal strategies.

Key words: Timed automata, timed games, optimal costs, weighted games.

1 Introduction

Timed automata [AD94] extend finite-state automata with real-valued clock variables, and have proved to be useful in modeling real-time systems. The canonical problem for timed automata is reachability, and can be solved in polynomial-space using a finite-state quotient—the so-called *region graph*—of the underlying infinite state-space. A natural generalization of reachability corresponds to *optimal reachability* that asks how soon a target state (or a set of states) can be reached from an initial state. This problem, and its variations, are theoretically interesting as decidability and finiteness of representation are not immediate from the region graph construction, and have been studied by

⋆ This research was supported in part by ARO URI award DAAD19-01-1-0473, and NSF awards CCR-9970925, ITR/SY 0121431, and CCR-030638.
A preliminary version appeared in the Proceedings of ICALP, 2004.
Email addresses: alur@cis.upenn.edu (Rajeev Alur), mberbads@cis.upenn.edu (Mikhail Bernadsky), madhusudan@cis.upenn.edu (P. Madhusudan).

many researchers [CY91,ACH97,ALP01,LBB+01]. In particular, in a *weighted timed automaton* (also known as a *priced timed automaton*), each discrete transition has an associated nonnegative integer denoting the cost to be paid when the transition is taken, and each location has an associated nonnegative integer denoting the cost rate with respect to the time spent at that location. The minimum-cost reachability problem for weighted timed automata can be solved in exponential-time [ALP01]. An alternative branch-and-bound solution is implemented in the verification tool UPPAAL with applications to scheduling problems [LBB+01,BHF+01].

In this paper, we consider games on weighted timed automata. Games are useful for synthesizing controllers, and for generating schedules in the context of real-time systems. In one round of our game, the controller chooses an input symbol a, and a time $t \geq 0$ at which it wants to supply the input. The adversary updates the state of the automaton either by executing an uncontrolled discrete transition at time $t' \leq t$, or by executing an a-labeled discrete transition at time t. Given a set of target locations, an initial state s, a cost bound C, and a bound k on the number of rounds, the (bounded) *optimal game-reachability problem* is to determine if the controller has a strategy to enforce the game started in state s into a target location within k rounds while ensuring that the cost of the run is bounded by C. In absence of costs and optimality, there is a long history of research on games for timed automata, and such games are known to be decidable [WH91,MPS95,DM02,CHR02]. Time-optimal games, that is, games in which the cost of a run equals the total time spent, are considered in [AM99], and are shown to be decidable (however, no complexity bounds, upper or lower, are reported and the solution technique does not generalize to weighted timed games). The general case for (bounded games) on weighted timed automata is considered in [TMM02], and the authors show that the problem can be encoded using the first-order theory of reals with addition [FR75], leading to a solution that takes doubly-exponential time. This solution only relies on linearity and quantifier elimination, and does not exploit any special structure of a timed automaton; also, it is known that first-order theory over reals with addition cannot be decided even in nondeterministic exponential time (i.e. it is not in NE) [FR74].

In this paper, we provide an exponential-time solution to the (bounded) optimal game-reachability problem. We show how to compute a sequence of functions opt_i such that for each i, for each state s, $opt_i(s)$ is the optimal cost of reaching a target location starting from s in i steps in the timed game, and the representation of opt_i is exponential in i and in the size of the automaton.

It is easy to show that each region of the timed game can be split into finitely many subregions (or cells) such that the optimal cost function opt_i is linear within each cell. The main technical challenge in this paper is getting a tight bound on the required splitting into cells. While computing the function opt_i

from opt_{i-1}, one source of complexity is the discrete *min-max* nature of the game: the controller picks the input symbol a, and the adversary picks an a-labeled edge. If f_1 and f_2 are functions with n pieces, then the *min* or *max* operation could result in a function that has $O(n^d)$ splits (where d is the number of clocks). However, this analysis only gives a doubly exponential bound on the growth of the number of cells. We show that the partitioning of a region into cells can be organized as a tree, where each node has an associated cell, a set of hyperplanes, and a child for every subcell formed by these hyperplanes. In this representation, *min-max* operation adds just one level to the tree, and the number of hyperplanes at a node of the tree for opt_i grows linearly. The second source of complexity is the continuous *inf-sup* nature of the game: the controller picks a time t and the adversary picks a time $t' \leq t$. In a timed automaton, all clocks increase uniformly, and hence, the state evolves along diagonal hyperplanes. Hence, intuitively, if we restrict attention to a diagonal tube where the same set of cells are relevant, the interesting choices of t and t' are at the cell boundaries. Our final analysis combines the tube-based and tree-based representations, and shows that each opt_i can be represented using at most exponentially many cells.

We also show that the bound on splitting a region into cells is tight: for every n, there exists a weighted timed automaton A_n, a region R of A_n, and exponentially many states within R, such that the discrete components of the optimal cost strategies are all different for games starting at these states, and thus, the region R must be split into exponentially many cells.

The paper is organized as follows. Section 2 defines weighted timed automata and optimal reachability games and illustrates the problem with an example. Section 3 defines regions and expressions that inductively compute the optimal costs for winning the game in each region in i steps; we also intuitively describe the various problems that arise in bounding the complexity and motivate the definition of nested partition functions. In Section 4, we prove the arguments that show the optimal game problem can be solved in exponential time. Section 5 constitutes the arguments that show the exponential lower bound on the number of cells a region can get split into. We conclude with a discussion of the results in Section 6.

2 Model

2.1 Weighted Timed Automata

Let X be a finite set of clocks. Let $\mathbb{R}_+$ denote the set of all non-negative reals and let $\mathbb{N}$ denote the set of all non-negative integers. A clock valuation is a map $\nu : X \to \mathbb{R}_+$. The set of *constraints* over X, denoted $G(X)$, is the set of boolean combinations of constraints of the form $x \sim \beta$ or $x - y \sim \beta$ where $x, y \in X$, $\beta \in \mathbb{N}$, and $\sim \in \{<, >, =, \leq, \geq\}$. The notion of when a clock valuation ν over X satisfies a constraint over X is the natural one; for example, if $\nu(x) = 3.3$, then ν satisfies the constraint $x > 3$. Let $\bar{0}$ denote the valuation that maps each clock in X to 0.

Definition 1 *A weighted timed automaton (WTA) is a tuple* $A = (Q, Q_F, X, \Sigma, u, \delta, Inv, W_Q, W_\delta)$ *where Q is a finite set of locations, $Q_F \subseteq Q$ is a set of target locations, X is a finite set of clocks, Σ is a finite set of actions that contains the special symbol u, $\delta \subseteq Q \times \Sigma \times G(X) \times 2^X \times Q$ is the transition relation, $Inv : Q \to G(X)$ is an* invariant function, *$W_Q : Q \to \mathbb{N}$ gives the cost for each location and $W_\delta : \delta \to \mathbb{N}$ gives the cost for each transition.* $\square$

For a transition $e = (q, a, g, Z, q') \in \delta$, the label of e is a, and is denoted by $Action(e)$. Transitions labeled u model uncontrolled transitions.

A *state* of A is a pair $s = (q, \nu)$ where $q \in Q$ and ν is a clock valuation over the set of clocks X. Let *States* denote the set of all states. For a clock valuation ν and $t \in \mathbb{R}_+$, let $\nu + t$ denote the clock valuation ν' where $\nu'(x) = \nu(x) + t$, for each $x \in X$. Also, for any clock valuation ν and a set of clocks $Z \subseteq X$, let $\nu/reset(Z)$ denote the valuation ν' where $\nu'(z) = 0$ for each $z \in Z$ and for each $x \notin Z$, $\nu'(x) = \nu(x)$.

We now define timed transitions and discrete transitions between states. A timed transition is of the form $(q, \nu) \xrightarrow{t} (q, \nu{+}t)$, where $(q, \nu), (q, \nu{+}t) \in States$ and $t \in \mathbb{R}_+$, such that for every $0 \leq t' \leq t$, $\nu + t'$ satisfies $Inv(q)$, the invariant at q.

A discrete transition is of the form $(q, \nu) \xrightarrow{e} (q', \nu')$, where e is a transition of the form $(q, a, g, Z, q') \in \delta$ such that ν satisfies g, $\nu' = \nu/reset(Z)$ and ν' satisfies $Inv(q')$. We say e is enabled at a state (q, ν) if there is a transition of the form $(q, \nu) \xrightarrow{e} (q', \nu')$, for some state (q', ν'). We say an action $a \in \Sigma$ is enabled at (q, ν) if some a-labeled transition is enabled at (q, ν).

A run of length k of a WTA A from a state s_1 is a sequence of $2k$ alternating timed and discrete transitions $\rho = s_1 \xrightarrow{t_1} s'_1 \xrightarrow{e_1} s_2 \xrightarrow{t_2} s'_2 \ldots s'_k \xrightarrow{e_k} s_{k+1}$. For such a run, we define the cost of ρ, denoted $W(\rho)$ to be the cost incurred along this run, i.e. if $s_i = (q_i, \nu_i)$, for each i, then $W(\rho) = (\sum_{i=1}^{k} W_Q(q_i) \cdot t_i) + (\sum_{i=1}^{k} W_\delta(e_i))$.

Let $\Sigma' = \Sigma \setminus \{u\}$. The timed game on WTA A is played by two players—the *controller* and the *adversary*. At any state, the controller first picks a time t and an action $a \in \Sigma'$ to signal that it would like to make an a-labeled transition after time t and not any transition before that time. This choice must be valid in the sense that it must be possible to wait for time t without violating the invariant and after time t, some a-labeled transition must be available. The adversary now has two choices: it can wait for some time $0 \leq t' \leq t$ and execute a transition labeled u or it can decide to wait for time t and choose to take some a-labeled transition. The game then evolves to a new state and the players proceed to play as before.

Formally, *a (controller) strategy* is a function $str : States \to \mathbb{R}_+ \times \Sigma'$. A run $\rho = s_1 \xrightarrow{t'_1} s'_1 \xrightarrow{e_1} s_2 \ldots \xrightarrow{t'_k} s'_k \xrightarrow{e_k} s_{k+1}$ of A is said to be a play according to a controller strategy str if for every i, if $str(s_i) = (t_i, a_i)$, then either $t'_i = t_i$ and $Action(e_i) = a$, or, $t'_i \leq t_i$ and $Action(e_i) = u$.

Let $\rho = s_1 \xrightarrow{t'_1} s'_1 \xrightarrow{e_1} s_2 \ldots \xrightarrow{t'_k} s'_k \xrightarrow{e_k} s_{k+1}$ be a run of length k. We say that ρ *wins within i steps* if there is some $i' \leq i$ such that $s_{i'} = (q, \nu)$ where $q \in Q_F$.

A controller strategy str is said to be winning from a state s_1 in k-steps and within cost $Cost$ if for every play $\rho = s_1 \xrightarrow{t'_1} s'_1 \xrightarrow{e_1} s_2 \ldots \xrightarrow{t'_k} s'_k \xrightarrow{e_k} s_{k+1}$ of length k according to str, there is an $i \leq k$ such that:

- ρ wins within i steps and the cost of the prefix run $\rho_i = s_1 \xrightarrow{t'_1} s'_1 \xrightarrow{e_1} s_2 \ldots \xrightarrow{t'_i} s'_i \xrightarrow{e_i} s_{i+1}$ is less than or equal to $Cost$, i.e. $W(\rho_i) \leq Cost$.
- For every $j \leq i$, if $s_i = (q_i, \nu_i)$ and $str(s_i) = (t, a)$, then $(q_i, \nu_i) \xrightarrow{t} (q_i, \nu_i + t)$ is a timed transition and a is enabled at $(q_i, \nu_i + t)$.

The first condition above formalizes the requirement that the controller must force the play to Q_F within k steps and while doing so incur cost less than $Cost$, and the second formalizes the condition that while playing the game, the controller must pick valid times and actions.

We can now state the main problem we consider:

Optimal bounded weighted timed game problem:
Given a weighted timed automaton A, an initial state q_{in} and a number k, find the optimal cost $Cost$ such that there is a controller strategy that wins from the state $(q_{in}, \bar{0})$ in k steps and within cost $Cost$.

The solution we give in fact solves a more general problem:

Uniform optimal bounded weighted timed game problem:
Given a weighted timed automaton A and a number k, find a function $f_k : States \to \mathbb{R}_+ \cup \{\infty\}$ such that $f_k(s)$ is the least cost such that there is a controller strategy that wins from s in k steps and within cost $f_k(s)$ ($f_k(s)$ is ∞ if there is no strategy that wins in k steps).

The more general timed game problem requires that the controller forces the game to reach the target within a finite number of discrete switches, but places

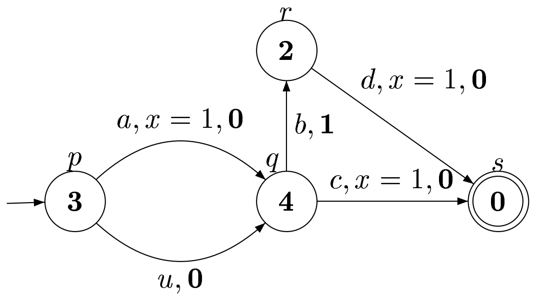


Fig. 1. The automaton A

no restriction on the number of these switches:

Optimal unbounded weighted timed game problem:
Given a weighted timed automaton A, find a function $f : States \to \mathbb{R}_+ \cup \{\infty\}$ such that $f(s)$ is the least cost such that there is a controller strategy that wins from s in k steps and within cost $f(s)$, for some $k \in \mathbb{N}$.

2.2 Example

Consider the WTA A in Figure 1. The weights of the nodes and the edges are given in the bold font. The symbols a, b, c, d denote the actions available to the controller.

The game starts at the initial node p, and the goal for the controller is to reach the final node s while paying the minimum cost. To determine the minimum cost and the controller strategy we will start at the final node s and work our way back to p. At s the cost is 0. Since the controller can reach the destination node only when $x = 1$, the only possible strategy at the node r is to wait until x becomes 1 (assuming the value of x is less than 1). Thus the minimum cost to reach s from r is $2(1 - \hat{x})$, where $\hat{x}$ is the valuation of x upon entering r.

At the node q the controller has to decide in which node q or r it will wait until x becomes 1. The node r is cheaper and thus more preferable for waiting than q, but the controller incurs a cost of unit 1 for the transition from q to r. Thus the minimum cost, assuming $\hat{x}$ is the value of the clock when entering q, is given by the expression:

$$Cost_q(x_0^q) = \min\{1 + 2(1 - \hat{x}), 4(1 - \hat{x})\} = \min\{3 - 2\hat{x}, 4(1 - \hat{x})\}$$
$$= \begin{cases} 3 - 2\hat{x}, \ \ \textit{if} \ \ 0 \le \hat{x} \le \frac{1}{2} \\ 4(1 - \hat{x}), \ \ \textit{if} \ \ \frac{1}{2} \le \hat{x} \le 1 \end{cases}$$

When $0 \leq x_0^q < 1/2$ the controller is better off by immediately moving to r, but when $1/2 < x_0^q \leq 1$ the controller must wait in q. For $x_0^q = 1/2$ the controller can choose either strategy. Note that the expressions that give the minimum cost can be different even if the clock valuations belong to the same region, which is in contrast to the setting of unweighted games.

Now consider the state p. The transition marked by the action symbol a prevents the adversary from stopping a game in the node p. What is the optimal time τ for the adversary to move from p to q, given the initial clock valuation of x is $\hat{x}$? The expression to determine τ is

$$\begin{aligned} Cost_p(\hat{x}) &= \sup_{\tau \geq 0, \tau + \hat{x} \leq 1} (3\tau + Cost_q(\hat{x} + \tau)) \\ &= \sup_{\tau \geq 0, \tau + \hat{x} \leq 1} (\min(3 + \tau - 2\hat{x}, 4 - \tau - 4\hat{x})) \end{aligned}$$

Analysis of the expression above gives the following expression for τ:

$$\tau = \begin{cases} \frac{1}{2} - \hat{x} & \mathit{if}\ 0 \leq \hat{x} \leq \frac{1}{2} \\ 0 & \mathit{if}\ \frac{1}{2} < \hat{x} \leq 1 \end{cases}$$

Substituting this into the expression of the cost, we obtain:

$$Cost_p(\hat{x}) = \begin{cases} \frac{7}{2} - 3\hat{x} & 0 \leq \hat{x} \leq \frac{1}{2} \\ 4(1 - \hat{x}) & \frac{1}{2} < \hat{x} \leq 1 \end{cases}$$

This example shows the optimal controller strategy may have to take different discrete choices even within different points in a region, and that it may have to wait to make make its move in the middle of region (for example, if $\hat{x} = 0$, then $\tau = 1/2$).

3 Optimal Cost Functions

3.1 Regions

Let us fix a WTA $A = (Q, Q_F, X, \Sigma, u, \delta, \mathit{Inv}, W_Q, W_\delta)$ for the rest of this subsection and let β_{max} be the largest constant mentioned in the constraints in A. Fix an order on the clocks, say $\langle x_1, \ldots, x_d \rangle$. Then clock valuations naturally correspond to points in $\mathbb{R}_+^d$; we use this correspondence freely.

A *clock region* of A is a nonempty subset R of $\mathbb{R}_+^d$ that satisfies a set of constraints $\tilde{R}$ where:

(1) for each clock x, there is a $\beta \in \mathbb{N}$, $0 \leq \beta \leq \beta_{max}$, such that $\tilde{R}$ contains a constraint of the form $\beta - 1 < x < \beta$, or $x = \beta$, or $x > \beta_{max}$.
(2) $\tilde{R}$ contains an ordering on the fractional parts of the clock variables, i.e. a set of constraints of the form $(x - \lfloor x \rfloor) \sim (y - \lfloor y \rfloor)$, where $\sim \in \{<, >, =\}$ for every $x, y \in X$. (The expression $\lfloor x \rfloor$ denotes the fractional part of x.)

We say a clock region R' is the *timed-successor* of a clock region R if R' is the first new clock region reached if we let time elapse from some point in R. That is, for any clock valuation ν in R, there exists $t > 0$ such that $\nu + t$ is in R' ($R' \neq R$) and for every $0 < t' \leq t$, $\nu + t'$ is in $R \cup R'$. The maximal clock regions (those that have constraints of the form $x > \beta_{max}$, for every $x \in X$) do not have timed successors. We can show, using the definition of clock regions, that timed-successor is a well-defined partial function.

Let us denote the set of regions as $\mathcal{R}$; note that the size of $\mathcal{R}$ is exponential in the number of clocks and the length of constants. Also, there are at most $O(d \cdot \beta_{max})$ successive timed-successors to any region, where d is the number of clocks, i.e. if $R_0, R_1, \ldots R_l$ are such that each R_{i+1} is a timed-successor of R_i, then $l = O(d \cdot \beta_{max})$. [1]

A pair (q, R), where q is a location and R is a clock region, is called a *region* of A. We say a state (q, ν) belongs to a region (q', R) if $q = q'$ and ν belongs to R. The regions hence partition the set of all states of the automaton A.

If g is any constraint that is mentioned in A, then for any region R, either all points in R satisfy g or none satisfy g. Hence, let $Enabled_\delta(q, R)$ denote the set of all transitions enabled at some (and hence all) states (q, ν) in (q, R). Let $Enabled_\Sigma(q, R)$ denote the set of actions enabled at some (and hence all) states (q, ν) in (q, R). If $e \in Enabled_\delta(q, R)$, let $succ((q, R), e)$ denote the region reached when the discrete transition e is taken from any state in (q, R) (this notion is well-defined).

We say a region R is *thin* if letting any time elapse from any point in the region leads to a clock valuation outside R, i.e. if all points in R satisfy a constraint of the form $x = \beta$ for some $x \in X$ and $\beta \in \mathbb{N}$. Note that thin regions always have timed-successors. If R is not thin but has a timed-successor region R', then for every clock valuation ν in R, the minimum time required such that $\nu + t$ is in R' is $\beta - \nu(x)$ where x is the clock that has the maximum fractional

[1] Note however that since β_{max} is given in binary, l could be exponential in the size of the input; however, we can define a slightly different notion of regions that uses only the constants mentioned in the automaton and hence get l to be linear in the size of the input.

value in R (i.e. x is such that for every y, $(x - \lfloor x \rfloor) \leq (y - \lfloor y \rfloor)$ holds in R) and β is the smallest constant such that for every point ν' in R, $\nu'(x) < \beta$. We then call $(\beta - x)$ the *critical clock expression* for the region R and denote it as $cce(R)$. If R is not thin and does not have a timed successor region (i.e. if it is a maximal region), we define the critical clock expression of R to be ∞.

3.2 Expressions for the optimal cost

We now wish to define a set of functions $opt_i^{(q,R)} : R \to \mathbb{R}_+ \cup \{\infty\}$ that is supposed to capture the optimal cost for the controller to win a game from any state in (q, R) in i steps. That is, we want that for any $\nu \in R$, $opt_i^{(q,R)}(\nu)$ is the minimum cost *Cost* such that the controller has a winning strategy that wins the game in i steps and within cost *Cost* from the state(q, ν). However, this will not be precisely true as such a *Cost* may not exist, in which case we take the infimum of all possible costs within which the controller can win (see Lemma 2 below).

The following is an inductive definition of $opt_i^{(q,R)}$, by induction on i. Further, for any fixed i, we define $opt_i^{(q,R)}$ inductively with respect to the partial order imposed by the timed-successor relation. That is, when defining $opt_i^{(q,R)}$, we assume the functions $opt_i^{(q,R')}$ have been defined, where R' is a transitive timed-successor of R.

- For every location $q \in Q_F$, $R \in \mathcal{R}$, $opt_0^{(q,R)} = 0$ and for every location $q \notin Q_F$, $R \in \mathcal{R}$, $opt_0^{(q,R)} = \infty$.
- Let $i \geq 1$ and let (q, R) be a region.
  If (q, R) is a thin region, then let R' be the timed-successor of R; otherwise, let $R' = R$. Note that R' is not thin. Let $T = cce(R')$ be the critical clock expression of R'.
  If R' has a timed-successor, let it be R''.
  Let $D = Enabled_\Sigma(q, R) \cap \Sigma'$ be the controller actions enabled at (q, R) and for any $a \in \Sigma$, let $Ev(a) = \{e \mid e \in Enabled_\delta(q, R), Action(e) = a\}$ be the a-labeled transitions enabled at (q, R). Similarly, let $D' = Enabled_\Sigma(q, R') \cap \Sigma'$, and $Ev'(a) = \{e \mid e \in Enabled_\delta(q, R'), Action(e) = a\}$. Then

$$opt_i^{(q,R)}(\nu) = \begin{cases} \min\{opt_{i-1}^{(q,R)}(\nu), h_1(\nu), h_2(\nu), h_3(\nu)\} & \text{(if } R' \text{ has a timed succ)} \\ \min\{opt_{i-1}^{(q,R)}(\nu), h_1(\nu), h_2(\nu)\} & \text{(otherwise)} \end{cases}$$

  where

$$h_1(\nu) = \min_{a \in D} \max_{e \in Ev(a) \cup Ev(u)} \{opt_{i-1}^{succ((q,R),e)}(\nu/reset(e)) + W_\delta(e)\}$$
$$h_2(\nu) = \inf_{0<t<T} \min_{a \in D'} \max\{g_1(\nu,t), \quad g_2(\nu,a,t)\}$$
$$h_3(\nu) = \max\{g_1(\nu,T), \quad W_Q(q) \cdot T + opt_i^{(q,R'')}(\nu+T)\}$$

$$g_1(\nu,t'') = \sup_{0<t' \leq t''} \max_{e \in Ev'(u)} \{opt_{i-1}^{succ((q,R'),e)}((\nu+t')/reset(e)) + W_Q(q) \cdot t' + W_\delta(e)\}$$

$$g_2(\nu,a,t'') = \max_{e \in Ev'(a)} \{opt_{i-1}^{succ((q,R'),e)}((\nu+t'')/reset(e)) + W_Q(q) \cdot t'' + W_\delta(e)\}$$

Intuitively, h_1 is an expression that evaluates to the optimal cost if the controller chooses to move immediately. h_2 is an expression that evaluates the optimal cost if the controller chooses to wait for some time $0 < t < T$ within the current region (or the next region if the current region is thin) and h_3 is the cost if the controller wants to let time elapse for time T to get to the next region. The definition hence says that $opt_i^{(q,R)}$ is defined as the function that minimizes these choices as well as the choice of reaching the goal in $i-1$ or less steps.

If the controller chooses to move immediately, then it picks an action a, the adversary picks a transition labelled a or u, and the game is updated to the resulting position where the controller must win in $i-1$ steps; the definition of h_1 captures this.

If the controller chooses to move after some time $0 < t < T$, then the expression for h_2 captures the fact that the controller picks a time t and an action a to minimize the cost. The adversary then has two choices: to switch before or at t using a u-labeled transition (captured by g_1) or to switch at t with an a-labeled or u-labeled transition (captured by g_2).

The expression for h_3 captures the scenario when the controller lets time T elapse to get to the next region—the adversary can switch before time T using a u-labeled transition (captured by g_1) or choose to let time elapse (captured by the second expression in h_3). The following is not hard to prove:

Lemma 2 *Let A be a weighted timed automaton, $k \in \mathbb{N}$, and s be a state in the region (q,R). Then,*

$$opt_k^{(q,R)}(s) = \inf\{Cost \mid controller\ wins\ from\ s\ in\ k\ steps\ and\ within\ cost\ Cost\}$$

□

In order to compute the functions $opt_i^{(q,R)}$, it turns out that we need to handle primarily only three functions—min and max of a set of functions and the function:

$$f(\bar{x}) = \inf_{t\in[0,T]} \max \begin{cases} f_1(\bar{x}+t)+wt \\ \sup_{0\leq t'\leq t}(f_2(\bar{x}+t')+wt') \end{cases} \tag{1}$$

where f_1 and f_2 are functions that have already been computed, T is a critical clock expression and w is a constant.

For example, let us show how to compute $h_2(\nu)$ using min's, max's and the expression in (1) above.

Given $opt_{i-1}^{succ((q,R'),e)}(\nu)$ we can easily compute $opt_{i-1}^{succ((q,R'),e)}((\nu)/reset(e))$ by removing from all linear cost expressions the terms that correspond to clocks being reset on the edge e. In the next step, we compute partition cost functions

$$\hat{G} = \max_{e\in Ev'(u)} \{opt_{i-1}^{succ((q,R'),e)}((\nu)/reset(e)) + W_\delta(e)\}$$

and

$$\hat{F} = \max_{e\in Ev'(a)} \{opt_{i-1}^{succ((q,R'),e)}((\nu)/reset(e)) + W_\delta(e)\}$$

Now observing that

$$h_2(\nu) = \inf_{0<t<T} \min_{a\in A'} \max\{g_1(\nu,t),\ g_2(\nu,a,t)\} = \min_{a\in A'} \inf_{0<t<T} \max\{g_1(\nu,t),\ g_2(\nu,a,t)\}$$

we conclude that

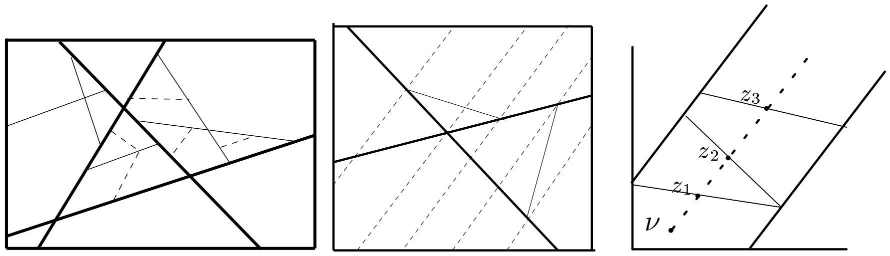
$$h_2(\nu) = \min_{a\in A'} \inf_{0<t<T} \max \begin{cases} \hat{F}(\nu+t)+wt \\ \sup_{0\leq t'\leq t} \hat{G}(\nu+t')+wt' \end{cases}$$

This expression is computable by applying using expression (1) first followed by a min operation.

In the subsequent sections, we show how to represent and compute these functions using a special structure called a *nested partition*. For the most part, we focus only on computations involving d-dimensional regions. Regions of lower dimensions can be handled similarly, but with some additional operations that allow us to project $d-m$ dimensional nested partitions to $d-m-1$ dimension partitions and others that allow us to extend $d-m-1$ dimensional nested partitions to $d-m$ dimensional nested partitions. These operations are discussed in Section 4.3.

3.3 Motivation for the definitions

In this section, we describe briefly the main difficulties that arise in showing that each function $opt_i^{(q,R)}$ is a piece-wise linear function with at most an



(a) A nested partition (b) A tube partition (c) An atomic tube

Fig. 2.

exponential number of pieces and informally describe the ideas to circumvent these. The next section gives a formal exposition of the required technical results.

We illustrate the ideas for the setting where we have two clocks $\{x_1, x_2\}$. Cost functions are hence functions from regions to $\mathbb{R}_+$, where a region is a subset of $\mathbb{R}^2_+$; these functions will be piece-wise linear and we refer to the 'pieces' as 'cells'.

As mentioned in the introduction, bookkeeping in terms of the number of cells does not suffice as superpositioning the cells of two functions f and g, each having n cells, could cause $O(n^2)$ cells and hence a double-exponential growth in cells over k iterations. We can circumvent this difficulty by instead keeping track of the number of lines that partition the region into cells. If the cells of f and g are defined using at most n lines each, the superposition of cells of f and g are formed using at most $2n$ lines. Moreover, n lines can form at most $O(n^2)$ cells (for d-dimensions, n hyperplanes can form at most $O(n^d)$ cells), and the bookkeeping works as far as superpositioning is concerned.

However, when we take $h = \min\{f, g\}$, each new cell formed by the intersection of a cell of f and a cell of g gets further split into two (along the line where $f = g$) and causes an extra line to be added. Hence there could be $O(n^2)$ new lines defining cells in h and, again, leads only to a double-exponential bound on cell growth.

The crucial observation is that the new lines that are added are contained *within* a cell and *do not intersect with lines added in other cells.* For example, in Figure 2(a), the cells formed by lines are cut by the dashed lines into at most two parts but the dashed lines do not extend beyond the cell. We exploit this structure by introducing the notion of a *nested partition* of cells. A nested partition is a tree structure where every level of the tree refines the partition of the region by dividing cells into subcells. More formally, each node is associated with a cell and also associated with a set of lines that partition this cell into subcells. Figure 2(a) illustrates a nested partition: the three bold lines partition the region into 7 cells, the thin lines partition each such cell into at most 4 cells, and the dotted lines partition each of these newly formed cells into at most 2 cells.

The complexity of a nested partition is written as a tuple of numbers $\langle n_1, \ldots, n_i \rangle$ which means the following: the region is split by n_1 lines; *each* cell formed is further split by at most n_2 lines; in general, a cell formed at the j'th level is split by n_{j+1} lines. For instance, the nested partition in Figure 2(a) has complexity $\langle 3, 2, 1 \rangle$.

Now, if we take $\min\{f, g\}$ where f and g have complexity $\langle n_1, \ldots, n_i \rangle$, then we get a function that has complexity at most $\langle 2n_1, \ldots, 2n_i, 1 \rangle$ as the lines at each level add up and each atomic cell formed by the superposition of f and g can be split by one line, which causes a new level with a single line. The number of cells formed by a nested partition of complexity $\langle n_1, \ldots, n_k \rangle$, is at most $3^k n_1^d n_2^d \ldots n_k^d$, the growth of cells is hence under control and min and max operations can be handled.

Now let us consider the expression in (1). For any clock valuation ν, when time elapses, the points $\nu + t$ lie along a *diagonal* line drawn upwards from ν. The relevant positions that need to be examined for evaluating the expression for ν hence depend on this diagonal line and the cells that this diagonal line passes through.

In order to group together points which are such that the diagonal lines from the points meet the same set of cells, we draw diagonal lines from *every intersection of lines that form cells*, as illustrated by the dotted lines in Figure 2(b). This results in a set of diagonally placed cells that we call *tubes*.

Now consider an atomic tube (i.e. a tube within which the lines forming cells do not intersect) as shown in Figure 2(c). For any point ν, we can show that (i) the distance to any of the lines along the diagonal from ν is linear and (ii) in order to optimize the expression in (1), the values of t and t' must be at the times that correspond to when the diagonal from ν meets one of these lines (depicted as z_1, z_2 and z_3 in the figure). This reduces the quantification of t and t' from possibly infinite sets to a finite set; this enables us to reduce the expression in (1) to an expression that involves just min and max and we can use the procedures developed before to handle this.

However, when evaluating this expression, the cells could further get split and this could create new intersection points from which we may need to draw diagonals again in order to evaluate (1) in cells below the current cell. But we show that splitting of cells can happen only along diagonal lines which avoids this complication.

Finally, the number of diagonal lines introduced could be large compared to the number of lines defining the cells; however diagonal lines once formed cannot contribute further to forming diagonal lines. So we enhance the nested representation so that diagonal lines are accounted for separately and use the above property to show a bound on the growth of cells.

In dimensions higher than two, diagonal hyperplanes could intersect and to control their growth, we need a nesting of tubes as well. A nested tube partition is the final structure we use and it has a complexity of the form $\langle l_1, \ldots, l_m, n_1, \ldots, n_k \rangle$ where the l_i's denote the complexity of diagonal lines that contribute to defining nested tubes and the n_i's denote how each tube thus formed is further partitioned into nested cells.

4 The Algorithm

In this section, we describe the various results that lead to an exponential time algorithm to solve the uniform optimal bounded weighted timed game problem. The first section defines the required geometric notions of cells, nested partitions, tube partitions and partition cost functions. Section 4.2 contains the two main theorems that show how the operations of min and max and that of computing the expression in (1) can be computed for partition cost functions with the appropriate complexities (Theorem 2 and Theorem 3). Section 4.3 discusses additional operations required to deal with translating partition cost functions from one region to another. Theorem 4 shows how the above theorems result in an algorithm of exponential time complexity for our problem.

4.1 Clock Space Geometry

In this section we introduce the geometric notions pertaining to our problem. Let $d \in \mathbb{N}$ denote the number of dimensions (d will be the number of clocks in the timed automaton) and consider the space $\mathbb{R}^d_+$. A *hyperplane* in $\mathbb{R}^d_+$ is a set of points that satisfy an equation of the form $a_1x_1 + a_2x_2 + \ldots + a_dx_d + b = 0$. We say that such a hyperplane is *diagonal* if $\sum_{i=1}^{d} a_i = 0$.

A *cell* is a (convex) set of points of $\mathbb{R}^d_+$ that is bounded by hyperplanes. Formally, a cell is a *d-dimensional* set of points defined by a finite set of inequalities of the form $h(x_1, \ldots, x_d) \leq 0$, where $h(x_1, \ldots, x_d)$ is a linear expression over the variables $\langle x_1, \ldots, x_d \rangle$.

Let c be a cell defined by the inequalities I and let H be a set of hyperplanes. Then the hyperplanes in H partition c into a number of subcells. Formally, the set of *subcells* of c induced by H, $\mathit{Subcells}(c, H)$, is the set of all minimal cells c', where each c' is defined by I in conjunction with a set of inequalities of the form $h'(x_1, \ldots, x_d) \leq 0$, where $h'(x_1, \ldots, x_d) = 0$ belongs to H.

Note that the cells in $\mathit{Subcells}(c, H)$ divide c into a number of subcells— however, this is not a partition since the points at the border of a subcell can belong to the border of other subcells as well. It is well known that $\mathit{Subcells}(c, H)$ contains $O(|H|^d)$ cells (in fact $3|H|^d$ cells; see for example [Mat02]).

Definition 3 *A* nested partition *of dimension d and depth i is a structure (Tr, η, H) where $Tr = (V, E)$ is a finite rooted tree, and for each $v \in V$, η is a function that maps v to a cell of dimension d and H is a function that maps v to a finite set of hyperplanes in $\mathbb{R}_+^d$ such that the following hold:*

- *A nested partition of depth 0 is (Tr, η, H) where Tr is a single node tree $(\{v\}, \emptyset)$, η maps v to a cell and $H(v) = \emptyset$.*
- *A nested partition of depth $i+1$ $(i \geq 0)$ is a tree (V, E) that satisfies the following. If the root is r, then let $c = \eta(r)$. Then for every $c' \in \mathit{Subcells}(c, H(r))$, there is precisely one child v of the root such that $\eta(v) = c'$ and these are the only children of the root. Also the tree rooted at the children of the root must be nested partitions of depth j, where $j \leq i$, and the tree rooted at at least one child is a nested partition of depth i.* □

The domain of a nested partition P is $D_P = \eta(root)$ where $root$ is the root of P, i.e. D_P is the cell that labels the root of the tree. For any tree, we say a vertex is at level i if its distance from the root is $i-1$ (the root is hence at level 1 and if a tree is of depth k, then there is a leaf at level $k+1$).

For a nested partition P, we say that P is of *complexity at most* $\langle n_1, \ldots n_k \rangle$, denoted $P \prec \langle n_1, \ldots n_k \rangle$, if P has depth k and for every vertex v, if v is at level i, then $H(v)$ contains at most n_i hyperplanes.

We are interested in the set of cells that are at the leaves of a nested partition P; let us call these *base cells* and denote the set of base cells as $BC(P) = \{c \mid \exists v \in V, v \textit{ is a leaf}, \textit{ and } \eta(v) = c\}$. It is not hard to see that if $P \prec \langle n_1, \ldots n_k \rangle$, then $|BC(P)| \leq 3^k n_1^d \ldots n_k^d$.

We define an operation on nested partitions that takes two nested partitions P_1 and P_2 and creates the coarsest nested partition that refines both P_1 and P_2. This operation $P_1 \oplus P_2$ creates a nested partition P where for every two base cells c_1 in P_1 and c_2 in P_2, if $c' = c_1 \cap c_2$ is nonempty, then c' is a base cell of P. It turns out that if $P_i \prec \langle n_1^i, \ldots, n_k^i \rangle$, $i = 1, 2$, then $P_1 \oplus P_2 \prec \langle n_1^1 + n_1^2, \ldots, n_k^1 + n_k^2 \rangle$.

Let us first define the operation $\oplus$ on nested partitions. Formally, let $P_i = (Tr_i, \eta_i, H_i)$, $i = 1, 2$, be two nested partitions with the same domain D. Assume without loss of generality that they are of the same depth k and in fact that all leaves at level k. (If a leaf v is of height $l < k$, then it can be extended to level k using a sequence of nodes, $v_{l+1}, \ldots v_k$ where $\eta(v_j) = \eta(v)$ and $H(v_j) = \emptyset$, for each $j = l+1, \ldots, k$).

Now, let $P = (Tr, \eta, H)$ be defined as follows. $Tr = (V, E)$ where $V = \{(v_1, v_2) \mid v_1, v_2$ belong to the same level$\}$ and $((v_1, v_2), (v_1', v_2')) \in E$ iff $(v_1, v_1') \in E_1$ and $(v_2, v_2') \in E_2$. Let $\eta((v_1, v_2)) = \eta(v_1) \cap \eta(v_2)$, i.e. the cell formed by the conjunction of the inequalities defining $\eta(v_1)$ and $\eta(v_2)$. Also, let $H((v_1, v_2)) = H(v_1) \cup H(v_2)$. Prune a vertex v (and the subtree below it) if the cell at v is empty or not d-dimensional or not minimal amongst the cells at the siblings of v. Then it is easy to show that P is indeed a nested partition (of the same depth k) and we define $P_1 \oplus P_2$ to be this nested partition. Also, it is easy to see that if $P_i \prec \langle n_1^i, \ldots, n_k^i \rangle$, $i = 1, 2$, then $P_1 \oplus P_2 \prec \langle n_1^1 + n_1^2, \ldots, n_k^1 + n_k^2 \rangle$.

A *partition cost function* is a pair (P, F) where P is a nested partition and F is a mapping that maps each leaf node v of P to a linear expression f_v over the variables $\{x_1, \ldots, x_d\}$ such that the following condition holds: let u belong to two different base cells at leaves v and v'; then, $F(v)(u) = F(v')(u)$.

In other words, a partition cost function defines linear cost functions at the base cells and if a point is present in many base cells, then the cost for this point is the same at all these base cells. A partition cost function (P, F) hence assigns a cost to each point in the domain given by $\hat{F} : D_P \to \mathbb{R}_+$ with $\hat{F}(u) = F(v)(u)$ where v is any leaf such that the base cell at v contains u.

A *tube* is a cell that is formed by diagonal hyperplanes. If H is a set of diagonal hyperplanes, let us call the set of cells it induces $\mathit{Tubes}(H)$.

Let $m_1, m_2 \in \mathbb{N}$. Then an (m_1, m_2) *nested tube partition* of dimension d is a nested partition P of dimension d that has depth $(m_1 + m_2)$ such that for all $i : 1 \le i \le m_1$, if v is a node at level i, then $H(v)$ contains only diagonal planes. If $P = ((V, E), \eta, H)$ is an (m_1, m_2) nested tube partition then the set of *base tubes* of P is $BT(P) = \bigcup\{\mathit{Tubes}(H(v)) \mid \exists v \in V$ and v is a m_1-level vertex$\}$.

4.2 Operations

In this section we consider operations on partition cost functions and the change in complexity that they result in. There are two main theorems: Theorem 2 shows the complexity for min/max operations and Theorem 3 shows the complexity for handling the expression (1).

For any partition cost function (P, F), let us denote the function it represents as $\hat{F}$. F maps the base cells to linear expressions of the form $y = a_1 x_1 + \ldots +$

$a_d x_d$. Below we will often use the fact that such expressions define hyperplanes in $\mathbb{R}_+^{d+1}$.

Given a finite set of hyperplanes $W = \{h_i : y = a_1^i x_1 + \ldots + a_d^i x_d + b^i\}_{i=1}^m$, we say that a hyperplane $h_u : y = a_1^u x_1 + \ldots + a_d^u x_d + b^u, h_u \in W$, is an *upper hyperplane* of W on a set $S \subseteq \mathbb{R}_+^d$ if it lies above all other hyperplanes in W when restricted to S, i.e. $a_1^u x_1 + \ldots + a_d^u x_d + b^u \geq a_1^i x_1 + \ldots + a_d^i x_d + b^i$ for all $i, 1 \leq i \leq m$, and all $(x_1, \ldots, x_d) \in S$. Similarly, we define a *lower hyperplane* h_l of W. The following is a simple but useful fact that intuitively says that if a set of hyperplanes defined on a cell do not intersect within the cell (they can intersect at the boundaries of the cell), then there is a lower and upper hyperplane on the cell.

Proposition 1 *For a set of hyperplanes, $W = \{h_i\}_{i=1}^m$, $h_i : y = a_1^i x_1 + \ldots + a_d^i x_d + b^i$, consider the set $X \subseteq \mathbb{R}_+^d$ such that $x \in X$ iff $\exists i, j : a_1^i x_1 + \ldots + a_d^i x_d + b^i = a_1^j x_1 + \ldots + a_d^j x_d + b^j$, where $(y = a_1^i x_1 + \ldots + a_d^i x_d + b^i), (y = a_1^j x_1 + \ldots + a_d^j x_d + b^j) \in W$.*

Then there exist upper and lower hyperplanes of W on a cell c iff $X \cap c \subseteq bd(c)$, where $bd(c)$ is the set of border points of c ($x \in bd(c)$ if any neighborhood of x contains a point y such that $y \notin c$). □

Theorem 2 *Let (P_i, F_i), $i = 1, \ldots m$, be m d-dimensional partition cost functions, defined over the same domain D, $P_i \prec \langle n_1, \ldots, n_k \rangle$ for all i. Then there exists a partition cost function (P, F) over D, $P \prec \langle m \cdot n_1, \ldots, m \cdot n_k, m^2 \rangle$ such that $\hat{F} = \min_{i=1,\ldots,m}\{\hat{F}_i\}$.*

Proof. Intuitively, we first take P' that refines all the partitions P_i, $i = 1, \ldots, m$, i.e. $P' = P_1 \oplus P_2 \ldots P_m$. Then $P' \prec \langle m \cdot n_1, \ldots, m \cdot n_k \rangle$. At each base cell of P', we take the minimum of the functions $\hat{F}_i$; since each pair of functions can cause a split of the cell along one hyperplane, there are at most m^2 hyperplanes added as a new level.

Let $P' = P_1 \oplus \ldots \oplus P_m$. For each $c_{base} \in BC(P')$ consider the set of linear expressions $L_{c_{base}} = \{F_i(v_i)\}_{i=1}^m$ such that v_i is a leaf of P_i, $c_i = \eta(v_i)$ (thus, $c_i \in BC(P_i)$) and $c_{base} = c_1 \cap \ldots \cap c_m$. It follows that $\hat{F}(x) = \min_{i=1,\ldots,m} F_i(v_i)(x)$ for all $x \in c_{base}$. $L_{c_{base}}$ defines a set of hyperplanes in $\mathbb{R}_+^{d+1}$. Suppose two $(d+1)$-hyperplanes $h' : y = a'_1 x_1 + \ldots + a'_d x_d + b'$ and $h'' : y = a''_1 x_1 + \ldots + a''_d x_d + b''$, $h', h'' \in L(c_{base})$, intersect. The intersection is described by the system of equations:

$$\begin{cases} y = a'_1 x_1 + \ldots + a'_d x_d + b' \\ y = a''_1 x_1 + \ldots + a''_d x_d + b'' \end{cases}$$

Eliminating y we obtain $(a'_1 - a''_1)x_1 + \ldots + (a'_d - a''_d)x_d + (b' - b'') = 0$ which defines a d-dimensional hyperplane. Let $L_{c_{base}}^{int}$ be a set of d-hyperplanes that correspond to all possible intersections of hyperplanes in $L_{c_{base}}$. Clearly, $|L^{int}_{c_{base}}| \leq \binom{m}{2} < m^2$.

We construct a nested partition P from P' by adding one more level at its leaves. For each k-level vertex $v \in Tr$ (where Tr is the tree for P), we set $H_P(v) = L^{int}_c$, where $c = \eta_{P'}(v)$. This splits c into a set of subcells; for each subcell $\tilde{c}$ we create a child $\tilde{v}$ of v, and set $\eta_P(\tilde{v}) = \tilde{c}$ and $H_P(\tilde{v}) = \emptyset$.

Since P is a refinement of P', for any cell $c \in BC(P)$ we can find $c' \in BC(P')$ such that $c \subseteq c'$. We define $F(c)$ as a lower hyperplane of the set $L_{c'}$. By our construction and Proposition 1 such mapping is well defined. It is also easy to see that F is defined correctly on the cell borders (i.e. if $x \in c_1$ and $x \in c_2$ then $F(c_1)(x) = F(c_2)(x)$). $\square$

The above theorem also holds for the max function. [2]

Theorem 3 *Let (P_F, F) and (P_G, G) be two partition cost functions over the same domain, where P_F and P_G are (k_1, k_2) nested tube partitions of complexities at most $\langle l_1, \ldots l_{k_1}, n_1, \ldots n_{k_2} \rangle$. Consider the function:*

$$S(\bar{x}) = \inf_{t:\bar{x}+t\in D} \max \begin{cases} \hat{F}(\bar{x}+t) + wt \\ \sup_{0\leq t' \leq t} \hat{G}(\bar{x}+t') + wt' \end{cases} \tag{2}$$

Then there exists a partition cost function (P_K, K) such that P_K is a d-dimensional (k_1+2, k_2+1) nested tube partition of complexity at most $\langle 2l_1, \ldots, 2l_{k_1}, (k_2+1)3^{k_2+1}(2n_1)^{d+1} \cdot \ldots \cdot (2n_{k_2})^{d+1}, (2n_1 + \ldots + 2n_{k_2} + 3)^2, 2n_1, \ldots, 2n_{k_2}, 7 \rangle$ and $\hat{K} = S$.

We prove the theorem using several lemmas. Below, we use the convention that if $\bar{x} = (x_1, \ldots x_m)$ is a vector then $\bar{x} + \alpha$ denotes $(x_1 + \alpha, \ldots x_m + \alpha)$.

Let P be the partition $P_F \oplus P_G$ with a new level created by taking the hyperplane along which F and G divide each cell, as in the proof of Theorem 2. First, we show that the values of t and t' that we need to consider to evaluate $S(\nu)$ can be constrained to belong to the points at which the diagonal from ν meets the various hyperplanes of P, or 0. We then want to "drop diagonal hyperplanes" (as in Figure 2(b)) from each point of intersection of hyperplanes that define the nested partition.

[2] For precise complexity bounds, we must also establish bounds on the growth of the coefficients used in the definition of hyperplanes. Typically, the representation of coefficients grows when we consider intersections of hyperplanes, but these grow slowly (linearly).

For any node v of a nested partition P, let L_v denote the union of the sets of all hyperplanes that label v and its ancestors. If h_1 and h_2 are two non-diagonal hyperplanes, we say that (h_1, h_2) is a *ridge* if there exists a node v such that h_1 and h_2 belong to L_v. A simple counting argument shows the following:

Lemma 4 *Let $P \prec \langle n_1, \dots n_k\rangle$ be a nested partition of dimension d. Then the number of ridges in P is bounded by $k \cdot 3^k \cdot n_1^{d+1} \cdot \dots \cdot n_k^{d+1}$.*

Proof. Let $P = (Tr, \eta, H)$ and let us denote by r_i the maximal possible number of ridges (for a partition of the given complexity) that are formed by a pair of hyperplanes such that one belongs to $H(v)$ for a vertex v at level i and the other belongs to $H(v')$, where v' is a predecessor of v in Tr or v itself.

Fix an i-level vertex v. There are at most n_i hyperplanes in $H(v)$ and thus they form at most $\binom{n_i}{2}$ ridges. Further each hyperplane in $H(v)$ can intersect with hyperplanes from the set $\{h \in H(v') |\ v' \text{ is a predecessor of } v \text{ in } T\}$. The maximal number of the corresponding ridges is $n_i(n_1 + \dots + n_{i-1})$.

The number of i-level vertices in T is at most $l = 3^{i-1}n_1^d \cdot \dots \cdot n_{i-1}^d$ and it follows that $r_i = l(\binom{n_i}{2} + n_i(n_1 + \dots + n_{i-1})) < 3^{i-1}n_1^d \cdot \dots \cdot n_{i-1}^d n_i(n_1 + \dots + n_{i-1} + n_i) < 3^{i-1}n_1^{d+1} \cdot \dots \cdot n_{i-1}^{d+1} \cdot n_i^{d+1}$. It easy to see that $r_1 \leq \dots \leq r_k$, thus $r = r_1 + \dots + r_k \leq kr_k < 3^k k n_1^{d+1} \cdot \dots \cdot n_k^{d+1}$, which proves the lemma. $\square$

Before proving the next lemma let us define an auxiliary operation *insert* on nested partitions. If P is a nested partition of depth m and if $s \in \mathbb{N}$ is such that $s \leq m$, then $P' = insert(P, s)$ is defined as follows. First, the nested partition P' is such that the tree Tr of P coincides with the tree Tr' of P' at the first s levels. Also, for each vertex v at the level $s+1$ we set $H_{P'}(v) = \emptyset$ and introduce a new vertex v', which is the only child of v. Now we make the subtree rooted at v in Tr coincide with the subtree rooted at the v' in Tr'. It easy to see that $BC(P') = BC(P)$ and if $P \prec \langle n_1, \dots, n_m\rangle$ then $P' \prec \langle n_1, \dots, n_s, 0, n_{s+1}, \dots, n_m\rangle$.

We say that a tube partitioning P of type (k_1, k_2) is *atomic* if hyperplanes that partition cells at the last k_2 levels do not intersect with each other in the interior of the tube they belong to. We now want to "drop" diagonal hyperplanes from each ridge so that the resulting tubes are atomic. Using Lemma 4 we can show the following:

Lemma 5 *Let (P, F) be a partition cost function, where P is a d-dimensional (k_1, k_2) nested tube partition and $P \prec \langle l_1, \dots l_{k_1}, n_1, \dots n_{k_2}\rangle$. Then there exists a partition cost function (P', F'), where P' is atomic, which defines the same function (i.e. $\hat{F} = \widehat{F'}$) such that P' is a $(k_1 + 1, k_2)$ nested tube partition and $P' \prec \langle l_1, \dots l_{k_1}, k_2 3^{k_2} n_1^{d+1} \dots n_{k_2}^{d+1}, n_1, \dots, n_{k_2}\rangle$.*

Proof. Consider a subtree t whose root v is a (k_1+1) level vertex of P. By Lemma 4 a subpartition that is defined by t has at most $k_2 3^{k_2} n_1^{d+1} \cdot \ldots \cdot n_{k_2}^{d+1}$ ridges. For each ridge r we construct a diagonal hyperplane h_r which goes through r (i.e. for any point $\bar{x} \in \mathbb{R}_+^d$: if $\bar{x} \in r$ then $\bar{x} \in h_r$). The construction is quite simple. Suppose that r is given as a system of two equations: $\{h_1(\bar{x}), h_2(\bar{x})\}$, where $h_1(\bar{x})$ and $h_2(\bar{x})$ are the equations of two intersecting hyperplanes constituting the ridge. Take any point $\bar{y} \in h_r$. Since h_r is a diagonal hyperplane, the two equations, $\{h_1(\bar{y}+\alpha), h_2(\bar{y}+\alpha)\}$, must hold for any α. Expressing α using one equation and substituting it in the other we obtain the equation for h_r.

Let G_v denote the set of diagonal hyperplanes constructed from the tree rooted at v. To complete the proof we need two auxiliary nested tube partitions A_1 and A_2. Partition A_1 coincides with P up to the level k_1. For each vertex v at the level k_1+1 we set $H_{A_1}(v) = G_v$ and children of v are leaves. It follows that $A_1 \prec \langle l_1, \ldots l_{k_1}, k_2 3^{k_2} n_1^{d+1} \cdot \ldots \cdot n_{k_2}^{d+1} \rangle$. $A_2 = insert(P, k_1)$ and thus $A_2 \prec \langle l_1, \ldots l_{k_1}, 0, n_1, \ldots, n_{k_2} \rangle$

Now let $P' = A_1 \oplus A_2$. Notice that $P' \prec \langle l_1, \ldots l_{k_1}, k_2 3^{k_2} n_1^{d+1} \cdot \ldots \cdot n_{k_2}^{d+1}, n_1, \ldots, n_{k_2} \rangle$ (Since the hyperplanes marking the first k_1 levels coincide in the two partitions, the complexity of these levels do not increase). P' is a refinement of P thus if $c' \in BC(P')$ then $\exists c \in BC(P)$, such that $c' \subseteq c$ and we define $F'(c') = F(c)$. It is left to show that P' is atomic. But this is obvious since by construction the ridge formed by the intersection of any two non-diagonal hyperplanes lies on a diagonal hyperplane which marks a vertex within the first k_1+1 levels. $\square$

We will write $P' = atomic(P)$ if P' is the atomic nested tube partition obtained from P as shown in the proof of Lemma 5.

We are now ready to prove Theorem 3. Intuitively the argument is as follows. First, note that if time elapses from two points within the same cell of an atomic tube, then they meet the same set of "border" hyperplanes (that partition the region). We can show that for any cell, the time required by points in the cell to reach a particular border is a linear expression. Using the fact that the values of t and t' in expression (2) have to be evaluated only for the values that correspond to hitting these borders, we show we can rewrite the expression for S using min's and max's. When evaluating this expression, a cell within an atomic tube could get split and hence cause additional ridges from which we may have to drop diagonal hyperplanes—however, we show that this cannot happen as these splits will be diagonal hyperplanes themselves. A careful analysis of the cost of evaluating the expression then yields the theorem.

Proof of Theorem 3

We start by computing $(P, M) = \min((P_F, F), (P_G, G))$. We are only interested in the resulting partition P ($P \prec \langle 2l_1, \ldots, 2l_{k_1}, 2n_1, \ldots, 2n_{k_2}, 1\rangle$ by Theorem 2), and not in M. Let $P_a = \mathit{atomic}(P)$. By Lemma 5, P_a is an atomic tube partition of type (k_1+1, k_2+1) and complexity $P_a \prec \langle 2l_1, \ldots, 2l_{k_1}, (k_2+1)3^{k_2+1}(2n_1)^{d+1} \cdot \ldots \cdot (2n_{k_2})^{d+1}, 2n_1, \ldots, 2n_{k_2}, 1\rangle$.

Let (P_a, F') and (P_a, G') be two cost partition functions such that $\hat{F}' = \hat{F}$ and $\hat{G}' = \hat{G}$. The functions are well-defined since P_a refines both P_F and P_G. Thus, the formula below is equivalent to (2):

$$S(\bar{x}) = \inf_{t:\bar{x}+t\in D} \max \begin{cases} \hat{F}'(\bar{x}+t) + wt \\ \sup_{0\leq t'\leq t} \hat{G}'(\bar{x}+t') + wt' \end{cases}.$$

Consider an atomic tube $c \in \mathit{Tubes}(P_a)$. It easy to see that if $\bar{x} \in c$ and t is such that $\bar{x}+t \in D$ then $\bar{x}+t \in c$. It follows that to compute $S(\bar{x})|_c$ one needs to know only $\hat{F}'(\bar{x})|_c$ and $\hat{G}'(\bar{x})|_c$, i.e. the function $\hat{F}'$ and $\hat{G}'$ restricted to c.

For a given $\bar{x} \in D$ let us denote by τ and τ' the values for t and t', respectively, that optimize the expression in $S(\bar{x})$. Without loss of generality assume that both τ and τ' exist [3].

We now show the following claim:
Claim: If $\tau > 0$, then the value of τ be chosen such that $\bar{x}+\tau$ lies on a border of a cell.

The proof of the claim is by contradiction. Suppose that $\tau > 0$ and τ could not be chosen in such a way that the point $\bar{x}+\tau$ lies on a border. Pick a base cell c such that $\bar{x}+\tau \in c$. Since $\bar{x}+\tau$ is not on a border then $\exists \tau_1, \tau_2 : \bar{x}+\tau_1 < \bar{x}+\tau < \bar{x}+\tau_2$, $\bar{x}+\tau_1, \bar{x}+\tau_2 \in c$, $\bar{x}+\tau_2$ lies on a border, and either $\tau_1 = 0$ or $\bar{x}+\tau_1$ lies on another border. Consider two functions over $[\tau_1, \tau_2]$: $f(t) = \hat{F}'(\bar{x}+t) + wt$ and $g(t) = \sup_{0\leq t'\leq t}(\hat{G}'(\bar{x}+t') + wt')$, for every $t \in [\tau_1, \tau_2]$. The function f is linear and hence must be either strictly increasing, or strictly decreasing or a constant function. Because of the sup operation in the definition of g, g is either strictly increasing (this happens if $\tilde{g}(t) = \hat{G}'(\bar{x}+t) + wt$ is strictly increasing in which case $g = \tilde{g}$) or a constant function (if $\tilde{g}$ is a non-increasing function and hence $g = \tilde{g}(0)$). There are six possible cases; let us consider four of them:

(1) g is a constant function and f is strictly decreasing or a constant function.

[3] We can show that when computing the cost function, for every unbounded cell of every unbounded region R, the cost for c must be a constant function. This validates our assumption.

Then it easy to see that choosing τ_2 yields a smaller or equal value for $S(\bar{x})$ than choosing τ which is contradiction.

(2) g is strictly increasing and f is a strictly decreasing or a constant function. Two subcases are possible:
   (a) $g > f$ for all $t \in [\tau_1, \tau_2]$. Then we can choose τ_1 instead of τ. Contradiction.
   (b) $g < f$ for all $t \in [\tau_1, \tau_2]$. Then we can choose τ_2 instead of τ. Contradiction.

The remaining two cases are similar. **End of claim.**

A similar argument shows that if $\bar{x} + \tau$ lies on a border then τ' can be chosen such that $\tau' = 0$ or $\bar{x} + \tau'$ lies on a border of some cell.

Given an atomic tube $tb \in BT(P_a)$ consider the set $U_{tb} = \{\ h_i^d : a_1^i x_1 + \ldots + a_d^i x_d + b^i = 0\}_{i=0}^{s_{tb}}$, of all d-dimensional non-diagonal hyperplanes which lie in tb (and correspond to descendants of v where v is defined by $tb = \eta(v)$). We know that $s_{tb} \leq (2n_1 + \ldots + 2n_{k_2} + 1) + 2$. Since P_a is an atomic partition we can impose an order on U_{tb} by requiring that if $\bar{z} \in h_i$ and $\bar{z} + t \in h_j$, $t > 0$ then $j > i$.

For each $h_i \in U_{tb}$ we construct two $(d+1)$-dimensional hyperplanes $y = \hat{F}_i(\bar{x})$ and $y = \hat{G}_i(\bar{x})$ (we will call them *border cost hyperplanes*) such that for all t, $\bar{x} + t \in h_i$, we have $\hat{F}_i(\bar{x}) = \hat{F}(\bar{x} + t) + wt$ and $\hat{G}_i(\bar{x}) = \hat{G}(\bar{x} + t) + wt$. We construct $y = \hat{F}_i(\bar{x})$ only, the construction of $y = \hat{G}_i(\bar{x})$ is similar.

Consider a base cell c which has its border lying on h_i. By definition a point $(y, \bar{x}) \in \mathbb{R}_+^{d+1}$ lies on the hyperplane $y = \hat{F}_i(\bar{x})$ if there exists α such that:

$$\begin{cases} h_i(\bar{x} + \alpha) \\ y = F(c)(\bar{x} + \alpha) + w\alpha \end{cases}$$

Eliminating α we obtain the equation $y = \hat{F}_i(\bar{x})$ in the explicit form. It easy to see that $\hat{F}_i(\bar{x})$ is a linear expression and thus $y = \hat{F}_i(\bar{x})$ is indeed a hyperplane.

For any $\bar{x} \in tb$ we denote as $u_{l(\bar{x})} \in U_{tb}$ the border which is "closest and above" $\bar{x}$, i.e. $\nexists t \geq 0$ such that $\bar{x} + t \in u_r$ and $r < l(\bar{x})$. Then from the discussion above it follows that:

$$S(\bar{x})|_{tb} = \min \begin{cases} \min_{l(\bar{x}) \leq i \leq s_{tb}} \max \begin{cases} \max \begin{cases} \hat{F}_i(\bar{x}) \\ \max_{l(\bar{x}) \leq j \leq i} \hat{G}_j(\bar{x}) \end{cases} \\ \hat{G}(\bar{x}) \end{cases} \\ \max\{\hat{F}(\bar{x}), \hat{G}(\bar{x})\} \end{cases} \quad (3)$$

Intuitively, either the controller can pick t to be 0 (this corresponds to $\max\{\hat{F}, \hat{G}\}$) or it picks a border i. In the latter case, the adversary picks a border j before i or can choose $t' = 0$. The above expression captures this.

Formula (3) is equivalent to:

$$S(\bar{x})|_{tb} = \min \begin{cases} \max \begin{cases} \min_{l(\bar{x}) \le i \le s_{tb}} \max \begin{cases} \hat{F}_i(\bar{x}) \\ \max_{l(\bar{x}) \le j \le i} \hat{G}_j(\bar{x}) \end{cases} \\ \hat{G}(\bar{x}) \end{cases} \\ \max\{\hat{F}(\bar{x}), \hat{G}(\bar{x})\} \end{cases}$$

which in turn is equivalent to:

$$S(\bar{x})|_{tb} = \max \begin{cases} \min \begin{cases} \min_{l(\bar{x}) \le i \le s_{tb}} \max \begin{cases} \hat{F}_i(\bar{x}) \\ \max_{l(\bar{x}) \le j \le i} \hat{G}_j(\bar{x}) \end{cases} \\ \hat{F}(\bar{x}) \end{cases} \\ \hat{G}(\bar{x}) \end{cases} \tag{4}$$

Since we know how to compute min and max over partition cost functions we can evaluate the formula for each tube separately and obtain $\hat{K}(\bar{x})$. However, min and max operations in (4) are computed over a large number of functions and thus require careful examination. The rest of the proof is devoted to finding the complexity of the tube arrangement P_K.

Consider a tube tb. First we show that if two $(d+1)$-dimensional hyperplanes from $BH(tb) = \{y = \hat{F}_i(\bar{x}), y = \hat{G}_i(\bar{x})\}_{i=1}^{s_{tb}}$ intersect then the projection of the intersection on D is a diagonal hyperplane.

We will use the simple fact that for $h \in BH(tb)$, if $(x_0, y_0, z_0) \in h$ then $(x_1 + \alpha, \ldots, x_d + \alpha, y - \alpha w) \in h$ for any α. It follows that if $a_1 x_1 + \ldots + a_d x_d + a_{d+1} y + b = 0$ is the equation for w then $\sum_{i=1}^{d} a_i - w a_{d+1} = 0$.

Suppose that $h', h'' \in BH(tb)$ are defined by $a'_1 x_1 + \ldots + a'_d x_d + a'_{d+1} y + b' = 0$ and $a''_1 x_1 + \ldots + a''_d x_d + a''_{d+1} y + b'' = 0$ respectively. Eliminating y we obtain $h_{proj} : (a''_{d+1} a'_1 - a'_{d+1} a''_1) x_1 + \ldots + (a''_{d+1} a'_d - a'_{d+1} a''_d) x_d + (a''_{d+1} b' - a'_{d+1} b'') = 0$. In order for h_{proj} to be a diagonal plane it should satisfy $(a''_{d+1} a'_1 - a'_{d+1} a''_1) + \ldots + (a''_{d+1} a'_d - a'_{d+1} a''_d) + (a''_{d+1} b' - a'_{d+1} b'') = 0$. But we know that $a'_1 + \ldots + a'_d - w a'_{d+1} = 0$ and $a''_1 + \ldots + a''_d - w a''_{d+1} = 0$ and eliminating the last term from both equations we get exactly what is required, i.e. $(a''_{d+1} a'_1 - a'_{d+1} a''_1) + \ldots + (a''_{d+1} a'_d - a'_{d+1} a''_d) + (a''_{d+1} b' - a'_{d+1} b'') = 0$.

There are at most $\binom{2(2n_1+\ldots+2n_{k_2}+3)}{2} \leq 2(2n_1+\ldots+2n_{k_2}+3)^2$ intersections of hyperplanes in $BH(tb)$ and each intersection corresponds to a diagonal hyperplane on D. Let denote the set of these diagonal hyperplanes as $R(tb)$.

In our next step we partition each $tb \in BT(P_a)$ with the diagonal hyperplanes from $R(tb)$. We will do this exactly in the same way as we did in the proof of Lemma 5. After this operation we obtain a new nested tube partition P'_a such that $P'_a \prec \langle 2l_1, \ldots, 2l_{k_1}, (k_2+1)3^{k_2+1}(2n_1)^{d+1} \cdot \ldots \cdot (2n_{k_2})^{d+1}, 2(2n_1+\ldots+2n_{k_2}+3)^2, 2n_1, \ldots, 2n_{k_2}, 1\rangle$. Take any cell $c \in P'_a$, $c \in tb'$, $tb' \in BT(P'_a)$, $tb' \subseteq tb$, $tb \in BT(P_a)$. The proposition 1 holds for the set of hyperplanes $BH(tb)$ and c. It follows that we can define a new partition cost function (P'_a, I) such that $I(c)(\bar{x}) = \min_{l(\bar{x}) \leq i \leq s_{tb}} \max \begin{cases} \hat{F}_i(\bar{x}) \\ \max_{l(\bar{x}) \leq j \leq i} \hat{G}_j(\bar{x}) \end{cases}$.

Thus, from formula (4):

$$\hat{K}(\bar{x}) = S(\bar{x}) = \max \begin{cases} \min \begin{cases} \hat{I}(\bar{x}) \\ \hat{F}(\bar{x}) \end{cases} \\ \hat{G}(\bar{x}) \end{cases}$$

Using operation *insert* applied to P_F and P_G let us define two auxiliary partitions $P'_F \prec \langle l_1, \ldots l_{k_1}, 0, 0, n_1, \ldots n_{k_2}, 0\rangle$ and $P'_G \prec \langle l_1, \ldots l_{k_1}, 0, 0, n_1, \ldots n_{k_2}, 0\rangle$ and two new partition cost functions (P'_F, F') and (P'_F, G') such that $\hat{F}' = \hat{F}$ and $\hat{G}' = \hat{G}$.

Now we can compute (P_K, K):

$$(P_K, K) = \max \begin{cases} \min \begin{cases} (P'_a, I) \\ (P'_F, F') \end{cases} \\ (P'_G, G') \end{cases}$$

It follows that $P_K \prec \langle 2l_1, \ldots, 2l_{k_1}, (k_2+1)3^{k_2+1}(2n_1)^{d+1} \cdot \ldots \cdot (2n_{k_2})^{d+1}, (2n_1+\ldots+2n_{k_2}+3)^2, 2n_1, \ldots, 2n_{k_2}, 1, 1, 1\rangle$ and after collapsing the last three levels of the partition tree into one: $P_K \prec \langle 2l_1, \ldots, 2l_{k_1}, (k_2+1)3^{k_2+1}(2n_1)^{d+1} \cdot \ldots \cdot (2n_{k_2})^{d+1}, (2n_1+\ldots+2n_{k_2}+3)^2, 2n_1, \ldots, 2n_{k_2}, 7\rangle$. □

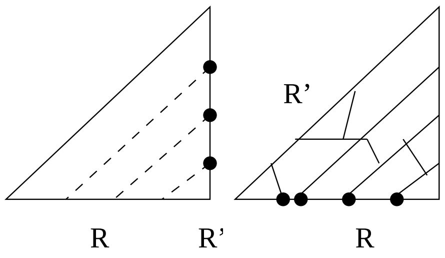


Fig. 3. An example of the *Ext* and *Proj* operations

4.3 Handling multiple dimensions

Computing the opt_i functions involves operations on the partition cost functions defined on regions of different dimensions. For example, when we consider a situation in which from the non-thin region $R = \{x < 1, y < 1, y < x\}$ both the controller and the adversary let time elapse until the next (thin) region $R' = \{x = 1, y < 1\}$ is reached, we need to calculate the partition cost function on R from that on R'. However, the operations on cost partition functions that we used in the previous section deal only partition cost functions defined on the same regions. To overcome this difficulty we introduce two operations, *Ext* and *Proj*.

The first operation, *Ext*, called the *extension operation*, intuitively extends a partition cost function on a thin region to a corresponding partition cost function over the non-thin region below it. The operation is constructed as follows (see the first illustration in Figure 3). Given a thin region R' with a partition cost function F' on it, let the time predecessor region of R' be R. Then, we create a tube partition P on R by drawing a diagonal (d-dimensional) hyperplanes through each of the $(d-1)$-dimensional hyperplanes that form the nested partition P' of F'. Complexity of the tube part of P' coincides with the complexity of the original partition (which does not have tubes).

The cost a point p in R is determined by two summands: the first summand is the cost to reach R' when neither the controller nor the adversary take a discrete transition; the second summand is given by F' at the point where R' is reached. It can be shown that the cost defined in this way is determined, in each tube, by a linear expression over the clock valuations.

The second operation is the *projection operation*, denoted *Proj* and illustrated in the second figure in Figure 3, that takes a partition cost function F' on a non-thin region R' and returns a cost partition function F on a thin region R, where R is a predecessor region of R'. To construct *Proj* we find projections of the hyperplanes that define cells in R' onto R. These projections form a nested partition P in R. Every cell in P is a projection of a cell in the nested partition of F' and thus we can define cost expressions on cells in P as projections on R of the linear cost expressions of the corresponding cells in F'.

We can extend *Ext* and *Proj* for thin and non-thin regions of dimensions lesser than d by observing that the time successor and predecessor of a non-thin region of the dimension $d-m$ are thin regions of the dimension $d-m-1$.

4.4 The Main Result

Theorem 4 *Given a WTA A and $k \in \mathbb{N}$, the (uniform) optimal bounded weighted timed game problem can be solved in time exponential in k and the size of A.*

Proof. We compute partition cost functions for each function $opt_i^{(q,R)}$. Each such computation is inductive with inductive depth $O(d\beta_{max}k)$ where β_{max} is the maximum constant that occurs in constraints of A. We can show, inductively, that after r operations of min's, max's or of the expression in (2), we have a partition cost function (P, F) where P is a tube partition of complexity at most $\langle l_1, \ldots, l_{k_1}, n_1, \ldots n_{k_2}\rangle$ where k_1 and k_2 are $O(r)$, each l_i is $O(m^{poly(r)})$ and each n_i is $O(m^r)$, where m is the outdegree of the game graph. The number of cells is hence at most exponential in r. □

5 A lower bound

Our algorithm to solve the optimal bounded weighted game problem runs in exponential time and the time complexity is primarily due to the fact that the number of cells a region can be split into is bounded above only by an exponential in k, where k is the number of steps within which the controller must make the game reach the target. While we do not know a matching complexity lower bound, we show in this section that there are games where a region can be split into exponentially many cells, where the controller must play differently from each cell in order to win the game.

Theorem 5 *For every $k \geq 0$, there is an* acyclic *WTA A_k with 3 clocks, where constants in the constraints of A_k are only 0 or 1, where all edges have weight zero and states have weights 0 or 1 and such that the number of states in A_k is bound by a fixed polynomial in k, such that the following holds:*

Let str be any optimal strategy for $(A_k, 3k)$. Then there is a region (q, R) and an exponential number of states $s_1, \ldots, s_{2^{k-1}}$, in (q, R) such that each of these states is visited by some play according to str and for every pair of distinct states s_i and s_j, the discrete components (i.e. the Σ labels) of the set of plays from s_i and from s_j according to str are different.

Let us fix a number k for the rest of this section. The rest of the section is devoted in constructing a timed game A_k that satisfies the requirements of the above theorem.

First, we need to construct two classes of gadgets, C_i^z and B_i', and prove the properties they possess. The next two subsections give details of these two gadgets. Then we construct the WTA A_k. Intuitively, the WTA A_k is a timed game where the adversary picks a number r ($r \leq 1$) and the controller, in order to win, must produce the most accurate binary representation of k bits that approximates the value of $(1 - r)$. Moreover, the way the controller encodes these bits is by making discrete choices in the game, and this results in a region getting split into cells that number exponential in k, one for each binary representation. The adversary continually interacts with the controller during these choices in order to ensure that the binary representation produced by the controller is the most accurate approximation of $(1 - r)$.

The gadget C_i^z

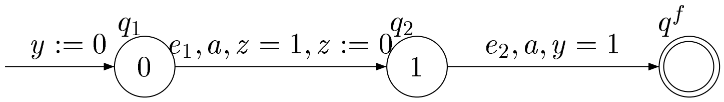


Fig. 4. WTA C_k^z

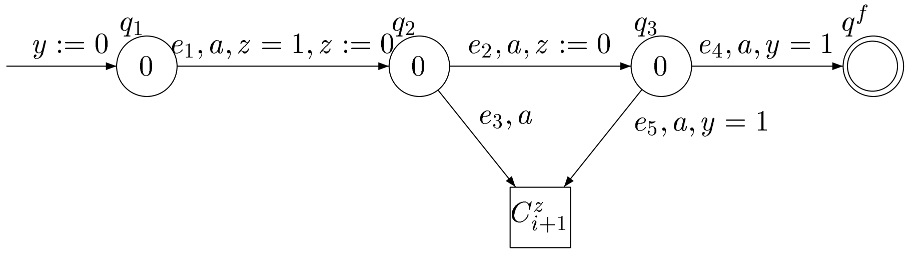


Fig. 5. WTA C_i^z, where $i \in \{1, \ldots, k-1\}$

Let us define a set of weighted timed automata $C_i^z, 1 \leq i \leq k$. The automaton C_k^z is depicted on Figure 4. The weight of node q_1 is 0 and the weight of q_2 is 1. Node q^f is the node the controller needs to reach to win a game. Automaton C_i^z, $1 \leq i \leq k-1$ is constructed from C_{i+1}^z as shown in Figure 5. The weights of q_1, q_2, and q_3 are equal to 0, edges e_3 and e_5 connect nodes q_2, q_3 of C_i^z to node q_1 of C_{i+1}^z. In all $C_i^z, 1 \leq i \leq k$, weights of all edges are 0 and for all nodes the controller has only one action available to it.

We want to show that C_i^z is a game where the controller's optimal strategy will have cost $2^{i-k}\nu_0(z)$, for any $i \in \{1, \ldots, k\}$, where $\nu_0(z)$ is the initial value of the clock z which we assume is in the range $[0,1]$. Hence C_i^z can be used as a gadget to "charge 2^{i-k} to z." Intuitively, in C_{i+1}, the step from q_1 to q_2 sets the value of the clock y to be $1-\nu_0(z)$. Since the edge from q_3 to q^f demands y be 1, the controller needs to split the remaining time $1-(1-\nu_0(z)) = \nu_0(z)$ into two parts determined by when it picks to move from q_2. The adversary can force the game to C_{i+1}^z at q_2 and q_3 and thereby choose to charge $2^{(i-k)+1}$ to either half. The optimal choice for the controller is to split the interval into two equal halves and hence pay a cost of $2^{i-k}\nu_0(z)$.

Lemma 6 *Let $\nu_0(z) \in [0,1]$ be the initial value of the clock z in a game played on the automaton C_i^z, $i = 1,\ldots,k$. Then the optimal cost for the controller to win the game in i steps is $2^{i-k}(\nu_0(z))$. Moreover there is only one optimal strategy that guarantees this cost.*

Proof. For $i = k$ the statement of the lemma is obvious. Suppose that the lemma holds for $i+1$. We want to show that then it holds for i. Notice that $y = 1-\nu_0(z)$ at the moment the game reaches q_2. It is easy to see that if the controller waits at the node q_2 for $\frac{\nu_0(z)}{2}$ units of time and then chooses the only available action a, then it wins within cost $2^{i-k}(\nu_0(z))$. Hence the controller can win within cost $2^{i-k}(\nu_0(z))$. If a controller strategy chooses to move at any other point, then one of the two halves will be greater than $\nu_0(z)/2$, and the adversary can force the play into the gadget C_i^z and force the game to cost more than $2^{i-k}\nu_0(z)$. Hence there is only one optimal strategy and it costs $2^{i-k}\nu_0(z)$. □

The gadget B_i'

Our next building block is WTA B_i' depicted on Figure 6. B_i' is a gadget in which the controller's optimal cost to win is 2^{-k}. This strategy has the property that if the adversary chooses to force the path from q_{in} to q_{exit}, then the *time* taken will be $1-2^{-i}$. In the final WTA we construct, the optimal cost for the WTA will be 2^{-k} and hence using this gadget will ensure that any optimal controller spends time exactly $1-2^{-i}$ from q_{in} to q_{exit}.

Lemma 7 *In a game played on B_i', $i = 1,\ldots,k$, the optimal cost for the controller is $\frac{1}{2^k}$. Also, there is only one such optimal strategy, and when playing according to this strategy the play could go from q_{in} to q_{exit}; in this case the time spent along this play is $1-\frac{1}{2^i}$ and the cost along this path is 0.*

Proof. Suppose that upon entering the block j (i.e. when entering q_{j1}), the valuation of the clock x is $\nu_j(x)$. Then the optimal strategy for the controller is to wait at q_{j1} for exactly $\frac{1-\nu_j(x)}{2}$ time units. If the controller does not do that then the adversary switches either at q_{j1} using the edge e_{j2} or at q_{j2} using e_{j4} incurring cost strictly larger than $\frac{1}{2^k}$. Thus, there is only one optimal strategy

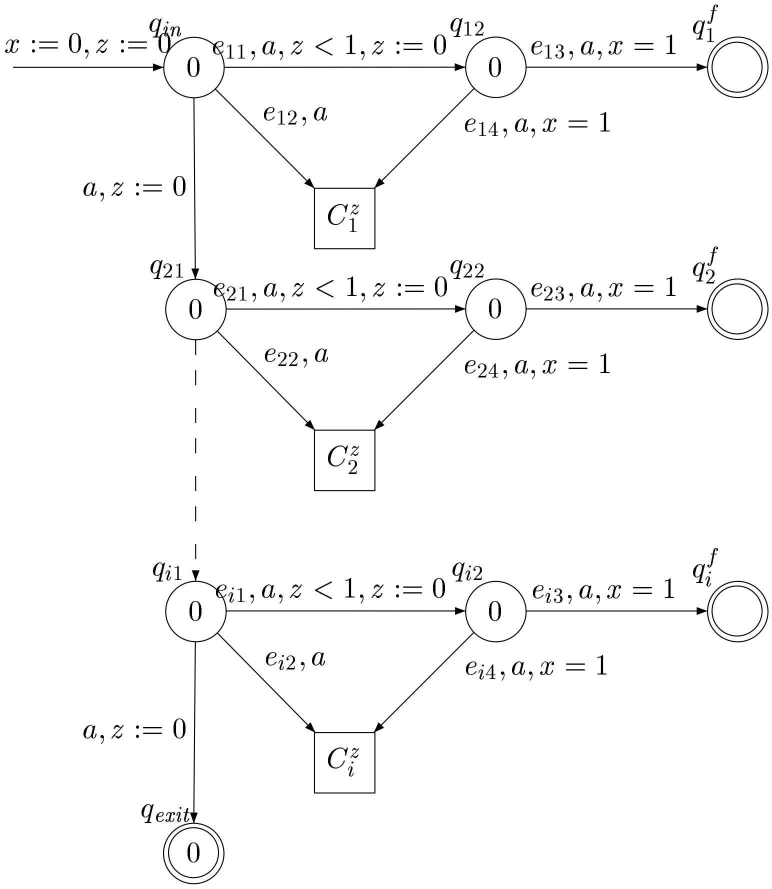


Fig. 6. WTA B'_i

S that guarantees cost less than $\frac{1}{2^k}$. It is easy to see that using S the controller reaches q_{exit} in exactly $\frac{1}{2} + \ldots + \frac{1}{2^i} = 1 - \frac{1}{2^i}$ time units and incurs 0 cost along this play. □

5.0.1 The WTA A_k

Our final construction is the WTA A_k depicted in Figure 7. Gadgets B_i in A_k are identical to B'_i except that they all share the same (non-final) node q_{exit}. We say that a game progresses along the *main line* if the only visited nodes are $q_0, q_1, \ldots, q_{k+1}, q_1^f$.

Intuitively, the adversary picks some time to switch from q_0 to q_1 which is less than or equal to one unit (otherwise the controller can switch at time 1). Let this switch happen after time r, $r \leq 1$. On the main line, the guard on the last transition is $x = 1$. Hence there is $1 - r$ time left and the controller has to pick some way of splitting this into k parts corresponding to the times spent

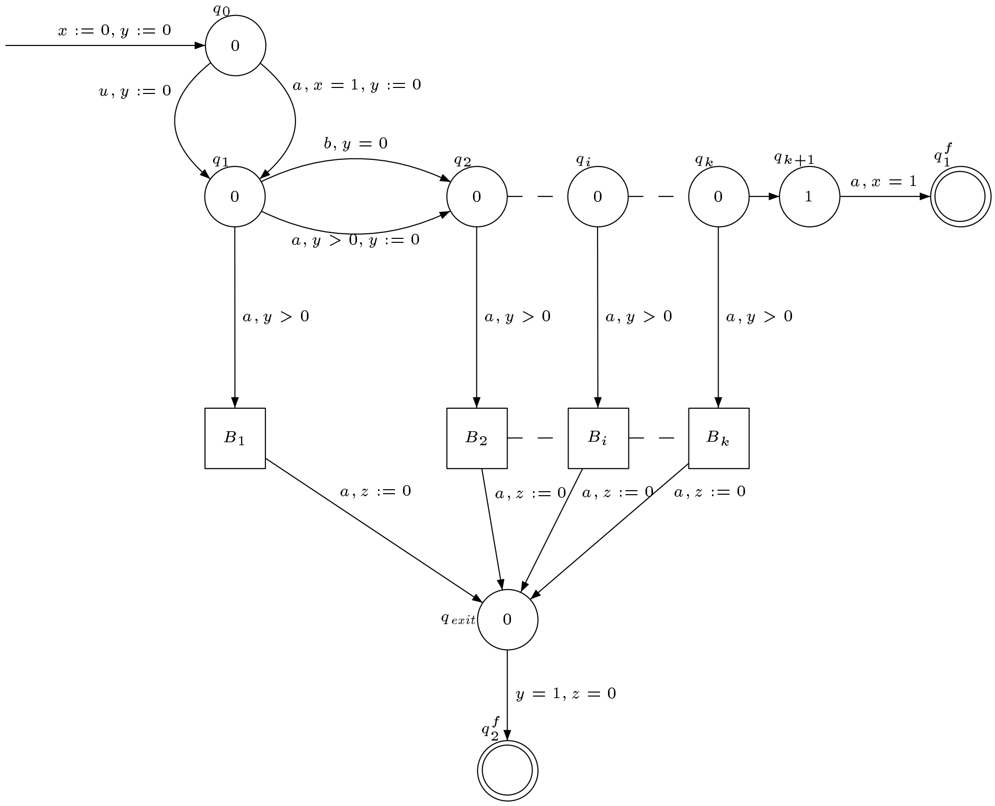


Fig. 7. WTA A_k

at $q_1, \ldots q_k$—let us call these intervals $y_1, \ldots, y_k$.

The idea of the game is to force a controller to pick the y_i's such that it gives an *encoding of the binary representation of* $1-r$. More precisely, we want each y_i to be either 0 or 2^{-i}. Since the sum of the y_i's must add up to $1-r$ (not quite, because one can wait for some small time in q_{k+1} paying cost at rate 1), this will cause the controller to behave differently on two values r and r' of x if $1-r$ and $1-r'$ have different k-bit binary representations. Since the choices for 0 and 2^{-i} are coded using different actions a and b, there will be an exponential number of states in the region corresponding to q_1 and $0 < x < 1$, where the discrete component of the plays from them will be different. We show the theorem using a sequence of lemmas.

Lemma 8 *In a game played on A_k the controller can win within cost $\frac{1}{2^k}$.*

Proof. If a transition between q_0 and q_1 happens at time r, let $s \leq 1-r$ be the largest value such that s can be written as $\sum_{i=1}^{k} \frac{d_i}{2^i}$ where each d_i is 0 or 1. Then $(1-r) - s < \frac{1}{2^k}$.

Take the strategy that plays the moves at times $y_i = \frac{d_i}{2^i}$ $(i = 1, \ldots, k)$ along the main line (choosing actions a or b appropriately), and plays the optimal strategy in the B_i gadgets. On the main line the cost of the strategy is $1 \cdot ($*time spent in* $q_{k+1}) = 1 \cdot (1 - r - s) < \frac{1}{2^k}$. If a game proceeds to B_i for some i then the cost is less or equal to $\frac{1}{2^k}$. If the play exits B_i, then the time elapsed in B_i is $1 - \frac{1}{2^i}$; thus the game reaches q_{exit} when $y = \frac{1}{2^i} + (1 - \frac{1}{2^i}) = 1$ and the controller wins. □

Lemma 9 *Let S be a controller strategy and let S choose to stay in q_i for time $t = \frac{1}{2^i} + \delta$ (where $|\delta| > 0$ and $t > 0$). Then the cost of S is greater than $\frac{1}{2^k}$.*

Proof. If the adversary switches to B_i then the controller has either to pay a cost more than $\frac{1}{2^k}$ in B_i or lose the game in B_i; if it plays the optimal strategy in B_i, the adversary can force it to exit B_i using $1 - \frac{1}{2^i}$ units of time but $\frac{1}{2^i} + \delta + (1 - \frac{1}{2^i}) \neq 1$ and the controller cannot reach q_2^f. □

Corollary 10 *For all $i = 1, \ldots, k$, any optimal strategy for A_k must stay in q_i for either 0 or $\frac{1}{2^i}$ time units. (By Lemmas 8 and 9).* □

Lemma 11 *Let S be the strategy for the controller for a game on A_k. Then cost of $S \geq \frac{1}{2^k}$.*

Proof. If cost of $S \leq \frac{1}{2^k}$, then by Corollary 10, for every i, S must stay in at q_i for either 0 or $\frac{1}{2^i}$ time units.

Consider the play where the u labelled transition between q_0 and q_1 is taken at $x = 1 - \frac{1}{2^k}$. The time left for the game is $\frac{1}{2^k}$. This time may be spent along the main line either at q_k or q_{k+1}. If spent at q_{k+1}, the cost incurred will be $\frac{1}{2^k}$; if spent at q_k, the adversary can force the game to B_k. In both cases cost will be $\frac{1}{2^k}$. Hence cost of $S \geq \frac{1}{2^k}$. □

Corollary 12 *The strategy presented in Lemma 8 is optimal for A_k.* □

Lemma 13 *Let S be an optimal strategy for A_k (We know that the cost of S is $\frac{1}{2^k}$). Let r_1 and r_2 be values such that each r_j is expressible as $\sum \frac{d_i^j}{2^i}$ where $d_i^j \in \{0, 1\}$ and let $|r_1 - r_2| > \frac{1}{2^k}$. Then the discrete labels (i.e. Σ-labels) of the play followed along main line if the adversary plays the u-labelled transition in time $(1 - r_1)$ is different from the discrete labels of the play followed when the adversary plays in time $(1 - r_2)$.*

Proof. The time spent in the penultimate state must be less than or equal to $\frac{1}{2^k}$ for any play. Assume that r_1 and r_2 satisfy the conditions in the lemma and that the discrete labels picked along the main line are the same when the environment picks the u-transition at times r_1 and r_2.

Then, the intervals of time spent at each q_i on the main line must also be the same (by Corollary 10). Since r_j is expressible as $\sum \frac{d_i^j}{2^i}$, the time spent in penultimate state must be $g_j = 0$ or $g_j = \frac{1}{2^k}$. Let y_i^j be the time spent at q_i when the u-transition is played at time r_k, $j = 1, 2$. Then $r_j = \sum y_i^j + g_j$. Since $y_i^1 = y_i^2$ for each i, $|r_1 - r_2| \leq |g_1 - g_2| \leq \frac{1}{2^k}$ which is a contradiction. □

The above lemma proves Theorem 5.

6 Discussion

We have established an exponential upper bound for computing the optimal cost for reachability games in weighted timed automata. Our algorithm can be modified so that it also computes the optimal winning strategy. To do this, we can keep track with every partition cost function for a region, the controller's move for each cell that realizes the appropriate cost.

The complexity of our procedure depends on the number of iterations, the bound on which must be specified by the user. However, the decidability of the more general problem of optimal *unbounded* weighted timed games (where the bound on the number of iterations is not give) is still open. But if we assume that in every cycle of the automaton, a bounded positive cost must be paid (such an assumption is typical to avoid problems with Zeno behaviors), then we can obtain a bound k on the number of iterations and use our procedure to solve the game. One way to compute this bound is as follows: first, we solve the unoptimized timed game problem where the controller's aim is simply to reach the target state, which is easy to solve. Then, we take the synthesized strategy and compute its cost, which gives an upper bound on the cost of reaching the target. Now, given that each cycle of the cost must incur a bounded cost, we can obtain a bound on the number of iterations.

In a recent work [BCFL04a], the authors explore solving the optimal timed game problem by reduction to games on linear hybrid automata games (without cost) and describe techniques to implement the solution using the tool HyTech [BCFL04b].

Finally, though we have shown that exponential splitting of a region is necessary for representing the optimal cost as a function of the initial state, the precise lower bound on the complexity of the decision problem remains open.

References

[ACH97] R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design*, 11(2):137–155, 1997.

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[ALP01] R. Alur, S. La Torre, and G. Pappas. Optimal paths in weighted timed automata. In *Hybrid Systems: Computation and Control, Fourth International Workshop*, LNCS 2034, pages 49–62, 2001.

[AM99] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Proceedings of the 2nd International Workshop on Hybrid Systems: Computation and Control*, LNCS 1569, pages 19 – 30. Springer, 1999.

[BCFL04a] P. Bouyer, F. Cassez, E. Fleury, and K.G. Larsen. Optimal strategies in priced timed game automata. Technical Report RS-04-4, BRICS, Denmark, 2004. Available at `http://www.brics.dk/RS/04/4/`.

[BCFL04b] P. Bouyer, F. Cassez, E. Fleury, and K.G. Larsen. Synthesis of optimal strategies using HyTech. In *Proceedings of First Workshop on Games in Design and Verification*, 2004. To appear.

[BHF$^+$01] G. Behrman, T. Hune, A. Fehnker, K. Larsen, P. Petersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *Hybrid Systems: Computation and Control, Fourth International Workshop*, LNCS 2034, pages 147–161, 2001.

[CHR02] F. Cassez, T.A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *Hybrid Systems: Computation and Control, Fifth International Workshop*, LNCS 2289, pages 134–148, 2002.

[CY91] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proceedings of the Third Workshop on Computer-Aided Verification*, LNCS 575, pages 399–409. Springer-Verlag, 1991.

[DM02] D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proceedings of the 19th Symposium on Theoretical Aspects of Computer Science*, LNCS 2285, pages 571–582. Springer, 2002.

[FR74] M.J. Fischer and M.O. Rabin. Super-exponential complexity of presburger arithmetic. In *Proceedings of SIAM-AMS Symposium in Applied Mathematics*, volume 7, pages 27–41, 1974.

[FR75] J. Ferrante and C. Rackoff. A decision procedure for the first order theory on real addition with order. *SIAM Journal of Computing*, 4(1):69–76, 1975.

[LBB+01] K. Larsen, G. Behrman, E. Brinksma, A. Fehnker, T. Hune, P. Petersson, and J. Romijn. As cheap as possible: Efficient cost-optimal reachability for priced timed automata. In *Proc. of the 13th International Conference on Computer Aided Verification*, LNCS 2102, pages 493–505. Springer, 2001.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry.* Springer, 2002.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proceedings of the 12th Annual Symposium on Theoretical Aspects of Computer Science*, LNCS 900, pages 229 – 242, 1995.

[TMM02] S. La Torre, S. Mukhopadhyay, and A. Murano. Optimal-reachability and control for acyclic weighted timed automata. In *Proceedings of the 17th IFIP World Computer Congress: TCS*, pages 485–497. Kluwer, 2002.

[WH91] H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *Proceedings of the 30th IEEE Conference on Decision and Control*, pages 1527–1528, 1991.