Optimal Reachability for Weighted Timed Games*

Rajeev Alur, Mikhail Bernadsky, and 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 the 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.

1 Introduction

Timed automata [2] 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 can a target set 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 many researchers (cf. [7, 1, 3, 11]). 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 [3]. An alternative branch-and-bound solution is implemented in the verification tool UPPAAL with applications to scheduling problems [11, 5].

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

* This research was partially supported by ARO URI award DAAD19-01-1-0473, and NSF awards ITR/SY 0121431 and CCR 0306382.

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 *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 (cf. [15, 12, 8, 6]). Time-optimal games, that is, games in which the cost of a run equals the total time spent, are considered in [4], 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 [14], and the authors show that the problem can be encoded using first-order theory of reals with addition [9], leading to a doubly-exponential solution (note that the first-order theory over reals with addition is not decidable in nondeterministic exponential time [10]).

In this paper, we provide an exponential-time solution to the 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 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. If f_1 and f_2 are functions with n pieces, then the *min* or *max* operation could result in a function which 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, 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 f_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.

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. 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.*

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 $(q, \nu) \xrightarrow{e} (q', \nu')$. 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 game 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 the more general *uniform* timed game problem, where we 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).

3 Optimal Cost Functions

Regions Let us fix a WTA $A = (Q, Q_F, X, \Sigma, u, \delta, Inv, W_Q, W_\delta)$ 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.

The notion of a *clock region* is standard in the theory of timed automata and is used to partition the clock space into partitions with respect to a timed bisimilar relation. Due to lack of space, we assume this notion and the notion of a *timed successor* of a region (see [2]).

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 successor of R_i, then $l = O(d \cdot \beta_{max})$.

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 an automaton A.

Let $\mathit{Enabled}_\delta(q, R)$ denote the set of all transitions enabled at some (and hence all) states (q, ν) in (q, R). Let $\mathit{Enabled}_\Sigma(q, R)$ denote the set of actions enabled at some (and hence all) states (q, ν) in (q, R). If $e \in \mathit{Enabled}_\delta(q, R)$, let $\mathit{succ}((q, R), e)$ 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 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 $\beta - x$ as the *critical clock expression* for the region R and denote it as $\mathit{cce}(R)$. 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 ∞.

Expressions for the Optimal Cost: We now wish to define a set of functions $\mathit{opt}_i^{(q,R)} : R \rightarrow \mathbb{R}_+ \cup \{\infty\}$ 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 $\nu \in R$, $\mathit{opt}_i^{(q,R)}(\nu)$ 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 1 below).

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

- For every location $q \in Q_F$, $R \in \mathcal{R}$, $\mathit{opt}_0^{(q,R)} = 0$ and for every location $q \notin Q_F$, $R \in \mathcal{R}$, $\mathit{opt}_0^{(q,R)} = \infty$.
- 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 $T = \mathit{cce}(R')$ be the critical clock expression of R'.
  If R' has a timed-successor, let it be R''.

Let $A = \mathit{Enabled}_\Sigma(q, R) \cap \Sigma'$ be the controller actions enabled at (q, R) and for any $a \in \Sigma$, let $\mathit{Ev}(a) = \{e \mid e \in \mathit{Enabled}_\delta(q, R), \mathit{Action}(e) = a\}$ be the a-labeled transitions enabled at (q, R). Similarly, let $A' = \mathit{Enabled}_\Sigma(q, R') \cap \Sigma'$, and $\mathit{Ev}'(a) = \{e \mid e \in \mathit{Enabled}_\delta(q, R'), \mathit{Action}(e) = a\}$. Then

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

where

$$h_1(\nu) = \min_{a \in A} \max_{e \in \mathit{Ev}(a) \cup \mathit{Ev}(u)} \{\mathit{opt}_{i-1}^{\mathit{succ}((q,R),e)}(\nu/\mathit{reset}(e)) + W_\delta(e)\}$$

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

$$h_3(\nu) = \max\{g_1(\nu, T), \quad W_Q(q) \cdot T + \mathit{opt}_i^{(q,R'')}(\nu + T)\}$$

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

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

The following is not hard to prove:

Lemma 1. *Let A be a weighted timed automaton, $k \in \mathbb{N}$, and s be a state in the region (q, R). Then,*

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

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

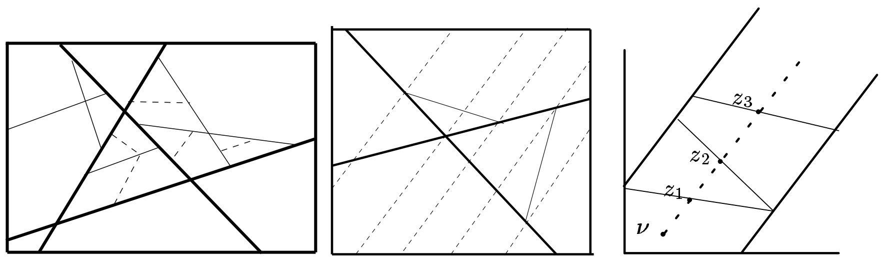
$$f(\bar{x}) = \inf_{t \in [0,T]} \max \begin{cases} f_1(\bar{x} + t) + wt \\ \sup_{0 \leq t' \leq t}(f_2(\bar{x} + t') + wt') \end{cases} \tag{1}$$

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

4 The Algorithm

4.1 Motivation of Definitions

In this section, we describe briefly the main difficulties that arise in showing that each function $\mathit{opt}_i^{(q,R)}$ is a piece-wise linear function with at most an exponential number of pieces and informally describe the ideas to circumvent these. The next section gives a formal but terse summary of the required technical results.



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

Fig. 1.

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. 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 1(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 1(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 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 1(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 in 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 1(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 1(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' over possibly infinite sets to a finite set; this leads 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.

4.2 Clock Space Geometry

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. It is well known that $\mathit{Subcells}(c, H)$ contains $O(|H|^d)$ cells (in fact $3|H|^d$ cells; see [13]).

Definition 2. *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 $D_P = \eta(\mathit{root})$ 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 $P_1 \oplus P_2 \prec \langle n_1^1 + n_1^2, \ldots, n_k^1 + n_k^2 \rangle$.

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. 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.

Let us now consider operations on partition cost functions and the change in complexity that the operations result in. For any partition cost function (P, F), let us denote the function it represents as $\hat{F}$.

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

The above theorem also holds for the max function.[1]

Theorem 2. *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 $\langle l_1, \dots l_{k_1}, n_1, \dots n_{k_2} \rangle$. Consider the function:*

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

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 $\langle 2l_1, \dots, 2l_{k_1}, (k_2 + 1)3^{k_2+1}(2n_1)^{d+1} \cdot \ldots \cdot (2n_{k_2})^{d+1}, (2n_1 + \ldots + 2n_{k_2} + 3)^2, 2n_1, \dots, 2n_{k_2}, 7 \rangle$ and $\hat{K} = S$.

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

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. 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 now want to "drop diagonal hyperplanes" (as in Figure 1(b)) from each point of intersection of hyperplanes that define the nested partition.

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:

[1] 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).

Lemma 2. *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 $k \cdot 3^k \cdot n_1^{d+1} \cdot \ldots \cdot n_k^{d+1}$.*

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 2 we can show the following:

Lemma 3. *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 $P' \prec \langle l_1, \dots l_{k_1}, k_2 3^{k_2} n_1^{d+1} \dots n_{k_2}^{d+1}, n_1, \dots, n_{k_2} \rangle$.*

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' 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.

4.3 The Main Results

Theorem 3. *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.*

We can also show that an optimal strategy may have to "split" a region into exponentially many parts:

Theorem 4. *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 $s_1, \dots, s_{2^{k-1}}$, 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.

5 Discussion

We have established an exponential upper bound on computing the optimal cost for reachability games in weighted timed automata. The complexity of our procedure depends on the number of iterations. A bound on the number of iterations can be specified by the user, or can be obtained from the automaton if we assume that in every cycle a positive cost must be paid (such an assumption is typical to avoid problems with Zeno behaviors). However, the complexity (and even decidability) of the problem in the absence of such an assumption is open. Also, 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

1. 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.
2. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
3. R. Alur, S. La Torre, and G. Pappas. Optimal paths in weighted timed automata. In *Hybrid Systems: Computation and Control*, LNCS 2034, pages 49–62, 2001.
4. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Hybrid Systems: Comp. and Control*, LNCS 1569, pages 19–30, 1999.
5. 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*, LNCS 2034, pages 147–161, 2001.
6. F. Cassez, T.A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *HSCC*, LNCS 2289, pages 134–148, 2002.
7. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proc. of Third Workshop on Computer-Aided Verification*, LNCS 575, pages 399–409. Springer-Verlag, 1991.
8. D. D'Souza and P. Madhusudan. Timed control synthesis for external specifications. In *Proc. STACS*, LNCS 2285, pages 571–582. Springer, 2002.
9. 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.
10. M.J. Fischer and M.O. Rabin. Super-exponential complexity of Presburger arithmetic. In *Proc. of SIAM-AMS Symp. in Appl. Math.*, vol. 7, pages 27–41, 1974.
11. 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 CAV*, LNCS 2102, pages 493–505. Springer, 2001.
12. 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.
13. J. Matoušek. *Lectures on Discrete Geometry.* Springer, 2002.
14. 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.
15. H. Wong-Toi and G. Hoffmann. The control of dense real-time discrete event systems. In *IEEE Conference on Decision and Control*, pages 1527–1528, 1991.