Optimal Paths in Weighted Timed Automata*

Rajeev Alur[1,2], Salvatore La Torre[1,3], and George J. Pappas[1]

[1]University of Pennsylvania
[2]Bell Labs
[3]Università degli Studi di Salerno
e-mail: {alur,pappasg}@cis.upenn.edu, latorre@seas.upenn.edu

Abstract. We consider an *optimal-reachability problem* for a timed automaton with respect to a linear cost function which results in a *weighted timed automaton*. Our solution to this optimization problem consists of reducing it to a (parametric) shortest-path problem in a finite directed graph. The directed graph we construct is a refinement of the region automaton due to Alur and Dill. We present an exponential time algorithm to solve the shortest-path problem for weighted timed automata starting from a single state, and a doubly-exponential time algorithm to solve this problem starting from a zone of the state space.

1 Introduction

Timed automata [AD94] are widely accepted as formal models of real-time systems: a discrete transition graph is equipped with a finite set of *clock variables* so expressing *timing constraints* becomes possible. Automated analysis of timed automata relies on the construction of a finite quotient of the infinite space of clock valuations. In particular, by this construction, the decidability of the *reachability* problem, and hence of the verification of *safety* properties, in timed automata is obtained. Given two states s and t of a timed automaton A, the reachability problem can be stated as the problem of determining if there exists a run of A from s to t.

In the theory of timed automata there are many decision problems which are undecidable, and decidability is hard in general. In this paper we are interested in an optimal-reachability problem. Time-optimal reachability was first considered in [CY91], where the problem of computing lower and upper bounds on time delays in timed automata was solved. Minimal-time reachability is also considered in [NTY00]. In [ACH93], a weight w is associated to each location q such that w gives the cost of a unit of time spent in q. Then given a cost interval I and two states s and t, the problem of determining if t is reachable from s at a cost in I is solved. Here we solve a more general optimal-reachability problem, that has been independently solved also in [BHF$^+$]. We consider *weighted timed*

* This work is partially supported by the DARPA/ITO MoBIES grant F33615-00-C-1707, the NSF Career award CCR97-34115, the SRC award 99-TJ-688, the MURST grant TOSCA, the DARPA JFACC grant N66001-99-C-8510, and the University of Pennsylvania Research Foundation.

automata, that is timed automata with weights (different costs) on both locations and transitions. The cost of a path is the sum of costs of the taken switches plus the sum of the costs associated with the visited locations multiplied for the time spent in each of them. Our optimization problem, which we call *optimal-run problem*, can be formalized as a tuple containing a weighted timed automaton, a *source zone* and a *target zone*. If the source zone contains only a state of the automaton, we refer to this problem as the *single-source* optimal-run problem.

Our solution to the optimal-run problem consists of two main steps: first we reduce the optimal-run problem to a shortest-path problem in directed graphs, then we solve it. The first step is obtained by constructing a finite graph which is a refinement of the region automaton [AD94]. Each clock region is split into several disjoint subregions relatively to a starting state and to sequences of resets that may occur in "potential" optimal runs. This construction is parameterized on the differences of two consecutive fractional parts from the clock valuation of the starting state. When we consider a general source zone, we leave unspecified these parameters and the above construction reduces the optimal-run problem for weighted timed automata to a parametric shortest-path problem in directed graphs. We give a fix-point computation algorithm to solve this problem, so obtaining a doubly-exponential time algorithm to solve the optimal-run problem. In case the input automaton has only one clock variable, this result can be improved to a single exponential by adapting to our case the algorithm given in [KO81,YTO91] for solving a particular case of parametric shortest-path problem. In case the source zone is a singleton we substitutes the parameters with the actual values from the starting state, and thus our optimization problem is reduced to a standard shortest-path problem. Solving this problem by Dijkstra's algorithm, we obtain an exponential time algorithm to solve the single-source optimal-run problem.

The optimal-reachability problem is strictly related to other decision problems. Among them, the problem of synthesizing an optimal controller is certainly one of the most interesting. The *optimal-control synthesis problem* can be informally stated as the problem of designing a control which is able to drive, at a minimum cost, a given system into a given target zone. In the literature, control synthesis problems have been considered in the context of discrete automata [Chu62,Tho95], timed automata [AMP95,MPS95,AM99], linear hybrid automata [WT97], and general hybrid systems [LTS99,SPS00]. The design of an optimal control for hybrid systems is not trivial and in general is undecidable. Our approach can be applied to solve the optimal-control synthesis problem for weighted timed automata, so generalizing the results obtained in [AM99] on the synthesis of a time-optimal controller for a timed automaton.

The rest of the paper is organized as follows. In section 2, we define the optimal-run problems and we give some examples. In section 3, we introduce a graph construction to reduce the optimal-run problems to the corresponding shortest-path problems in directed graphs. In section 4, we present our solutions to the single-source optimal-run problem and to the general case.

2 Preliminaries

In this section we define the single-source and the parametric optimal-run problems. We start introducing some notation and the definition of timed automaton.

Given a set C of n variables, a k-zone is a subset of $\mathbb{R}^n$ that can be obtained as a boolean combination of inequalities of the form $x \leq y + c$, $x < y + c$, $x \leq c$, and $x < c$ where $x, y \in C$ and $c \in \{0, 1, \ldots, k\}$. We denote by TRUE the clock constraint which is true for any clock values. We denote by $Z(C)$ the set of all the k-zones, for all $k \in \mathbb{N}$. A function $\lambda : \mathbb{R}^n \longrightarrow \mathbb{R}^n$ is called a reset function if it is equal to the identity on some of the coordinates and zero on the others. We denote by Λ_n the set of all reset functions over $\mathbb{R}^n$. A *timed automaton*[1] A is a tuple $(Q, C, \Delta, \mathrm{Inv})$ where:

- Q is a finite set of locations;
- C is a finite set of n clock variables;
- Δ is a finite subset of $Q \times Z(C) \times \Lambda_n \times Q$;
- $\mathrm{Inv} : Q \longrightarrow Z(C)$ maps each location q to its invariant $\mathrm{Inv}(q)$.

A state is a tuple (q, ν) where $q \in Q$ and $\nu \in \mathbb{R}^n$. We denote by $S = Q \times \mathbb{R}^n$ the set of states for A. A *discrete step* is $(q, \nu) \xrightarrow[e]{} (q', \nu')$ where $e = (q, \delta, \lambda, q') \in \Delta$, ν satisfies δ, $\nu' = \lambda(\nu)$, and ν' satisfies $\mathrm{Inv}(q')$. A *time step* is $(q, \nu) \xrightarrow{t} (q, \nu')$ where $\nu' = \nu + t$, $t \geq 0$, and $\nu + t'$ satisfies $\mathrm{Inv}(q)$ for all $0 \leq t' \leq t$. A *step* is $(q, \nu) \xrightarrow[e]{t} (q', \nu')$ where $(q, \nu) \xrightarrow{t} (q, \nu'')$ and $(q, \nu'') \xrightarrow[e]{} (q', \nu')$, for some $\nu'' \in \mathbb{R}^n$, that is a transition e taken after spending some time t in the current location. A run r of a timed automaton A is a finite sequence $(q_0, \nu_0) \xrightarrow[e_1]{t_1} (q_1, \nu_1) \xrightarrow[e_2]{t_2} \ldots \xrightarrow[e_{k-1}]{t_{k-1}} (q_{k-1}, \nu_{k-1}) \xrightarrow{t_k} (q_k, \nu_k)$. We say that r starts at (q_0, ν_0) and ends at (q_k, ν_k). The definition of r allows time to be spent after taking the last transition e_{k-1}. A *weighted timed automaton* is a timed automaton A with the following cost functions:

- $J_s : \Delta \longrightarrow \mathbb{N}$ (*switch cost*), and
- $J_d : Q \longrightarrow \mathbb{N}$ (*duration cost*).

Given a run r of A and cost functions J_s, and J_d, we associate costs to r as follows:

- $J_s(r) = \sum_{i=1}^{k} J_s(e_i)$, and
- $J_d(r) = \sum_{i=0}^{k-1} t_i \cdot J_d(q_i)$.

The total cost associated to a run r is then $J(r) = J_s(r) + J_d(r)$. We are interested in determining optimal-cost runs for a timed automaton. In the following examples we informally introduce some notions that we will formalize in the rest of the section.

[1] The standard definition of timed automata requires also an acceptance condition and a symbol alphabet. Since we are not interested in studying languages accepted by timed automata we omit these features here.

Example 1. Consider the timed automaton defined in Figure 1 such that $J_d(0) = 3$, $J_d(1) = 1$, and the switch costs are all 1. Suppose that we start from state $s = (0, x, y)$ for $0 \leq x, y < 2$ and we want to reach a state in location 2. Possible minimal-cost runs from s to a state $s' = (2, x', y')$ are either $r_1 = (0, x, y) \xrightarrow[e_1]{t_1} (1, x_1, y_1) \xrightarrow[e_3]{t_2} (2, x + 2 - y, 2)$, or $r_2 = (0, x, y) \xrightarrow[e_2]{t_3} (2, 2, y + 2 - x)$ for $t_3 = (2 - x)$ (obviously, staying in location 2 longer might only increase the overall cost). According to the cost function J, the cost of r_1 is $J_s(r_1) + J_d(r_1) = 2+3t_1+(2-y-t_1) = 4-y+2t_1$ and the cost of r_2 is $J_s(r_2)+J_d(r_2) = 1+3(2-x) = 7-3x$. Clearly, $J(r_1)$ is minimized when $t_1 = 0$, that is the transition from 0 to 1 is taken immediately. Moreover, assuming $t_1 = 0$, $J(r_1) \leq J(r_2)$ if $y \geq 3(x - 1)$, and $J(r_1) > J(r_2)$, otherwise. Thus, a minimal-cost run from s to a state in location 2 depends on the clock valuation of state s.

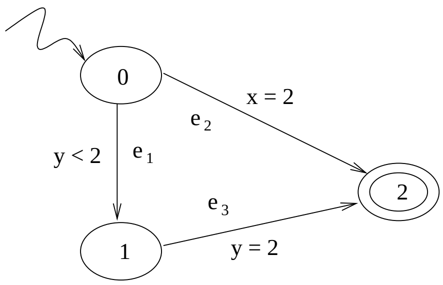

Fig. 1. A timed automaton with more than an optimal run from a same location.

Example 2. Consider the timed automaton defined in Figure 2 such that $J_d(0) = 1$, $J_d(1) = 2$, and the switch costs are all 1. Suppose that we start from state $s = (0, x)$ for $0 \leq x < 2$ and we want to reach a state in location 2. Possible minimal-cost runs from s to a state $s' = (2, x')$ are given by $r_t = (0, x) \xrightarrow[e_1]{t} (1, x_1) \xrightarrow[e_2]{t'} (2, 2)$. Notice that r_t is a run parameterized by t, where t is the time at which the first edge is taken. Thus $J(r_t) = J_s(r_t) + J_d(r_t) = 2+t+2(2-t-x) = 6-t-2x$. Hence the cost of r_t is minimized if t is maximized. Since $t < (2 - x)$ must hold, the optimal cost for a run starting at s is $(4 - x)$, but none of the runs starting at s has such a cost. In fact, for any actual run r_t there exists a $\xi > 0$ such that $t = (2-x-\xi)$, and $J(r_t) = (4-x+\xi)$. Vice-versa, for any $\xi > 0$ there exists a run r such that $J(r) = (4 - x + \xi)$. Clearly, there is not a minimal-cost run but we can determine a run whose cost is arbitrarily close to the optimal one.

Now we formalize the notion of optimal cost, optimal run, and approximation of an optimal run. Given a timed automaton A, a state s, and a target zone T, an *optimal cost* for a run from s to T is a J^* such that $J^* \leq J(r)$ for any run r from s to a state in T, and for any $\xi > 0$ there is a run r such that $J(r) \leq J^* + \xi$.

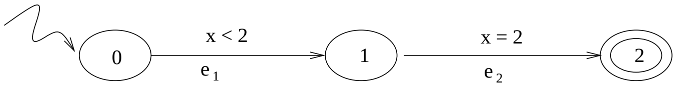

Fig. 2. A timed automaton with no optimal runs from a location.

If there exists a run r^* such that $J(r^*) = J^*$, then r^* is said to be an *optimal* run. As shown in Example 2, sometimes an optimal run from a state s to a target zone T does not exist. In these cases, we are interested in a family R of runs such that all the runs coincide on the sequence of switches and for any $\xi \in \mathbb{R}_+$ there exists a run $r \in R$ such that $J(r) < J^* + \xi$, where J^* is the optimal cost over all runs from s to T. That is we can determine a sequence of runs in R whose costs are arbitrarily close to J^*. We call such a family of runs R an *approximation* of an optimal run. Given a timed automaton A, a source zone S, and a target zone T, we consider the problem of determining an optimal run from a given state $s \in S$ to T, if one exists, or an approximation of an optimal run, otherwise. We call this problem a *single-source optimal-run problem*. We also consider a more general problem, a *zone optimal-run problem*, defined as the problem of determining a symbolic representation of the solution to the single-source optimal-run problem for all states in S. In Example 1, if we consider as target region all the states in location 2 and as only source state $(0, 0, 0)$, then a solution to the corresponding instance of the single-source optimal-run problem is r_1 with $t_1 = 0$. As observed in Example 1, if we consider as source zone the set of states $(0, x, y)$ such that $0 \leq x, y \leq 1$, then the solution of the corresponding instance of the zone optimal-run problem is r_1 with $t_1 = 0$ if $y \geq 3(x - 1)$, and r_2, otherwise.

We end this section with an example on an air-traffic control problem that we will use subsequently in the paper.

Example 3. Consider the timed automaton in Figure 3. It models a scenario in which two aircraft send a landing request to an airport, and our goal is to allow both the aircraft to land safely and at minimum cost. Safety requires that only one aircraft at a time must be acknowledged for landing, thus there are two possible choices: aircraft 1 waits for the landing of aircraft 2 to be completed, or vice-versa. There are costs c_1 and c_2 to pay for forcing respectively aircraft 1 and aircraft 2 to wait. Moreover, there is also a cost, expressed by w_i, which is related to the time spent waiting. Alternatively, aircraft i can make, at a cost c'_i, a maneuver that allows to spend w'_i instead of w_i per each time unit. This maneuver takes at least time 1. Since it is realistic to reduce the time a runway stays unused, we penalize this event by a cost c_0 per time unit. Finally, we assume that the landing of each aircraft takes at least time 1 since the related acknowledgement was issued by the control tower.

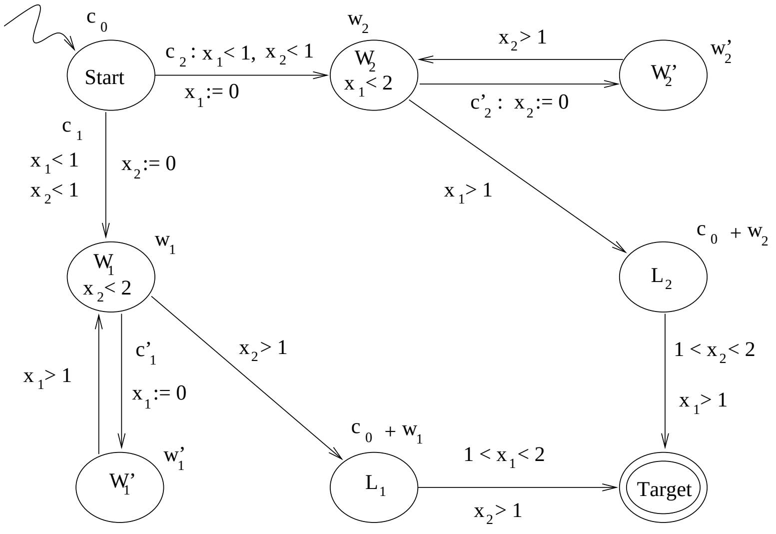

Fig. 3. An air-traffic control problem.

3 The graph construction

In this section we give the graph construction underlying the reduction of the single-source optimal-run problem to the shortest-path problem and the zone optimal-run problem to a parametric shortest-path problem. The obtained graph is a refinement of the region automaton [AD94] of a timed automaton, in the sense that each vertex v carries more information than a region. This additional information mainly concerns the sequence of resets needed to reach v from a starting vertex, and the construction preserves the transitions of the region automaton. Via this construction we emphasize the states of the timed automaton that might be visited in some optimal runs. We start by recalling the concepts of labelled directed graph and region automaton, then we describe our graph construction.

Let Θ be a set of real-valued parameters, we denote by D the set of linear expressions over Θ. Given an alphabet Σ, a *D-labelled directed graph* G is a pair (V, E), where V is a set of vertices, and $E \subseteq V \times D \times V$ is a set of D-labelled edges. A path π from v_0 to v_n in G is a sequence $v_0 \xrightarrow{f_1} v_1 \xrightarrow{f_2} \ldots \xrightarrow{f_{n-1}} v_{n-1} \xrightarrow{f_n} v_n$ such that $v_{i-1} \xrightarrow{f_{i-1}} v_i \in E$ for $i = 1, \ldots, n$. For a path π, the cost of π is given by $\sum_{i=1}^{n} f_i$. A path π from v to v' is a *shortest path* if π is the path with minimum cost among those connecting v to v'. Notice that varying the values of parameters in Θ the shortest path of a graph may change, that is to different valuations of parameters may correspond different sets of shortest paths in the graph.

Consider now a timed automaton A. By definition its set of states is infinite. However, they can be partitioned in a finite number of equivalence classes, called *regions*, which are defined by a location and a *clock region*. Denoted by c_x the largest constant in clock constraints involving the clock variable x, a clock region is described by:

- a constraint of type $c-1 < x < c$, $x > c_x$, or $x = c$ for each clock variable x and $c \leq c_x$;
- the ordering of the fractional parts of the clock variables x such that $x < c_x$.

Thus a clock region denotes a set of clock valuations. Given a clock valuation ν, $[\nu]$ denotes the clock region containing ν. A state (q, ν) belongs to a region $\langle q', \alpha \rangle$ if $q = q'$ and $\nu \in \alpha$. A clock region α is said to be *open* if for any clock variable x and $c \leq c_x$, $x = c$ does not hold in α. Otherwise α is said to be a *boundary* clock region. These definitions apply to regions in an obvious way. The key property of this equivalence, is that all the valuations belonging to a region satisfy the same set of clock constraints from the given timed automaton. Consistently we say that a clock region α satisfies a constraint δ if ν satisfies δ for any $\nu \in \alpha$. A clock region α' is said to be a *time-successor* of a clock region α if and only if for any $\nu \in \alpha$ there is a $d \in \Re_+$ such that $\nu + d \in \alpha'$. The *region automaton* of A is a transition system defined by:

- the set of states $R(S) = \{\langle q, \alpha \rangle \mid q \in Q$ and α is a clock region for $A\}$;
- the transition rules $R(\Delta)$ such that: $(\langle q, \alpha \rangle, \langle q', \alpha' \rangle) \in R(\Delta)$ if and only if $(q, \lambda, \delta, q') \in \Delta$ and there is a time-successor α'' of α such that α'' satisfies δ and $\alpha' = [\lambda \rightarrow 0]\alpha''$.

We denote the region automaton corresponding to A as $R(A)$. For the sake of simplicity, in the following when no confusion can arise we refer to the value of a clock variable x by x itself. With $\overline{x}$ we denote the fractional part of a clock variable x. Let $s = (q, \nu)$ be a state of A and $(0 \approx_1 \overline{x}'_1 \approx_2 \ldots \approx_N \overline{x}'_N \approx_{N+1} 1)$ be the ordering of the fractional parts of the region containing a clock valuation ν (notice that $\approx_i$ is either $=$ or $<$). With $\vartheta(s) = (\vartheta_1, \ldots, \vartheta_{N+1})$ we denote the differences between consecutive values in the above ordering, that is $\vartheta_1 = \overline{x}'_1$, $\vartheta_{N+1} = 1 - \overline{x}'_N$, and $\vartheta_i = \overline{x}'_i - \overline{x}'_{i-1}$ for $i = 2, \ldots, N$. In the following we will use $(\vartheta_1, \ldots, \vartheta_{N+1})$ to denote these differences in the starting state. The graph we are going to define is parameterized over $(\vartheta_1, \ldots, \vartheta_{N+1})$. Moreover, for $i, j \leq N$, we denote by $I(i,j)$ the set of integers $\{i, \ldots, j-1\}$, if $i < j$, and $\{i, \ldots, N\} \cup \{1, \ldots, j-1\}$, otherwise.

The region automaton does not carry enough information to solve our optimization problems. Thus we define a labelled directed graph whose vertices correspond to "sub-states" of the region automaton. For a given state $\langle q, \alpha' \rangle$ of the region automaton, a sub-state $\langle q, \alpha \rangle$ is such that α is a convex region contained in α'. Denoted by $(0 \approx_1 \overline{x}'_1 \approx_2 \ldots \approx_h \overline{x}'_h \approx_{h+1} 1)$ the ordering of the fractional parts in a clock region α', we consider sub-regions α of α' such that for some of the $\approx_i$'s which are equal to $<$, the difference between $\overline{x}'_{i-1}$ and $\overline{x}'_i$ is very close to 0. Thus we represent α by α' and specifying in the ordering of the fractional parts if a $<$ is relative to a "small" difference (denoted by $\lesssim$) or to a "large" difference (denoted by $<$). We call each such sub-region α a *boundary sub-region*. Intuitively, the reason we are interested in boundary sub-regions is that the cost functions we consider are linear, and their infimum over a given region is reached on the boundary. Thus optimal runs leave open regions from states which are arbitrarily close to their boundaries. As a consequence optimal runs visit also states characterized by having clocks values either with arbitrarily close fractional parts or with fractional parts which reflects the starting state and the reset history of the computation. For this reason, we add to each boundary sub-region a tuple of indices $(i_1, \ldots, i_k)$ from $\{1, \ldots, n+1\}$ such that: k is the number of large differences in the ordering of the fractional parts, i_l corresponds to the l-th large difference in the ordering of the fractional parts, and there exists a $d \in \{1, \ldots, k\}$ such that $i_{d+h} < i_{d+h+1}$ for $h = 0, \ldots, k-1$, where the sums $(d+h+1)$ and $(d+h)$ are modulo k. We call such tuples *distance tuples*, since they are used to store the difference between two consecutive fractional parts when this difference is "large" (i.e., they are not arbitrarily close). We define the set of vertices V as the set of tuples $\langle q, \alpha, (i_1, \ldots, i_k)\rangle$ where q is a location, α is a boundary sub-region, and $(i_1, \ldots, i_k)$ is a distance tuple from $\{1, \ldots, n+1\}$. For a vertex $\langle q, \alpha, (i_1, \ldots, i_k)\rangle$, the sum $\sum_{l \in I(i_k, i_1)} \vartheta_l$ gives the time to leave the region since this subregion is entered.

The set of edges E contains three types of edges: *immediate switches*, *time edges* and *delayed switches*. Informally, immediate switches correspond to transitions taken in the current state, time edges correspond to letting time elapse until the next region is reached, and delayed switches correspond to transitions taken at the "beginning" or at the "end" of the closest open region (this region if it is an open region, the next otherwise).

Given two vertices $v = \langle q, \alpha, (i_1, \ldots, i_h)\rangle$ and $v' = \langle q', \beta, (j_1, \ldots, j_k)\rangle$), there is an immediate switch $v \xrightarrow{J_s(e)} v'$ if there exists a transition e of $R(A)$ from $\langle q, \alpha'\rangle$ to $\langle q', \beta'\rangle$, where α' and β' are respectively the regions of $R(A)$ containing α and β, and the sequence $(j_1, \ldots, j_k)$ is obtained from $(i_1, \ldots, i_h)$ by deleting all the indices i_l such that all the clocks between the l-th and the $(l+1)$-th large differences (in the ordering of the fractional parts of α') are reset in e.

Consider a vertex $v = \langle q, \alpha, (i_1, \ldots, i_h)\rangle$ and let $(0 \approx_1 \overline{y}_1 \approx_2 \ldots \approx_k \overline{y}_k \approx_{k+1} 1)$ be the ordering of the fractional parts in α. If we assume that $\alpha(y_k)+1$ is not larger than the largest constant in the timing constraints involving y_k (i.e., when time elapses the first integer value reached by y_k is at most this constant), we add to E a time edge $v \xrightarrow{c} v'$ for $v' = \langle q, \beta, (j_1, \ldots, j_{h'})\rangle$ where β is the closest time-successor of α such that the conditions expressed by one of the rows of the following Table 1 are satisfied (where $(0 \approx'_1 \overline{y}'_1 \approx_2 \ldots \approx'_k \overline{y}'_k \approx_{k+1} 1)$ denotes the ordering of the fractional parts in β, and $l = 2, \ldots, k$):

	$\approx_1$	$\approx_{k+1}$	$\approx'_1$	$\approx'_2$	$\approx'_{l+1}$	$(j_1, \ldots, j_{h'})$	c
1.	$<$	$<$	$=$	$<$	$\approx_l$	$(i_h, i_2, \ldots, i_{h-1})$	$J_d(q)\ \sum_{l \in I(i_h, i_1)} \vartheta_l$
2.	$\lesssim$ or $=$	$<$	$=$	$<$	$\approx_l$	$(i_h, i_1, \ldots, i_{h-1})$	$J_d(q)\ \sum_{l \in I(i_h, i_1)} \vartheta_l$
3.	$<$	$\lesssim$	$=$	$<$	$\approx_l$	$(i_1, \ldots, i_h)$	0
4.	$\lesssim$ or $=$	$\lesssim$	$=$	$\lesssim$	$\approx_l$	$(i_1, \ldots, i_h)$	0

In the other case, time edges are defined in the same way except for the fact that the clock y_k does not appear in the ordering of the fractional parts of

v' since it has reached its highest constant. To see an example of a time edge, consider a vertex $v = \langle q,\ 0 < x < y < z < 1,\ (1,2,3,4)\rangle$. By row 1 of the above table we have a time edge from v to $\langle q,\ 0 < x < y < 1 \wedge z = 1,\ (4,2,3)\rangle$. The distance tuple $(4,2,3)$ captures the fact that time $(1-z)$ has elapsed and thus the distance in time from x to 0 is increased by $(1-z)$, the fractional part of z is now 0, and all the other distances stay unchanged.

Given a vertex $v \in V$ as above, we add to E a delayed switch $v \xrightarrow{c} v''$ for any vertex $v'' \in V$ such that there exists an immediate switch $v' \xrightarrow{J_s(e)} v''$ and $c = c' + J_s(e)$, where $v' = \langle q, \beta, (j_1, \ldots, j_{h'})\rangle$ and β is the closest time-successor of α such that the conditions expressed by one of the rows of the following Table 2 are satisfied (where $(0 \approx'_1 \overline{y}'_1 \approx_2 \ldots \approx'_k \overline{y}'_k \approx_{k+1} 1)$ denotes the ordering of the fractional parts in β, and $l = 2, \ldots, k$):

	$\approx_1$	$\approx_{k+1}$	$\approx'_1$	$\approx'_l$	$\approx'_{k+1}$	$(j_1, \ldots, j_{h'})$	c'
1.	$<$	$<$	$<$	$\approx_l$	$\lesssim$	$(i_h, i_2, \ldots, i_{h-1})$	$J_d(q)\ \sum_{l \in I(i_h, i_1)} \vartheta_l$
2.	$\lesssim$	$<$	$<$	$\approx_l$	$\lesssim$	$(i_h, i_1, \ldots, i_{h-1})$	$J_d(q)\ \sum_{l \in I(i_h, i_1)} \vartheta_l$
3.	$=$	$<$	$\lesssim$	$\approx_l$	$<$	$(i_1, \ldots, i_h)$	0
4.	$=$	$<$	$<$	$\approx_l$	$\lesssim$	$(i_h, i_1, \ldots, i_{h-1})$	$J_d(q)\ \sum_{l \in I(i_h, i_1)} \vartheta_l$
5.	$=$	$\lesssim$	$\lesssim$	$\approx_l$	$\lesssim$	$(i_1, \ldots, i_h)$	0

For a given tuple of parameters $\vartheta = (\vartheta_1, \ldots, \vartheta_{N+1})$, we denote by $G_A(\vartheta)$ the D-labelled directed graph (V, E). We recall that for our purposes ϑ represents the differences between the fractional parts of two consecutive clocks in the ordering of the fractional parts in the starting state. The construction of $G_A(\vartheta)$ is general in the sense that it does not depend on the particular source and target zones of the problem, but only on the timed automaton. This allows us to use it for solving both the single-source optimal-run problem (for a fixed ϑ) and the zone optimal-run problem (ϑ belongs to a convex set). As an example of application of the above construction, we discuss a fragment of the graph $G_A(\vartheta)$ for the weighted timed automaton modelling the air-traffic control problem from Example 3 (see Figure 4). For the sake of simplicity, we have marked with $1, \ldots, 5$ the vertices of $G_A(\vartheta)$ in Figure 4, and we refer to them by these numbers. Consider vertex 1. Since in the timed automaton from Figure 3 there is a transition from W_1 to W'_1 resetting clock x_1, we have in $G_A(\vartheta)$ an immediate switch from 1 to 2. Edges from 1 to 3 and from 1 to 4 are delayed switches obtained by the same transition above and respectively rows 3 and 4 of Table 2. The edge from 1 to 5 is a time edge and is defined by row 2 of Table 1. Notice that for a given state $s = (q, \nu)$, we have corresponding vertices of $G_A(\vartheta(s))$ of form $\langle q, \alpha, (i_1, \ldots, i_k)\rangle$, where $\nu \in \alpha$. Moreover, each edge is labelled by the actual cost of the corresponding "activity" in A, that is for immediate switches we have just the cost of the A transition, for time edges the cost of spending the time upto the end of the current region in the current A location, and for delayed switches the cost corresponding to the A transition plus the cost for the time spent in the current location before that the transition is taken. We have the following lemma.

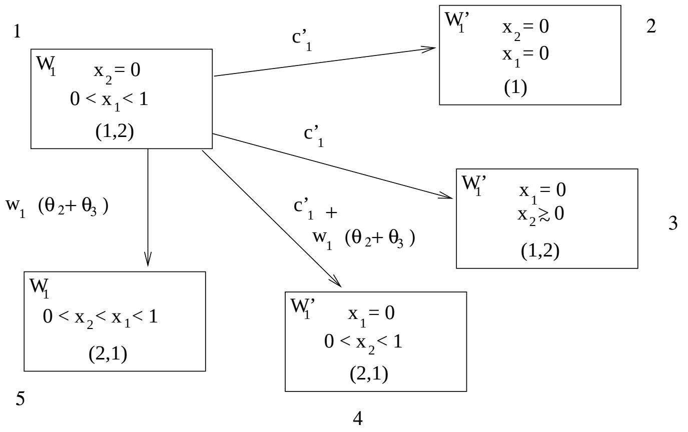

Fig. 4. A fragment of $G_A(\vartheta)$ for the weighted timed automaton in Example 3.

Lemma 1. *Given a timed automaton A, the size of $G_A(\vartheta)$ is exponential in the length of clock constraints of A.*

Proof. In [AD94] the authors proved that the size of the region automaton is $O(|A|\, 2^{|\delta(A)|})$, where $|\delta(A)|$ denotes the length of the clock constraints. A simple counting argument gives that the number of ways to substitute $<$ with $\lesssim$ in the ordering of the fractional parts of a clock region is at most 2^{n+1} and the number of tuples of indices we use to represent the relative differences between the fractional parts is at most $n2^n$. Thus the size of $G_A(\vartheta)$ is at most $O(|A|\, n\, 2^{2n+1}\, 2^{|\delta(A)|})$, and since $n = O(|\delta(A)|)$, it is exponential in the length of the clock constraints.

4 Optimal-runs in in weighted timed automata

4.1 Single-source case

In this section we prove that the single-source optimal-run problem in timed automata can be reduced to the shortest path problem in a weighted directed graph. To see this we introduce first some notation. Let s_0 be a state (q_0, ν_0) of a weighted timed automaton A and $\vartheta(s_0) = (\vartheta_1, \ldots, \vartheta_{N_0+1})$, we denote by $g(s_0)$ the vertex $\langle q_0, \alpha_0, (i_{0,1}, \ldots, i_{0,N_0})\rangle$ of $G_A(\vartheta(s_0))$ such that $\nu_0 \in \alpha_0$ and $i_{0,j}$ is the j-th largest distance in the ordering of the fractional parts in α_0. Given a positive real $\xi << 1$ and a path $\pi = \langle q_0, \alpha_0, (i_{0,1}, \ldots, i_{0,N_0})\rangle \xrightarrow{c_1} \langle q_1, \alpha_1, (i_{1,1}, \ldots, i_{1,N_1})\rangle \xrightarrow{c_2} \ldots \xrightarrow{c_h} \langle q_h, \alpha_h, (i_{h,1}, \ldots, i_{h,N_h})\rangle$ in $G_A(\vartheta(s_0))$, we denote by $R_\pi(\xi)$ the set of runs of A starting at s_0 and obtained by replacing with $(q_j, \nu_j) \xrightarrow[e_j]{t_j} (q_k, \nu_k)$ each portion $\langle q_j, \alpha_j, (i_{j,1}, \ldots, i_{j,N_j})\rangle \xrightarrow{c_{j+1}} \ldots \xrightarrow{c_k} \langle q_k, \alpha_k, (i_{k,1}, \ldots, i_{k,N_k})\rangle$ of π such that:

- $\langle q_{j-1}, \alpha_{j-1}, (i_{j-1,1}, \ldots, i_{j-1,N_{j-1}})\rangle \xrightarrow{c_{j-1}} \langle q_j, \alpha_j, (i_{j,1}, \ldots, i_{j,N_j})\rangle$ is either an immediate or a delayed switch;
- for $l = j \ldots, k-2$, $\langle q_l, \alpha_l, (i_{l,1}, \ldots, i_{l,N_l})\rangle \xrightarrow{c_{l+1}} \langle q_{l+1}, \alpha_{l+1}, (i_{l+1,1}, \ldots, i_{l+1,N_{l+1}})\rangle$ is a time edge;
- $\langle q_{k-1}, \alpha_{k-1}, (i_{k-1,1}, \ldots, i_{k-1,N_{k-1}})\rangle \xrightarrow{c_k} \langle q_k, \alpha_k, (i_{k,1}, \ldots, i_{k,N_k})\rangle$ is either an immediate or a delayed switch. Let $t_j = \tau' + \tau''$ and $\nu_j + \tau' \in \alpha_{k-1}$. In the case of an immediate switch $\tau'' = 0$, while in the other case τ'' is such that:
  - if the delayed switch is obtained by rows 1 and 2 of Table 2, then $\nu_j + t_j \in \alpha_{k-1}$ and the largest fractional part in $\nu_j + t_j$ is greater than $(1 - \xi)$;
  - otherwise, denoted as α' the time-successor of α_{k-1} which is first entered by letting time elapse from a valuation in α_{k-1}, it holds that $\nu_j + t_j \in \alpha'$, moreover if the delayed switch is obtained by rows 4 and 5 of Table 2, the largest meaningful fractional part in $\nu_j + t_j$ is greater than $(1 - \xi)$, and if the delayed switch is obtained by rows 3 and 5 of Table 2, the smallest meaningful fractional part in $\nu_j + t_j$ is less than ξ;
- e_j is the transition corresponding to $\langle q_{k-1}, \alpha_{k-1}, (i_{k-1,1}, \ldots, i_{k-1,N_{k-1}})\rangle \xrightarrow{c_{k-1}} \langle q_k, \alpha_k, (i_{k,1}, \ldots, i_{k,N_k})\rangle$.

In the following we assume that ξ is a positive real number such that $\xi << 1$. By the definition of $G_A(\vartheta)$ and $R_\pi(\xi)$, we have the following lemma.

Lemma 2. *Given a timed automaton A and a state $s = (q, \nu)$ of A, if π is a path of $G_A(\vartheta(s))$ from $g(s)$ of cost c_π then $R_\pi(\xi)$ is a set of runs of A such that for any $\varepsilon > 0$ there exists an $r \in R_\pi(\xi)$ such that $c_\pi \leq J(r) < c_\pi + \varepsilon$.*

To complete our reduction we need the following lemma.

Lemma 3. *Given a run r of A from a state s to a target zone T, there exists a path π of $G_A(\vartheta(s))$ from $g(s)$ to a vertex corresponding to a state in T such that the cost of π is not larger than $J(r)$.*

Proof. The interesting case is when transitions in r are from states that do not belong to any of the subregions encoded by $G_A(\vartheta(s))$ vertices. Assume that A in run r takes a transition e from an open region α after spending some time in it, and e is the first transition in r with this property. Clearly, upto e, r has a corresponding path π in $G_A(\vartheta(s))$ whose cost is not more than $J(r)$. We observe that by definition there must be two delayed transitions e_1 and e_2 of $G_A(\vartheta(s))$ corresponding respectively to the cases e is taken as soon as α is entered and e is taken just before leaving α. Moreover, consider two A runs r_1 and r_2 that differ from r only for the fact that in r_1 A takes e after an arbitrarily short time spent in α, while in r_2 A takes e after an arbitrarily short time before leaving α. Clearly, $J(r) \geq \min\{J(r_1), J(r_2)\}$ holds. Thus we can add to π the transition corresponding to the run r_i with the least cost between r_1 and r_2. Applying iteratively this argument, we determine a path π in $G_A(\vartheta(s))$ of cost $c \leq J(r)$.

As a direct consequence of Lemmas 2 and 3, we have the following theorems.

Theorem 1. *Given a timed automaton A, a state s of A, a target zone T, π is a shortest path of $G_A(\vartheta(s))$ starting from $g(s)$ to a vertex corresponding to a state in T if and only if $R_\pi(\xi)$ is an approximation of an optimal run of A from s to T.*

Theorem 2. *Given a timed automaton A, a state s of A, a target zone T, there exists an optimal run of A from s to T if and only if for a shortest path π of $G_A(\vartheta(s))$ from $g(s)$ to a vertex corresponding to a state in T there exists a run $r \in R_\pi(\xi)$, such that the cost of π is equal to $J(r)$. Moreover, r is an optimal run of A from s to T.*

Given a timed automaton A, a source state s, and a target zone T, the following algorithm solves the single-source optimal-run problem:

1. Let G be the graph obtained from $G_A(\vartheta(s))$ by collapsing all the vertices corresponding to a state in T in a single vertex v_t.
2. Solve the single-source shortest-path problem on G from $g(s)$.
3. Let π be a shortest path from v_s to v_t. Output[2] $R_\pi(\xi)$ and the cost of π.

Theorem 3. *The single-source optimal-run problem can be solved in time exponential in the size of the timed automaton.*

4.2 The algorithm for the general case

In this section we consider the zone optimal-run problem. We give an exponential time algorithm to solve this problem for timed automata with at most 1 clock and a fix-point algorithm in doubly-exponential time, for the general case.

We start considering the general case. Since we want to solve the problem of determining the optimal runs from any state of the source zone S to a state of a target zone T, for parameters ϑ in $G_A(\vartheta)$ we consider only values given by $\vartheta = \vartheta(s)$ for a state in $s \in S$. Thus it holds that $\vartheta_1 + \ldots + \vartheta_{N+1} = 1$ and we can eliminate a parameter by the substitution $\vartheta_{N+1} = 1 - \sum_{i=1}^{N} \vartheta_i$. From now on, we will assume that $\vartheta(s)$ is the tuple $(\vartheta_1, \ldots, \vartheta_N)$ and $G_A(\vartheta(s))$ is the graph obtained after the substitution $\vartheta_{N+1} = 1 - \sum_{i=1}^{N} \vartheta_i$. The algorithm that we are giving, labels the vertices of $G_A(\vartheta)$ with sets of linear expressions on $\vartheta = (\vartheta_1, \ldots, \vartheta_N)$. The meaning of these expressions is that given a state $s \in S$ the minimum over these expressions gives the optimal cost of a run from s. An *expression* is a first-degree polynomial in $\vartheta_1, \ldots, \vartheta_N$, and $(1 - \sum_{i=1}^{N} \vartheta_i)$ with integer coefficients. That is, an expression has the form $f(\vartheta) = a_0 + a_1\vartheta_1 + \ldots + a_N\vartheta_N + a_{N+1}(1 - \sum_{i=1}^{N} \vartheta_i)$, where $a_0, \ldots, a_{N+1}$ are nonnegative integer constants. We denote expressions by $(N+2)$-tuples of coefficients and write $(a_0, \ldots, a_{N+1})$ for the above expression $f(\vartheta)$. We denote by $\prec$ the natural extension to tuples of the total ordering $<$ over reals. Moreover,

[2] This step needs a further refinement to distinguish between an approximate solution and an optimal solution. It is not entirely straightforward, but it can be handled at the same complexity. We defer the reader to the full version of the paper.

let f, f' be two expressions, and v, v' be two vertices of $G_A(\vartheta)$, $(f, v) \prec (f', v')$ if and only if $f \prec f'$. A set X of tuples of type (f, v), for an expression f and a vertex v, is said to be *minimized* (with respect to $\prec$) if for any $(f, v), (f', v') \in X$, (f, v) and (f', v') are not comparable with respect to $\prec$.

The algorithm we present computes a labelling function l that maps any vertex u of $G_A(\vartheta)$ to a minimized set of pairs (f, v) for which there exist a path π and a state $s \in S$ such that:

- π is a shortest path of $G_A(\vartheta(s))$ from u to a vertex corresponding to T,
- the first edge e of π connects u to v, and
- the cost of π is given by $f(\vartheta(s))$.

We can summarize our algorithm in the following steps:

1. Initialize l by assigning $l(u) = \{(0, \ldots, 0, u)\}$ for u corresponding to a state in T, and $l(u) = \emptyset$ for all remaining vertices.
2. **repeat**
   $l' \leftarrow l$; $l \leftarrow$ UPDATE(l')
   until $l' = l$
3. Output l.

We just need to specify the function UPDATE. Consider an edge e a vertex $u = \langle q, \alpha, (i_1, \ldots, i_h) \rangle$. We have the following cases:

- e is an immediate switch from u to v: for $(a_0, \ldots, a_{N+1}, v') \in l'(v)$, define $(a'_0, \ldots, a'_{N+1}, v)$ such that $a'_0 = a_0 + c_e$, and $a'_i = a_i$ for $i = 1, \ldots, (N+1)$, where c_e is the cost of e;
- e is a time edge from u to v: for any $(a_0, \ldots, a_{N+1}, v') \in l'(v)$, define $(a'_0, \ldots, a'_{N+1}, v)$ such that if e is obtained by rows 1 and 2 of Table 1 and $i \in I(i_h, i_1)$, then $a'_i = a_i + J_d(q)$, otherwise $a'_i = a_i$;
- the edge e is a delayed switch from u to v: for any $(a_0, \ldots, a_{N+1}, v') \in l'(v)$, define $(a'_0, \ldots, a'_{N+1}, v)$ such that if e is obtained by rows 1, 2 and 4 of Table 2 and $i \in I(i_h, i_1)$, then $a'_i = a_i + J_d(q)$, otherwise $a'_i = a_i$.

Let $l''(u)$ be the set of all the tuples generated for u. After executing $l \leftarrow$ UPDATE(l'), $l(u)$ contains the set obtained deleting from $l'(u) \cup l''(u)$ all the tuples (f, v) such that $f' \prec f$ for some $(f', v') \in l'(u) \cup l''(u)$. Moreover, once the function l is output, it is easy to determine the optimal cost and generate the corresponding solution from l and the graph $G_A(\vartheta)$, given ϑ. We observe that each of the tuples (f, v) belonging to $l(u)$ corresponds to a path from u to a target vertex. Thus the cardinality of $l(u)$ is bounded above by the number of simple paths in $G_A(\vartheta)$. Hence we have the following theorem.

Theorem 4. *The zone optimal-run problem can be solved in doubly-exponential time.*

If we restrict to timed automata with just one clock variable, it is possible to solve the zone optimal-run problem in singly exponential time. We consider the algorithm given in [KO81,YTO91] to solve a particular shortest-path problem with only a parameter ϑ and edge costs given by $(c - \vartheta)$, for constants c. This algorithm runs in polynomial time and can be modified in order to obtain a polynomial time algorithm to solve the parametric shortest-path problem with edge costs given by a first-degree polynomial of ϑ ($\vartheta \in [0, 1]$).

Theorem 5. *The zone optimal-run problem for automata with one clock variable can be solved in exponential time.*

References

[ACH93] R. Alur, C. Courcoubetis, and T.A. Henzinger. Computing accumulated delays in real-time system. In *Proc. of the Fifth International Conference on Computer-Aided Verification, CAV'93*, LNCS 697, pages 181 – 193, 1993.

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

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

[AMP95] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Proc. of the 2nd International Workshop on Hybrid Systems*, LNCS 999, pages 1 – 20, 1995.

[BHF+] G. Behrman, T. Hune, A. Fehnker, K. Larsen, P. Pettersson, R. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *this Volume*.

[Chu62] A. Church. Logic, arithmetic, and automata. In *Proc. of the International Congress of Mathematics*, pages 23–35, 1962.

[CY91] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. In *Proc. of the 3rd International Conference on Computer Aided Verification*, LNCS 575, pages 399 – 409, 1991.

[KO81] R. M. Karp and J. R. Orlin. Parametric shortest path algorithm with an application to cyclic staffing. *Discrete Applied Math.*, 3:37 – 45, 1981.

[LTS99] J. Lygeros, C. Tomlin, and S.S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3):349–370, March 1999.

[MPS95] O. Maler, A. Pnueli, and J. Sifakis. On the synthesis of discrete controllers for timed systems. In *Proc. of the 12th Annual Symposium on Theoretical Aspects of Computer Science, STACS'95*, LNCS 900, pages 229 – 242, 1995.

[NTY00] P. Nierbert, S. Tripakis, and S. Yovine. Minimum-time reachability for timed automata. In *Proc. of the 8-th IEEE Mediterranean Conference on Control and Automation*, 2000.

[SPS00] O. Shakernia, G. J. Pappas, and S. Sastry. Decidable controller synthesis for classes of linear systems. In *Proc. of the 3rd International Workshop on Hybrid Systems: Computation and Control, HSCC'00*, LNCS 1790, pages 407 – 420, 2000.

[Tho95] W. Thomas. On the synthesis of strategies in infinite games. In Ernst W. Mayr and Claude Puech, editors, *12th Annual Symposium on Theoretical Aspects of Computer Science, STACS'95*, LNCS 900, pages 1 – 13, 1995.

[WT97] H. Wong-Toi. The synthesis of controllers for linear hybrid automata. In *Proc. of the 36th IEEE CDC*, San Diego, CA, December 1997.

[YTO91] N. E. Young, R. Tarjan, and J. Orlin. Faster parametric shortest path and minimum balance algorithms. *Networks*, 21 (2):205 – 221, 1991.