Inference of Delayed Biological Regulatory Networks from Time Series Data

Emna Ben Abdallah[1(✉)], Tony Ribeiro[1], Morgan Magnin[1,2], Olivier Roux[1], and Katsumi Inoue[3]

[1] Institut de Recherche en Communications et Cybernétique de Nantes, LUNAM Université, École Centrale de Nantes, IRCCyN UMR CNRS 6597, 1 rue de la Noë, 44321 Nantes, France
{emna.ben-abdallah,olivier.roux}@irccyn.ec-nantes.fr
[2] National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
[3] Department of Computer Science, Tokyo Institute of Technology, 2-12-1 Oookayama, Meguro-ku, Tokyo 152-8552, Japan

Abstract. The modeling of Biological Regulatory Networks (BRNs) relies on background knowledge, deriving either from literature and/or the analysis of biological observations. But with the development of high-throughput data, there is a growing need for methods that automatically generate admissible models. Our research aim is to provide a logical approach to infer BRNs based on given time series data and known influences among genes. In this paper, we propose a new methodology for models expressed through a timed extension of the Automata Networks [22] (well suited for biological systems). The main purpose is to have a resulting network as consistent as possible with the observed datasets. The originality of our work consists in the integration of quantitative time delays directly in our learning approach. We show the benefits of such automatic approach on dynamical biological models, the DREAM4 datasets, a popular reverse-engineering challenge, in order to discuss the precision and the computational performances of our algorithm.

Keywords: Inference model · Dynamic modeling · Delayed biological regulatory networks · Automata network · Time series data

1 Introduction

With both the spread of numerical tools in every part of daily life and the development of NGS methods (New Generation Sequencing methods), like DNA microarrays in biology, a large amount of time series data is now produced [6,8, 18]. This means that the produced data from the experiments led on a biological system grows drastically. The newly produced data - as long as the associated noise does not raise an issue with regard to the precision and relevance of the corresponding information - can give us some new insights on the behavior of a system. This justifies the urge to design efficient methods for inference.

E. Bartocci et al. (Eds.): CMSB 2016, LNBI 9859, pp. 30–48, 2016.
DOI: 10.1007/978-3-319-45177-0_3

Reverse engineering of gene regulatory networks from expression data have been handled by various approaches [14,16,29,34]. Most of them are only static. However, other researchers are rather focusing on incorporating temporal aspects in inference algorithms. The relevance of these various algorithms have been recently assessed in [15]. The authors of [17] tackled the inference problem of time-delayed gene regulatory networks through Bayesian networks. As this is a complex problem, in [33], the authors propose a Time-Window-Extension Technique based on time series segmentation in different successive phases. These approaches take gene expression data into account as input and generate the associated regulations. But the discrete approaches that simplify this problem by abstractions, need to determine the relevant thresholds of each gene to define its active and inactive state. Various approches have been designed to tackle the discretization problem. We can cite for example [33], in which the authors have proposed an alternative methodology that considers not a concentration level, but the way the concentration changes (in other words: the derivative of the function giving the concentration w.r.t time) in the presence/absence of one regulator. On the other hand, the major problem for modeling lies on the quality of the expression data. Indeed, noisy data may be the main origin of the errors in the inference process. Thus, the pre-processing of the biological data is crucial pertinence of the inferred relations between components. In this work, the input data is considered to be pre-processed and the result is reliable discretized time series data.

In this paper, we aim to provide a logical approach to tackle the learning of qualitative models of biological dynamic systems, like gene regulatory networks. In our context, we assume the set of interacting components as fixed and we consider the learning of the interactions between those components. As in [3], in which the authors targeted the completion of stationary Boolean networks, we suppose that the topology of the network is given, providing us the influences among each gene as background knowledge. From time series data of the evolution of the system, given its topology, we learn the dynamics of the system. The main originality of our work is that we address this problem in a timed setting, with quantitative delays potentially occurring between the moment an interaction activated and the moment its effect is visible.

During the past decade, there has been a growing interest for the hybrid modeling of gene regulatory networks with delays. These hybrid approaches consider various modeling frameworks. In [19], the authors hybridize Petri Nets: the advantage of hybrid with regard to discrete modeling lies in the possibility of capturing biological factors, e.g., the delay for the transcription of RNA polymerase. The merits of other hybrid formalisms in biology have been studied, for instance timed automata [28], hybrid automata [2] and boolean representation [21]. Finally, in [7], the authors investigate a direct extension of the discrete René Thomas' modeling approach by introducing quantitative delays. These delays represent the compulsory time for a gene to turn from a discrete qualitative level to the next (or previous) one. They exhibit the advantage of such a framework for the analysis of mucus production in the bacterium Pseudomonas aeruginosa. The approach we propose in this paper inherits from this idea that some models need to capture these timing features.

2 Background

The definition and semantics of automata networks is presented in Sect. 2.1. The enrichment of the automata networks with delays and the corresponding new semantics is presented in Sect. 3.

2.1 Automata Network

Definition 1 introduces the Automata Network (AN) [22–24] as a model of finite-state machines having transitions between their *local state* conditioned by the state of other automata in the network. A *local state* of an automaton is noted by a_i, where a is the automaton identifier, and i is the expression level within a. At any time, each automaton has exactly one *active local state*, and the set of *active local states* is called the *global state* of the network.

The concurrent interactions between automata are defined by a set of *local transitions*. Each *local transition* has this form $\tau = a_i \xrightarrow{\ell} a_j$, with a_i, a_j being local states of an automaton a called respectively *origin* and *destination* of t and ℓ is a (possibly empty) set of local states of automata other than a (with at most one local state per automaton).

Notation: Given a finite state A, $\wp(A)$ is the power set of A. Given a network N, $\mathsf{state}(N, t)$ is the state of N at a time step $t \in \mathbb{N}$.

Definition 1 (Automata Network). *An* Automata Network *is a triple* $(\Sigma, \mathcal{S}, \mathcal{T})$ *where:*

- $\Sigma = \{a, b, \dots\}$ *is the finite set of automata identifiers;*
- *For each* $a \in \Sigma$, $\mathcal{S}(a) = \{a_i, \dots, a_j\}$, *is the finite set of local states of automaton* a; $\mathcal{S} = \prod_{a \in \Sigma} \mathcal{S}(a)$ *is the finite set of global states;* $\boldsymbol{LS} = \cup_{a \in \Sigma} \mathcal{S}(a)$ *denotes the set of all the local states.*
- $\mathcal{T} = \{a \mapsto \mathcal{T}_a \mid a \in \Sigma\}$, *where* $\forall a \in \Sigma, \mathcal{T}_a \subset \mathcal{S}(a) \times \wp(\boldsymbol{LS} \setminus \mathcal{S}(a)) \times \mathcal{S}(a)$ *with* $(a_i, \ell, a_j) \in \mathcal{T}_a \Rightarrow a_i \neq a_j$, *is the mapping from automata to their finite set of local transitions.*

Example 1. The Fig. 1 represents an Automata Network, $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ with 4 automata ($\Sigma = \{a, b, c, d\}$) such that $\mathcal{S}(a) = \{a_0, a_1\}$, $\mathcal{S}(b) = \{b_0, b_1\}$, $\mathcal{S}(c) = \{c_0, c_1, c_2\}$, $\mathcal{S}(d) = \{d_0, d_1, d_2\}$ and 5 local transitions,
$\mathcal{T} = \{\ b_0 \xrightarrow{\{a_1\}} b_1,\ a_1 \xrightarrow{\{b_1, d_2\}} a_0,\ c_2 \xrightarrow{\{a_1\}} c_1,\ d_2 \xrightarrow{\{a_0\}} d_1,\ b_1 \xrightarrow{\{a_1, c_2\}} b_0, \}$.

A *global state* of a given AN consists in a set of all active *local states* of each automaton in the network. The active *local state* of a given automaton $a \in \Sigma$ in a state $\zeta \in \mathcal{S}$ is noted $\zeta[a]$. For any given *local state* a_i we also note, $a_i \in \zeta$ if and only if $\zeta[a] = a_i$. For each automaton, it cannot have more than one active *local state* at one *global state*.

Definition 2 (Playable Local Transition). *Let* $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ *be an Automata Network and* $\zeta \in \mathcal{S}$, *with* $\zeta = \mathsf{state}(\mathcal{AN}, t)$. *We note* P_t *the set of playable local transitions in* $\mathcal{AN}$ *at time step* t *by:*
$P_t = \{\ a_i \xrightarrow{\ell} a_j \in \mathcal{T} \mid \ell \subseteq \zeta \wedge a_i \in \zeta$ *with* $state(\mathcal{AN}, t) = \zeta\}$.

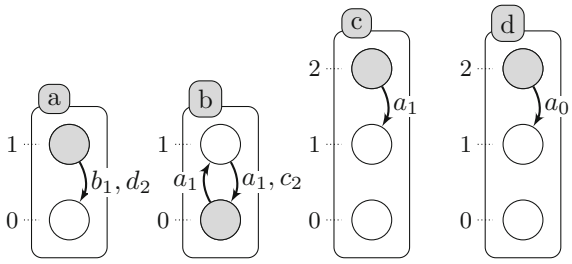

Fig. 1. Example of Automata Network with 4 automata: a, b, c and d presented by labeled boxes and their local states are presented by circles (for instance a is either at level 0 or 1). A local transition is a labeled directed edge between two local states within the same automaton: its label stands for the set of necessary conditions local states of the automata to play the transition. The grayed circles stand for the global state: $\langle a_1, b_0, c_2, d_2 \rangle$.

The dynamics of the AN is performed thanks to the *global transitions*. Indeed, the transition from one state ζ_1 to its successor ζ_2 is satisfied by a set of the playable local transitions (Definition 2) at ζ_1.

Definition 3 (Global Transitions). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be an Automata Network and $\zeta_1, \zeta_2 \in \mathcal{S}$, with $\zeta_1 = \mathsf{state}(\mathcal{AN}, t)$ and $\zeta_2 = \mathsf{state}(\mathcal{AN}, t+1)$. Let P_t be the set of playable local transitions at t. We note G_t the power set of global transitions at t:*

$$G_t := \wp(P_t)$$

In the semantics that we are based on, parallel application of local transitions in different automata is permitted but it is not enforced. Thus the set of *global transitions* is a power set of all the playable *local transitions* (also empty set).

3 Timed Automata Networks

In some dynamics it is crucial to have information about the *delays* between two events (two states of a AN). The discrete transition, described above, cannot exhibit this information: we just process chronological information, that the state ζ_2 will be after ζ_1 in the next step but it is not possible to know chronometry, i.e., how much time this transition takes to occur and whether it blocks some transitions during this time. In fact some local transitions could not be played any more because of concurrency about shared resources (necessary components to play the transition) between them. We thus need to restrain the general dynamics to capture more realistic behavior w.r.t biology. So we propose in this section to add the *delays* in the *local transitions* attributes and give the associated semantics that we based on to infer biological networks.

Definition 4 (Timed Automata Network (T-AN)). Timed Automata Network *is a triple $(\Sigma, \mathcal{S}, \mathcal{T})$ where:*

- $\Sigma = \{a, b, \dots\}$ *is the finite set of automata identifiers;*
- *For each* $a \in \Sigma$, $\mathcal{S}(a) = \{a_i, \dots, a_j\}$, *is the finite set of local states of automaton* a; $\mathcal{S} = \prod_{a \in \Sigma} \mathcal{S}(a)$ *is the finite set of global states;* $\boldsymbol{LS} = \cup_{a \in \Sigma} \mathcal{S}(a)$ *denotes the set of all the local states.*
- $\mathcal{T} = \{a \mapsto \mathcal{T}_a \mid a \in \Sigma\}$, *where* $\forall a \in \Sigma, \mathcal{T}_a \subset \mathcal{S}(a) \times \wp(\boldsymbol{LS} \setminus \mathcal{S}(a)) \times \mathcal{S}(a) \times \mathbb{N}$ *with* $(a_i, \ell, a_j, \delta) \in \mathcal{T}_a \Rightarrow a_i \neq a_j$, *is the mapping from automata to their finite set of timed local transitions.*

To model biological networks where quantitative time plays a major role, we will use T-AN (Timed Automata Network). This formalism enriches $\mathcal{AN}$ with timed local transitions: $a_i \xrightarrow[\delta]{\ell} a_j$. In the latter, δ is called a *delay* and represents the time needed for the transition to be performed. When modeling a regulation phenomenon, this allows to capture the delay between the activation order of the production of the protein and its effective synthesis. and the synthesis of the product.

We note $a_i \xrightarrow[\delta]{\ell} a_j \in \mathcal{T} \Leftrightarrow (a_i, \ell, a_j, \delta) \in \mathcal{T}(a)$ and $a_i \xrightarrow{\ell} a_j \in \mathcal{T} \Leftrightarrow \exists \delta \in \mathbb{N}, a_i \xrightarrow[\delta]{\ell} a_j \in \mathcal{T}$. Given $\tau = a_i \rightarrow a_j \in \mathcal{T}$, $orig(\tau) = a_i$, $dest(\tau) = a_j$. Definition 2 also applies to timed local transitions.

Considering delays in the evolution of timed automata networks creates concurrency between the timed local transitions. This concurrency is mainly justified by the shared resources between local transitions. Indeed, transitions that have the same origins and/or destinations could not be fired synchronously. Besides, during the delay of the execution of a transition τ_1, it is possible that another transition τ_2 could be activated. Then we need to take care of the following possible conflicts between resources: transition τ_2 may change the local states of the automata participating in τ_1. We make the following assumptions, that is similar to the one adapted in [12]: we consider τ_2 needs to be blocked until the current transition τ_1 finishes. Nevertheless, we allow the resources of τ_1 to participate to other transitions. In addition we do not forbid the process involved in $orig(\tau_1)$ to participate to other transition τ_2 if and only if that the remaining $delay(\tau_1)$ is greater than $delay(\tau_2)$ (see Definition 5). Those considerations lead to the followings definitions.

Definition 5 (Blocked Timed Local Transition). *Let* $AN = (\Sigma, \mathcal{S}, \mathcal{T})$ *be a T-AN and* $t \in \mathbb{N}$. *Let* P *be a set of pairs* $\mathcal{T} \times \mathbb{N}$. *The* set of blocked timed local transitions *of* $\mathcal{AN}$ *by* P *at* t *is defined as follows:*

$B(\mathcal{AN}, P, t) := \{a_i \xrightarrow[\delta]{\ell} a_j \in \mathcal{T} \mid \exists (b_k \xrightarrow[\delta']{\ell'} b_l, t') \in P$ *such that* $(a = b) \vee (a_i \in \ell' \wedge \delta' > t' - (t + \delta)) \vee (b_k \in \ell \wedge \delta' < t' - (t + \delta))\}$

In Definition 5, if P is the set of currently ongoing timed local transition, it allows us to prevent the execution of transitions that would alternate the resources currently being used or that would rely on resources that will be modified before the end of those transitions. Let t_1 be a transition such that $\tau_1 = a_i \xrightarrow[\delta]{\ell} a_j$ is fired at time step t. So $t + \delta$ is the ending time of τ_1 and $(t' - (t + \delta))$ is the interval

of time between the ending of the transition $\tau_2 = b_k \xrightarrow[\delta']{\ell} b_l$ and the beginning of transition τ_1 with $t' > t$. According to the Definition 5, τ_2 is blocked if a_i (resp. b_k) is a necessary resource for τ_2 (resp. τ_1) and the τ_1 (resp. τ_2) finishes before τ_2 (resp. τ_1): $\delta' > t' - (t + \delta)$ (resp. $\delta' < t' - (t + \delta)$) i.e. a_i (resp. b_k) is not available to participate in the transition τ_2 (resp. τ_1) during δ' (δ).

Definition 6 (Fireable Timed Local Transition). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a T-AN, $\zeta \in \mathcal{S}$ the state of $\mathcal{AN}$ at $t \in \mathbb{N}$. Let P be a set of pairs $\mathcal{T} \times \mathbb{N}$ and $B(\mathcal{AN}, P, t)$ be the set of blocked timed local transitions of $\mathcal{AN}$ by P at t. The* set of fireable local transitions *of $\mathcal{AN}$ in ζ w.r.t. P at t is defined as follows:*

$$F(\mathcal{AN}, \zeta, P, t) := \{a_i \xrightarrow{\ell} a_j \in \mathcal{T} \setminus B(\mathcal{AN}, P, t) \mid \ell \subseteq \zeta, a_i \in \zeta\}$$

Definition 6 extends the notion of playable transition by considering concurencies with the currently ongoing transition of P.

Definition 7 (Set of Fireable Sets of Timed Local Transition). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a T-AN, $\zeta \in \mathcal{S}$ the state of $\mathcal{AN}$ at $t \in \mathbb{N}$. Let P be a set of pairs $\mathcal{T} \times \mathbb{N}$ and $F(\mathcal{AN}, \zeta, P, t)$ the set of fireable local transitions of $\mathcal{AN}$ in ζ w.r.t. P at t. The* set of firable sets of timed local transition *of $\mathcal{AN}$ in ζ w.r.t. P at t is defined as:*

$$SFS(\mathcal{AN}, \zeta, P, t) := \{FS \subseteq F(\mathcal{AN}, \zeta, P, t) \mid$$

$$(\forall \tau = (b_k \xrightarrow[\delta]{\ell} b_l) \in FS, \nexists (b_k \xrightarrow[\delta']{\ell'} b_{l'}) \in FS, b_l \neq b_{l'}, \tau \notin B(\mathcal{AN}, FS \setminus \{\tau\}, t)\}$$

Definition 7 prevents the execution of two transitions that would affect the same automaton.

Definition 8 (Active Timed Local Transitions). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a T-AN, $\zeta \in \mathcal{S}$ the state of $\mathcal{AN}$ at $t \in \mathbb{N}$. Let $SFS(\mathcal{AN}, \zeta, P, t)$ be the set of firable sets of timed local transition. The set of* active timed local transitions *of $\mathcal{AN}$ at t is:*

$$A(\mathcal{AN}, t) := \begin{cases} \{(\tau \in FS, t) \mid FS \in SFS(\mathcal{AN}, \zeta, \emptyset, t)\} & if\ t = 0 \\ \{(\tau \in FS, t) \mid FS \in SFS(\mathcal{AN}, \zeta, A(\mathcal{AN}, t-1), t)\} & \\ \cup \{(b_k \xrightarrow[\delta']{\ell'} b_l, t') \in A(\mathcal{AN}, t-1) \mid t - t' < \delta\} & if\ t > 0 \end{cases}$$

Definition 8 provides us the evolution of the possible set of ongoing actions. Supposing that in the initial state of a trajectory (at $t = 0$) no transition is blocked and all playable timed transitions are fireable. Then, when $t > 0$, at each time step it should be verified that a playable timed transition is also fireable, in other words that it is not blocked by the active timed local transitions fired in previous steps. Furthermore the timed local transitions fired at the same time should not block each other.

Delays of local transitions can now be represented in an Automata Network thanks to *timed local transitions*. Note that if all delays of local transitions are set to 0 it is equivalent to an AN without delays (original AN). The way these new local transitions should be used is described as follows.

At any time, each automaton has one and only one *local state*, forming the *global state* of the network. Choosing arbitrary ordering between automata identifiers, the set of *global states* of the network is referred to as $\mathcal{S}$ with $\mathcal{S} = \prod_{a \in \Sigma} \mathcal{S}(a)$. Given a global state $\zeta \in \mathcal{S}$, $\zeta(a)$ is the local state of automaton a in ζ, i.e., the a-th coordinate of ζ. We write also $a_i \in \zeta \Leftrightarrow \zeta(a) = a_i$; and for any $ls \in \mathbf{LS}$, $ls \subset \zeta \Leftrightarrow \forall a_i \in \zeta, \zeta(a) = a_i$. In this paper, we allow, but do not force, applying in parallel transitions in different automata such in Definition 3 but adding delays in the local transitions and considering concurrency between transitions require further study of the semantics of the model (Definition 9).

Definition 9 (Semantics of Timed Automata Network). *Let* $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ *be a T-AN and* $t \in \mathbb{N}$. *The set of timed local transition fired at* t *is:* $FS := \{(a_i \xrightarrow[\delta]{\ell} a_j) \mid ((a_i \xrightarrow[\delta]{\ell} a_j), t) \in A(\mathcal{AN}, t)\}$ *then*

$$(a_i \xrightarrow[\delta]{\ell} a_j) \in FS \implies \zeta(a) = a_j \, with \, \zeta = \mathsf{state}(\mathcal{AN}, t + \delta).$$

The state of $\mathcal{AN}$ *at* $t + 1$ *is denoted* $\zeta_{t+1} = \mathsf{state}(\mathcal{AN}, t + 1)$ *and defined according to the set of timed local transitions that finished at* $t + 1$*:*

$$F_{t+1} := \{(b_k \xrightarrow[\delta']{\ell'} b_l) \mid ((b_k \xrightarrow[\delta']{\ell'} b_l), t') \in A(\mathcal{AN}, t), t + 1 - t' = \delta\}$$

then $\forall c \in \Sigma$, *such that* $\nexists (c_k \xrightarrow[\delta'']{\ell''} c_l) \in F_{t+1} \implies \zeta_{t+1}(c) = \zeta_t(c)$ *with* $\zeta_t = \mathsf{state}(\mathcal{AN}, t)$ *and* $\zeta_{t+1} = \mathsf{state}(\mathcal{AN}, t + 1)$.

We note that at any time step t such that $\zeta = \mathsf{state}(\mathcal{AN}, t)$ and P the set of ongoing transitions, we have: $FS \in F(\mathcal{AN}, \zeta, P, t) \in \mathcal{T} \setminus B(\mathcal{AN}, P, t)$.

Where synchronous biological regulatory networks have been studied, little has been done on the asynchronous counterpart [31], although there is evidence that most living systems are governed by synchronous and asynchronous updating. According to Harvey and Bossomaier [13], asynchronous systems are biologically more plausible for many phenomena than their synchronous counterpart and observed global synchronous behavior in nature usually simply arises from the local asynchronous behavior. In this paper, we defend these assumptions and we consider an asynchronous behavior for each automata in one hand and a synchronous behavior in the global network.

The assumptions in the synchronous model that all components could change at the same time and take an equivalent amount of time in changing their expression levels, is biologically unrealistic. But there is seldom enough informations to be able to discern the precise order and duration of state transitions. The timed extension of Automata Network we propose in this paper allows both asynchronous and synchronous behavior by proposing a non-deterministic application of

the timed local transitions. Figure 2 shows a trajectory of a Timed Automata Network when we choose to apply timed local transition in a synchronous maner.

We presented above the semantics of the T-AN that we are based on to modeling BRNs from experimental data. Even if it already exists a few hybrid formalisms like time Petri Nets, hybrid automata, etc., we propose this extension of the AN framework for several reasons. First, AN is a general framework that, although it was mainly used for biological networks [9,23], allows to represent any kind of dynamical models, and converters to several other representations are available. Indeed, a T-AN is a subclass of time Petri nets [10]. Finally, the particular form of the timed local transition in a AN model allows to easily represent them in Answer Set Programming (ASP), with one fact per timed local transition, as described in this work [1]. Later we propose a new approach to resolve the generation problem of T-AN models from time series data.

Taking the following timed automata network as an example, we generate a possible trajectory of the network starting from a known initial state.

Example 2. Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T}))$ be a timed automata extended with delays from Example 1. Such that $\mathcal{T} = \{\tau_1 = b_0 \xrightarrow[2]{\{a_1\}} b_1,\ \tau_2 = a_1 \xrightarrow[3]{\{b_1,d_2\}} a_0,\ \tau_3 = c_2 \xrightarrow[5]{\{a_1\}} c_1,\ \tau_4 = d_2 \xrightarrow[2]{\{a_0\}} d_1,\ \tau_5 = b_1 \xrightarrow[2]{\{a_1,c_2\}} b_0,\}$.

T	0	1	2	3	4	5	6	7
$B(\mathcal{AN}, P, t)$	$\emptyset$	$\{\tau_1, \boldsymbol{\tau_2}, \tau_3\}$	$\{\boldsymbol{\tau_2}, \tau_3\}$	$\{\tau_2, \tau_3\}$	$\{\tau_2, \tau_3, \tau_5\}$	$\emptyset$	$\{\tau_4\}$	$\emptyset$
$F(\mathcal{AN}, \zeta, P, t)$	$\{\tau_1, \tau_3\}$	$\emptyset$	$\{\tau_2, \tau_5\}$	$\emptyset$	$\emptyset$	$\{\tau_4\}$	$\emptyset$	$\emptyset$
$SFS(\mathcal{AN}, \zeta, P, t)$	$\{\emptyset, \{\tau_1\}, \{\tau_3\}, \{\tau_1, \tau_3\}\}$	$\{\emptyset\}$	$\{\emptyset, \{\tau_2\}, \{\tau_5\}\}$	$\{\emptyset\}$	$\{\emptyset\}$	$\{\emptyset, \{\tau_4\}\}$	$\{\emptyset\}$	$\{\emptyset\}$
FS	$\{\tau_1, \tau_3\}$	$\emptyset$	$\{\tau_2\}$	$\emptyset$	$\emptyset$	$\{\tau_4\}$	$\emptyset$	$\emptyset$
$A(\mathcal{AN}, t)$	$\{(\boldsymbol{\tau_1}, \mathbf{0}), (\boldsymbol{\tau_3}, \mathbf{0})\}$	$\{(\tau_1, 0), (\tau_3, 0)\}$	$\{(\tau_3, 0), (\boldsymbol{\tau_2}, \mathbf{2})\}$	$\{(\tau_3, 0), (\tau_2, 2), (\boldsymbol{\tau_5}, \mathbf{3})\}$	$\{(\tau_3, 0), (\tau_2, 2), (\tau_5, 3)\}$	$\{(\tau_4, \quad)\}$	$\{(\tau_4, 5)\}$	$\emptyset$
state$(\mathcal{AN}, T)$	$< a_1, b_0, c_2, d_2 >$	$< a_1, b_0, c_2, d_2 >$	$< a_1, \mathbf{b_1}, c_2, d_2 >$	$< a_1, b_1, c_2, d_2 >$	$< a_1, b_1, c_2, d_2 >$	$< \mathbf{a_0}, \mathbf{b_0}, \mathbf{c_1}, d_2 >$	$< a_0, b_0, c_1, d_2 >$	$< a_0, b_0, c_1, \mathbf{d_1} >$

Fig. 2. Example of a trajectory of the timed automata network of Example 2 starting from an initial state $<a_1, b_0, c_2, d_2>$ (at $t = 0$) to a stable state $< a_0, b_1, c_1, d_1 >$ (at $t = 10$). With $P = A(AN, t-1)$ as in Definition 8.

4 Learning Timed Automata Networks

This algorithm takes as input a model expressed as a Timed Automata Network, which the set of local transitions is empty, and time series data capturing the dynamics of the studied system. Given the influences between the components (or assuming all possible influences if no background knowledge is available), this algorithm generates the *timed local transitions* that could result in the the same changes of the model than the ones observed through the observation data.

4.1 Algorithm

In this section we propose an algorithm to build Timed Automata Networks from time series data. We assume that the latter data observations are provided as a *chronogram* of size T: the value of each variable is given for each time point t, $0 \leq t \leq T$, through a time interval discretization (see Definition 10 below).

Definition 10 (Chronogram). *A chronogram is a discretization of the time series data for each component of a biological regulatory network. It is presented by the following function Γ,*

$$\begin{aligned} \Gamma : [0,T] \subset \mathbb{N}^+ &\longrightarrow \{0,...,n\} \\ t &\longmapsto i \end{aligned}$$

with T is the maximum time point regarding the time series data called the size of the chronogram and n is the maximum level of discretization.

We note Γ_a a chronogram of the time series data for a component a.

Algorithm 1, `MoT-AN` (Modeling Timed Automata networks) shows the pseudo code of our implemented algorithm. It will generate all possible *timed local transitions* that can realize each observed change. Because of the delays and the non-determinism of the semantics, it is not possible to decide whether a timed local transition is absolutly correct or not. But we can output the minimal sets of time local transitions necessary to realize all the changes.

Theorem 1 (Completeness). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a Timed Automata Network, Γ be a chronogram of the components of $\mathcal{AN}$, $i \in \mathbb{N}$ and $R \in \mathcal{T}$ be the set of timed local transitions that realized the chronogram Γ such that $(a_i, l, a_j, \delta) \in R \implies |l| \leq i$. Let χ be the regulation influences of all $a \in \Sigma$. Let $\mathcal{AN}' = (\Sigma, \mathcal{S}, \emptyset)$ be a Timed Automata Network. Given $\mathcal{AN}'$, Γ, χ and i as input, Algorithm 1 is complete: it will output a set of Timed Automata Networks ϕ, such that $\exists \mathcal{AN}'' = (\Sigma, \mathcal{S}, \varphi') \in \phi$ with $R \subseteq \varphi'$.*
Proof is given in appendix.

Theorem 2 (Complexity). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a Timed Automata Network, $|\Sigma|$ be the number of automata of $\mathcal{AN}$ and η be the total number of local states of an automaton of $\mathcal{AN}$. Let Γ be a chronogram of the components of $\mathcal{AN}$ over τ units of time, such that c is the number of changes of Γ. The memory use of Algorithm 1 belongs to $O(\tau \cdot i^{|\Sigma|+1} \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$ that is bounded by $O(\tau \cdot |\Sigma|^{T \cdot |\Sigma|^{|\Sigma|+1}})$. The complexity of learning $\mathcal{AN}$ by generating timed local transitions from the observations of Γ with Algorithm 1 belongs to $O(c \cdot i^{|\Sigma|+1} + 2^{2 \cdot \tau \cdot i^{|\Sigma|+1}} + c \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$, that is bounded by $O(\tau \cdot 2^{3 \cdot \tau \cdot |\Sigma|^{|\Sigma|+1}})$.*
Proof is given in appendix.

4.2 Case Study

In this section we show how this method generates a T-AN model consistent with the set of biological regulatory time series data. First, the method uses discretized observations as an input (i.e. *chronogram*), thus it is necessary to treat first the time series data with another method in order to discretize it.

Our method may be summarized as follows:

- Detect biological components changes;
- Compute the *candidate timed local transitions* responsible for the network changes;
- Generate minimal subset of *candidate timed local transitions* that can realize all changes.

We apply the Algorithm 1 on learning the timed local transition $\tau \in \mathcal{T}$ of a simple example of a network $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ with 3 components ($|\Sigma| = 3$) whose chronogram is detailed in Fig. 3:

The first change occurs at $t_{min} = t_1 = 2$, denoted by `change(2)`. It is the gene z whose value changes from 0 to 1, thus the timed local transition that has realized this change has this form $z_0 \xrightarrow[\delta]{\ell} z_1$, where ℓ can be any combination of the values of the regulators at $t_1 - 1$ of z.

Let $\chi = \{b \to z, a \to z, a \to a\}$ be the set of regulation influences among the components of the network. According to χ the set of genes having influence on z is $\chi_z = \{a, b\}$. It means that $\ell = \{a_?, b_?\}$ or $\ell = \{a_?\}$ or $\ell = \{b_?\}$. The expression

Algorithm 1. MoT-AN: Modeling Timed Automata Networks

INPUT:
- Timed Automata Network $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ with $\mathcal{T} = \emptyset$;
- a chronogram $\Gamma = \bigcup_{a \in \Sigma} \Gamma_a$;
- the regulation influences $\chi = \bigcup_{a \in \Sigma} \chi_a$ and
- a maximal in-degree $i \in \mathbf{N}^*$

OUTPUT: ϕ a set of Timed Automata Networks that realize the time series data.

- Let $\varphi := \emptyset$
- Step 1: According to the chronogram Γ, for each time step where a component a changes its value from a_i to a_j, with $a_i, a_j \in \mathcal{S}(a)$:
  - Let $\delta(b) < t$ be the last time step where b has changed with $b \in \chi_a$
  - For each $l' \in \wp(\chi_a), |l'| \leq i$ generates all timed local transitions:

$$\tau := (a_k, l, a_l, t - \delta)$$

    such that $\delta = \delta(b'), b' \in l', \nexists b'' \in l', \delta(b'') > \delta(b')$ and $l = \{b_i \in \zeta(\delta) \mid b \in l'\}$
  - Add all timed local transition τ in φ
- Step 2: Generate ϕ the set of all Timed Automata Networks $\mathcal{AN}' = (\Sigma, \mathcal{S}, \varphi')$ with $\varphi' \subseteq \varphi$ a set of timed local transitions that can realize Γ such that φ' is minimal:

$$\forall \mathcal{AN}' = (\Sigma, \mathcal{S}, \varphi') \in \phi, \nexists \varphi'' \subseteq \varphi, \varphi'' \subset \varphi', \text{such that } \varphi' \text{can realize } \Gamma$$

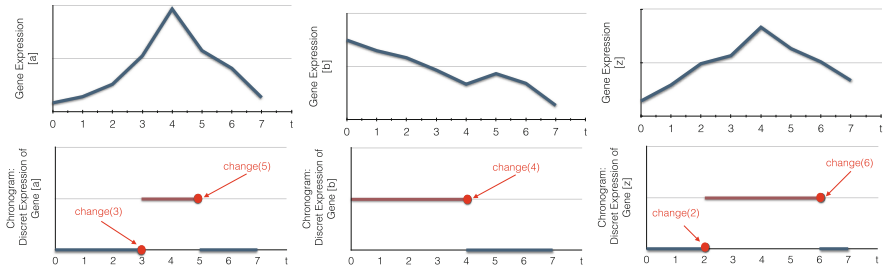

Fig. 3. Examples of the discretization of continous time series data into bi-valued *chronograms*. Abscissa (resp. ordinate) represents time (resp. gene expression levels). In this example, the expression level is discretized according to a threshold fixed to the half of the maximum gene expression value. `change(t)` indicates that the expression level of a biological component, here a gene, changes its value at a time point `t`.

level of the genes of χ_z when the researched candidate timed local transition (τ_i) is ongoing, i.e. during the partial steady state between two successive changes (t_i and t_{i-1}). This level is computed from the chronograms as follows:

- $a \in \chi_z$: $[a]_t = 0\ \forall t \in [0,2]$ - $b \in \chi_z$: $[b]_t = 1\ \forall t \in [0,2]$

Thus ℓ=$\{a_0, b_1\}$ or ℓ=$\{a_0\}$ or ℓ=$\{b_1\}$ and the set of *candidate timed local transitions* is: $\mathcal{T}_{change(2)} = \{\tau_1 = z_0 \xrightarrow[\delta_1]{a_0} z_1, \tau_2 = z_0 \xrightarrow[\delta_2]{b_1} z_1, \tau_3 = z_0 \xrightarrow[\delta_3]{a_0 \wedge b_1} z_1\}$. Since it is the first change, the delay of each timed local transition is the same: $\delta_1 = \delta_3 = \delta_3 = 2$.

The second change occurs at $t_2 = 3$ and denoted by $change(3)$. Here it is the gene a whose state changes from a_0 to a_1, thus the timed local transition that realize this change has this form $\tau = a_0 \xrightarrow[\delta]{\ell} a_1$ where ℓ can be any combination of the regulators value at t_1 of z. According to χ the genes influencing a are $\chi_a = \{a\}$. It means that $\ell = \{a_?\}$ and the expression level of a between t_1 and t_2 is a_0. So $\ell = \{a_0\}$. Thus there is only one *candidate timed local transition*: $\mathcal{T}_{change(3)} = \{\tau = a_0 \xrightarrow[1]{\emptyset} a_1\}$.

The third change occurs at $t_3 = 4$, $change(4)$. Here it is the gene b whose value changes from b_1 to b_0, thus the timed local transition that realize this change is of this form, $\tau = b_1 \xrightarrow[\delta]{\ell} b_0$ where ℓ can be any combination of the regulators value at $t_3 - 1$ of b. According to χ there is no gene that can influence b, thus no timed local transition can realize this change.

The fourth change occurs at $t_4 = 5$, $change(5)$. Here it is a whose expression decreases and changes from a_1 to a_0, thus the candidate timed local transition that could realize this change has this form, $\tau = a_1 \xrightarrow[\delta]{\ell} a_0$ where ℓ can be any combination of the regulators value at $t_4 - 1$ of a. According to χ the set of genes having influences on a is $\chi_a = \{a\}$. Again $\ell = \{a_?\}$ and since the expression level

of a since its last change is a_1. we have $A = \{a_1\}$ and there is only one candidate timed local transition: $\mathcal{T}_{change(5)} = \{\tau = a_1 \xrightarrow[1]{\emptyset} a_0\}$.

The fifth change occurs at $t_5 = 6$, $change(6)$. Here it is z whose value changes from z_1 to z_0, thus the time local transition that has realized this change has the form of: $\tau = z_1 \xrightarrow[\delta]{\ell} z_0$ where ℓ can be any combination of the regulators value at $t_3 - 1$ of b. Since $\chi_z = \{a, b\}$, it means that $\ell = \{a_?, b_?\}$ or $\ell = \{a_?\}$ or $\ell = \{b_?\}$ The expression level of a and b at $t_5 - 1$ is respectively a_0 and b_0. Thus $\ell = \{a_0, b_1\}$ or $\ell = \{a_0\}$ or $\ell = \{b_0\}$. The candidate timed local states are:
$\mathcal{T}_{change(6)} = \{\tau_1 = z_1 \xrightarrow[\delta_1]{a_0} z_0,\ \tau_2 = z_1 \xrightarrow[\delta_2]{b_0} z_0,\ \tau_3 = z_1 \xrightarrow[\delta_3]{a_0 \wedge b_0} z_0\}$.

The last change of a is at $t_4 = 5$ and the last change of b is at $t_3 = 4$. Thus $\delta_1 = t_5 - t_4 = 1$, $\delta_2 = t_5 - t_3 = 2$, $\delta_3 = t_5 - max(t_4, t_3) = 1$.

After processing all changes, the set of *timed local transitions* that could realize the chronograms are:
$\mathcal{T}_{change(2)} = \{\tau_1 = z_0 \xrightarrow[2]{a_0} z_1, \tau_2 = z_0 \xrightarrow[2]{b_1} z_1, \tau_3 = z_0 \xrightarrow[2]{a_0 \wedge b_1} z_1\}$
$\mathcal{T}_{change(3)} = \{\tau_4 = a_0 \xrightarrow[1]{\emptyset} a_1\}$, $\mathcal{T}_{change(5)} = \{\tau_5 = a_1 \xrightarrow[1]{\emptyset} a_0\}$
$\mathcal{T}_{change(6)} = \{\tau_6 = z_1 \xrightarrow[1]{a_0} z_0,\ \tau_7 = z_1 \xrightarrow[2]{b_0} z_0,\ \tau_8 = z_1 \xrightarrow[1]{a_0 \wedge b_0} z_0\}$.

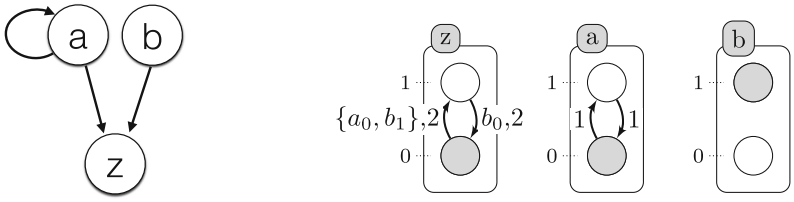

Fig. 4. Left: influence graph modeling of the case study example (Fig. 3). Right, one of the Timed Automata Networks generated by the Algorithm 1. The labels of each local transition stands for the local states of the automata which make the transition playable and its delay (time needed for the transition to be performed).

All *timed local transitions* learned are consistent with all observed time series data and the regulation influences given as input. The used method ensures completeness, we have the full set of timed local transitions that can explain the observations. By generating all minimal subsets of this set of timed local transitions, one of those subset will be the set who realized the observations (Fig. 4).

5 Evaluation

In this section, we provide two evaluations of Algorithm 1. We evaluate the capacity of our algorithm[1] to construct models for prediction and the impact of the quantity of observations on run time. Here we process chronograms obtained from time series data of the DREAM4 challenge [25].

5.1 DREAM4

In this section, we assess the efficiency of our algorithm through case studies coming from the DREAM4 challenge. DREAM challenges are annual reverse engineering challenges that provide biological case studies. In this section, we focus on the datasets coming from DREAM4. It provides data for systems of different size (10 genes on one hand, 100 genes on the other hand), allowing us to assess the scalability of our approach. The input data that we tackle here consists of the following: 5 different systems each composed of 100 genes, all coming from E. coli and yeast networks. For every such system, the available data are the following: (i) 10 time series data with 21 time points and 1000 is the duration of each time series; (ii) steady state for wild type; (iii) steady states after knocking out each gene; (iv) steady states after knocking down each gene (i.e. forcing its transcription rate at 50 %); (v) steady states after some random multifactorial perturbations. We processed all the data. Here, we focus on the management of time series data.

Each time series includes different perturbations that are maintained all time along during the first 10 time points and applied to at most 30 % of the genes. In this setting, a perturbation means a significant increase or decrease of the gene expression. In the raw data of the time series, gene expression values are given as real numbers between 0 and 1. To apply our approach, we chose to discretize those data into two to six qualitative values. Increasing the number of qualitatives values from 2 to 4 improves the precision, but then the score decrease from 5, must likely because of overfitting: the relations learned become too precised and can't be applied on something else than the training data. The best score we obtain were with 4 qualitatives values and are reported in Fig. 5. Each gene is discretized in an independent manner, with respect to the following procedure: we compute the average value of the gene expression among all data of a time series, then the values between the average and the maximal/minimal value are divided into as many levels. Discretizing the data according to the average value of expression is expected to reduce the impact of perturbation on the discretization and thus on the model learned.

The DREAM4 challenge offers two different problems, which consist in predicting (i) the structure of the gene interactions (in terms of an unsigned directed graph); (ii) attractors in some given conditions. Our method is not designed to tackle the first issue, indeed we need to know those influences. But the models

[1] All programs, described in this article, for Timed Automata Network generation are implemented in ASP and are available online at: http://www.irccyn.ec-nantes.fr/~benabdal/modeling-biological-regulatory-networks.zip.

we learn can be applied to predict trajectories and thus attractors. Here we use the influences graphs expected in the first problem as background knowledge (given in appendix) to tackle the attractor prediction part of the challenge.

5.2 Results

For this evaluation, we are given an initial state and 5 different dual gene knockouts conditions. The goal is to predict the attractor in which the system will fall from the initial state for each dual knockout. Here, we just choose the first model that our algorithm output and use the biggest set of fireable timed local transition at each time step to produce a trajectory until a cycle is detected. The first state of this cycle is reverse discretized and proposed as the predicted state. In the challenge, the quality of the prediction is evaluated by computing the mean square error (MSE) between the predicted state and the expected one. As shown in Fig. 5, the precision we achieved in those experiments is quite good considering the results of the competitors of the DREAM4 challenge [1]. Their results range between 0.010 and 0.075 for the same evaluation settings, which we are comparable to (0.033 to 0.086) giving us encouraging results. Regarding run time, learning and predicting the trajectories of the benchmarks of 10 genes took less than 30 s and the same experiements for the benchmarks of 100 genes took about 3 h and 20 min on one processor Intel Core2 Duo (P8400, 2.26 GHz).

Benchmark	Number of genes	MSE
insilico_size10_1	10	0.086
insilico_size10_2	10	0.080
insilico_size10_3	10	0.076
insilico_size10_4	10	0.039
insilico_size10_5	10	0.076

Benchmark	Number of genes	MSE
insilico_size100_1	100	0.052
insilico_size100_2	100	0.042
insilico_size100_3	100	0.033
insilico_size100_4	100	0.033
insilico_size100_5	100	0.052

Fig. 5. Evaluation of our method on learning and prediction of the evolution of gene regulatory network benchmarks from the DREAM4 challenge through the Mean Square Error (MSE): 10 variables benchmarks (left) and 100 variables benchmarks (right).

To achieve this score, we had to perform several tests by varying the discretization precision and the complexity of the dynamics learned. Those tests also allows us to assess the scalability of our approach in practice. Figure 6 shows the impact of both timed local transition indegree and discretization level on run time.

In the results obtained from the experimentation of our algorithm on the time series data of the DREAM4 we can see the exponential influence on the run time of the indegree per local transition considered as well as the level of discretization chosen for all the 5 different networks. But it also shows that in practice our approach can tackle big network, here 100 genes.

5.3 Discussion

We propose a new method `MoT-AN` (Algorithm 1) to automatically infer models that could explain the dynamic evolution of the biological system. We illustrated the merits of this method by applying it on a large real biological system

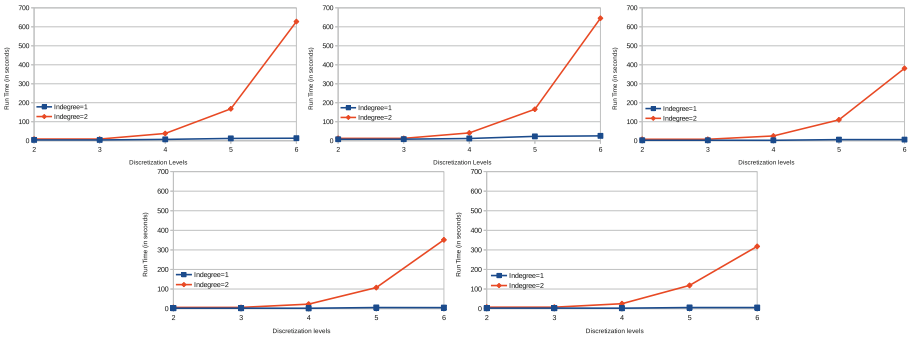

Fig. 6. Evolution of run time on processing different models inferred from time series data of DREAM4 (100 variables benchmarks), varying indregree of local timed transitions and discretization levels. These tests were performed on a processor Intel Core i7 (4700, 3 GHz) with 16 GB of RAM.

(DREAM4 challenge). As a result we obtain in few seconds models that are proved to be relevant (this relevance is qualified in terms of mean square error with gold standard network) This algorithm is implemented using Answer Set Programming [4,5], thus provides the exhaustive enumeration of all models.

The main limit of the approach presented in this paper is the fact that topology of the network is considered as granted. As discussed in the introduction of the paper, there is a wide range of algorithms designed to address this issue. Furthermore, such interaction graphs could be deduced from the available reliable databases of biological networks. Some examples of data bases for human regulatory knowledge are: Pathways Interaction Database [20], Human Integrated Pathway DB [32] and Causal Biological Network Database [30].

Various inference approaches [11,20,26] from time series data based on prior knowledge about component interactions have been proposed. But they share a common limit: they focus on static characterization of the interactions and they do not allow to infer dynamic behaviors where delays are involved. The merits of our contribution lie in the fact that we overcome such limits, and we infer delays in a qualitative dynamic modeling of the network.

6 Conclusion and Perspectives

In this paper, we propose an approach takes a background knowledge under the form of regulation graph and time series data as an input. The contribution of our method lies in the fact that it identifies the set of interactions between biological components by (1) concertizing the signs (negative or positive) (2) providing thresholds and associating the quantitative time delays. As a result, we have a set of Timed Automata networks that explain the biological network

evolution. The Algorithm 1 is implemented in ASP. We illustrated the applicability and limits of the proposed method through benchmarks from DREAM4. This opens the way to promising applications in the cooperation between biologists and computer scientists. Further works now consist in discussing the kind of information one can get on Timed Automata Network by analyzing the associated untimed model. We also plan to improve our implementation to make it robust against noisy and scarse data as like in the DREAM8: Heritage-DREAM breast cancer network inference challenge.

A Appendixes

A.1 Proof of Theorem 1

Theorem 3 (Completeness). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a Timed Automata Network, Γ be a chronogram of the components of $\mathcal{AN}$, $i \in \mathsf{N}$ and $R \in \mathcal{T}$ be the set of timed local transitions that realized the chronogram Γ such that $(a_i, l, a_j, \delta) \in R \implies |l| \leq i$. Let χ be the regulation influences of all $a \in \Sigma$. Let $\mathcal{AN}' = (\Sigma, \mathcal{S}, \emptyset)$ be a Timed Automata Network. Given $\mathcal{AN}'$, Γ, χ and i as input, Algorithm 1 is complete: it will output a set of Timed Automata Network ϕ, such that $\exists \mathcal{AN}'' = (\Sigma, \mathcal{S}, \varphi') \in \phi$ with $R \subseteq \varphi'$.*

Proof. Let us suppose that the algorithm is not complete, then there is a timed local transition $h \in R$ that realized Γ and $h \notin \varphi'$. In Algorithm 1, after step 1, φ contains all timed local transitions that can realize each change of the chronogram Γ. Here there is no timed local transition $h \in R$ that realizes Γ which is not generated by the algorithm, so $h \in \varphi$. Then it implies that at step 2, $\forall \varphi', h \notin \varphi'$. But since h realizes one of the change of Γ and h is generated at step 1, then it will be present in one of the minimal subset of timed local transitions. Such that h will be in one of the networks outputted by the algorithm. □

A.2 Proof of Theorem 2

Theorem 4 (Complexity). *Let $\mathcal{AN} = (\Sigma, \mathcal{S}, \mathcal{T})$ be a Timed Automata Network, $|\Sigma|$ be the number of automaton of $\mathcal{AN}$ and η be the total number of local state of a automaton of $\mathcal{AN}$. Let Γ be a chronogram of the components of $\mathcal{AN}$ over τ units of time, such that c is the number of changes of Γ. The memory use of Algorithm 1 belongs to $O(\tau \cdot i^{|\Sigma|+1} \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$ that is bounded by $O(\tau \cdot |\Sigma|^{T \cdot |\Sigma|^{|\Sigma|+1}})$. The complexity of learning $\mathcal{AN}$ by generating timed local transitions from the observations of Γ with Algorithm 1 belongs to $O(c \cdot i^{|\Sigma|+1} + 2^{2 \cdot \tau \cdot i^{|\Sigma|+1}} + c \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$, that is bounded by $O(\tau \cdot 2^{3 \cdot \tau \cdot |\Sigma|^{|\Sigma|+1}})$.*

Proof. Let i be the maximal indegree of a timed local transition in $\mathcal{AN}$, $0 \leq i \leq |\Sigma|$. Let p be an automaton local state of $\mathcal{AN}$ then $|\Sigma|$ is maximal the number of automaton that can influence p. There is $i^{|\Sigma|}$ possible combinations of those regulators that can influences p at the same time forming a timed local transition. There is at most τ possible delays, so that there are $\tau \cdot |\Sigma| \cdot i^{|\Sigma|}$ possibles

timed local transitions, thus in Algorithm 1 at step 1, the memory is bounded by $O(\tau \cdot i^{|\Sigma|+1})$*, which belongs to* $O(\tau \cdot |\Sigma|^{|\Sigma|+1})$ *since* $0 \leq i \leq |\Sigma|$*. Generating all minimal subsets of timed local transitions* φ *of* $\mathcal{AN}$ *that can realize* Γ *can require to generate at most* $2^{\tau \cdot |\Sigma| \cdot i^{|\Sigma|+1}}$ *set of rules. Thus, the memory of our algorithm belongs to* $O(\tau \cdot i^{|\Sigma|+1} \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$ *and is bounded by* $O(\tau \cdot |\Sigma|^{|\Sigma|+1} \cdot 2^{\tau \cdot |\Sigma|^{|\Sigma|+1}})$*.*

The complexity of this algorithm belongs to $O(c \cdot i|\Sigma| + 1)$*. Since* $0 \leq i \leq |\Sigma|$ *and* $0 \leq c \leq \tau$ *the complexity of Algorithm 1 is bounded by* $O(\tau \cdot |\Sigma|^{|\Sigma|+1}))$*.*

Generating all minimal subsets of timed local transitions φ *of* $\mathcal{AN}'$ *that realize* Γ *can require to generate at most* $2^{\tau \cdot i^{|\Sigma|+1}}$ *set of timed local transitions. Each set has to be compared with the others to keep only the minimal ones, which costs* $O(2^{2 \cdot \tau \cdot i^{|\Sigma|+1}})$*. Furthermore, each set of timed local transitions has to realize each change of* Γ*, it requires to check* c *changes and it costs* $O(c \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$*. Finally, the total complexity of learning* $\mathcal{AN}$ *by generating timed local transitions from the observations of* Γ *belongs to* $O(c \cdot i^{|\Sigma|+1} + 2^{2 \cdot \tau \cdot i^{|\Sigma|+1}} + c \cdot 2^{\tau \cdot i^{|\Sigma|+1}})$*. that is bounded by* $O(3\tau \cdot 2^{2 \cdot \tau \cdot |\Sigma|^{|\Sigma|+1}})$*.*

□

A.3 DREAM4: Influence Network

The Fig. 7 presents the regulatory graph that we are based on to identify the signs (negative or positive), the thresholds and the quantitative time delays of the learned transitions.

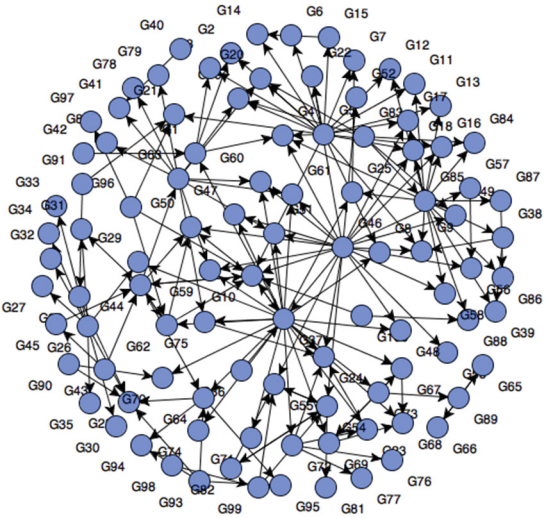

Fig. 7. The influence network of the DREAM4 challenge model (100 genes) given by GeneNetWeaver (GNW) data generator [27]. Each node is a gene and each edge is an influence from the source to the target gene.

References

1. Ben Abdallah, E., Folschette, M., Roux, O., Magnin, M.: Exhaustive analysis of dynamical properties of biological regulatory networks with answer set programming. In: IEEE International Conference on Bioinformatics and Biomedicine (BIBM), pp. 281–285. IEEE (2015)
2. Ahmad, J., Bernot, G., Comet, J.-P., Lime, D., Roux, O.: Hybrid modelling and dynamical analysis of gene regulatory networks with delays. ComPlexUs **3**(4), 231–251 (2006)
3. Akutsu, T., Tamura, T., Horimoto, K.: Completing networks using observed data. In: Gavaldà, R., Lugosi, G., Zeugmann, T., Zilles, S. (eds.) ALT 2009. LNCS, vol. 5809, pp. 126–140. Springer, Heidelberg (2009)
4. Anwar, S., Baral, C., Inoue, K.: Encoding higher level extensions of petri nets in answer set programming. In: Cabalar, P., Son, T.C. (eds.) LPNMR 2013. LNCS, vol. 8148, pp. 116–121. Springer, Heidelberg (2013)
5. Baral, C.: Knowledge representation, reasoning and declarative problem solving. Cambridge University Press, New York (2003)
6. Callebaut, W.: Scientific perspectivism: a philosopher of sciences response to the challenge of big data biology. Stud. Hist. Philos. Sci. Part C. Stud. Hist. Philos. Biol. Biomed. Sci. **43**(1), 69–80 (2012)
7. Comet, J.-P., Fromentin, J., Bernot, G., Roux, O.: A formal model for gene regulatory networks with time delays. In: Chan, J.H., Ong, Y.-S., Cho, S.-B. (eds.) CSBio 2010. CCIS, vol. 115, pp. 1–13. Springer, Heidelberg (2010)
8. Fan, J., Han, F., Liu, H.: Challenges of big data analysis. Nat. Sci. Rev. **1**(2), 293–314 (2014)
9. Folschette, M., Paulevé, L., Inoue, K., Magnin, M., Roux, O.: Identification of biological regulatory networks from process hitting models. Theoret. Comput. Sci. **568**, 49–71 (2015)
10. Freedman, P.: Time, petri nets, and robotics. IEEE Trans. Robot. Autom. **7**(4), 417–433 (1991)
11. Gallet, E., Manceny, M., Le Gall, P., Ballarini, P.: An LTL model checking approach for biological parameter inference. In: Merz, S., Pang, J. (eds.) ICFEM 2014. LNCS, vol. 8829, pp. 155–170. Springer, Heidelberg (2014)
12. Goldstein, Y.A.B., Bockmayr, A.: A lattice-theoretic framework for metabolic pathway analysis. In: Gupta, A., Henzinger, T.A. (eds.) CMSB 2013. LNCS, vol. 8130, pp. 178–191. Springer, Heidelberg (2013)
13. Harvey, I., Bossomaier, T.: Time out of joint: attractors in asynchronous random boolean networks. In: Proceedings of the Fourth European Conference on Artificial Life, pp. 67–75. MIT Press, Cambridge (1997)
14. Kim, S.Y., Imoto, S., Miyano, S.: Inferring gene networks from time series microarray data using dynamic bayesian networks. Briefings Bioinf. **4**(3), 228–235 (2003)
15. Koh, C., Fang-Xiang, W., Selvaraj, G., Kusalik, A.J.: Using a state-space model and location analysis to infer time-delayed regulatory networks. EURASIP J. Bioinf. Syst. Biol. **2009**(1), 1 (2009)
16. Koksal, A.S., Yewen, P., Srivastava, S., Bodik, R., Fisher, J., Piterman, N.: Synthesis of biological models from mutation experiments. ACM SIGPLAN Not. **48**, 469–482 (2013). ACM
17. Liu, T.-F., Sung, W.-K., Mittal, A.: Learning multi-time delay gene network using bayesian network framework. In: 16th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2004, pp. 640–645. IEEE (2004)

18. Marx, V.: Biology: the big challenges of big data. Nature **498**(7453), 255–260 (2013)
19. Matsuno, H., doi, A., Nagasaki, M., Miyano, S.: Hybrid petri net representation of gene regulatory network. In: Pacific Symposium on Biocomputing, vol. 5, p. 87. World Scientific Press, Singapore (2000)
20. Ostrowski, M., Paulevé, L., Schaub, T., Siegel, A., Guziolowski, C.: Boolean network identification from multiplex time series data. In: Roux, O., Bourdon, J. (eds.) CMSB 2015. LNCS, vol. 9308, pp. 170–181. Springer, Heidelberg (2015)
21. Paoletti, N., Yordanov, B., Hamadi, Y., Wintersteiger, C.M., Kugler, H.: Analyzing and synthesizing genomic logic functions. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 343–357. Springer, Heidelberg (2014)
22. Paulevé, L.: Goal-oriented reduction of automata networks. In: CMSB 2016–14th Conference on Computational Methods for Systems Biology (2016)
23. Paulevé, L., Chancellor, C., Folschette, M., Magnin, M., Roux, O.: Logical Modeling of Biological Systems, chapter Analyzing Large Network Dynamics with Process Hitting, pp. 125–166. Wiley, Hoboken (2014)
24. Paulevé, L., Magnin, M., Roux, O.: Refining dynamics of gene regulatory networks in a stochastic π-calculus framework. In: Priami, C., Back, R.-J., Petre, I., de Vink, E. (eds.) Transactions on Computational Systems Biology XIII. LNCS, vol. 6575, pp. 171–191. Springer, Heidelberg (2011)
25. Prill, R.J., Saez-Rodriguez, J., Alexopoulos, L.G., Sorger, P.K., Stolovitzky, G.: Crowdsourcing network inference: the dream predictive signaling network challenge. Sci. Signal. **4**(189), mr7 (2011)
26. Saez-Rodriguez, J., Alexopoulos, L.G., Epperlein, J., Samaga, R., Lauffenburger, D.A., Klamt, S., Sorger, P.K.: Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. Mol. Syst. Biol. **5**(1), 331 (2009)
27. Schaffter, T., Marbach, D., Floreano, D.: Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics **27**(16), 2263–2270 (2011)
28. Siebert, H., Bockmayr, A.: Temporal constraints in the logical analysis of regulatory networks. Theoret. Comput. Sci. **391**(3), 258–275 (2008)
29. Sima, C., Hua, J., Jung, S.: Inference of gene regulatory networks using time-series data: a survey. Curr. Genomics **10**(6), 416–429 (2009)
30. Talikka, M., Boue, S., Schlage, W.K.: Causal biological network database: a comprehensive platform of causal biological network models focused on the pulmonary and vascular systems. Comput. Syst. Toxicol. **2015**, 65–93 (2015)
31. Thomas, R.: Regulatory networks seen as asynchronous automata: a logical description. J. Theoret. Biol. **153**(1), 1–23 (1991)
32. Namhee, Y., Seo, J., Rho, K., Jang, Y., Park, J., Kim, W.K., Lee, S.: Hipathdb: a human-integrated pathway database with facile visualization. Nucleic Acids Res. **40**(D1), D797–D802 (2012)
33. Zhang, Z.-Y., Horimoto, K., Liu, Z.: Time series segmentation for gene regulatory process with time-window-extension (2008)
34. Zhao, W., Serpedin, E., Dougherty, E.R.: Inferring gene regulatory networks from time series data using the minimum description length principle. Bioinformatics **22**(17), 2129–2135 (2006)