Minimization of the Number of Clocks for Timed Scenarios

Neda Saeedloei[1] and Feliks Kluźniak[2]

[1] Towson University
nsaeedloei@towson.edu
[2] LogicBlox
feliks.kluzniak@logicblox.com

Abstract. We present a new optimization algorithm for timed scenarios that achieves the minimal number of clocks when timed scenarios are viewed as timed automata.

1 Introduction

Using scenarios for specification and implementation of complex systems (including real time systems), and synthesizing formal models of systems from scenarios have been active areas of research for several decades [15, 8, 9, 16, 2, 7, 5, 10, 11].

In our earlier work [12] we developed, from first principles, a formal, yet simple notation for timed scenarios. We want to use such scenarios to automatically synthesize formal models in the form of timed automata.

For a timed automaton with $|K|$ clocks, the number of clock regions is at most $R = |K|!4^{|K|}\Pi_{x\in K}(\mu_x + 1)$, where μ_x is the maximum constant with which clock x is compared [4]. Verification of a timed automaton can be computationally expensive, and the cost depends on the number of regions of the automaton.

So, we want the value of R in our synthesized automaton to be as low as possible. Before tackling that problem it behooves us to study it in the more limited setting of a single scenario: this is the topic of the current paper.

As part of our earlier work [12] we obtained a canonical representation (a "stable distance table") for the entire class of scenarios that are equivalent to a given one. We used stable tables as a linchpin of a very simple algorithm for "optimizing" scenarios ("optimization" was used in the sense of "improvement", but did not necessarily lead to a result that was optimal in some sense).

The goal of that algorithm was to decrease the number of constraints as well as the maximum constants appearing in constraints.

In our later work [13] we studied optimization of scenarios in more depth and developed a general algorithm that would optimize scenarios according to some given strategy. Our main goal was to minimise the maximum constants associated with the clocks that would be needed after converting a scenario to a timed automaton, while decreasing the number of clocks. Similarly to our earlier algorithm, this new algorithm relied on some particular ordering of constraints. While we had more control over—and insight into—how the algorithm worked,

we did not obtain an entirely satisfactory result: we did not achieve minimality in the number of clocks.

The current paper summarizes our final approach to optimizing scenarios. Given a scenario, our new optimization algorithm replaces the time constraints of the scenario, represented by a stable distance table, with an equivalent set that would require the smallest number of clocks in the entire class of equivalent scenarios, when the scenarios are viewed as timed automata.

2 Preliminaries

2.1 Timed automata

A *timed automaton* [3] is a tuple $\mathcal{A} = \langle \Sigma, Q, q_0, Q_f, C, T \rangle$, where Σ is a finite alphabet, Q is the (*finite*) set of locations, $q_0 \in Q$ is the initial location, $Q_f \subseteq Q$ is the set of final locations, C is a finite set of *clock* variables (clocks for short), and $T \subseteq Q \times Q \times \Sigma \times 2^C \times 2^{\Phi(C)}$ is the set of transitions. In each transition $(q, q', e, \lambda, \phi)$, λ is the set of clocks to be reset with the transition and $\phi \subset \Phi(C)$ is a set of clock constraints over C of the form $c \sim a$ (where $\sim \in \{\leq, <, \geq, >, =\}$, $c \in C$ and a is a constant in the set of rational numbers, $\mathbb{Q}$).

A *clock valuation* ν for C is a mapping from C to $\mathbb{R}^{\geq 0}$. ν *satisfies* a set of clock constraints ϕ over C iff every clock constraint in ϕ evaluates to true after each clock c is replaced with $\nu(c)$. For $\tau \in \mathbb{R}, \nu + \tau$ denotes the clock valuation which maps every clock c to the value $\nu(c) + \tau$. For $Y \subseteq C$, $[Y \mapsto \tau]\nu$ is the valuation which assigns τ to each $c \in Y$ and agrees with ν over the rest of the clocks.

A *timed word* over an alphabet Σ is a pair (σ, τ) where $\sigma = \sigma_1\sigma_2...$ is a finite [1, 6] or infinite [3] word over Σ and $\tau = \tau_1\tau_2...$ is a finite or infinite sequence of (time) values such that (i) $\tau_i \in \mathbb{R}^{\geq 0}$, (ii) $\tau_i \leq \tau_{i+1}$ for all $i \geq 1$, and (iii) if the word is infinite, then for every $t \in \mathbb{R}^{\geq 0}$ there is some $i \geq 1$ such that $\tau_i > t$.

A run ρ of $\mathcal{A}$ over a timed word (σ, τ) is a sequence of the form $\langle q_0, \nu_0 \rangle \xrightarrow[\tau_1]{\sigma_1} \langle q_1, \nu_1 \rangle \xrightarrow[\tau_2]{\sigma_2} \langle q_2, \nu_2 \rangle \xrightarrow[\tau_3]{\sigma_3} \ldots$, where for all $i \geq 0$, $q_i \in Q$ and ν_i is a clock valuation such that (i) $\nu_0(c) = 0$ for all clocks $c \in C$ and (ii) for every $i > 1$ there is a transition in T of the form $(q_{i-1}, q_i, \sigma_i, \lambda_i, \phi_i)$, such that $(\nu_{i-1} + \tau_i - \tau_{i-1})$ satisfies ϕ_i, and ν_i equals $[\lambda_i \mapsto 0](\nu_{i-1} + \tau_i - \tau_{i-1})$. The set $inf(\rho)$ consists of $q \in Q$ such that $q = q_i$ for infinitely many $i \geq 0$ in the run ρ.

A run over a finite timed word is *accepting* if it ends in a final location [6]. A run ρ over an infinite timed word is *accepting* iff $inf(\rho) \cap Q_f \neq \emptyset$ [3]. The *language* of $\mathcal{A}$, $L(\mathcal{A})$, is the set $\{(\sigma, \tau) \mid \mathcal{A}$ has an accepting run over $(\sigma, \tau)\}$.

2.2 Timed scenarios

(This subsection briefly recounts our earlier work [12, 13].)

Let Σ be a finite set of symbols called *events*. A *behaviour*[1] over Σ is a sequence $(e_0, t_0)(e_1, t_1)(e_2, t_2)\ldots$, such that $e_i \in \Sigma$, $t_i \in \mathbb{R}^{\geq 0}$ and $t_{i-1} \leq t_i$ for

[1] The notion of "behaviour" is equivalent to that of Alur's "timed word" [3].

ξ	η
$L_0:\ a;$ $L_1:\ b;$ $L_2:\ c\ \{L_0 \le 9, L_0 \ge 9, L_1 \le 3\};$ $\quad d\ \{L_2 \le 2\}.$	$L_0:\ a;$ $\quad b\ \{L_0 \ge 6\};$ $L_2:\ c\ \{L_0 \le 9, L_0 \ge 9\};$ $\quad d\ \{L_2 \le 2\}.$

	1	2	3
0	(6, 9)	(9, 9)	(9, 11)
1		(0, 3)	(0, 5)
2			(0, 2)

Fig. 1. Two equivalent scenarios with their stable table

$i \in \{1, 2 \dots\}$. For a finite behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1) \dots (e_{n-1}, t_{n-1})$ of length n, and for any $0 \le i < j < n$, we use $t^{\mathcal{B}}_{ij}$ to denote the *distance*, in time units, of event j from event i in $\mathcal{B}$. That is, $t^{\mathcal{B}}_{ij} = t_j - t_i$.

Given a natural number n, we use $\Phi(n)$ to denote the set of *constraints* of the form $b \sim a$, where b is the symbol $\tau_{i,j}$ (for some integers $0 \le i < j < n$), $\sim \in \{\le, \ge\}$[2] and a is a constant in the set of rational numbers, $\mathbb{Q}$. A *timed scenario* (*scenario* for short) of length $n \in \mathbb{N}$ over Σ is a pair $(\mathcal{E}, \mathcal{C})$, where $\mathcal{E} = e_o e_1 ... e_{n-1}$ is a sequence of events, and $\mathcal{C} \subset \Phi(n)$ is a finite set of constraints.

A scenario will be written as a sequence of events, separated by semicolons and terminated by a period. If the scenario contains a constraint such as $\tau_{i,j} \le a$, then event i in the sequence will be labelled by a unique symbol L_i, and event j will be annotated with a set of constraints that contains $L_i \le a$. We refer to this as the external representation of the scenario. η in Fig. 1 is the external representation of scenario $(abcd, \{\tau_{0,1} \ge 6, \tau_{0,2} \le 9, \tau_{0,2} \ge 9, \tau_{2,3} \le 2\})$.

A behaviour $\mathcal{B} = (e_0, t_0)(e_1, t_1) \dots (e_{n-1}, t_{n-1})$ over Σ is *allowed* by scenario $\xi = (\mathcal{E}, \mathcal{C})$ iff $\mathcal{E} = e_0 \dots e_{n-1}$ and every $\tau_{i,j} \sim a$ in $\mathcal{C}$ evaluates to true after $\tau_{i,j}$ is replaced by $t^{\mathcal{B}}_{ij}$.

The constraints $\tau_{i,j} \ge 0$ and $\tau_{i,j} \le \infty$, which always evaluate to true after we replace them with some $t^{\mathcal{B}}_{ij}$, will be called *default constraints*.

The *semantics* of scenario ξ, denoted by $[\![\xi]\!]$, is the set of behaviours that are allowed by ξ. For scenario η in Fig. 1 $[\![\eta]\!] = \{(a, t_0)(b, t_1)(c, t_2)(d, t_3) \mid t_3 \ge t_2 \ge t_1 \ge t_0 \wedge t_1 - t_0 \ge 6 \wedge t_2 - t_0 \le 9 \wedge t_2 - t_0 \ge 9 \wedge t_3 - t_2 \le 2\}$.

A scenario ξ is *consistent* iff $[\![\xi]\!] \neq \emptyset$. It is *inconsistent* iff $[\![\xi]\!] = \emptyset$.

If $\xi = (\mathcal{E}, \mathcal{C})$ is a scenario of length n, and $\mathcal{C}$ contains a constraint $\tau_{i,j} \sim a$ for some $0 \le i < j < n$, then the index i is an *anchor*. We sometimes say constraint $\tau_{i,j} \sim a$ "begins" at anchor i. For an anchor i, if $0 < j < n$ is the largest number such that $\tau_{i,j} \sim a$ is a constraint in $\mathcal{C}$, then $[i, j)$ is the *range* of anchor i. If i_1 and i_2 are two anchors with ranges $[i_1, j_1)$ and $[i_2, j_2)$ in ξ, then the two ranges *overlap* iff $i_1 < i_2 < j_1$ or $i_2 < i_1 < j_2$.

An anchor i in ξ corresponds to a referenced label in the external representation of ξ. For example, in scenario η of Fig. 1, i.e., $(abcd, \{\tau_{0,1} \ge 6, \tau_{0,2} \le 9, \tau_{0,2} \ge 9, \tau_{2,3} \le 2\})$, the anchors 0 and 2 correspond to labels L_0 and L_2, respectively. The range of 0 is $[0, 2)$ and the range of 2 is $[2, 3)$: these are non-overlapping.

By $Anch_\xi$ we denote the set of anchors of ξ. We assume the existence of a set X of clock variables. A clock allocation for ξ is a relation $alloc_\xi \subset Anch_\xi \times X$.

[2] To keep the presentation compact, equality is expressed in terms of $\le$ and $\ge$ [12].

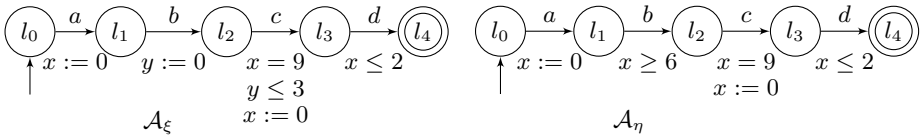

Fig. 2. Two equivalent timed automata corresponding to the scenarios of Fig. 1

$alloc_\xi$ is *complete* iff for every anchor $i \in Anch_\xi$ there is a clock $x \in X$ such that $(i, x) \in alloc_\xi$. $alloc_\xi$ is *incorrect* iff there exist two different anchors i and j in $Anch_\xi$ whose ranges overlap, such that $(i, x) \in alloc_\xi$ and $(j, x) \in alloc_\xi$ for some $x \in X$. $alloc_\xi$ is *correct* iff it is not incorrect. A correct and complete clock allocation is *optimal* if there is no other correct and complete allocation that uses fewer clocks.

$\{(0, x), (2, x)\}$ is an optimal clock allocation for scenario η of Fig. 1.

For a scenario $\xi = (e_0 e_1 \dots e_{n-1}, \mathcal{C})$ and an optimal clock allocation $alloc_\xi$, its corresponding timed automaton, $\mathcal{A}_\xi$, is defined as follows: $\{l_0, l_1, \dots l_n\}$ is the set of locations of $\mathcal{A}_\xi$; l_0 is the initial location and l_n is the final location; there is a transition r_i from l_i to l_{i+1} labeled with e_i, for each $0 \le i < n$; $K = \{x \mid \exists_{i \in Anch_\xi}(i, x) \in alloc_\xi\}$ is the set of clocks of $\mathcal{A}_\xi$; if $(i, x) \in alloc_\xi$, then there is a clock reset $x := 0$ on transition r_i; if $\tau_{i,j} \sim a$ is a constraint in $\mathcal{C}$ and $(i, x) \in alloc_\xi$, then there is a clock constraint $x \sim a$ on transition r_j.

The two automata of Fig. 2 correspond to scenarios of Fig. 1.

For a consistent scenario ξ of length n, and for $0 \le i < j < n$, we define $m^\xi_{ij} = min\{t^\mathcal{B}_{ij} \mid \mathcal{B} \in [\![\xi]\!]\}$ and $M^\xi_{ij} = max\{t^\mathcal{B}_{ij} \mid \mathcal{B} \in [\![\xi]\!]\}$. If there is no upper bound for i and j we will use $M^\xi_{ij} = \infty$. We will often write just m_{ij} and M_{ij} when ξ is understood. Obviously, for any behaviour in $[\![\xi]\!]$, $0 \le m_{ij} \le t_{ij} \le M_{ij} \le \infty$.

For a consistent scenario ξ of length n, and for any $0 \le i < j < k < n$ the following inequations hold:

$$m_{ij} + m_{jk} \le m_{ik} \le \left\{ \begin{matrix} m_{ij} + M_{jk} \\ M_{ij} + m_{jk} \end{matrix} \right\} \le M_{ik} \le M_{ij} + M_{jk} \tag{1}$$

Let $\xi = (\mathcal{E}, \mathcal{C})$ be a scenario of length n, such that, for any $0 \le i < j < n$, C contains at most one constraint of the form $\tau_{i,j} \ge c$ and at most one of the form $\tau_{i,j} \le c$. A *distance table* for ξ is a representation of $\mathcal{C}$ in the form of a triangular matrix $\mathcal{D}^\xi$. For $0 \le i < j < n$, $\mathcal{D}^\xi_{ij} = (l_{ij}, h_{ij})$, where l_{ij} and h_{ij} are rational numbers. If $\tau_{i,j} \ge c \in \mathcal{C}$ then $l_{ij} = c$, otherwise $l_{ij} = 0$; if $\tau_{i,j} \le c \in \mathcal{C}$ then $h_{ij} = c$, otherwise $h_{ij} = \infty$. (See the example in Fig. 3.)

A distance table of size n is *valid* iff $l_{ij} \le h_{ij}$, for all $0 \le i < j < n$. A table that is not valid is *invalid*. If $\mathcal{D}^\xi$ is invalid, then ξ is obviously inconsistent.

A distance table of size n is *stable* iff, for all $0 \le i < j < k < n$, the inequations in (1) hold when m_{ij}, m_{jk}, m_{ik} are replaced by l_{ij}, l_{jk}, l_{ik} and M_{ij}, M_{jk}, M_{ik} are replaced by h_{ij}, h_{jk}, h_{ik}. If $\mathcal{D}^\xi$ is stable then ξ is consistent.

L_0 : e;
L_1 : f;
L_2 : g $\{L_0 \leq 10, L_0 \geq 10\}$;
h $\{L_1 \leq 8, L_1 \geq 8, L_2 \leq 2\}$.

	1	2	3
0	(0, ∞)	(10, 10)	(0, ∞)
1		(0, ∞)	(8, 8)
2			(0, 2)

	1	2	3
0	(2, 4)	(10, 10)	(10, 12)
1		(6, 8)	(8, 8)
2			(0, 2)

Fig. 3. A scenario, its initial distance table and its stable distance table

To *stabilise* $\mathcal{D}^\xi$ we repeatedly apply the following six rules until the table becomes either invalid or stable.

$$l_{ij} + l_{jk} > l_{ik} \longrightarrow l_{ik} := l_{ij} + l_{jk} \qquad l_{ik} > l_{ij} + h_{jk} \longrightarrow l_{ij} := l_{ik} - h_{jk}$$
$$l_{ik} > h_{ij} + l_{jk} \longrightarrow l_{jk} := l_{ik} - h_{ij} \qquad l_{ij} + h_{jk} > h_{ik} \longrightarrow h_{jk} := h_{ik} - l_{ij}$$
$$h_{ij} + l_{jk} > h_{ik} \longrightarrow h_{ij} := h_{ik} - l_{jk} \qquad h_{ik} > h_{ij} + h_{jk} \longrightarrow h_{ik} := h_{ij} + h_{jk}$$

At least one of these rules is applicable if and only if some inequation in (1) does not hold. The purpose of each rule is to tighten a constraint just enough to establish a particular inequation.

A stabilised distance table has two properties. First, all the constraints represented by the table are as *tight* as possible, i.e., $l_{ij} = m_{ij}$ and $h_{ij} = M_{ij}$ for every $0 \leq i < j < n$. Second, as a result of applying the rules above, the table includes all the constraints that are "implied" by the initial set of constraints.

The right hand side of Fig. 1 shows the stable distance table obtained from the constraints of either of the two scenarios in the figure (which shows that they are equivalent). The right-hand side of Fig. 3 shows the result of stabilising the original distance table (shown in the middle).

Given a scenario ξ with its stable distance table $\mathcal{D}_s^\xi$, we use $\mathcal{C}(\mathcal{D}_s^\xi)$ to denote the set of constraints represented by $\mathcal{D}_s^\xi$.

Definition 1. *Let ξ be a scenario of length n, $\mathcal{D}_s^\xi$ be its stable table, $c \in \mathcal{C}(\mathcal{D}_s^\xi)$ be a non-default constraint, $S \subset \mathcal{C}(\mathcal{D}_s^\xi)$, and $0 \leq i < j < k < n$. We say that c is* directly supported *by S, denoted by $S \rightsquigarrow c$, iff c and S satisfy one of the following six conditions:*

1. $c = \tau_{i,k} \geq u$, $S = \{\tau_{i,j} \geq v, \tau_{j,k} \geq w\}$, *and* $u = v + w$.
2. $c = \tau_{i,j} \geq u$, $S = \{\tau_{i,k} \geq v, \tau_{j,k} \leq w\}$, *and* $u = v - w$.
3. $c = \tau_{j,k} \geq u$, $S = \{\tau_{i,k} \geq v, \tau_{i,j} \leq w\}$, *and* $u = v - w$.
4. $c = \tau_{j,k} \leq u$, $S = \{\tau_{i,k} \leq v, \tau_{i,j} \geq w\}$, *and* $u = v - w$.
5. $c = \tau_{i,j} \leq u$, $S = \{\tau_{i,k} \leq v, \tau_{j,k} \geq w\}$, *and* $u = v - w$.
6. $c = \tau_{i,k} \leq u$, $S = \{\tau_{i,j} \leq v, \tau_{j,k} \leq w\}$, *and* $u = v + w$.

Each of the cases in the definition corresponds to one of the six rules above. For example, if $l_{13} = 3$, $l_{36} = 4$ and $l_{16} = 0$ (i.e., the corresponding constraint is missing), then the first rule will force l_{16} to be 7. In other words, the constraint $l_{16} = 7$ is directly supported (i.e., "implied") by the other two constraints.

Definition 2. *(quasi-transitivity) Let $\mathcal{D}_s^\xi$ be a stable table. $\rightsquigarrow^+ \subset 2^{\mathcal{C}(\mathcal{D}_s^\xi)} \times \mathcal{C}(\mathcal{D}_s^\xi)$ is the smallest relation that satisfies the following two conditions:*

1. *If* $S \rightsquigarrow c$ *then* $S \rightsquigarrow^+ c$*;*
2. *If* $S \rightsquigarrow^+ c$ *and there is a constraint* $d \in S$ *such that, for some* S'*,* $S' \rightsquigarrow^+ d$ *and* $c \notin S'$*, then* $(S \setminus \{d\}) \cup S' \rightsquigarrow^+ c$*.*

When $S \rightsquigarrow^+ c$*, we say that* c *is* supported *by* S*.* S *is then called a* support *of* c*.*

We sometimes say that c *has a support*, when there is no need to specify S.

Intuitively, if a constraint d has a supporting set, then d can be removed from the scenario, because stabilisation of the distance table will restore it: d is implied by its support. The removed d can be a member of the supports of other constraints, e.g., d can appear in a set S that supports c. As long as d has a support S' that does not include c, S can be updated by replacing d with S'. The relation $\rightsquigarrow^+$ captures all the possible supports for the constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$. As constraints are removed, the set of constraints will decrease and $\rightsquigarrow^+$ (as well as other relation derived from it) should be understood as restricted to the current set of constraints: we will not be pedantic about capturing it in the notation.

Observation 1 *If* $S \rightsquigarrow^+ c$*, then* $c \notin S$*.*

3 A new optimization algorithm

Given a collection of scenarios, our ultimate goal is to synthesize a timed automaton [3] whose language would be the set of behaviours allowed by appropriate combinations of the scenarios [14]. The first step towards that goal is optimization of single scenarios.

A timed scenario ξ can be trivially converted to a simple timed automaton $\mathcal{A}_\xi$ (Sec. 2.1). An equivalent scenario η would yield a timed automaton $\mathcal{A}_\eta$. $\mathcal{A}_\xi$ and $\mathcal{A}_\eta$ will be language-equivalent, but each of them might have a different number of clocks. As an example consider the two equivalent scenarios of Fig. 1 along with their corresponding equivalent automata shown in Fig. 2. Notice that $\mathcal{A}_\eta$ has only one clock, while $\mathcal{A}_\xi$ requires two clocks.

Our task is as follows: given a timed scenario ξ, find an equivalent scenario ξ' such that the number of clocks in $\mathcal{A}_{\xi'}$ is minimal in the class of automata that are obtained from all scenarios equivalent to ξ.

We do so by starting from a scenario that contains $\mathcal{C}(\mathcal{D}_s^\xi)$ (i.e., all the constraints that are implied by the constraints of the original scenario ξ), and then successively removing constraints that are implied by other constraints. From a bird's eye perspective, the process can be described as follows (see Sec. 3.5 for details):

- Let $C = \mathcal{C}(\mathcal{D}_s^\xi)$;
- While there is a $c \in C$ such that $\exists_{S \subset C}(S \rightsquigarrow^+ c)$:
  - remove c from C;
- C is the solution, i.e., the final set of constraints.

This involves making choices, and the solution may depend on how these choices are made. For example, removing constraint a may prevent us from removing constraint b, and vice versa.

In the remainder of this section we carefully analyze the situations that may arise during this process, and formulate a set of rules that, when applied, guarantee reaching a solution that is optimal in terms of the number of clocks.

We begin by examining cyclic dependencies, because it turns out they are the only source of those choices that actually affect the outcome.

3.1 Cyclic dependencies

Definition 3. *Let $\mathcal{D}_s^\xi$ be a stable table and let a and b be constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$. A* support of b with respect to a, *denoted by $S_a^{\triangleright b}$, is any set $S \subset \mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \in S$ and $S \rightsquigarrow^+ b$.*

Given a and b, $S_b^{\triangleright a}$ need not be unique. Notice that $a \notin S_b^{\triangleright a}$ (Observation 1).

Definition 4. *If there is some $S_a^{\triangleright b}$, then we say that b* depends *on a, and write $a \to b$.*

Observation 2 *Let a, b and c be three different constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$, such that $a \to b$ and $b \to c$. If there is some $S_a^{\triangleright b}$ that does not contain c, then $a \to c$.*

Proof. The above follows directly from Definitions 4, 3 and 2. □

Definition 5. *We say that a and b are in a* cyclic dependency, *denoted by $a \leftrightarrow b$, if $a \to b$ and $b \to a$.*

As an example consider scenario ξ of Fig. 1 along with its distance table. Let $a = \tau_{0,1} \geq 6$, $b = \tau_{1,2} \leq 3$ and $c = \tau_{1,3} \leq 5$. $\{\tau_{0,2} \leq 9, \tau_{0,1} \geq 6\} \rightsquigarrow \tau_{1,2} \leq 3$, so we have a $S_a^{\triangleright b}$, hence $a \to b$. Moreover, $\{\tau_{1,2} \leq 3, \tau_{2,3} \leq 2\} \rightsquigarrow \tau_{1,3} \leq 5$, so we have a $S_b^{\triangleright c}$, hence $b \to c$. Since we have a $S_a^{\triangleright b}$ that does not contain c, we can conclude that $a \to c$.

Also, $a \leftrightarrow b$: let $S_a^{\triangleright b}$ be $\{\tau_{0,2} \leq 9, \tau_{0,1} \geq 6\}$ and let $S_b^{\triangleright a}$ be $\{\tau_{0,2} \geq 9, \tau_{1,2} \leq 3\}$.

Observation 3 *Let a, b and c be three different constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$, such that $a \to b$ and $b \to c$. If, additionally, for every pair of constraints x and y, $x \not\leftrightarrow y$, then $a \to c$.*

Proof. Assume the condition of Observation 2 is not satisfied, i.e., c is present in every $S_a^{\triangleright b}$. There is at least one $S_a^{\triangleright b}$, but if it includes c then it is also $S_c^{\triangleright b}$. Therefore $b \leftrightarrow c$: a contradiction. □

Observation 4 *Let $\mathcal{D}_s^\xi$ be a stable distance table. Let a, b and c be three different constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \leftrightarrow b$ and $b \leftrightarrow c$. Then $a \leftrightarrow c$ if there is some $S_a^{\triangleright b}$ that does not contain c and some $S_c^{\triangleright b}$ that does not contain a.*

Proof. This follows directly from Observation 2 and Definition 5. □

In the scenario of Fig. 3 let $a = \tau_{0,1} \leq 4$, $b = \tau_{1,2} \geq 6$, and $c = \tau_{2,3} \leq 2$.

$\{\tau_{0,2} \leq 10, \tau_{1,2} \geq 6\} \rightsquigarrow \tau_{0,1} \leq 4$, and $\{\tau_{0,2} \geq 10, \tau_{0,1} \leq 4\} \rightsquigarrow \tau_{1,2} \geq 6$. So we have a $S_b^{\triangleright a}$ and a $S_a^{\triangleright b}$. That is, $a \leftrightarrow b$. On the other hand, we also have a $S_c^{\triangleright b}$ and a $S_b^{\triangleright c}$: $\{\tau_{1,3} \geq 8, \tau_{2,3} \leq 2\} \rightsquigarrow \tau_{1,2} \geq 6$ and $\{\tau_{1,3} \leq 8, \tau_{1,2} \geq 6\} \rightsquigarrow \tau_{2,3} \leq 2$. So $b \leftrightarrow c$. Notice that the $S_a^{\triangleright b}$ does not include c, while the $S_c^{\triangleright b}$ does not include a. By Observation 4, $a \leftrightarrow c$. Indeed, $S_c^{\triangleright a}$ and $S_a^{\triangleright c}$ exist: $\{\tau_{0,2} \leq 10, \tau_{1,3} \geq 8, \tau_{2,3} \leq 2\} \rightsquigarrow^+ \tau_{0,1} \leq 4$ and $\{\tau_{1,3} \leq 8, \tau_{0,2} \geq 10, \tau_{0,1} \leq 4\} \rightsquigarrow^+ \tau_{2,3} \leq 2$.

Observation 5 *Let $\mathcal{D}_s^\xi$ be a stable distance table. Let a, b and c be three different constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \leftrightarrow b$ and $a \not\leftrightarrow c$. If $a \rightarrow c$, then, after a is removed, c is still supported.*

Proof. $a \leftrightarrow b$, so there is some $S_b^{\triangleright a}$: call it S_1. Clearly, $c \notin S_1$, otherwise we would have $c \rightarrow a$, and therefore $a \leftrightarrow c$.

$a \rightarrow c$, so there is a set $S \subset \mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \in S$ and $S \rightsquigarrow^+ c$. $c \notin S$ (by Observation 1). So, by Definition 2, $(S \setminus \{a\}) \cup S_1 \rightsquigarrow^+ c$. □

In other words: if two constraints a and b are in a cyclic dependency and during the course of optimization one of them, say a, is removed, then all the constraints that were previously supported by a continue to be supported. The only exception is b: if b has only one support which is its support with respect to a, then b can no longer be removed after the removal of a.

Obviously, if two constraints are in a cyclic dependency, either one of them can be removed. However, both of them cannot be removed unless at least one of them has a support other than its supports with respect to the other constraint.

Observation 6 *Let $\mathcal{D}_s^\xi$ be a stable distance table. Let a and b be two constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \leftrightarrow b$. If there is some $S \subset \mathcal{C}(\mathcal{D}_s^\xi)$ such that $S \rightsquigarrow a$ and $b \notin S$, then both a and b can be removed.*

Proof. $a \leftrightarrow b$, so there is a $S_a^{\triangleright b}$: call it S_1.

We consider two cases:

(1) If a is removed first, then, by Definition 2, $(S_1 \setminus \{a\}) \cup S \rightsquigarrow^+ b$. Since b has a support, it can be removed.

(2) If b is removed first, then a is supported by S and can be removed. □

In the example of Fig. 1 $\tau_{0,1} \geq 6$ and $\tau_{1,2} \leq 3$ do not have any supports other than their supports with respect to each other, so one of them must be retained. Scenario η of Fig. 1 shows an equivalent scenario where $\tau_{0,1} \geq 6$ is retained. Observe that after removal of $\tau_{1,2} \leq 3$ label L_1 is no longer needed.

Theorem 1. *Let $\mathcal{D}_s^\xi$ be a stable distance table. Let a, b and c be three constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \rightarrow b$, $b \rightarrow c$, $a \not\leftrightarrow b$, and $b \not\leftrightarrow c$. Then $c \not\rightarrow a$, and therefore $a \not\leftrightarrow c$.*

Proof. Assume that $c \rightarrow a$. Then there must exist some $S_c^{\triangleright a}$. It cannot contain b, otherwise we would have $b \rightarrow a$, and therefore $a \leftrightarrow b$. But if so, then by Observation 4 we would have $c \rightarrow b$, and therefore $b \leftrightarrow c$: a contradiction. □

Intuitively, Theorem 1 states that there is no cycle of length three consisting of members of $\mathcal{C}(\mathcal{D}_s^\xi)$ between which there is no cyclic dependency. Moreover, because of the quasi-transitivity of $\rightsquigarrow^+$ (Definition 2), it follows that there are then also no cycles of length greater than three.

3.2 Cyclic dependencies and equality constraints

Henceforth, we will refer to the pair of constraints, $\tau_{i,j} \leq a$ and $\tau_{i,j} \geq a$, for some a, as an *equality constraint* $\tau_{i,j} = a$.

It turns out that if there is a cyclic dependency among the constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$, then there must be at least one equality constraint in $\mathcal{C}(\mathcal{D}_s^\xi)$.

Observation 7 *Let ξ be a scenario of length n and $\mathcal{C}(\mathcal{D}_s^\xi)$ be the set of constraints in its stable distance table. If $\forall_{0\leq i<j<n}(\tau_{i,j} \geq u \in \mathcal{C}(\mathcal{D}_s^\xi) \wedge \tau_{i,j} \leq v \in \mathcal{C}(\mathcal{D}_s^\xi) \implies u \neq v)$, then there is no cyclic dependency between the members of $\mathcal{C}(\mathcal{D}_s^\xi)$.*

Proof. First, we show that there is no "cyclic dependency of length one", i.e., for any two constraints a and b in $\mathcal{C}(\mathcal{D}_s^\xi)$, there are no $S_a^{\triangleright b}$ and $S_b^{\triangleright a}$ such that $S_a^{\triangleright b} \rightsquigarrow b$ and $S_b^{\triangleright a} \rightsquigarrow a$ (note that $\rightsquigarrow$ is the direct support relation: see Definition 1). The general observation then follows from Theorem 1.

Assume that there are two constraints a and b such that $a \leftrightarrow b$ and that there is a $S_a^{\triangleright b}$ and a $S_b^{\triangleright a}$ such that $S_a^{\triangleright b} \rightsquigarrow b$ and $S_b^{\triangleright a} \rightsquigarrow a$.

We show the proof for the case when constraint a is of type minimum. Then there are three cases to consider:

1. Assume a is of the form $\tau_{i,k} \geq u$, and $S_b^{\triangleright a} = \{\tau_{i,j} \geq v, \tau_{j,k} \geq w\}$, where $i < j < k$ and $u = v + w$ (pt. 1 of Definition 1). One of the two constraints in $S_b^{\triangleright a}$ must be b.
   (a) Assume $b = \tau_{i,j} \geq v$. Since $a \in S_a^{\triangleright b}$, $S_a^{\triangleright b} = \{\tau_{i,k} \geq u, \tau_{j,k} \leq w'\}$ for some w' such that $v = u - w'$ (pt. 2 of Definition 1). But then $v + w = v + w'$, which implies $w = w'$. Therefore both $\tau_{j,k} \geq w$ and $\tau_{j,k} \leq w$ are constraints (in $S_b^{\triangleright a}$ and $S_a^{\triangleright b}$, respectively). But this contradicts the assumptions of Observation 7.
   (b) Assume $b = \tau_{j,k} \geq w$. Since $a \in S_a^{\triangleright b}$, $S_a^{\triangleright b} = \{\tau_{i,k} \geq u, \tau_{i,j} \leq w'\}$ for some w' such that $w = u - w'$ (pt. 3 of Definition 1). But then $v + w = w + w'$, which implies $v = w'$. Therefore both $\tau_{i,j} \geq v$ and $\tau_{i,j} \leq v$ are constraints (in $S_b^{\triangleright a}$ and $S_a^{\triangleright b}$, respectively). But this is a contradiction.
2. Assume a is of the form $\tau_{i,j} \geq u$ and $S_b^{\triangleright a} = \{\tau_{i,k} \geq v, \tau_{j,k} \leq w\}$, where $i < j < k$ and $u = v - w$ (pt. 2 of Definition 1).
   (a) If $b = \tau_{i,k} \geq v$, then $S_a^{\triangleright b} = \{\tau_{i,j} \geq u, \tau_{j,k} \geq w'\}$, where $v = u + w'$ (pt. 1 of Definition 1). So $u + w = u + w'$, hence $w = w'$ and both $\tau_{j,k} \leq w$ and $\tau_{j,k} \geq w$ are constraints: contradiction.
   (b) If $b = \tau_{j,k} \leq w$, then $S_a^{\triangleright b} = \{\tau_{i,j} \geq u, \tau_{i,k} \leq w'\}$, where $w = w' - u$ (pt. 3 of Definition 1). So $v - w = w' - w$, hence $v = w'$ and both $\tau_{i,k} \geq v$ and $\tau_{i,k} \leq v$ are constraints: contradiction.

3. Assume a is of the form $\tau_{j,k} \geq u$ and $S_b^{\triangleright a} = \{\tau_{i,k} \geq v, \tau_{i,j} \leq w\}$, where $i < j < k$ and $u = v - w$ (pt. 3 of Definition 1).
   (a) If $b = \tau_{i,k} \geq v$, $S_a^{\triangleright b} = \{\tau_{j,k} \geq u, \tau_{i,j} \geq w'\}$, where $v = u + w'$ (pt. 1 of Definition 1). So $v - w = v - w'$, hence $w = w'$, and both $\tau_{i,j} \leq w$ and $\tau_{i,j} \geq w$ are constraints: a contradiction.
   (b) If $b = \tau_{i,j} \leq w$, $S_a^{\triangleright b} = \{\tau_{j,k} \geq u, \tau_{i,k} \leq w'\}$, where $w = w' - u$ (pt. 2 of Definition 1). So $v - w = w' - w$, hence $v = w'$ and both $\tau_{i,k} \geq v$ and $\tau_{i,k} \leq v$ are constraints: a contradiction.

We omit the very similar proof for the case when a is of the form maximum. □

Theorem 2. *Let ξ be a scenario and let $\mathcal{D}_s^\xi$ be its stable distance table. If, for every pair of constraints x and y in $\mathcal{C}(\mathcal{D}_s^\xi)$, $x \not\leftrightarrow y$, then $\rightarrow$ defined on $\mathcal{C}(\mathcal{D}_s^\xi)$ is a strong partial order.*

Proof. We must show that $\rightarrow$ *i*s irreflexive, antisymmetric and transitive.

By Observation 1, $\rightarrow$ *i*s irreflexive. To see that $\rightarrow$ *i*s antisymmetric, we assume it is symmetric. Let a and b be two constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \rightarrow b$. Then $b \rightarrow a$, therefore $a \leftrightarrow b$: a contradiction.

For transitivity, let a and b be two constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$ such that $a \rightarrow b$ and $b \rightarrow c$. There are no cyclic dependencies: $a \rightarrow c$ follows from Observation 3. □

Intuitively, in the absence of cyclic dependencies $(\mathcal{C}(\mathcal{D}_s^\xi), \rightarrow)$ is a partially ordered set. Since $\mathcal{C}(\mathcal{D}_s^\xi)$ is finite, it has at least one minimal element, i.e., an element that does not have any support and hence cannot be removed. The set of such minimal elements is equivalent to the set of constraints represented by $\mathcal{C}(\mathcal{D}_s^\xi)$. Moreover, this is the smallest set equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$: any scenario equivalent to ξ must include this minimal set in its set of constraints.

It should be obvious that when a scenario with just this minimal set of constraints is converted to a timed automaton, the latter has the smallest number of clocks in the class of all equivalent timed automata.

Another important consequence of Theorem 2 is that if $\mathcal{C}(\mathcal{D}_s^\xi)$ does not include cyclic dependencies, then all of its members that do have supports can be removed in any order.

3.3 $\mathcal{C}(\mathcal{D}_s^\xi)$ with cyclic dependencies

$\mathcal{C}(\mathcal{D}_s^\xi)$ can include equality constraints which might give rise to cyclic dependencies (Observation 7). In that case $(\mathcal{C}(\mathcal{D}_s^\xi), \rightarrow)$ is not a partially ordered set. We could then have $a \leftrightarrow b$ such that we can remove a or b, but not both. As a result there could be more than one minimal set of constraints equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$.

To avoid this we will introduce a set of rules that would make the choice for removal between two constraints that are in a cyclic dependency deterministic. More importantly, the choice will result in a set of constraints that would require the smallest number of clocks in the class of all timed automata that are obtained from scenarios that are equivalent to ξ. These rules are presented below.

3.4 Resolving cyclic dependencies

We consider all the cases that give rise to cyclic dependencies. These cases (which involve equality constraints) are summarized as observations in the remainder of this subsection.

Observation 8 *Let $i < j$ be some event indices such that $(\tau_{i,j} = a) \in \mathcal{C}(\mathcal{D}_s^\xi)$ and, for any $i < p < j$, a_1 and a_2, $(\tau_{i,p} = a_1) \notin \mathcal{C}(\mathcal{D}_s^\xi)$ and $(\tau_{p,j} = a_2) \notin \mathcal{C}(\mathcal{D}_s^\xi)$. Then the clock allocated to anchor i can always be allocated to anchor j if needed.*

Proof. We consider the following three cases (see Fig. 4):

1. If every constraint that begins at i is of the form $\tau_{i,r} \geq c$ (or $\tau_{i,r} \leq c$), for some $r < j, c \leq a$, then the ranges of i and j will be non-overlapping, so one clock can be allocated to both.
2. If there is a constraint of the form $\tau_{i,k} \geq c$, for some $k > j, c \geq a$, then we must have $\{\tau_{i,j} \geq a, \tau_{j,k} \geq c - a\} \rightsquigarrow \tau_{i,k} \geq c$ (pt. 1 of Definition 1) and $\{\tau_{i,k} \geq c, \tau_{i,j} \leq a\} \rightsquigarrow \tau_{j,k} \geq c - a$ (pt. 3 of Definition 1). That is, $\tau_{i,k} \geq c \leftrightarrow \tau_{j,k} \geq c - a$. In this case, $\tau_{i,k} \geq c$ can be removed, while $\tau_{j,k} \geq c - a$ is retained. Then the ranges of i and j are non-overlapping.
   Similarly, if there is a constraint of the form $\tau_{i,k} \leq c$, for some $k > j, c \geq a$, then $\tau_{i,k} \leq c \leftrightarrow \tau_{j,k} \leq c - a$ and $\tau_{i,k} \leq c$ can be removed.
3. If there is a constraint of the form $\tau_{i,k} = c$, for some $k > j, c \geq a$, then we must also have $\tau_{j,k} = c - a$. According to Definition 1 the direct supports of these three equality constraints (six constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$) are

$$\{\tau_{i,j} \leq a, \tau_{j,k} \leq c - a\} \rightsquigarrow \tau_{i,k} \leq c, \qquad \{\tau_{i,j} \geq a, \tau_{j,k} \geq c - a\} \rightsquigarrow \tau_{i,k} \geq c,$$
$$\{\tau_{i,k} \leq c, \tau_{i,j} \geq a\} \rightsquigarrow \tau_{j,k} \leq c - a, \qquad \{\tau_{i,k} \geq c, \tau_{i,j} \leq a\} \rightsquigarrow \tau_{j,k} \geq c - a,$$
$$\{\tau_{i,k} \leq c, \tau_{j,k} \geq c - a\} \rightsquigarrow \tau_{i,j} \leq a, \text{ and } \{\tau_{i,k} \geq c, \tau_{j,k} \leq c - a\} \rightsquigarrow \tau_{i,j} \geq a.$$

   Observe that $\tau_{i,j} \leq a \leftrightarrow \tau_{j,k} \geq c - a \leftrightarrow \tau_{i,k} \geq c \leftrightarrow \tau_{i,j} \geq a \leftrightarrow \tau_{j,k} \leq c - a \leftrightarrow \tau_{i,k} \leq c \leftrightarrow \tau_{i,j} \leq a$.
   The dependencies can be resolved by removing some of the constraints. We can remove $\tau_{i,k} \geq c$, then $\tau_{i,j} \geq a$ and $\tau_{j,k} \geq b$ will both lose their direct supports. If we follow this by removing $\tau_{i,k} \leq c$, then $\tau_{i,j} \leq a$ and $\tau_{j,k} \leq b$ will lose their direct supports. That is, the equality between i and k can be removed as long as the equalities between i and j, and between j and k remain. Then the ranges of i and j become non-overlapping. □

It is worth noticing that resolving the cyclic dependencies as described in case 3 of Observation 8 (see the diagram on the left of Fig. 5) leads to the most satisfactory result: the other possibility, where the equality between j and k is retained (shown in the diagram on the right of Fig. 5), would require two clocks associated with anchors i and j.

Observe also that, for example, $\tau_{i,j} \leq a \not\leftrightarrow \tau_{i,k} \geq c$: Observation 4 cannot be applied, because $\tau_{j,k} \geq b$ has only one support, and that includes both $\tau_{i,j} \leq a$ and $\tau_{i,k} \geq c$.

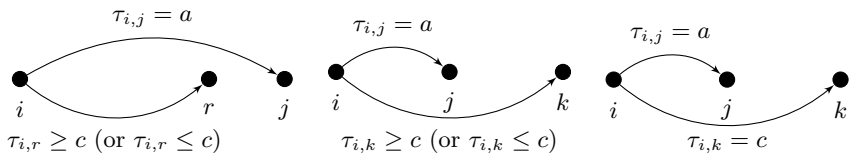

Fig. 4. Observation 8

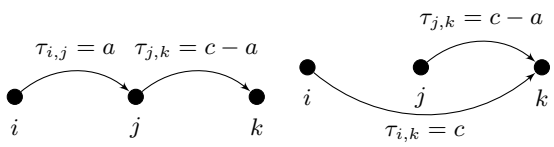

Fig. 5. Alternatives of case 3 of Observation 8

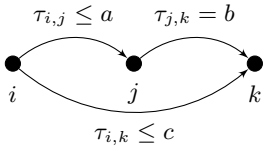

Fig. 6. Observation 10

Observation 9 *An equality constraint can only be supported by a pair of equality constraints.*

Proof. This is a direct consequence of Definition 1 (see the discussion above). □

In the first diagram of Fig. 5 (after $\tau_{i,k} = c$ has been removed) neither $\tau_{i,j} = a$, nor $\tau_{j,k} = c - a$ can be removed.

Observation 10 *Let $i < j < k$ be some event indices such that $(\tau_{j,k} = b) \in \mathcal{C}(\mathcal{D}_s^\xi)$ and, for any $j < p < k$, b_1 and b_2, $(\tau_{j,p} = b_1) \notin \mathcal{C}(\mathcal{D}_s^\xi)$ and $(\tau_{p,k} = b_2) \notin \mathcal{C}(\mathcal{D}_s^\xi)$. Moreover, either (a) $\tau_{i,j} \leq a$, and $\tau_{i,k} \leq c$, or (b) $\tau_{i,j} \geq a$, and $\tau_{i,k} \geq c$, such that $a + b = c$. If there is no unsupported constraint of the form $\tau_{i,l} \sim d$ such that $l > j$, then one clock can be associated with both anchors i and j.*

Proof. In case (a), illustrated in Fig. 6, from Definition 1 (pts. 6 and 5) we have $\{\tau_{i,j} \leq a, \tau_{j,k} \leq b\} \rightsquigarrow \tau_{i,k} \leq c$ and $\{\tau_{i,k} \leq c, \tau_{j,k} \geq b\} \rightsquigarrow \tau_{i,j} \leq a$. So $\tau_{i,k} \leq c \leftrightarrow \tau_{i,j} \leq a$. After removing $\tau_{i,k} \leq c$, if there is no unsupported constraint of the form $\tau_{i,l} \sim d$, such that $l > j$, then the ranges for i and j will be non-overlapping and one clock can be assigned to both i and j. Otherwise ranges for i and j will overlap and two clocks will be needed.

Case (b) is very similar. □

Observation 11 *Let ξ be a scenario of length n and $\mathcal{D}_s^\xi$ be its stable table. Let $i < j_1 < j_2 < \cdots < j_{m-1} < k$ (where $0 \leq i, k \leq n, m > 1$) be indices of events and $\tau_{i,k} = c$, $\tau_{i,j_1} = a_1, \tau_{j_1,j_2} = a_2, \ldots, \tau_{j_{m-1},k} = a_m$ be constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$, such that $a_1 + a_2 + \ldots a_m = c$. Moreover, for any $i < p < k$, such that $p \neq j_l$ $(1 \leq l \leq m - 1)$, and for any b_1, b_2, $\mathcal{C}(\mathcal{D}_s^\xi)$ contains no constraints of the form $(\tau_{i,p} = b_1)$ or $(\tau_{p,k} = b_2)$. Then, after all the supported constraints have been removed, an allocation to anchors $i, j_1, j_2, \ldots, j_{m-1}$ will require only one clock.*

Proof. For any three event indices between $i, j_1, j_2, \ldots, j_{m-1}, k$, there is in $\mathcal{C}(\mathcal{D}_s^\xi)$ a supported equality constraint between the earliest and the latest event. After

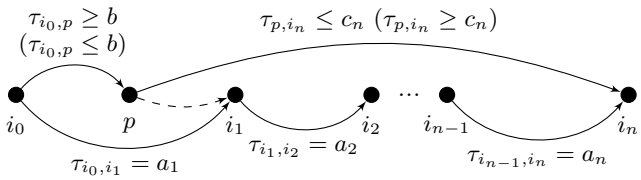

Fig. 7. An illustration of Observation 12

removing every such constraint, all constraints $\tau_{i,k} = c$, $\tau_{i,j_1} = a_1, \tau_{j_1,j_2} = a_2, \ldots, \tau_{j_{m-1},k} = a_m$ will have lost their supports, hence cannot be removed. So anchors $i, j_1, j_2, \ldots, j_{m-1}$ must be allocated clocks. But by Observation 8 the ranges of all these anchors are non-overlapping, therefore the same clock can be allocated to all of them. □

By Observation 11 the only unsupported constraints that begin at anchors $i, j_1, j_2, \ldots, j_{m-1}$ are the equality constraints between any two adjacent events. The next two observations consider the constraints that begin at other anchors.

Observation 12 *Let $i_0 < i_1 < i_2 < \cdots < i_{n-1} < i_n$ (where $n \geq 1$) be indices of events and $\tau_{i_0,i_1} = a_1, \tau_{i_1,i_2} = a_2, \ldots, \tau_{i_{n-1},i_n} = a_n$ be constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$. Let $i_0 < p < i_1$ be an arbitrary event index such that $\tau_{i_0,p} \geq b$ (or $\tau_{i_0,p} \leq b$). Then, after all the supported constraints have been removed, an allocation to anchors $i_0, p, i_1, \ldots, i_{n-1}$ will require at most two clocks.*

Proof. Let us assume $\tau_{i_0,p} \geq b$. (We omit the very similar case of $\tau_{i_0,p} \leq b$.)

Then *for every i_m such that* $1 \leq m \leq n$ we must have $\tau_{p,i_m} \leq c_m$, where c_m satisfies $a_1 + a_2 + \cdots + a_m = b + c_m$ (see the diagram in Fig. 7).
By Definition 1 pts. 4 and 2 (with $i = i_0$, $j = p$ and $k = i_1$):

$$\{\tau_{i_0,i_1} \leq a_1, \tau_{i_0,p} \geq b\} \rightsquigarrow \tau_{p,i_1} \leq c_1$$
$$\{\tau_{i_0,i_1} \geq a_1, \tau_{p,i_1} \leq c_1\} \rightsquigarrow \tau_{i_0,p} \geq b$$

By Definition 1 pts. 4 and 2 (with $i = i_0$, $j = p$ and $k = i_m$) and Definition 2:

$$\{\tau_{i_0,i_1} \leq a_1, \tau_{i_1,i_2} \leq a_2, \ldots, \tau_{i_{m-1},i_m} \leq a_m, \tau_{i_0,p} \geq b\} \rightsquigarrow^+ \tau_{p,i_m} \leq c_m$$
$$\{\tau_{i_0,i_1} \geq a_1, \tau_{i_1,i_2} \geq a_2, \ldots, \tau_{i_{m-1},i_m} \geq a_m, \tau_{p,i_m} \leq c_m\} \rightsquigarrow^+ \tau_{i_0,p} \geq b$$

By Definition 1 pts. 5 and 6 (with $i = p$, $j = i_1$ and $k = i_m$) and Definition 2:

$$\{\tau_{p,i_m} \leq c_m, \tau_{i_1,i_2} \geq a_2, \ldots, \tau_{i_{m-1},i_m} \geq a_m\} \rightsquigarrow^+ \tau_{p,i_1} \leq c_1$$
$$\{\tau_{p,i_1} \leq c_1, \tau_{i_1,i_2} \leq a_2, \ldots, \tau_{i_{m-1},i_m} \leq a_m\} \rightsquigarrow^+ \tau_{p,i_m} \leq c_m$$

That is, $\tau_{i_0,p} \geq b \leftrightarrow \tau_{p,i_1} \leq c_1$, $\tau_{i_0,p} \geq b \leftrightarrow \tau_{p,i_m} \leq c_m$, and $\tau_{p,i_1} \leq c_1 \leftrightarrow \tau_{p,i_m} \leq c_m$. By Observation 6, two of the constraints among the three can be removed (for any $1 \leq m \leq n$). Observe that $\tau_{p,i_1} \leq c_1$ and $\tau_{p,i_m} \leq c_m$ begin at anchor p. We consider three cases:

1. If there are in $\mathcal{C}(\mathcal{D}_s^\xi)$ no other constraints that begin at p, then after removing $\tau_{p,i_1} \leq c_1$ and $\tau_{p,i_m} \leq c_m$, p is no longer an anchor.

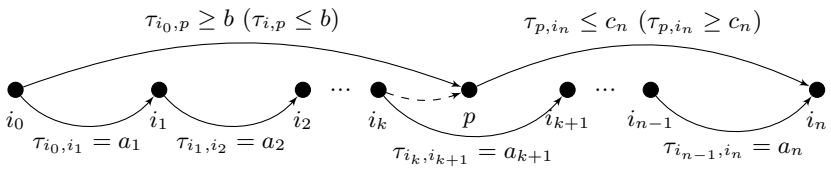

Fig. 8. An illustration of Observation 13

2. If there are in $\mathcal{C}(\mathcal{D}_s^\xi)$ some other constraints of the form $\tau_{p,j} \leq d$ or $\tau_{p,j} \geq d$ (where $j > p$, $j \neq i_m$ and $1 \leq m \leq n$) and they are all supported, then, after removing $\tau_{p,i_1} \leq c_1$ and $\tau_{p,i_m} \leq c_m$, they continue to be supported (Observation 5). So all all the constraints that begin at p can be removed and p is no longer an anchor.
3. If there is in $\mathcal{C}(\mathcal{D}_s^\xi)$ an unsupported constraint of the form $\tau_{p,j} \leq d$ or $\tau_{p,j} \geq d$ (where $j > p$, $j \neq i_m$, and $1 \leq m \leq n$), then after removing $\tau_{p,i_1} \leq c_1$ and $\tau_{p,i_m} \leq c_m$ p is still an anchor p.

If p is no longer an anchor, then by Observation 11 an allocation to anchors $i_0, i_1, \ldots, i_{n-1}$ requires exactly one clock. If p remains an anchor, we will need an extra clock for p. □

Observation 13 *Let $i_0 < i_1 < i_2 < \cdots < i_{n-1} < i_n$ (where $n \geq 1$) be indices of events and $\tau_{i_0,i_1} = a_1, \tau_{i_1,i_2} = a_2, \ldots, \tau_{i_{n-1},i_n} = a_n$ be constraints in $\mathcal{C}(\mathcal{D}_s^\xi)$. Let $i_k < p < i_{k+1}$ (where $1 \leq k < n$) be an arbitrary event index such that $\tau_{i_0,p} \geq b$ (or $\tau_{i_0,p} \leq b$). Then, after all the supported constraints have been removed, an allocation to anchors $i_0, i_1, \ldots, i_{n-1}, p$ will require at most two clocks.*

Proof. Let us assume $\tau_{i_0,p} \geq b$. Then for every i_m such that $1 < m \leq n$ we must have $\tau_{p,i_m} \leq c_m$, where c_m satisfies $a_1 + a_2 + \cdots + a_m = b + c_m$ (see the diagram in Fig. 8).

The argument is quite similar to that for Observation 12. It can be shown that $\tau_{i_0,p} \geq b \leftrightarrow \tau_{p,i_m} \leq c_m$, $\tau_{p,i_m} \leq c_m \leftrightarrow \tau_{i_k,p} \geq d_k$, and $\tau_{i_k,p} \geq d_k \leftrightarrow \tau_{i_0,p} \geq b$, for $k < m$, $d_k < a_m$, and $d_k < b$. By Observation 6, two of the constraints among the three can be removed (for any $1 \leq m \leq n$). It is not difficult to see that retaining $\tau_{i_k,p} \geq d_k$ is the best choice and that at most two clocks will be needed. □

In all the cases that we have considered so far in this section (observations 8–13), the cyclic dependencies are between constraints of the form $\tau_{r_1,r_2} \sim c$ and $\tau_{l_1,l_2} \sim d$ (for some c and d) such that either $r_2 = l_1$, or $r_2 = l_2$, or $r_1 = l_1$. That is, the two constraints "shared" an event index. But other forms of cyclic dependencies might exist.

As an example consider the scenario in Fig. 3 along with its stable table. As we mentioned before, $a = \tau_{0,1} \leq 4 \leftrightarrow b = \tau_{1,2} \geq 6$ and $b \leftrightarrow c = \tau_{2,3} \leq 2$:
$\{\tau_{0,2} \geq 10, \tau_{0,1} \leq 4\} \rightsquigarrow \tau_{1,2} \geq 6$, $\{\tau_{0,2} \leq 10, \tau_{1,2} \geq 6\} \rightsquigarrow \tau_{0,1} \leq 4$,
$\{\tau_{1,3} \geq 8, \tau_{2,3} \leq 2\} \rightsquigarrow \tau_{1,2} \geq 6$, and $\{\tau_{1,3} \leq 8, \tau_{1,2} \geq 6\} \rightsquigarrow \tau_{2,3} \leq 2$.

Observe that constraints a and b share index 1, while b and c share index 2.

But we also have a cyclic dependency between a and c that is not due to direct support:

$\{\tau_{0,2} \leq 10, \tau_{1,3} \geq 8, \tau_{2,3} \leq 2\} \rightsquigarrow^+ \tau_{0,1} \leq 4$,
$\{\tau_{1,3} \leq 8, \tau_{0,2} \geq 10, \tau_{0,1} \leq 4\} \rightsquigarrow^+ \tau_{2,3} \leq 2$.

So $\tau_{0,1} \leq 4 \leftrightarrow \tau_{2,3} \leq 2$. Only one of them can be removed, and the question is: which one? To answer this question we evaluate the two options by comparing the number of clocks that would be needed if one converted the resulting scenarios to their corresponding automata.

In this particular example removing either one of the constraints would result in the same outcome: two clocks are needed. But this might not always be the case. In general:

Observation 14 *Let $i < j < k < l$ be some event indices.*

1. *If there is a cyclic dependency between one of $\tau_{i,j} \geq a/\tau_{i,j} \leq a$ and one of $\tau_{k,l} \geq b/\tau_{k,l} \leq b$ (for some a and b), and if $\tau_{i,k} = c$ for some c, then we remove $\tau_{k,l} \geq b$ (or $\tau_{k,l} \leq b$): i is needed as an anchor for the equality.*
2. *If there is a cyclic dependency between one of $\tau_{i,k} \geq a/\tau_{i,k} \leq a$ and one of $\tau_{j,l} \geq b/\tau_{j,l} \leq b$ (for some a and b), and if $\tau_{i,j} = c$ for some c, then by Observation 6 we remove both, because $\tau_{i,k} \geq a/\tau_{i,k} \leq a$ will be supported (Observation 8) and the support does not include $\tau_{j,l} \geq b/\tau_{j,l} \leq b$.*
3. *If there is a cyclic dependency between one of $\tau_{i,l} \geq a/\tau_{i,l} \leq a$ and one of $\tau_{j,k} \geq b/\tau_{j,k} \leq b$ (for some a and b), and if $\tau_{i,j} = c$ for some c, then we remove $\tau_{i,l} \geq a$ (or $\tau_{i,l} \leq a$): this will shorten the range of anchor i.*

After resolving the cyclic dependencies in $\mathcal{C}(\mathcal{D}_s^\xi)$ by removing some of the constraints, we have obtained a smaller set $C_{acyclic} \subset \mathcal{C}(\mathcal{D}_s^\xi)$. The dependency relation $\rightarrow$, when restricted to $C_{acyclic}$, is a partial order (Observation 2). The set of the minimal elements of this partial order will be equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$.

We are now ready to present our new optimization algorithm, which is based upon our previous algorithm [13].

3.5 The optimization algorithm

Given a scenario $\xi = (\mathcal{E}, \mathcal{C})$, our goal is to find $\mathcal{C}' \subseteq \mathcal{C}(\mathcal{D}_s^\xi)$ that is equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$, such that, if $\xi' = (\mathcal{E}, \mathcal{C}')$, then the number of clocks in $\mathcal{A}_{\xi'}$ is the smallest in the entire class of language-equivalent timed automata.

We define the *direct support relation on* $\mathcal{C}(\mathcal{D}_s^\xi)$ by $DSupp = \{(c, S) \mid S \rightsquigarrow c\}$ and the *support relation on* $\mathcal{C}(\mathcal{D}_s^\xi)$ by $Supp = \{(c, S) \mid S \rightsquigarrow^+ c\}$.

If (c, S) is a member of $Supp$, then $\mathcal{C}(\mathcal{D}_s^\xi) \setminus \{c\}$ is equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$.

The optimization algorithm uses three data structures, C, WS, and CD. C represents the current set of constraints, WS ("working support") contains information about whether and how the constraints are supported by subsets of C, while CD ("cyclic dependencies") contains information about constraints in C that are in cyclic dependencies with other constraints in C.

We initialize C to $\mathcal{C}(\mathcal{D}_s^\xi)$, WS to $DSupp$, and CD to $\leftrightarrow$ restricted to WS.

The optimization process is carried out in two phases. During the first phase the algorithm takes pairs of constraints from CD, one at a time, and resolves the cyclic dependency between the elements of the pair (see Sec. 3.4). This involves removing the appropriate constraint from C, updating WS with the supports of this constraint (see below) and removing the entry from CD. As WS is updated, new cyclic dependencies may be uncovered and added to CD.

This phase consists of two steps. First, the algorithm examines dependencies that are described by Observation 11 and removes every equality constraint that is supported by a pair of equality constraints on smaller distances. In the second step it resolves dependencies described by Observations 8, 10 and 12–14.

At the end of this step CD becomes empty: there is no cyclic dependency between the members of $\mathcal{C}$, hence there is a partial order on $\mathcal{C}$ (Theorem 2).

Then the algorithm proceeds to the second phase, where it takes any constraint that has a support in WS, removes it from $\mathcal{C}$ and updates WS. The order in which the constraints are considered for removal does not matter. Observe that CD remains empty during this phase.

The algorithm terminates once WS becomes empty. At this point $\mathcal{C}$ includes the final minimal set of constraints equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$.

The four important invariants are:

1. C is equivalent to $\mathcal{C}(\mathcal{D}_s^\xi)$;
2. WS is a subset of $Supp$, the support relation associated with $\mathcal{C}(\mathcal{D}_s^\xi)$;
3. WS contains only those tuples in $Supp$ that do not contain constraints from outside C(but not necessarily all such tuples);
4. CD contains only tuples with constraints that appear as the first elements of some tuples in WS.

Clearly, the initialization establishes these invariants.

Thanks to the second and third invariant, a constraint c that has support in WS can be removed from C without violating the first invariant. The resulting new version of C will not contain c, therefore WS must be updated to restore the third invariant, in a way that does not violate the second invariant. CD must also be updated accordingly.

Every time that a supported constraint c is removed from $\mathcal{C}$:

1. For each $(c', S') \in WS$, such that $c \in S'$:
   - remove (c', S') from WS;
   - for each $(c, S) \in WS$, if $c' \notin S$, add $(c', S' \setminus \{c\} \cup S)$ to WS.
2. Remove from WS every tuple whose first element is c.
3. Remove from CD every tuple whose first or second element is c.

Notice that the first step above generates new tuples in WS according to pt. 2 of Definition 2.

It should be clear that this method of updating ensures that WS remains within $Supp$ (restricted to the current C) and that we do not lose information about indirect supports in C. Moreover, every constraint that appears in some tuple in CD must also appear as the first element of some tuple in WS.

Termination is assured, because at each step we remove a constraint from a finite set of constraints. Correctness is ensured by the invariants.

Scenario η of Fig. 1, obtained by our algorithm, is equivalent to ξ.

In summary, every time there is a choice between constraints that are involved in a cyclic dependency, we retain the one that will reduce the number of anchors, or—if that is impossible—reduce the number of overlapping ranges of anchors (Observations 8–14). This process does not remove supports from those constraints that are not involved in cyclic dependencies (Observation 5).

Once the cyclic dependencies are resolved, we retain only those of the remaining constraints that have no support, and that must therefore be included in all the equivalent sets of constraints. It follows that the number of clocks required for the resulting automaton cannot be decreased by choosing another equivalent set of constraints.

This can be summarized as follows.

Theorem 3. *Let $\xi = (\mathcal{E}, \mathcal{C})$ be a scenario and $\mathcal{D}_s^\xi$ be its stable distance table. Let $\mathcal{C}_{opt}$ be the set of constraints obtained from $\mathcal{C}(\mathcal{D}_s^\xi)$ by our optimization algorithm. Then $\mathcal{A}_{\xi'}$, where $\xi' = (\mathcal{E}, \mathcal{C}_{opt})$, has the smallest number of clocks in the entire class of timed automata that are language equivalent to $\mathcal{A}_\xi$.*

4 Conclusions

We present a new optimization algorithm that achieves the minimal number of clocks when timed scenarios are viewed as timed automata.

That is, given a scenario $\xi = (\mathcal{E}, \mathcal{C})$, the algorithm finds a set of constraints, $\mathcal{C}'$, such that $\xi' = (\mathcal{E}, \mathcal{C}')$ is equivalent to ξ, and the automaton derived from ξ' has the smallest number of clocks in the entire class of equivalent timed automata.

References

1. Abdulla, P.A., Deneux, J., Ouaknine, J., Worrell, J.: Decidability and complexity results for timed automata via channel machines. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) Automata, Languages and Programming. pp. 1089–1101. Springer, Berlin, Heidelberg (2005)
2. Akshay, S., Mukund, M., Kumar, K.N.: Checking Coverage for Infinite Collections of Timed Scenarios. In: CONCUR 2007 - Concurrency Theory, 18th International Conference, CONCUR 2007, Proceedings. pp. 181–196
3. Alur, R., Dill, D.L.: A theory of timed automata. Theor. Comput. Sci. 126(2), 183–235 (Apr 1994)
4. Alur, R., Madhusudan, P.: Decision Problems for Timed Automata: A Survey. In: Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT, Revised Lectures. pp. 1–24 (2004)
5. Alur, R., Martin, M., Raghothaman, M., Stergiou, C., Tripakis, S., Udupa, A.: Synthesizing Finite-State Protocols from Scenarios and Requirements. In: Yahav, E. (ed.) Hardware and Software: Verification and Testing. pp. 75–91. Springer International Publishing, Cham (2014)

6. Baier, C., Bertrand, N., Bouyer, P., Brihaye, T.: When are timed automata determinizable? In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) Automata, Languages and Programming. pp. 43–54. Springer, Berlin, Heidelberg (2009)
7. Bollig, B., Katoen, J., Kern, C., Leucker, M.: Replaying Play In and Play Out: Synthesis of Design Models from Scenarios by Learning. In: Proceedings of the 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS. pp. 435–450 (2007)
8. Chandrasekaran, P., Mukund, M.: Matching scenarios with timing constraints. In: Asarin, E., Bouyer, P. (eds.) Formal Modeling and Analysis of Timed Systems. pp. 98–112. Springer, Berlin, Heidelberg (2006)
9. Harel, D., Kugler, H., Pnueli, A.: Synthesis Revisited: Generating Statechart Models from Scenario-Based Requirements, pp. 309–324. Springer Berlin Heidelberg (2005)
10. Heitmeyer, C.L., Pickett, M., Leonard, E.I., Archer, M.M., Ray, I., Aha, D.W., Trafton, J.G.: Building high assurance human-centric decision systems. Autom. Softw. Eng. 22(2), 159–197 (2015)
11. Saeedloei, N., Kluźniak, F.: From Scenarios to Timed Automata. In: Formal Methods: Foundations and Applications - 20th Brazilian Symposium, SBMF 2017, Proceedings. pp. 33–51
12. Saeedloei, N., Kluźniak, F.: Timed Scenarios: Consistency, Equivalence and Optimization. In: Formal Methods: Foundations and Applications - 21st Brazilian Symposium, SBMF 2018, Proceedings. pp. 215–233
13. Saeedloei, N., Kluźniak, F.: Optimization of timed scenarios. In: Carvalho, G., Stolz, V. (eds.) Formal Methods: Foundations and Applications - 23rd Brazilian Symposium, SBMF 2020, Ouro Preto, Brazil, November 25-27, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12475, pp. 119–136. Springer (2020)
14. Saeedloei, N., Kluzniak, F.: Synthesizing clock-efficient timed automata. In: Dongol, B., Troubitsyna, E. (eds.) Integrated Formal Methods - 16th International Conference, IFM 2020, Lugano, Switzerland, November 16-20, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12546, pp. 276–294. Springer (2020)
15. Somé, S., Dssouli, R., Vaucher, J.: From Scenarios to Timed Automata: Building Specifications from Users Requirements. In: Proceedings of the Second Asia Pacific Software Engineering Conference. pp. 48–57. APSEC '95, IEEE Computer Society
16. Uchitel, S., Kramer, J., Magee, J.: Synthesis of Behavioral Models from Scenarios. IEEE Trans. Softw. Eng. 29(2), 99–115 (Feb 2003)