Jan Długosz University in Częstochowa
Scientific Issues, Mathematics XV, Częstochowa 2010

SAT-BASED SEARCHING FOR k-QUASI-OPTIMAL RUNS IN WEIGHTED TIMED AUTOMATA

Bożena Woźna-Szcześniak, Andrzej Zbrzezny

Institute of Mathematics and Computer Science
Jan Długosz University of Częstochowa
al. Armii Krajowej 13/15, 42-200 Częstochowa, Poland
e-mail: {b.wozna, a.zbrzezny}@ajd.czest.pl

Abstract. In the paper we are concerned with an optimal cost reachability problem for weighted timed automata, and we use a translation to SAT to solve the problem. In particular, we show how to find a run of length $k \in \mathbb{N}$ that starts at the initial state and terminates at a state containing a target location, its total cost belongs to the interval $[c, c+1)$, for some natural number $c \in \mathbb{N}$, and the cost of each other run of length k, which also leads from the initial state to a state containing the target location, is greater or equal to c. This kind of runs is called *k-quasi-optimal*. We exemplify the use of our solution to the mentioned problem by means of the *air traffic control problem*, and we provide some preliminary experimental results.

1. Introduction

In automatic verification of hardware and software systems, the *reachability* problem is a core decision problem. This is because it can be used to detect *deadlocks*, or a violation of a *safety* property, which means that nothing bad will ever happen. For real-time systems like, for example, an air traffic control, or process controllers in manufacturing plants, it is also reasonable to ask questions about the minimum cost of reaching a desirable state of the system. Therefore, in the paper, we deal with the *k-optimal cost reachability* problem for *weighted timed automata* [3], in particular, we are interested in using SAT-methods to solve the problem.

A timed automaton [2] is a formalism that can be used to model the behaviour of a real-time system. It extends a finite automaton by adding a finite

set of variables that are able to measure real-time, and express timing constraints; these variables are called clocks. The semantics of a timed automaton is given in terms of an infinite labelled transition system with two kinds of transitions: a discrete transition and a time transition. The first one correspond to a change of a location, and the second one to the passage of time. However, in order to define the *k-optimal cost reachability* problem for timed automata we need to associate costs with transitions and locations. The costs assigned to transitions (switch costs) will give the cost of discrete transitions, and the costs assigned to locations (duration costs) will define the cost of time spent in these locations. Such timed automata augmented with costs are known as *weighted timed automata* [3], or *priced timed automata* [5].

Our solution to the k-optimal cost reachability problem relies on combining the well-know *forward reachability* analysis and the *bounded model checking* (BMC) method [6, 13, 14]. The forward reachability algorithm searches the state space using the breadth first mode, whereas the BMC performs a verification on a part of the automata model exploiting SAT solvers.

The rest of the paper is organised as follows. In the next section we provide the main formalisms used throughout the paper, i.e. weighted timed automata. In Section 3 we define and solve the k-optimal cost reachability problem for weighted timed automata. In Section 4 we show how our solution to the considered reachability problem works by means of the *air traffic control problem*. We conclude in Section 5 by discussing related work.

2. Weighted Timed Automata

Let us start by fixing names of the sets of numbers used in the rest of the paper. By $\mathbb{N} = \{0, 1, 2, 3, \dots\}$ we denote the set of natural numbers, by $\mathbb{Q}$ the set of non-negative rational numbers, and by PV a set of propositional variables.

To define weighted timed automata formally, we need to say what type of clock constraints are allowed as guards and invariants, and what are the cost functions. This is introduced in the following subsection.

2.1. Clocks and clock valuation

For a finite set $\mathcal{X}$ of real variables, called *clocks*, the set $\mathcal{C}(\mathcal{X})$ of all the *clock constraints* over $\mathcal{X}$ is defined by the following grammar:

$$\mathbf{cc} ::= \mathit{true} \mid x \sim c \mid x - y \sim c \mid \mathbf{cc} \wedge \mathbf{cc},$$

where $x, y \in \mathcal{X}$, $c \in \mathbb{N}$, and $\sim \in \{\leq, <, =, >, \geq\}$.

A *clock valuation* is a total mapping $\mathbf{c} : \mathcal{X} \to \mathbb{Q}$. Satisfiability of a clock constraint $\mathbf{cc} \in \mathcal{C}(\mathcal{X})$ by a clock valuation $\mathbf{c}$ ($\mathbf{c} \models \mathbf{cc}$) is defined inductively as follows:

- $\mathbf{c} \models \mathit{true}$,
- $\mathbf{c} \models (x \sim c)$ iff $\mathbf{c}(x) \models c$,
- $\mathbf{c} \models (x - y \sim c)$ iff $\mathbf{c}(x) - \mathbf{c}(y) \sim c$,
- $\mathbf{c} \models \mathbf{cc}_1 \wedge \mathbf{cc}_2$ iff $\mathbf{c} \models \mathbf{cc}_1$ and $\mathbf{c} \models \mathbf{cc}_2$.

In what follows, the set of all the clock valuations satisfying a clock constraint $\mathbf{cc}$ is denoted by $[\![\mathbf{cc}]\!]$. Given a clock valuation $\mathbf{c}$ and $\delta \in \mathbb{Q}$, by $\mathbf{c}+\delta$ we denote a clock valuation $\mathbf{c}'$ such that $\mathbf{c}'(x) = \mathbf{c}(x) + \delta$, for all $x \in \mathcal{X}$. Moreover, for a subset of clocks $X \subseteq \mathcal{X}$, $\mathbf{c}[X := 0]$ denotes the valuation $\mathbf{c}'$ such that for all $x \in X$, $\mathbf{c}'(x) = 0$ and for all $x \in \mathcal{X} \setminus X$, $\mathbf{c}'(x) = \mathbf{c}(x)$. Finally, by $\mathbf{c}^0$ we denote the *initial* clock valuation, i.e. the valuation such that $\mathbf{c}^0(x) = 0$ for all $x \in \mathcal{X}$.

2.2. Syntax and semantics

We assume the definition of weighted timed automata from [3] but augmented to include a special rational variable z.

Definition 1. (Weighted timed automaton). *A* weighted timed automaton *is a tuple* $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$, *where* Σ *is a finite set of labels (actions),* L *is a finite set of locations,* l^0 *is an initial location,* $\mathcal{X}$ *is a finite set of clocks,* $E \subseteq L \times \Sigma \times \mathcal{C}(\mathcal{X}) \times 2^{\mathcal{X}} \times L$ *is a transition relation,* $\mathcal{I} : L \to \mathcal{C}(\mathcal{X})$ *is an invariant function,* $J_s : E \to \mathbb{N}$ *is a switch cost function,* $J_d : L \to \mathbb{N}$ *is a duration cost function,* z *is a rational variable, and* $\mathcal{V} : L \to 2^{PV}$ *is a valuation function assigning to each location a set of atomic propositions true in that location.*

The switch cost function assigns to each transition a cost expressing the price of taking the transition. The duration cost function assigns to each location a cost expressing the price of staying in this location for one time unit. The invariant function assigns to each location a clock constraint expressing the condition under which $\mathcal{A}$ can stay in this location. Each element $t = (l, \sigma, \mathbf{cc}, X, l') \in E$ represents a transition from the location l to the location l', where σ is the label of the transition t, $\mathbf{cc}$ defines the enabling conditions for t, and X is a set of clocks to be reset.

The semantics of the weighted timed automaton is defined by associating to it a *dense model* as defined below.

Definition 2. *Let* $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$ *be a weighted timed automaton,* $\mathbf{z} : \{z\} \to \mathbb{Q}$ *a* valuation for z, *and* $\mathbf{z}^0$ *denote the* initial *valuation for* z, *i.e.,* $\mathbf{z}^0(z) = 0$. *A* dense model *for* $\mathcal{A}$ *is a tuple* $M(\mathcal{A}) = (\Sigma \cup \mathbb{Q}, S, s^0, \to, \mathcal{V}')$, *where* $\Sigma \cup \mathbb{Q}$ *is a set of labels,* $S = \{(l, \mathbf{c}, \mathbf{z}) \mid l \in L,\ \mathbf{c} \in \mathbb{Q}^{|\mathcal{X}|},\ \mathbf{c} \models \mathcal{I}(l), \mathbf{z} \in \mathbb{Q}\}$ *is a set of states,* $s^0 = (l^0, \mathbf{c}^0, \mathbf{z}^0)$ *is the initial state,* $\mathcal{V}' : S \to 2^{PV}$ *is a valuation function such that* $\mathcal{V}'((l, \mathbf{c}, \mathbf{z})) = \mathcal{V}(l)$, *and* $\to \subseteq S \times \Sigma \cup \mathbb{Q} \times S$ *is the smallest transition relation defined by the following rules:*

- *for* $\sigma \in \Sigma$, $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\sigma} (l', \mathbf{c}', \mathbf{z}')$ *iff there exists a transition* $t = (l, \sigma, \mathbf{cc}, X, l') \in E$ *such that* $\mathbf{c} \models \mathbf{cc}$, $\mathbf{c} \models \mathcal{I}(l)$, $\mathbf{c}[X := 0] \models \mathcal{I}(l')$, *and* $\mathbf{z}' = \mathbf{z} + J_s(t)$ (action transition),
- *for* $\delta \in \mathbb{Q}$, $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\delta} (l, \mathbf{c} + \delta, \mathbf{z}')$ *iff* $\mathbf{c}, \mathbf{c} + \delta \models \mathcal{I}(l)$, *and* $\mathbf{z}' = \mathbf{z} + J_d(l) \cdot \delta$ (time transition).

Intuitively, an action transition corresponds to an action performed by the automaton under consideration. The action can be performed only if the underlying enabling condition is satisfied. Moreover, all the clocks that are associated with the action are set to zero, its locations change accordingly, and the value of the variable z is increased by the switch cost. A time transition causes an equal increase in the value of all the clocks, and does not involve a location change. Obviously, the new clock valuations have to still satisfy all the location invariants, and the value of the variable z is increased by the duration cost.

Let $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\delta,\sigma} (l', \mathbf{c}', \mathbf{z}')$ denote that $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\delta} (l'', \mathbf{c}'', \mathbf{z}'')$ and $(l'', \mathbf{c}'', \mathbf{z}'') \xrightarrow{\sigma} (l', \mathbf{c}', \mathbf{z}')$, where $\sigma \in \Sigma$ and $\delta \in \mathbb{Q}$. A *run* ρ of a weighted timed automaton $\mathcal{A}$ is a finite sequence of states:

$$(l_0, \mathbf{c}_0, \mathbf{z}_0) \xrightarrow{\delta_1,\sigma_1} (l_1, \mathbf{c}_1, \mathbf{z}_1) \xrightarrow{\delta_2,\sigma_2} \dots \xrightarrow{\delta_{k-1},\sigma_{k-1}} (l_{k-1}, \mathbf{c}_{k-1}, \mathbf{z}_{k-1}) \xrightarrow{\delta_k,\sigma_k} (l_k, \mathbf{c}_k, \mathbf{z}_k)$$

such that $(l_i, \mathbf{c}_i, \mathbf{z}_i) \in S$, $\sigma_i \in \Sigma$, and $\delta_i \in \mathbb{Q}$ for each $i \in \{1, \dots, k\}$. Hereafter, we refer to a run ρ of length k as k-run.

Given a k-run ρ of $\mathcal{A}$ and cost functions J_s and J_d, we associate cost to ρ as follows:

- $J_s(\rho) = \sum_{i=0}^{k-1} J_s(t_i)$, where $t_i := (l_i, \mathbf{c}_i, \mathbf{z}_i) \xrightarrow{\delta_{i+1},\sigma_{i+1}} (l_{i+1}, \mathbf{c}_{i+1}, \mathbf{z}_{i+1})$,
- $J_d(\rho) = \sum_{i=1}^{k} \delta_i \cdot J_d(l_i)$.

The *total cost* associated to a k-run ρ is defined as $J(\rho) = J_d(\rho) + J_s(\rho)$.

The *k-optimal cost* for k-runs that start at a state containing location l and end at a state containing location l' is defined as $J_k^*(l, l') = inf\{J(\rho) | \rho$ is a k-run from a state containing location l to a state containing location $l'\}$.

A k-run ρ from a state containing location l to a state containing location l' such that $\lfloor J(\rho) \rfloor = \lfloor J_k^*(l, l') \rfloor$ is called *k-quasi-optimal*.

In this paper, for given two locations l and l' we are interested in finding the greatest integer lower bound (g.i.l.b. for short) of the k-optimal cost for k-runs starting at a state s containing location l and terminating at a state t containing location l', where k is the length of a shortest run from s to t. Moreover, we are interested in finding *k-quasi-optimal* runs. Therefore, in Section 3 we define k-optimal cost reachability problem, and we show how to solve it using SAT-methods.

2.3. Discrete semantics

In real-time systems modeled by (weighted) timed automata, in order to use SAT-techniques to test reachability or other properties, it is customary to discretise the set of all the clocks valuations. Here we take the discretisation scheme that is based on the one introduced in [15], but here we use the discretisation step that depends not only on the length of considered runs, but also on the maximal duration cost. It uses the following set of discretised clock's values and labels as primitives. Let c_{max} be the largest constant c appearing in all the invariants and guards of a weighted timed automaton $\mathcal{A}$. For every $m \in \mathbb{N}$ we define $A_m = \{a \in \mathbb{Q} \mid (\exists j \in \mathbb{N})\ a \cdot 2^m = j\}$ and $B_m = \{b \in \mathbb{Q} \mid (\exists j \in \mathbb{N})\ b \cdot 2^m = j \text{ and } b < c_{max} + 1\}$. Then, $A = \bigcup_{m=0}^{\infty} A_m$ defines the set of discretised clock's values, and $B = \bigcup_{m=1}^{\infty} B_m$ defines the set of labels. We use this technique to define a *discretised model* for a weighted timed automaton. This model is crucial for the translation of the k-optimal cost reachability problem to the SAT-problem as described in the next section.

To give a definition of a discretised model that supports clock constraints of the form $x - y \sim c$, we first recall the notion of *weak region equivalence* [15].

Definition 3. (Weak region equivalence). *Assume a set of clocks $\mathcal{X}$, and for any $t \in \mathbb{Q}$ let $\langle t \rangle$ denote the fractional (respectively integral) part of t (respectively $\lfloor t \rfloor$). The* weak region equivalence *is a relation $\cong \subseteq \mathbb{Q}^{\mathcal{X}} \times \mathbb{Q}^{\mathcal{X}}$ defined as follows. For two clock valuations u and v in $\mathbb{Q}^{\mathcal{X}}$, $u \cong v$ iff all the following conditions hold:*

[1] $\lfloor u(x) \rfloor = \lfloor v(x) \rfloor$, *for all* $x \in \mathcal{X}$.

[2] $\langle u(x) \rangle = 0$ *iff* $\langle v(x) \rangle = 0$, *for all* $x \in \mathcal{X}$.

[3] $\langle u(x) \rangle < \langle u(y) \rangle$ *iff* $\langle v(x) \rangle < \langle v(y) \rangle$, *for all* $x, y \in \mathcal{X}$.

Definition 4. (Discretised model). *Let $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$ be a weighted timed automaton. A* discretised model *for $\mathcal{A}$ is a tuple*

$M_d(\mathcal{A}) = (\Sigma \cup B, S_d, s_d^0, \rightarrow_d, \mathcal{V}_d)$, where $S_d = L \times A^{\mathcal{X}} \times B$ is a set of states, $s_d^0 = (l^0, \mathbf{c}^0, \mathbf{z}^0)$ is the initial state, $\mathcal{V}_d : S_d \rightarrow 2^{PV}$ is a valuation function defined by $\mathcal{V}_d((l, \mathbf{c}, \mathbf{z})) = \mathcal{V}(l)$, and $\rightarrow_d \subseteq S_d \times (\Sigma \cup B) \times S_d$ is a time/action transition relation defined by:

- *Time transition: for any $\delta \in B$, $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\delta}_d (l, \mathbf{c} + \delta, \mathbf{z}')$ iff $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\delta} (l, \mathbf{c} + \delta, \mathbf{z}')$ in $M(\mathcal{A})$ and $(\forall \delta' \leq \delta)$ $\mathbf{c} + \delta' \cong \mathbf{c}$ or $\mathbf{c} + \delta' \cong \mathbf{c} + \delta$,*
- *Action transition: for any $\sigma \in \Sigma$, $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\sigma}_d (l', \mathbf{c}', \mathbf{z}')$ iff $(l, \mathbf{c}, \mathbf{z}) \xrightarrow{\sigma} (l', \mathbf{c}', \mathbf{z}')$ in $M(\mathcal{A})$.*

The theorem below shows that the k-optimal cost reachability problem for a weighted timed automaton $\mathcal{A}$ can be solved using the discretised model $M_d(\mathcal{A})$ instead of the dense model $M(\mathcal{A})$.

In what follows, we denote by $\rho(s,t)$ a run that starts at state s and ends at state t. Moreover, for two states $s = (l, \mathbf{c}, \mathbf{z})$ and $t = (l', \mathbf{c}', \mathbf{z}')$, we write $s \cong t$ if and only if $l = l'$, $\mathbf{c} \cong \mathbf{c}'$, $\lfloor \mathbf{z}(z) \rfloor = \lfloor \mathbf{z}'(z) \rfloor$ and $\langle \mathbf{z}(z) \rangle = 0 \iff \langle \mathbf{z}'(z) \rangle = 0$.

Theorem 1. *Let $\mathcal{A}$ be a weighted timed automaton, s and t two states in $M(\mathcal{A})$, and $\rho(s,t)$ a k-quasi-optimal run in $M(\mathcal{A})$, where $k \in \mathbb{N}$ is the length of a shortest run that starts at s and ends at t. Then, there exist two states s' and t' in $M_d(\mathcal{A})$ and there exists a k-quasi-optimal run $\rho'(s', t')$ in $M_d(\mathcal{A})$ such that $s \cong s'$ and $t \cong t'$.*

Proof (Idea). The proof is an extension of the proof of Theorem 3.1 in [15], and it is conducted by means of induction on k. The induction step consists in showing that for each $q = (l, \mathbf{c}_q, \mathbf{z}_q), r = (l', \mathbf{c}_r, \mathbf{z}_r) \in M(\mathcal{A})$, $q' = (l, \mathbf{c}_{q'}, \mathbf{z}_{q'}), r' = (l', \mathbf{c}_{r'}, \mathbf{z}_{r'}) \in M_d(\mathcal{A})$, $\delta \in \mathbb{Q}$, $\delta' \in B$, if $q \cong q'$, $\delta \cong \delta'$ and there exist transitions $q \xrightarrow{\delta,\sigma} r$, $q' \xrightarrow{\delta',\sigma} r'$, then $r \cong r'$. The crucial part of the induction step is rather tedious, and relies on showing that $\mathbf{z}_q + J_d(l) \cdot \delta \cong \mathbf{z}_{q'} + J_d(l) \cdot \delta'$, what requires using some techniqal facts concerning the underling discretisation.

3. k-optimal cost reachability problem

In this section we formally define the k-optimal cost reachability problem for weighted timed automata, and we present a solution to the problem which uses SAT-solvers. We start by defining the problem, then we describe our solution informally, and finaly we show our algorithm.

The k-optimal cost reachability problem for weighted timed automata is defined as follows.

Definition 5. (k-optimal cost reachability). *Given a weighted timed automaton $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$, and a desirable location $l^p \in L$ satisfying a property p.* k-optimal cost reachability *consists in finding a k-quasi-optimal run ρ starting at $s_d^0 \in M_d(\mathcal{A})$ and terminating at a state in $M_d(\mathcal{A})$ containing location l^p.*

Note that if ρ is a k-quasi-optimal run, then there exists $c \in \mathbb{N}$ such that: $c \leq J(\rho) < c + 1$, and for all the $k-$runs ρ' that starts at s_d^0 and terminates at a state in $M_d(\mathcal{A})$ containing location l^p, $J(\rho') \geq c$ holds.

3.1. Our solution – an informal explanation

We begin with an informal explanation of our solution to the k-optimal cost reachability problem, which will help to understand the formal description presented later on in this section.

To solve the k-optimal cost reachability problem we proceed as follows. We first encode by propositional formulae both the property p, and the unfolding of the transition relation of $M_d(\mathcal{A})$ up to the depth k (for $k \in \mathbb{N}$). Let φ_k be the conjunction of the two above formulae. We test φ_k for the propositional satisfiability using a SAT-solver. If the test for φ_k is positive, we calculate the cost $r_0 \in \mathbb{Q}$ of the resulting witness ρ_0, and we know that $J(\rho_0) < \lceil r_0 \rceil$. Next, we set $c_0 = \lceil r_0 \rceil - 1$, and we run the propositional satisfiability test once again, but for the formula $\phi_k(c_0) = \varphi_k \wedge (z < c_0)$[1]. If the test for $\phi_k(c_0)$ is positive, we calculate the cost $r_1 \in \mathbb{Q}$ of the resulting witness ρ_1, and we know that $r_1 < c_0$. Next, we set $c_1 = \lceil r_1 \rceil - 1$, and we run the propositional satisfiability test once again, but for the formula $\phi_k(c_1) = \varphi_k \wedge (z < c_1)$, and so on. We stop testing if the test for $\phi_k(c_i)$ is negative or $r_i = 0$.

Notice, that if the test for $\phi_k(c_i)$ is negative, we can perform one more test for the formula $\psi_k(c_i) = \varphi_k \wedge (z = c_i)$. If the test for $\psi_k(c_i)$ is positive, we can conclude that k-optimal cost is equal to c_i. Otherwise, we can only conclude that the g.i.l.b. of the k-optimal cost is equal to c_i.

3.2. Translation to propositional formulae

Let $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$ be a weighted timed automaton, $M_d(\mathcal{A}) = (\Sigma \cup B, S_d, s_d^0, \rightarrow_d, \mathcal{V}_d)$ a discretised model, and $k \in \mathbb{N}$. Each state s of $M_d(\mathcal{A})$ reachable on a k-run can be encoded by a bit-vector whose length, say n, depends on the number of locations, the constant c_{max}, the maximal duration cost, and the number k. Thus, each state s of $M_d(\mathcal{A})$ can be represented by a vector $\mathtt{w} = (\mathtt{w}[1], \ldots, \mathtt{w}[n])$ of propositional variables (usually

[1] The notation $z < c_i$, for $i = 0, 1, 2, \ldots$ appearing in this section, denotes a propositional formula encoding the fact that the value of the variable z is less than c_i.

called *state variables*) to which we refer to as a *global state variable* [2]. A finite sequence $(\mathtt{w}_0, \ldots, \mathtt{w}_k)$ of global state variables is called a *symbolic k-path*.

For two global state variables $\mathtt{w}, \mathtt{w}'$, we define the following propositional formulae:

- $I_s(\mathtt{w})$ is a formula over $\mathtt{w}$ that is true for a valuation s_w of $\mathtt{w}$ iff $s_w = s$.
- $p(\mathtt{w})$ is a formula over $\mathtt{w}$ that is true for a valuation s_w of $\mathtt{w}$ iff $p \in \mathcal{V}(s_w)$ (encodes a set of states of $M_d(\mathcal{A})$ in which $p \in PV$ holds).
- $T(\mathtt{w}, \mathtt{w}')$ is a formula over $\mathtt{w}$ and $\mathtt{w}'$ that is true for two valuations s_w of $\mathtt{w}$ and $s_{w'}$ of $\mathtt{w}'$ iff $(s_w, s_{w'}) \in \rightarrow_d$ (encodes the transition relation of $M_d(\mathcal{A})$).

  The definition of the formula T involves the Boolean encoding of addition and multiplication of rational numbers, which has been described in [16].

We can now define the propositional formula φ_k, introduced in Subsection 3.1. As it was mentioned in Subsection 3.1, φ_k is a conjunction of two formulae. The first one, denoted by $p(\mathtt{w})$, is a translation of a propositional variable p that represents a location in question. The second one, denoted by $[M_d^{s_d^0}]_k$, encodes the unfolding of the transition relation of $M_d(\mathcal{A})$ up to depth $k \in \mathbb{N}$.

The formula $[M_d^{s_d^0}]_k$ is defined over global state variables w_i, for $0 \leq i \leq k$, and it constrains the symbolic k-path to be valid k-run of $M_d(\mathcal{A})$. Namely,

$$[M_d^{s_d^0}]_k := I_{s_d^0}(w_0) \wedge \forall_{i=0}^{k-1} T(w_i, w_{i+1})$$

3.3. Our solution – a formal algorithm

Now we give an algorithm that formalises the method for finding the greatest integer lower bound of k-optimal cost informally described above.

In Algorithm 1 we use the procedure $checkSAT(\gamma)$ that for any given propositional formula γ returns a pair (X, W), where W denotes the valuation returned by a SAT solver, and X can be one of the following three values: $TRUE$, $FALSE$, and $UNKNOWN$. The meanings of the values $TRUE$ and $FALSE$ are self-evident. The value $UNKNOWN$ is returned either if the procedure $checkSAT$ is not able to decide satisfiability of its argument

[2] Notice that we distinguish between states s encoded as sequences of 0's and 1's and their representations in terms of propositional variables $w[i]$.

within some preset timeout period, or has to terminate itself due to exhaustion of available memory. We also use the procedure $getCOST(W)$ that for the valuation W, which represents a $k-$run ρ, returns a natural number c such that the cost of ρ is less than c. Further, for a given propositional formula φ_k, we denote by $\phi_k(c)$ the formula $\varphi_k \wedge (z < c)$, and by $\psi_k(c)$ the formula $\varphi_k \wedge (z = c)$.

Algorithm 1 An algorithm for finding g.i.l.b. of k-optimal cost

1: $k \leftarrow 0$
2: **repeat**
3: $(result, W) \leftarrow checkSAT(\varphi_k)$
4: **if** $result = FALSE$ **then**
5: $k \leftarrow k + 1$
6: **else if** $result = UNKNOWN$ **then**
7: **return** $UNKNOWN$
8: **end if**
9: **until** $result = TRUE$
{there exists a witness of the length k for a desirable property}
10: $c \leftarrow getCOST(W)$
11: **repeat**
12: **if** $c = 0$ **then**
13: **return** k-optimal cost is equal to 0
14: **end if**
15: $(result, W) \leftarrow checkSAT(\phi_k(c-1))$
16: **if** $result = TRUE$ **then**
17: $c \leftarrow getCOST(W)$
18: **else if** $result = UNKNOWN$ **then**
19: **return** $UNKNOWN$
20: **end if**
21: **untill** $result = FALSE$
{optimal cost of any k-run is greater or equal to c}
22: $(result, W) \leftarrow checkSAT(\psi_k(c))$
23: **if** $result = TRUE$ **then**
24: **return** k-optimal cost is equal to c
25: **else**
26: **return** g.i.l.b. of k-optimal cost is equal to c
27: **end if**

4. Case study

4.1. An air traffic control problem

Weighted timed automata with the rational variable are suitable formalism for modelling several optimisation problems, for example, scheduling problems or air traffic control problems. In this section we take a closer look at the later problem.

Assume a situation in which two aircrafts send a landing request to an airport, and they are approaching the same runway. The goal is to allow both the aircrafts 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. This waiting can be implemented either by slowing down an aircraft (this concerns a situation, in which the aircrafts share the same trajectory, and the aircraft that is following is faster), or by forcing one of them to change its trajectory (this concerns a situation, in which the aircrafts reach the joining point of their trajectories almost at the same time).

Consider the automaton in Figure 1 [3]. It models the above scenario, i.e. the discrete values c_1 and c_2 are the costs of the choice of forcing, respectively, aircraft 1 and aircraft 2 to wait. These costs label the transitions, respectively, from location *Start* to location W_1, and from location *Start* to location W_2. The cost w_i, attached to location W_i, is related to the time spent on waiting by aircraft i. For the aircraft that has to wait for the clearance, we model two possible manoeuvres. A first one is to reduce the speed, and in this case the aircraft stays in location W_i. Another possibility is to change the original trajectory, which is modelled by the loop trough location W_i'. Doing this manoeuvre requires a fixed cost c_i', takes at least one time unit, and allows to pay w_i' instead of w_i per each time unit. Since it is realistic to reduce the time a runway stays unused, we penalise this event by a cost c_0 per time unit. Finally, we assume that the landing of each aircraft takes at least one time unit since the related acknowledgement was issued by the control tower.

4.2. Experimental results

All of the experiments have been performed on a computer equipped with the processor Intel Core 2 Duo (2 GHz), 2 GB main memory and the operating system Linux.

In Tables 1 and 2 we present experimental results for the air traffic control problem modeled by the automaton on Figure 1 with the following costs: $c_0 = 20$, $w_2 = 20$, $w_2' = 40$, $w_1 = 60$, $w_1' = 40$, $c_1 = 20$, $c_1' = 20$, $c_2 = 20$, $c_2' = 20$; we refer to this automaton as Automaton 1.

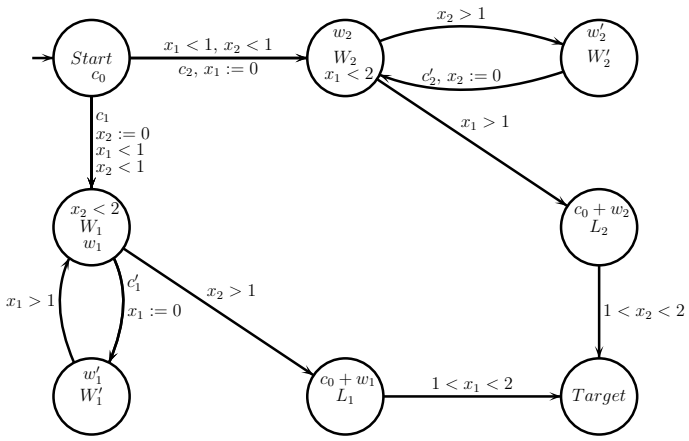

Figure 1: A weighted timed automaton for an air traffic control problem.

Table 1 shows how we get a shortest run of Automaton 1 that leads from the initial state $s_d^0 = (Start, < 0,0 >, 0)$ to a state containing the location $Target$. The cost of the 6-run is equal to $64\frac{12}{512}$, i.e., is less or equal than 65. Table 2 shows how we get the 6-quasi-optimal run of Automaton 1 such that the g.i.l.b. of 6-optimal cost is equal to 40. Table 3 shows the 6-quasi-optimal run of Automaton 1 that leads from the initial state to a state containing the location $Target$ with the g.i.l.b. of 6-optimal cost equal to 40.

BMC4WTA					**RSat**		
k	**variables**	**clauses**	**sec**	**MB**	**sec**	**MB**	**satisfiable**
0	133	190	0.00	1.9	0.0	1.3	NO
2	2278	6881	0.12	2.4	0.0	1.8	NO
4	4901	15057	0.12	3.1	0.0	2.4	NO
6	7275	22480	0.19	3.7	0.0	2.9	YES

Table 1: The shortest run of Automaton 1 on which the $Target$ location is reachable. Its total cost is $64\frac{12}{512}$.

5. Conclusions and related work

In this paper we have defined the k-optimal cost reachability problem for weighted timed automata, and presented a SAT-based method consisting in reducing this problem to the SAT-problem. In particular, we have shown how to find a k-quasi-optimal run that starts at the initial state and terminates at a desirable target state, and how to calculate g.i.l.b. of k-optimal cost for it.

Experimental results, which we have performed, show that the proposed algorithm can be very useful in finding g.i.l.b. of k-optimal cost. Obviously,

our method allows for finding only lower and upper bounds on the cost, to which the k-quasi-optimal run belongs (an unit interval $[c, c+1)$, for $c \in \mathbb{N}$), but in many real-time settings such a cost optimal approximation is sufficient.

BMC4WTA						**RSat**		
$z < c$	**cost**	**variables**	**clauses**	**sec**	**MB**	**sec**	**MB**	**satisfiable**
$z < 64$	$59\frac{132}{512}$	8194	25412	0.22	3.8	0.0	3.2	YES
$z < 59$	$48\frac{444}{512}$	8304	25784	0.21	3.9	0.1	3.2	YES
$z < 48$	$46\frac{508}{512}$	8238	25558	0.22	3.9	0.0	3.2	YES
$z < 46$	$43\frac{504}{512}$	8297	25756	0.22	3.9	0.0	3.2	YES
$z < 43$	$42\frac{36}{512}$	8318	25826	0.25	3.9	0.0	3.2	YES
$z < 42$	$41\frac{368}{512}$	8311	25798	0.22	3.9	0.0	3.2	YES
$z < 41$	$40\frac{480}{512}$	8339	25889	0.22	3.9	0.0	3.2	YES
$z < 40$	-	8267	25652	0.22	3.9	0.0	3.2	NO
$z = 40$	-	7552	23311	0.20	3.7	0.0	3.0	NO

Table 2: Searching for 6-quasi-optimal run of Automaton 1 that leads from the initial state to a state containing the location $Target$. The g.i.l.b. of 6-optimal cost is equal to 40.

k:	**location**	value of **z**	**delay**	values of **x1**, **x2**
0:	$Start$	$< 0 + \frac{0}{512} >$	$< 0 + \frac{0}{512} >$	$< 0 + \frac{0}{512}, 0 + \frac{0}{512} >$
1:	$Start$	$< 0 + \frac{20}{512} >$	$< 0 + \frac{1}{512} >$	$< 0 + \frac{1}{512}, 0 + \frac{1}{512} >$
2:	W_2	$< 20 + \frac{20}{512} >$	$< 0 + \frac{0}{512} >$	$< 0 + \frac{0}{512}, 0 + \frac{1}{512} >$
3:	W_2	$< 40 + \frac{480}{512} >$	$< 1 + \frac{23}{512} >$	$< 1 + \frac{23}{512}, 1 + \frac{24}{512} >$
4:	L_2	$< 40 + \frac{480}{512} >$	$< 0 + \frac{0}{512} >$	$< 1 + \frac{23}{512}, 1 + \frac{24}{512} >$
5:	L_2	$< 40 + \frac{480}{512} >$	$< 0 + \frac{0}{512} >$	$< 1 + \frac{23}{512}, 1 + \frac{24}{512} >$
6:	$Target$	$< 40 + \frac{480}{512} >$	$< 0 + \frac{0}{512} >$	$< 1 + \frac{23}{512}, 1 + \frac{27}{512} >$

Table 3: A 6-quasi-optimal run of Automaton 1 leading to a state containing the location $Target$.

The optimal reachability problem was considered by many researchers and several approaches treating the problem in the context of timed or hybrid automata have been described in the literature, but none of them used SAT-methods. In particular, in [9] the problem of computing lower and upper bounds on time delays in timed automata was addressed. In [1] a *duration-bounded reachability* problem for timed automata augmented to include the duration cost function is considered. This problem asks if there is a run of the timed automaton from the initial state to the given final state such that the duration of the run satisfies an arithmetic constraint (an optimal cost).

The *duration-bounded reachability* problem has been also analysed in [10]. This is because the problem can be reduced to checking whether a duration formula, which defines an optimal cost, is satisfied by a integer computation of an integration graph (a kind of a timed automaton). The solution is based on constructing a set of equations that characterises the length of time a computation spends in each automaton location.

The work [4] also tackles the optimal (minimum-time) reachability problem for timed automata. In particular, here, the problem is formulated in terms of a timed game automaton (TGA), and solved by constructing an optimal strategy using a backward fixed-point calculation on the state-space of the TGA. Minimum-time reachability problem for timed automata is also solved in [12]. However here, the solution is based on the forward fixed-point algorithm that generates on-the-fly a forward reachability graph for a given timed automaton.

The paper [5] introduces *priced timed automata* as an extension of timed automata with prices on both transitions and locations, and shows how to solve the minimum cost reachability problem; this sort of automata we have used in the paper. In [3] such reachability problem is called as the single-source optimal reachability problem, and it is solved by a reduction of the problem to a parametric shortest-path problem. The methods presented in both papers [5] and [3] are based on clock region graphs; in [3] the authors refer to priced timed automata as weighted timed automata.

Further, the paper [7] addresses the optimal reachability problem for weighted timed automata with cost functions allowing for both positive and negative costs on edges and locations, and apply the proposed method to timed games. In [11] the decidability of the optimal (minimum and maximum cost) reachability problems for multi-priced timed automata (an extension of timed automata with multiple cost variables evolving according to given rates for each location) is proved, and in [8] cost-optimal infinite schedules in terms of minimal (or maximal) cost per time ratio in the limit is considered.

References

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

[2] R. Alur, D. Dill. A theory of Timed Automata. *Theoretical Computer Science*, **126** (2), 183–235, 1994.

[3] R. Alur, S. La Torre, G. J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, **318** (3), 297–322, 2004.

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

[5] G. Behrmann, A. Fehnker, T.S. Hune, K. G. Larsen, P. Pettersson, J. M. T. Romijn, F.W. Vaandrager, F. W. Va, G. Behrmann, A. Fehnker, T. Hune, K. Larsen, P. Pettersson, J. Romijn. Minimum-cost reachability for priced timed automata. In: *Proc. HSCC'01*, vol. 2034 of *LNCS*, pp. 147–161. Springer, 2001.

[6] A. Biere, A. Cimatti, E. Clarke, O. Strichman, Y. Zhu. Bounded model checking. In: *Highly Dependable Software*, vol. 58 of *Advances in Computers*. Academic Press, 2003. Pre-print.

[7] P. Bouyer, T. Brihaye, V. Bruyère, J. Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods System Design*, **31** (2), 135–175, 2007.

[8] P. Bouyer, E. Brinksma, K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, **32** (1), 3–23, 2008.

[9] C. Courcoubetis, M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, **1** (4), 385–415, 1992.

[10] Y. Kesten, A. Pnueli, J. Sifakis, S. Yovine. Decidable integration graphs. *Information and Computation*, **150** (2), 209–243, 1999.

[11] K. G. Larsen, J. I. Rasmussen. Optimal reachability for multi-priced timed automata. *Theoretical Computer Science*, **390**, 197–213, 2008.

[12] P. Niebert, S. Tripakis, S. Yovine. Minimum-time reachability for Timed Automata. In: *Proc. 8th IEEE Mediterranean Conf. on Control and Automation (MED'2000)*, Patros, Greece, July 2000.

[13] W. Penczek, B. Woźna, A. Zbrzezny. Bounded model checking for the universal fragment of CTL. *Fund. Informaticae*, **51** (1-2), 135–156, 2002.

[14] B. Woźna, A. Zbrzezny, W. Penczek. Checking reachability properties for Timed Automata via SAT. *Fund. Informaticae*, **55**, 223–241, 2003.

[15] A. Zbrzezny. SAT-based reachability checking for timed automata with diagonal constraints. *Fund. Informaticae*, **67** (1-3), 303–322, 2005.

[16] A. Zbrzezny. A boolean encoding of arithmetic operations. In: *Proc. Int. Workshop on Concurrency, Specification and Programming*, vol. 170 of *Informatik-Berichte*, pp. 536–547. Humboldt University, 2008.