SMT-based Searching for k-quasi-optimal Runs in Weighted Timed Automata[⋆]
Extended Abstract

Bozena Wozna-Szczesniak, Agnieszka M. Zbrzezny, and Andrzej Zbrzezny

IMCS, Jan Dlugosz University Al. Armii Krajowej 13/15, 42-200 Czestochowa, Poland
{b.wozna,agnieszka.zbrzezny,a.zbrzezny}@ajd.czest.pl

Abstract. We investigate an optimal cost reachability problem for weighted timed automata, and we use a translation to SMT 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 the 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 we call *k-quasi-optimal*. We exemplify the use of our solution to the mentioned problem by means of the *weighted timed generic pipeline protocol*, and we provide some preliminary experimental results.

1 Introduction

Reachability is a core decision problem that appears in several different contexts, for example: concurrent systems [8, 17], time critical systems [12], and probabilistic systems [10]. For computational models of time critical systems like *weighted timed automata* [2] (see Section 2 for the formal definition), or *priced timed automata* [5], it is reasonable to make inquiries about the minimum (optimal) cost of reaching a desirable state of the system, i.e., to investigated *the optimal reachability problem*.

The optimal reachability problem was considered by numerous researchers and several methodologies treating the issue in the setting of timed automata have been described in the literature, but none of them used SMT- or SAT-based methods (see the Related Work section). The acronym SMT means *satisfiability modulo theories*. SMT-solvers are tools for deciding the satisfiability of formulae in a number of theories [4]. Nevertheless, in the paper [19] we made the first attempt to solve, so called, the *k-optimal cost reachability problem* for weighted timed automata (see Section 3 for the formal definition). The proposed solution used the translation to SAT, and it could only be applied to systems modelled by a single weighted timed automaton.

In this paper we also deal with the *k-optimal cost reachability* problem, but for time critical systems modelled by a network of weighted timed automata. Moreover, we are interested in using SMT-based verification methods to solve the problem instead of the SAT-based verification. The use of a translation to SMT allows us to avoid an intermediate discretised model.

[⋆] Partly supported by National Science Centre under the grant No. 2014/15/N/ST6/05079.

The solution which we propose for the k-optimal cost reachability problem is the combination of a well-known *forward reachability* algorithm with the *bounded model checking* (BMC) method [18, 19] that uses SMT solvers instead of SAT solvers (see Section 3 for the informal and formal description of the solution). 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 SMT solvers.

How our solution to the k-optimal cost reachability problem is effective we show by means of the *weighted timed generic pipeline protocol* (see Section 4).

2 Preliminaries

Let us start by introducing the key sets of numbers and variables used in the rest of the paper. The first is the set $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$ of natural numbers. The second is the set $\mathbb{R}$ of non-negative real numbers, and the third is the set $\mathbb{R}_+$ of positive real numbers. The next is the set $\mathcal{PV}$ of propositional variables, and the final is a finite set $\mathcal{X}$ of real variables, called *clocks*.

For the set $\mathcal{X}$ of clocks, $x, y \in \mathcal{X}$, $c \in \mathbb{N}$ and $\sim \in \{\leq, <, =, >, \geq\}$, the set $\mathcal{C}(\mathcal{X})$ of all the *clock constraints* over $\mathcal{X}$ is defined by the following grammar:

$$\mathtt{cc} ::= true \mid x \sim c \mid x - y \sim c \mid \mathtt{cc} \wedge \mathtt{cc}$$

Furthermore, a *clock valuation* is a total mapping $\mathbf{cv} : \mathcal{X} \to \mathbb{R}$, and *satisfiability* of a clock constraint $\mathtt{cc} \in \mathcal{C}(\mathcal{X})$ by a clock valuation $\mathbf{cv}$ ($\mathbf{cv} \models \mathtt{cc}$) is defined inductively as follows:

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

Given a clock valuation $\mathbf{cv}$ and $\delta \in \mathbb{R}_+$, by $\mathbf{cv} + \delta$ we denote a clock valuation $\mathbf{cv}'$ such that $\mathbf{cv}'(x) = \mathbf{cv}(x) + \delta$, for all $x \in \mathcal{X}$. Moreover, for a subset of clocks $X \subseteq \mathcal{X}$, $\mathbf{cv}[X := 0]$ denotes the valuation $\mathbf{cv}'$ such that for all $x \in X$, $\mathbf{cv}'(x) = 0$ and for all $x \in \mathcal{X} \setminus X$, $\mathbf{cv}'(x) = \mathbf{cv}(x)$. Finally, by $\mathbf{cv}^0$ we denote the *initial* clock valuation, i.e., the valuation such that $\mathbf{cv}^0(x) = 0$ for all $x \in \mathcal{X}$.

Definition 1. *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 actions,* L *is a finite set of locations,* $l^0 \in L$ *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,* $\mathcal{I} : L \mapsto \mathcal{C}(\mathcal{X})$ *is an invariant function,* $J_s : \Sigma \mapsto \mathbb{N}$ *is a switch cost function,* $J_d : L \to \mathbb{N}$ *is a duration cost function,* z *is a real variable, and* $\mathcal{V} : L \mapsto 2^{\mathcal{PV}}$ *is a valuation function assigning to each location a set of propositional variables true in that location.*

The switch cost function assigns to each action a cost expressing the price of taking the action. 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 $t = (l, \sigma, \mathtt{cc}, X, l') \in E$ represents a transition from the location l to the location l', where σ is the action of the transition t, $\mathtt{cc}$ defines the enabling conditions for t, and X is a set of clocks to be reset.

Note that the syntax of weighted timed automata is borrowed from [2, 19], but it is extended to a special real variable z. Moreover, we have changed the domain of the switch cost function. Namely, we assume that the domain of the switch cost function is the set of actions instead of the set of transitions. Such a change is necessary, since we considered systems that are modelled through a network of weighted timed automata instead of a single weighted timed automaton.

In general, weighted timed automata are often composed into a network of weighted timed automata consisting of n weighted timed automata $\mathcal{A}_i = (\Sigma_i, L_i, l_i^0, \mathcal{X}_i, E_i, \mathcal{I}_i, J_s^i, J_d^i, z_i, \mathcal{V}_i)$, for $i = 1, \ldots, n$, that run in parallel and communicate via synchronised actions. A formal definition of a parallel composition of the weighted timed automata is the following.

Definition 2. *Let $\Sigma = \bigcup_{i=1}^n \Sigma_i$, $\sigma \in \Sigma$, and $\Sigma(\sigma) = \{1 \leq i \leq n \mid \sigma \in \Sigma_i\}$ be the set of indexes of automata that synchronise at σ, and $z_1 = \ldots = z_n$ (i.e., all the automata have to share the variable). For $i = 1, \ldots, n$,* a parallel composition of weighted timed automata *$\mathcal{A}_i$ is the weighted timed automaton $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$, where $\Sigma = \bigcup_{i=1}^n \Sigma_i$, $L = \prod_{i=1}^n L_i$, $l^0 = (l_1^0, \ldots, l_n^0)$, $\mathcal{X} = \bigcup_{i=1}^n \mathcal{X}_i$, a transition $((l_1, \ldots, l_n), \sigma, \bigwedge_{j \in \Sigma(\sigma)} \mathtt{cc}_j, \bigcup_{j \in \Sigma(\sigma)} X_j, (l_1', \ldots, l_m')) \in E$ iff $(\forall j \in \Sigma(\sigma))$ $(l_j, \sigma, \mathtt{cc}_j, X_j, l_j') \in E_j$ and $(\forall i \in \{1, \ldots, n\} \setminus \Sigma(\sigma))$ $l_i' = l_i$, $\mathcal{I}(l_1, \ldots, l_n) = \bigwedge_{i=1}^n \mathcal{I}_i(l_i)$, $J_s(\sigma) = \sum_{i \in \Sigma(\sigma)} J_s^i(\sigma)$, $J_d(l_1, \ldots, l_n) = \sum_{i=1}^n J_d^i(l_i)$, $z = z_1$, and $\mathcal{V}(l_1, \ldots, l_n) = \bigcup_{i=1}^n \mathcal{V}_i(l_i)$.*

The semantics of weighted timed automata is defined by associating to them *dense models* as defined below.

Definition 3. *Let $\mathcal{A} = (\Sigma, L, l^0, \mathcal{X}, E, \mathcal{I}, J_s, J_d, z, \mathcal{V})$ be a weighted timed automaton, $\mathbf{z} : \{z\} \to \mathrm{I\!R}$ a* valuation *for z, and $\mathbf{z}^0$ the* initial *valuation for z (i.e., $\mathbf{z}^0(z) = 0$). A* dense model *for $\mathcal{A}$ is the tuple $M = (\Sigma \cup \mathrm{I\!R}_+, S, s^0, \to, \mathcal{V}')$, where $\Sigma \cup \mathrm{I\!R}_+$ is a set of labels, $S = \{(l, \mathbf{cv}, \mathbf{z}) \mid l \in L,\ \mathbf{cv} \in \mathrm{I\!R}^{|\mathcal{X}|},\ \mathbf{cv} \models \mathcal{I}(l), \mathbf{z} \in \mathrm{I\!R}\}$ is a set of states, $s^0 = (l^0, \mathbf{cv}^0, \mathbf{z}^0)$ is the initial state, $\mathcal{V}' : S \to 2^{\mathcal{PV}}$ is a valuation function such that $\mathcal{V}'((l, \mathbf{cv}, \mathbf{z})) = \mathcal{V}(l)$, and $\to \subseteq S \times \Sigma \cup \mathrm{I\!R} \times S$ is the smallest transition relation defined by the following two rules:*

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

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 us denote by $s \overset{\delta,\sigma}{\rightarrow} s'$ the sequence of the following time and action transitions: $s \overset{\delta}{\rightarrow} s''$ and $s'' \overset{\sigma}{\rightarrow} s'$, where $\sigma \in \Sigma$, $\delta \in \mathbb{R}_+$, and $s, s', s'' \in S$.

Definition 4. *Let $k \in \mathbb{N}$, $i \in \{0, \ldots, k\}$, $s_i \in S$, $j \in \{1, \ldots, k\}$, $\sigma_j \in \Sigma$, and $\delta_j \in \mathbb{R}_+$. A k-*run *ρ of a weighted timed automaton $\mathcal{A}$ is a finite sequence of transition: $s_0 \overset{\delta_1,\sigma_1}{\rightarrow} s_1 \overset{\delta_2,\sigma_2}{\rightarrow} \ldots \overset{\delta_{k-1},\sigma_{k-1}}{\rightarrow} s_{k-1} \overset{\delta_k,\sigma_k}{\rightarrow} s_k$.*

In other words, a k-run is a finite path of $\mathcal{A}$, where action transitions are taken finitely often and time transitions are aggregated. Moreover, the k-run does not permit two consecutive actions to be performed one after the other; such a run is called *strongly monotonic*.

Given a k-run ρ of $\mathcal{A}$ and cost functions J_s and J_d, we associate cost to ρ as follows: $J_s(\rho) = \sum_{i=1}^{k} J_s(\sigma_i)$, and $J_d(\rho) = \sum_{i=1}^{k} \delta_i \cdot J_d(l_{i-1})$. Furthermore, the *total cost* associated to a k-run ρ is defined as $J(\rho) = J_d(\rho) + J_s(\rho)$. Finally, we recall the definitions of both the *k-optimal cost* and *k-quasi-optimal* that have been introduced in [19].

Definition 5. *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 $l'\}$.*

If $\lfloor J(\rho) \rfloor = \lfloor J_k^*(l, l') \rfloor$, then a k-run ρ from a state containing location l to a state containing location l' is 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 SMT-methods.

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 SMT-solvers. We begin with defining the problem, and then we describe our solution informally, which will help to understand the formal algorithm presented later on in this section.

Definition 6 (**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 *problem consists in finding out a k-quasi-optimal run ρ starting at $s^0 \in M$ and terminating at a state in M 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^0 and terminates at a state in M containing location l^p, $J(\rho') \geq c$ holds.

An informal explanation. To solve the k-optimal cost reachability problem we proceed as follows. We first encode by quantifier-free first-order formulae with individual variables ranging over the real numbers both the property p, and the unfolding of the transition relation of M up to depth k (for $k \in \mathbb{N}$). Let φ_k be the conjunction of the two above formulae. We test φ_k for satisfiability using an SMT-solver. If the test for φ_k is positive, we calculate the cost $r_0 \in \mathbb{R}$ 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 satisfiability test once again, but for the formula $\phi_k(c_0) = \varphi_k \wedge (z < c_0)$. If the test for $\phi_k(c_0)$ is positive, we calculate the cost $r_1 \in \mathbb{R}$ 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 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.

A formal algorithm for finding the g.i.l.b. of k-optimal cost. Algorithm 5 uses the procedure $checkSMT(\gamma)$ that for any given quantifier-free first-order formula γ returns a pair (W, X), where W denotes the valuation returned by a SMT 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 $checkSMT$ is not able to decide satisfiability of its argument within some preset timeout period, or has to terminate itself due to exhaustion of available memory. Algorithm 5 also uses 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. Finally, for a given quantifier-free first-order formula φ_k, the symbol $\phi_k(c)$ denotes the formula $\varphi_k \wedge (z < c)$, and the symbol $\psi_k(c)$ denotes the formula $\varphi_k \wedge (z = c)$.

Translation to quantifier-free first-order 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 that is a parallel composition of n weighted timed automata $\mathcal{A}_i$, $M = (\Sigma \cup \mathbb{R}, S, s^0, \rightarrow, \mathcal{V}')$ a dense model for $\mathcal{A}$, and $k \in \mathbb{N}$. Each state $s \in S$ of M can be represented by a valuation of a *symbolic state* $\mathbf{w} = ((\mathtt{l}_1, \ldots, \mathtt{l}_n), (\mathtt{x}_1, \ldots, \mathtt{x}_{|\mathcal{X}|}), (\mathtt{d}_1, \ldots, \mathtt{d}_n), \mathtt{z})$ that consists of *symbolic local states, symbolic clock valuations, symbolic duration costs*, and *symbolic valuation of variable* z. Each symbolic local state $\mathtt{l}_i$ ($1 \leq i \leq n$) is an individual variable ranging over the natural numbers. Each symbolic clock valuations $\mathtt{x}_i$ ($1 \leq i \leq |\mathcal{X}|$) is an individual variable ranging over the real numbers. Each symbolic duration cost $\mathtt{d}_i$ ($1 \leq i \leq n$) is an individual variable ranging over the natural numbers, and symbolic valuation of variable z is an individual variable ranging over the real numbers. Similarly, each action $\sigma \in \Sigma$ can be represented by a valuation of a *symbolic action* $\mathtt{a}$ that is an individual variable ranging over the natural numbers, each real number $r \in \mathbb{R}$ can be represented by a valuation of a *symbolic number* $\mathtt{r}$ that is an individual variable ranging over the real numbers, and finally each switch cost s can be represented by a valuation of a *symbolic switch cost* $\mathtt{s}$ that is an individual variable ranging over the natural numbers.

A finite sequence $(\mathbf{w}_0, \ldots, \mathbf{w}_k)$ of symbolic states is called a *symbolic k-path*.

Algorithm 5: An algorithm for finding g.i.l.b. of k-optimal cost

```
1: $k \leftarrow 0$
2: **repeat**
3:    $(W, result) \leftarrow checkSMT(\varphi_k)$
4:    **if** $result = FALSE$ **then**
5:       $k \leftarrow k + 2$
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:    $(W, result) \leftarrow checkSMT(\phi_k(c - 1))$
16:    **if** $result = TRUE$ **then**
17:       $c \leftarrow getCOST(W)$
18:    **else if** $result = UNKNOWN$ **then**
19:       **return** $UNKNOWN$
20:    **end if**
21: **until** $result = FALSE$
    {optimal cost of any $k$-run is greater or equal to $c$}
22: $(W, result) \leftarrow checkSMT(\psi_k(c))$
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**
```

For two symbolic states $\mathbf{w}, \mathbf{w}'$, we define the following propositional formulae:

- $I_s(\mathbf{w})$ is a formula that encodes the state s of M
- $p(\mathbf{w})$ is a formula that encodes the set of states of M in which $p \in \mathcal{PV}$ holds.
- $T_{\mathtt{a}}(\mathbf{w}, (\mathtt{a}, \mathtt{s}), \mathbf{w}')$ is a formula that encodes the action transition relation of M.
- $T_{\mathtt{r}}(\mathbf{w}, \mathtt{r}, \mathbf{w}')$ is a formula that encodes the time transition relation of M.

We can now define the quantifier-free first-order formula φ_k, introduced in the informal description section. The formula φ_k is a conjunction of two formulae. The first formula $p(\mathbf{w})$ is a translation of a propositional variable p that represents a location in question. The second formula $[M^{s^0}]_k$ encodes the unfolding of the transition relation of M up to depth $k \in \mathbb{N}$.

The formula $[M^{s^0}]_k$ is defined over symbolic states $\mathbf{w}_i$, symbolic actions $\mathtt{a}_i$, symbolic switch costs $\mathtt{s}_i$, and symbolic numbers $\mathtt{r}_i$, for $0 \leq i \leq k$, and it constrains the symbolic k-path to be valid k-run of M. Namely, let $T(\mathbf{w}_i, \mathbf{w}_{i+1})$=$T_{\mathtt{r}}(\mathbf{w}_i, \mathtt{r}_i, \mathbf{w}_{i+1})$ if i is even, and $T(\mathbf{w}_i, \mathbf{w}_{i+1})$=$T_{\mathtt{a}}(\mathbf{w}_i, (\mathtt{a}_i, \mathtt{s}_i), \mathbf{w}_{i+1})$ if i is odd. Then,

$$[M^{s^0}]_k := I_{s^0}(\mathbf{w}_0) \wedge \bigwedge_{i=0}^{k-1} T(\mathbf{w}_i, \mathbf{w}_{i+1})$$

4 Experimental Results

In this section we experimentally evaluate the performance of our SMT-based solution to the k-optimal cost reachability problem by means of the *weighted timed generic pipeline protocol* (WTGPP).

WTGPP. The WTGPP (adapted from [16]) consists of $n+2$ automata: Producer P that is able to produce data within certain time interval $[a, b]$ or being inactive , Consumer C that is able to receive data within certain time interval $[c, d]$, to consume data within certain time interval $[g, h]$ or being inactive, and a chain of n intermediate Nodes N_i which can be ready for receiving data within certain time interval $[c, d]$, processing data within certain time interval $[e, f]$, sending data, or being inactive. The local locations, the possible local actions, the local clocks, the clock constraints, invariants and the local transitions for each automaton are shown in Fig. 1. Moreover, we assume the following two switch cost functions for each automaton:

- $J_s^P(Produce) = 4$, $J_s^P(send_1) = 2$, $J_s^C(Consume) = 4$, $J_s^C(send_{n+1}) = 2$, $J_s^{N_i}(send_i) = J_s^{N_i}(send_{i+1}) = J_s^{N_i}(Proc_i) = 2$.
- $J_s^P(Produce) = 4000000$, $J_s^P(send_1) = 2000000$, $J_s^C(Consume) = 4000000$, $J_s^C(send_{n+1}) = 2000000$, $J_s^{N_i}(send_i) = J_s^{N_i}(send_{i+1}) = J_s^{N_i}(Proc_i) = 2000000$.

Finally, we assume the following duration cost function for each automaton:

- $J_d^P(ProdReady) = 4$, $J_d^C(ConsFree) = 4$, and
- $J_d^i(j) = 1$ for $j \in \{ProdSend, ConsStart, ConsReady\} \cup \bigcup_{m=1}^n L_{N_m}$, and $i \in \{P, C, N_1, \ldots, N_n\}$.

We can define the set of global states S for the protocol as the product $((L_P \times \prod_{i=1}^n L_{N_i} \times L_C) \times (\prod_{i=1}^{n+2} \mathbb{R}) \times \mathbb{R})$, and we consider the following initial state s^0 = $((ProdReady, Node_1Start, \ldots, Node_nStart, ConsStart), (\underbrace{0, \ldots, 0}_{\text{n+2}}), 0)$.

The example can be scaled by adding Nodes, or by changing the length of intervals (i.e., the parameters a, b, c, d, e, f, g, h) that are used to adjust the time properties of Producer P, Consumer C, and Nodes N_i ($i = 1, .., n$), or by changing the switch cost functions or by changing the duration cost functions.

It should be straightforward to infer the model that is induced by the above description of WTGPP. Next, in the dense model of the protocol we assume the following set of proposition variables: $\mathcal{PV} = \{ConsFree\}$, and the following definition of valuation functions for Consumer: $\mathcal{V}_C(ConsFree) = ConsFree$.

The k-optimal reachability property we consider is the following. Given a weighted timed automata model of the WTGPP system, decide whether there is a k-optimal run of the weighted timed automaton from the initial state of the system to the given global state of the system that contains the Consumer location $ConsFree$.

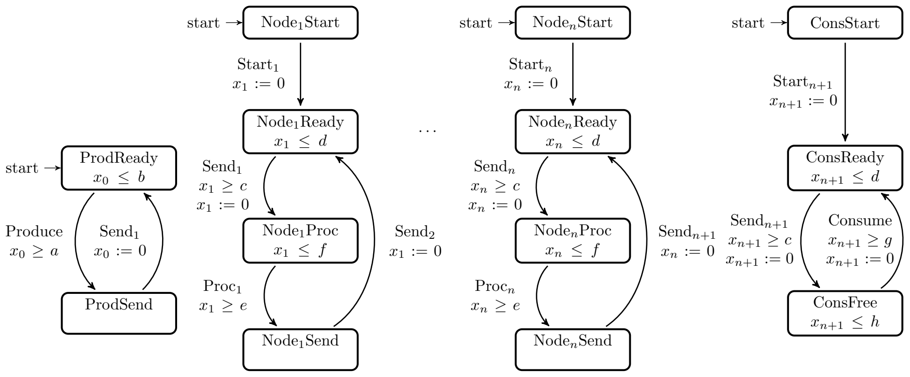

Fig. 1. A WTGPP protocol.

Performance evaluation. We have computed our experimental results on a computer equipped with I7-3770 processor, 32 GB of RAM, and the operating system Arch Linux with the kernel 3.15.3. Moreover, we used the state of the art SMT-solver Z3 [14].

In Tables 1 and 2 we present experimental results for the WTGPP system modelled by the network of automata on Figure 1 and for $k-$optimal reachability property. As can be seen from these tables, our method allows us to locate k-optimal path for 5 Nodes within a reasonable time. This result does not change if we drastically increase the values of the considered switch cost function, which may suggest that the effectiveness of our method does not depend on values of the switch cost function. In addition, a noticeable enormous increase in time for 5 Nodes suggests that a significant impact on the effectiveness of our method has a number of arithmetic operations performed, especially multiplication.

No. of		**BMC**		**Z3**		**BMC + Z3**	
Nodes	k	sec.	MB	sec.	MB	sec.	MB
1	12	0.0	1.8	0.2	15.5	0.2	15.5
2	18	0.0	1.8	0.7	17.2	0.7	17.2
3	26	0.0	1.9	7.3	22.3	7.3	22.3
4	34	0.2	2.1	211.6	33.6	211.8	33.6
5	44	0.3	2.1	7594.7	59.1	7595.0	59.1

Table 1. Time and memory used for k-optimal path and a number of Nodes n. Assumed the first switch cost function.

5 Conclusions and Related Work

In this paper we defined, solved, and implemented the k-optimal cost reachability problem for a network of weighted timed automata. The proposed solution is based on the reduction to the satisfiability problem of the quantifier-free first-order formulae, and it

Table 2. Time and memory used for k-optimal path and a number of Nodes n. Assumed the second switch cost function.

No. of		**BMC**		**Z3**		**BMC + Z3**	
Nodes	k	sec.	MB	sec.	MB	sec.	MB
1	12	0.0	1.8	0.1	15.5	0.1	15.5
2	18	0.0	1.8	0.5	17.2	0.5	17.2
3	26	0.0	1.9	6.4	22.6	6.4	22.6
4	34	0.1	2.1	270.5	33.8	270.7	33.8
5	44	0.3	2.1	17715.3	71.6	17715.6	71.6

uses an external tool that is the state of the art SMT-solver Z3 [14]. Experimental results, which we carried out, show that the proposed algorithm can be extremely helpful in finding g.i.l.b. of k-optimal cost.

Clearly, our method takes into account discovering just 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.

Related Work. The issue of processing lower and upper bounds on time delays in timed automata was addressed in [9]. A *duration-bounded reachability* issue for timed automata expanded to incorporate the duration cost function is considered in [1]. This issue inquires as to whether there is a run of the timed automaton from the initial state to the given last state such that the duration of the run fulfils an arithmetic requirement (an optimal cost). The *duration-bounded reachability* issue has been also analysed in [11]. This is because the problem can be reduced to checking whether a duration formula, which defines an optimal cost, is fulfilled by a integer computation of an integration graph (a sort 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 location of the given automaton.

The paper [3] also handles the optimal (minimum-time) reachability problem for timed automata. Specifically, here, the issue is formulated in terms of a timed game automaton (TGA), and solved by constructing an optimal strategy utilizing a backward fixed-point calculation on the state-space of the TGA. The minimum-time reachability problem for timed automata is likewise illuminated in [15]. 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] defines *priced timed automata* as an extension of timed automata with costs on both transitions and locations, and demonstrates how to solve the minimum cost reachability problem; this kind of automata we used in our paper. In [2] 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 exhibited in both papers [5] and [2] are taking into account clock region graphs; in [2] the authors refer to priced timed automata as weighted timed automata.

Furthermore, the paper [6] solves 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. Next, the paper [13] deals with 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). Finally, the paper [7] handles cost-optimal infinite schedules in terms of minimal (or maximal) cost per time ratio in the limit.

References

1. R. Alur, C. Courcoubetis, and T. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design*, 11(2):137–155, 1997.
2. R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318(3):297–322, June 8 2004.
3. E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *Proceedings of the 2nd International Workshop on Hybrid Systems: Computation and Control*, volume 1569 of *LNCS*, pages 19–30. Springer-Verlag, 1999.
4. Clark Barrett, Roberto Sebastiani, Sanjit Seshia, and Cesare Tinelli. Satisfiability modulo theories. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 26, pages 825–885. IOS Press, 2009.
5. G. Behrmann, A. Fehnker, T.S. Hune, K. G. Larsen, P. Pettersson, J. M. T. Romijn, F.W. Vaandrager, F. W. Va, Gerd Behrmann, Ansgar Fehnker, Thomas Hune, Kim Larsen, Paul Pettersson, and Judi Romijn. Minimum-cost reachability for priced timed automata. In *Proceedings of HSCC'01*, volume 2034 of *LNCS*, pages 147–161. Springer-Verlag, 2001.
6. P. Bouyer, T. Brihaye, V. Bruyère, and J. Raskin. On the optimal reachability problem of weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, 2007.
7. P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):3–23, 2008.
8. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
9. C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in real-time systems. *Formal Methods in System Design*, 1(4):385–415, 1992.
10. P.R. D'Argenio, B. Jeannet, H. E. Jensen, and K.G. Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, volume 2165 of *LNCS*, pages 39–56. Springer-Verlag, 2001.
11. Y. Kesten, A. Pnueli, J. Sifakis, and S. Yovine. Decidable integration graphs. *Information and Computation*, 150(2):209–243, 1999.
12. R. Koymans. *Specifying Message Passing and Time-Critical Systems with Temporal Logic*, volume 651 of *LNCS*, chapter Time-critical systems, pages 99–142. Springer Berlin Heidelberg, 1992.
13. K. G. Larsen and J. I. Rasmussen. Optimal reachability for multi-priced timed automata. *Theoretical Computer Science*, 390(2-3):197–213, 2008.
14. L. De Moura and N. Bjørner. Z3: an efficient SMT solver. In *Proceedings of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'2008)*, volume 4963 of *LNCS*, pages 337–340. Springer-Verlag, 2008.
15. P. Niebert, S. Tripakis, and S. Yovine. Minimum-time reachability for Timed Automata. In *Proceedings of the 8th IEEE Mediterranean Conference on Control and Automation (MED'2000)*, Patros, Greece, July 2000. IEEE Computer Society.
16. D. Peled. All from one, one for all: On model checking using representatives. In *Proceedings of the 5th International Conference on Computer Aided Verification (CAV'93)*, volume 697 of *LNCS*, pages 409–423. Springer-Verlag, 1993.
17. A.W. Roscoe. *Understanding Concurrent Systems*. Springer-Verlag, 2010.

18. B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for Timed Automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.
19. B. Woźna-Szcześniak and A. Zbrzezny. SAT-based searching for k-quasi-optimal runs in weighted timed automata. *Scientific Issues of Jan Długosz University in Częstochowa: Mathematica*, XV:149–162, 2010.