Learning normal and delayed behavior of max-plus linear systems from input and output event data

Ibis Velasquez, Euriell Le Corronc, Euriell Le Corronc, Yannick Pencolé

Learning normal and delayed behavior of max-plus linear systems from input and output event data

Ibis Velasquez[1], Yannick Pencolé[1], and Euriell Le Corronc[1]

***Abstract*— In this paper, we introduce the problem of learning max-plus linear models from event data available through unlabeled logs. We present a method for obtaining these models when the logs contain input and output event dates generated by a system in both normal conditions and abnormal conditions caused by failures. The properties of the method are presented, as well as results from a simulated example.**

I. Introduction

Timed Event Graphs (TEGs) are a type of discrete event system that can be represented using linear equations in algebraic structures called dioids [1]. They are especially useful when modeling systems with synchronizations and delays, like manufacturing systems and transportation networks. We are interested in the problem of learning the models of these systems from data available through logs. The acquired models could then be used to apply control methods to the system [1] or to perform fault diagnosis [8].

The subject of obtaining these linear models has been frequently discussed in the literature as an identification problem, and several approaches have been introduced. Examples include state-space identification from input and output data with knowledge of the structure of the system, discussed in [3], or from state data, presented in [7] as a regression problem. Another example is the identification of stochastic systems from input and state data discussed in [9] and [4]. However, these methods estimate parameters of a unique underlying system from a single set of measurements. We propose a simple method for learning complete input-output models of a system for normal behavior and abnormal behavior deriving from failures; the starting point is data available through unlabeled logs containing input and output event dates obtained in both operating conditions.

II. Timed event graphs in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$

A. Timed Event Graphs

Timed Event Graphs (TEGs) are a subclass of timed Petri nets in which each place has precisely one upstream transition and one downstream transition, all arcs have weight one, and transitions are fired as soon as they are enabled (earliest firing rule). The *events* of a TEG are the firings of its transitions. The following definition introduces the notations associated with TEGs that will be used throughout this paper.

Definition 1: A *timed event graph* [1], [10] $\mathcal{G}$ is a 5-tuple $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{A}, M_0, D)$ such that $\mathcal{P}$ (resp. $\mathcal{T}$) is a finite set of nodes called *places* (resp. *transitions*) ($\mathcal{P} \cap \mathcal{T} = \varnothing$); $\mathcal{A} \subseteq (\mathcal{P} \times \mathcal{T}) \cup (\mathcal{T} \times \mathcal{P})$ is the set of arcs going from places to transitions and from transitions to places, and every place has precisely one incoming arc and one outgoing arc; $M_0 : \mathcal{P} \to \mathbb{N}$ is the initial marking; $D : \mathcal{P} \to \mathbb{N}$ is the holding time (or duration) for a token in a place.

The transitions in a TEG are divided in three disjoint subsets: internal transitions (denoted x_i) have both incoming and outgoing arcs, output transitions (denoted y_i) have incoming arcs but do not have outgoing arcs, and input transitions (u_i) have outgoing arcs but no incoming arcs. This paper deals with non-autonomous TEGs, meaning that neither the set of input transitions nor the set of output transitions are empty. Input events are triggered by sources external to the TEG (as input transitions are considered to always be enabled).

Example 1: Fig. 1 represents a TEG with two inputs (u_0, u_1), two outputs (y_0, y_1), three internal transitions (x_0, x_1, x_2) and eight places ($p_0, ..., p_7$). The holding times of all places except p_1 and p_6 are equal to zero.

B. Dioids

The structure of a TEG and the firings of its transitions can be described using dioids. A *dioid* is as a semiring with addition $\oplus$ and product $\otimes$ where $\oplus$ is idempotent ($a \oplus a = a$). The zero element is denoted ε and the identity element is denoted e. In a dioid $(\mathcal{D}, \oplus, \otimes)$, the definition of addition $\oplus$ induces a partial order $\succeq : \forall a, b \in \mathcal{D}, \quad a \succeq b \Leftrightarrow a = a \oplus b$. Dioids are described in detail in [1] and papers on the subject of TEGs and max-plus linear systems; only some essential notions will be described in this section.

Definition 2: A dioid $(\mathcal{D}, \oplus, \otimes)$ is *complete* if it is closed for infinite sums and the left and right distributivity of the

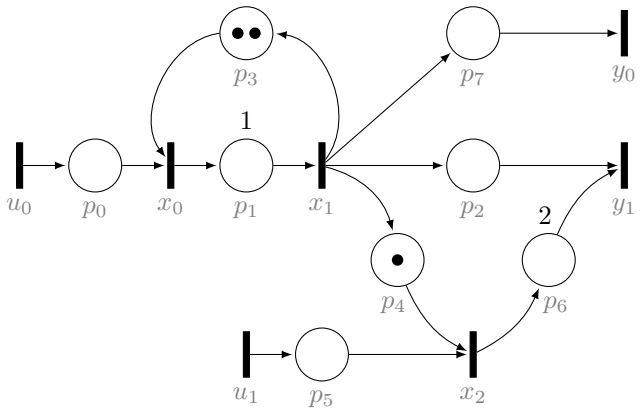

Fig. 1. A TEG with two inputs and two outputs.

[1]LAAS-CNRS, Université de Toulouse, CNRS, INSA, UT3. Toulouse, France.
{ibis.velasquez, yannick.pencole, euriell.le.corronc}@laas.fr

product with respect to addition holds for infinite sums.

Example 2: The Boolean set $\mathbb{B} = \{0, 1\}$ with the logical or as the addition $\oplus$ and the logical and as the multiplication $\otimes$, with $\varepsilon = 0$ and $e = 1$, is a complete dioid: $(\mathbb{B}, \text{or}, \text{and})$.

A complete dioid $(\mathcal{D}, \oplus, \otimes)$ admits a top element denoted $\top = \bigoplus_{a \in \mathcal{D}} a \in \mathcal{D}$ and an associative, commutative, idempotent *greatest lower bound* (or infimum) operator denoted $\wedge$: $\forall a, b \in \mathcal{D}, \quad a \wedge b = \bigoplus \{x \in \mathcal{D} \,|\, a \succeq x \oplus a, b \succeq x \oplus b\}$, which admits $\top$ as neutral element. Also, $a \succeq b \Leftrightarrow b = a \wedge b$.

We denote $a^0 = e$ and $\forall i \in \mathbb{N}^+, a^i = aa^{i-1}$. The *Kleene star* operator * is defined as $a^* = \bigoplus_{i \geq 0} a^i$; and for matrices $M \in \mathcal{D}^{n \times n}$, $M^* = \bigoplus_{i \geq 0} M^i$. In a complete dioid $(\mathcal{D}, \oplus, \otimes)$, the least solution of $ax \oplus b = x$ is $x = a^*b$.

C. Residuation

Let $f : \mathcal{D} \to \mathcal{C}$ be an isotone mapping between two complete dioids $\mathcal{D}$ and $\mathcal{C}$. Mapping f is *residuated* if for every $b \in \mathcal{C}$, there exists a greatest solution for $f(x) \preceq b$ denoted $f^\sharp(b) = \bigoplus_{x \in \mathcal{D}, f(x) \preceq b} x$ (the *residual* of f). It is the unique isotone mapping such that $f \circ f^\sharp \preceq I_\mathcal{C}$ and $f^\sharp \circ f \succeq I_\mathcal{D}$ where $I_\mathcal{C}$ (resp. $I_\mathcal{D}$) is the identity mapping on $\mathcal{C}$ (resp. $\mathcal{D}$).

Example 3: The mapping $R_a : x \mapsto x \otimes a$ over a dioid $\mathcal{D}$ (right multiplication by a) is a non-invertible residuated mapping. Its residual, $R_a^\sharp(b)$ (*right division*), acts as a pseudo inverse and is denoted $b \not/ a = \bigoplus_{x \in \mathcal{D}, xa \preceq b} x$. $b \not/ a$ is therefore the greatest solution to $xa \preceq b$. Note that $(xa) \not/ a \succeq x$.

D. Dioids of power series

Let $\mathbb{B}[\![\gamma, \delta]\!]$ be the set of formal power series s in two commutative variables γ and δ with exponents in $\mathbb{Z}$ (series of monomials $\gamma^n \delta^t$; $n, t \in \mathbb{Z}$) and coefficients in $\mathbb{B} = \{0, 1\}$. s is denoted $s = \bigoplus_{(n,t) \in \mathbb{Z}^2} c(n,t) \gamma^n \delta^t$ with $c : \mathbb{Z}^2 \to \mathbb{B}$. Let $s_{1(\text{resp.}2)} = \bigoplus_{(n,t) \in \mathbb{Z}^2} c_{1(\text{resp.}2)}(n,t) \gamma^n \delta^t \in \mathbb{B}[\![\gamma, \delta]\!]$ denote any pair of formal power series, we have $s_1 \oplus s_2 = \bigoplus_{(n,t) \in \mathbb{Z}^2} (c_1(n,t) \text{ or } c_2(n,t)) \gamma^n \delta^t$ and $s_1 \otimes s_2 = \bigoplus_{(n,t) \in \mathbb{Z}^2} c(n,t) \gamma^n \delta^t$ with $c(n,t)$ defined in $(\mathbb{B}, \text{or}, \text{and})$ as $c(n,t) = \bigoplus_{n = n_1 + n_2, t = t_1 + t_2} c_1(n_1, t_1) \otimes c_2(n_2, t_2)$.

$(\mathbb{B}[\![\gamma, \delta]\!], \oplus, \otimes)$ is a complete commutative dioid with $\varepsilon = \bigoplus_{(n,t) \in \mathbb{Z}^2} 0 \gamma^n \delta^t$ and $e = 1\gamma^0\delta^0 \oplus \bigoplus_{(n,t) \in \mathbb{Z}^2} 0 \gamma^n \delta^t$. In the previous expressions, 0 stands for the absence of a monomial in the series and 1 stands for its presence, so the notation will be simplified; for instance, $e = \gamma^0 \delta^0$.

Definition 3: Let us define the congruence relation $\equiv$ as $s_1 \equiv s_2 \Leftrightarrow \gamma^*(\delta^{-1})^* s_1 = \gamma^*(\delta^{-1})^* s_2$. Dioid $(\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!], \oplus, \otimes)$ is the quotient of dioid $(\mathbb{B}[\![\gamma, \delta]\!], \oplus, \otimes)$ induced by the congruence relation $\equiv$.

An equivalent class $[s]_{\gamma^*(\delta^{-1})^*} \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ covers all the series of $\mathbb{B}[\![\gamma, \delta]\!]$ that are equivalent modulo $\gamma^*(\delta^{-1})^*$. The zero element in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ is $[\varepsilon]_{\gamma^*(\delta^{-1})^*}$ and the identity element is $[e]_{\gamma^*(\delta^{-1})^*}$. In the following, $[s]_{\gamma^*(\delta^{-1})^*}$ will simply be denoted $s \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$. The following rule applies to the addition of monomials in $(\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!], \oplus, \otimes)$:

$$\gamma^n \delta^t \oplus \gamma^{n'} \delta^t = \gamma^{\min(n,n')} \delta^t. \tag{1}$$

Since $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ is complete, any couple $s_1, s_2 \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ accepts an infimum denoted $s_1 \wedge s_2$ which can be computed based on the following rule for monomials: $\gamma^n \delta^t \wedge \gamma^{n'} \delta^{t'} = \gamma^{\max(n,n')} \delta^{\min(t,t')}$.

E. Linear systems in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$

Let $\mathcal{G}$ be a TEG containing n_u input transitions, n_x internal transitions and n_y output transitions. $\mathcal{G}$ can be modeled as a linear system in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ through the state representation:

$$\begin{cases} X = AX \oplus BU \\ Y = CX \end{cases} \tag{2}$$

where $A \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]^{n_x \times n_x}$ details the connections among internal transitions, and $B \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]^{n_x \times n_u}$ (resp. $C \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]^{n_y \times n_x}$) details the connections from input (resp. internal) transitions to internal (resp. output) transitions. The places making these connections are represented in entries of A, B and C through the addition of monomials $\gamma^n \delta^d$ where d is the holding time and n is the number of initial tokens. When there are no places connecting two given transitions, the respective entry is ε and denoted $\cdot$ in the matrix. X, U and Y in (2) are column vectors of dimension n_x, n_u and n_y respectively. Their components are series detailing the firings of the transitions (see Example 7 in Section IV-B for an illustration).

Example 4: The matrices of the state representation of the TEG in Fig. 1 are:

$$A = \begin{pmatrix} \cdot & \gamma^2\delta^0 & \cdot \\ \gamma^0\delta^1 & \cdot & \cdot \\ \cdot & \gamma^1\delta^0 & \cdot \end{pmatrix}, \quad B = \begin{pmatrix} \gamma^0\delta^0 & \cdot \\ \cdot & \cdot \\ \cdot & \gamma^0\delta^0 \end{pmatrix},$$
$$C = \begin{pmatrix} \cdot & \gamma^0\delta^0 & \cdot \\ \cdot & \gamma^0\delta^0 & \gamma^0\delta^2 \end{pmatrix}.$$

The relation between inputs and outputs in a TEG is

$$Y = HU \quad (\text{with } H = CA^*B \in \mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]^{n_y \times n_u}). \tag{3}$$

Therefore, the outputs of a TEG depend on both its structure (through matrices A, B and C) and the applied input (U). In this paper, only input and output events are considered observable, meaning that internal events cannot be measured. Equation (3) will therefore be essential in learning the behavior of these systems.

Definition 4: A TEG is in *in phase* [10] if $\forall i \in \{0, ..., n_y - 1\}$, $\forall j \in \{0, ..., n_u - 1\}$, $H_{ij} \neq \varepsilon \Rightarrow (H_{ij} \succeq \gamma^0\delta^0)$. This means that the output transitions can only be fired once the input transitions preceding them are fired.

Example 5: The transfer matrix of the TEG in Fig. 1 is

$$H = \begin{pmatrix} \gamma^0\delta^1(\gamma^2\delta^1)^* & \cdot \\ \gamma^0\delta^1 \oplus (\gamma^1\delta^3)(\gamma^2\delta^1)^* & \gamma^0\delta^2 \end{pmatrix}. \tag{4}$$

This TEG is in phase. In the following, only TEGs that are in phase will be considered.

III. Failures in Timed Event Graphs

This section introduces the essential notions regarding failures in TEGs based on the algebraic and structural properties presented in the previous section.

A TEG is considered to be failing if it produces series or *trajectories* in X, U or Y that are not part of its expected behavior as described by the state representation (2). The unexpected trajectories describe *abnormal runs* caused by *failures* in the TEG. These failures are defined as changes in

the parameters related to the places of the TEG, and will be divided in two kinds: *time failures* and *resource failures*.

Definition 5: A *time failure* [10] held by a place $p \in \mathcal{P}$ whose normal duration is $d = D(p)$ is a relative delay $\theta \in \mathbb{N}^+$ so that the real duration associated with p is $d+\theta \in \mathbb{N}^+$.

In practice, examples of a time failure include the deterioration of a machine resulting in longer processing durations than normal. The definition of a *resource failure* naturally derives from that of a time failure.

Definition 6: A *resource failure* held by a place $p \in \mathcal{P}$ whose normal initial marking is $n = \mathcal{M}_0(p) \geq 1$ is a relative shortfall $\phi \in \{1, ..., n\}$ so that the real initial marking of p is $n - \phi \in \mathbb{N}$.

Intuitively speaking, resource failures constitute removals of the tokens initially present in a TEG. In practice, such a failure may represent the deterioration of a resource, like a machine being able to process fewer products simultaneously than normal. Only resource failures which do not block the system will be considered (so every run of the system produces an output).

The abnormal runs caused by failures can only be detected if they generate observations that differ from the observations of normal runs. The max-plus linear systems in this paper are partially observable, so observations include all firings of input and output transitions (U and Y respectively), but not the firings of internal transitions (X).

Definition 7: A failure in a system with transfer matrix H is *detectable* [10] if it leads to the production of a real output Y that is different from the normal output HU for the same input.

Failures that do not generate abnormal observations are undetectable and will not be covered in the following.

The behavior of a max-plus linear system of normal transfer matrix H which contains failures may be described with a new matrix $H' \succeq H$, meaning that failures can generate abnormal runs that are *delayed* with respect to normal runs with the same input. This was proved for time failures in [10], the next proposition extends the proof to all failures considered in this paper.

Proposition 1: Let H be the normal transfer matrix of a system. If the system produces an abnormal run with input U and output Y generated by sets of time and resource failures, then there exists a matrix $H' \succeq H$ such that $Y = H'U$.

Proof: The proposition has been proved for time failures in [10]. Regarding resource failures, let the normal system be modeled as a TEG $\mathcal{G} = (\mathcal{P}, \mathcal{T}, \mathcal{A}, M_0, D)$ with transfer matrix $H = CA^*B$ (3). A set of resource failures results in a new TEG $\mathcal{G}' = (\mathcal{P}, \mathcal{T}, \mathcal{A}, M'_0, D)$ by altering the initial marking of the places containing the failures. Let us denote this set of places $\mathcal{P}_f$, then $\forall p_i \in \mathcal{P}_f, \mathcal{M}'_0(p_i) = \mathcal{M}_0(p_i) - \phi_i$ and $\forall p \in \mathcal{P} \setminus \mathcal{P}_f, \mathcal{M}'_0(p) = \mathcal{M}_0(p)$. Let $H' = C'(A')^*B'$ be the matrix associated with $\mathcal{G}'$, then $Y = H'U$ in the abnormal run.

By construction, every place p_i in $\mathcal{G}$ is associated with a monomial $m_i = \gamma^{n_i}\delta^{t_i}$ in either A, B or C. This monomial becomes $m'_i = \gamma^{n_i-\phi_i}\delta^{t_i}$ for a place containing a failure in $\mathcal{G}'$. Since $\gamma^{n_i}\delta^{t_i} \oplus \gamma^{n_i-\phi_i}\delta^{t_i} = \gamma^{\min(n_i,n_i-\phi_i)}\delta^{t_i} = \gamma^{n_i-\phi_i}\delta^{t_i}$ (1), then $m'_i \succeq m_i$. If m'_i is present in matrix A', then $A' \succeq A$ and $(A')^* \succeq A^*$. The same is true if m'_i is in C' or B' ($C' \succeq C$ and $B' \succeq B$). Therefore, $H' = C'(A')^*B' \succeq H = CA^*B$.

Finally, let us define a new TEG $\mathcal{G}'' = (\mathcal{P}, \mathcal{T}, \mathcal{A}, M'_0, D')$ resulting from a set of both time and resource failures on the original normal TEG $\mathcal{G}$, with an associated matrix H''. When different places contain different types of failures, $H'' \succeq H$ follows from the previous proof. This leaves the scenario of different types of failure occurring in the same places. In this case, the monomials associated with those places become $m''_i = \gamma^{n_i-\phi_i}\delta^{t_i+\theta_i}$. We have $m''_i \oplus m'_i = \gamma^{n_i-\phi_i}\delta^{t_i+\theta_i} \oplus \gamma^{n_i-\phi_i}\delta^{t_i} = \gamma^{n_i-\phi_i}\delta^{\max(t_i,t_i+\theta_i)} = \gamma^{n_i-\phi_i}\delta^{t_i+\theta_i}$ so $m''_i \succeq m'_i$. Since $m'_i \succeq m_i$ and the relation $\succeq$ is transitive, then $m''_i \succeq m_i$. Following the logic of the precedent paragraph, this results in $H'' \succeq H$ for TEG $\mathcal{G}''$ as well. ■

IV. Learning max-plus linear systems from logs

The purpose of this paper is to propose a method for estimating the transfer matrix H and the *delayed behavior matrices* (caused by failures, previously denoted H' and H'') of a system whose dynamics are initially unknown. This section describes how the available information on the system is presented through a log and how the proposed method takes advantage of this log to achieve its goals.

A. Assumptions regarding the logs

The proposed method uses data presented as different runs (i.e. an input vector U and its respective output vector Y) which together constitute a log. A log is assumed to contain at least one normal run of the system (produced during normal behavior). The log may contain delayed runs, generated during abnormal behavior caused by either the same set of failures or by different sets of failures for different runs, potentially mixing both kinds (see Definitions 5 and 6). The runs are finite and independent of each other.

For instance, let us consider a group of machines that are restarted every morning and shut down every night, working at different failing stages. For each machine, every day would be considered a different run, and all the runs of all the machines can be combined in a single log.

B. Describing logs in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$

In practice, the runs considered in this paper contain couples (v, d) where v is the name of the event and d is its date of occurrence (see Example 6 below). Each run details an input that has been applied to the system and its respective output (so v can be an input or an output event). A log contains N runs numbered from 0 to N-1.

Example 6: The following log is composed of two runs, one for each line, produced by a system modeled by the TEG in Fig. 1. In the first run, each transition is fired twice. In the second run, u_0, y_0 and y_1 are fired once and u_1 twice.

$$(\mathrm{u}_1, 1)(\mathrm{u}_1, 1)(\mathrm{u}_0, 2)(\mathrm{u}_0, 2)(\mathrm{y}_0, 3)(\mathrm{y}_0, 3)(\mathrm{y}_1, 3)(\mathrm{y}_1, 5)$$
$$(\mathrm{u}_0, 0)(\mathrm{u}_1, 0)(\mathrm{u}_1, 0)(\mathrm{y}_0, 1)(\mathrm{y}_1, 2)$$

Obtaining trajectories in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ from runs is straightforward. A log containing N runs of the system can be

described as two N-tuples: $\mathbf{U} = (U_0, ..., U_{N-1})$ for the input vectors and $\mathbf{Y} = (Y_0, ..., Y_{N-1})$ for their respective output vectors. Note that input vector U_i of run i in the log is not to be confused with transition u_i in the TEG.

In each vector, the first occurrence of an event will be translated as the first monomial of its respective series, $\gamma^0\delta^t$, where t is its date. Subsequent occurrences follow the same rule with an ascending exponent of γ. All series start with γ^0 (and not γ^n with $n > 0$) because all runs are considered to be independent and the system is in phase (so the outputs are not ahead of the inputs and the numbering of their events also starts at zero) (see Definition 4). Finally, the last monomial of a series describing n occurrences of an event is $\gamma^n\delta^{+\infty}$, meaning that occurrence n never happens.

Example 7: The log in Example 6 contains two runs numbered 0 and 1. It is translated as $\mathbf{U} = (U_0, U_1)$ and $\mathbf{Y} = (Y_0, Y_1)$. Each vector U_i (resp. Y_i) has two components containing series describing the firings of u_0 and u_1 (resp. y_0 and y_1). The vectors in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$ from this log are:

$$U_0 = \begin{pmatrix} \gamma^0\delta^2 \oplus \gamma^2\delta^{+\infty} \\ \gamma^0\delta^1 \oplus \gamma^2\delta^{+\infty} \end{pmatrix}, Y_0 = \begin{pmatrix} \gamma^0\delta^3 \oplus \gamma^2\delta^{+\infty} \\ \gamma^0\delta^3 \oplus \gamma^1\delta^5 \oplus \gamma^2\delta^{+\infty} \end{pmatrix},$$

for the first line (run 0) and, for the second line (run 1):

$$U_1 = \begin{pmatrix} \gamma^0\delta^0 \oplus \gamma^1\delta^{+\infty} \\ \gamma^0\delta^0 \oplus \gamma^2\delta^{+\infty} \end{pmatrix}, \; Y_1 = \begin{pmatrix} \gamma^0\delta^1 \oplus \gamma^1\delta^{+\infty} \\ \gamma^0\delta^2 \oplus \gamma^1\delta^{+\infty} \end{pmatrix}.$$

C. Estimating transfer matrices from a log

Once the information in the available log has been translated in terms of $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$, it is possible to use it explain the dynamics of the underlying system. Let us consider a system with n_u inputs, n_y outputs, and unknown normal transfer matrix H. As previously stated, a log containing N runs of this system is expressed as $\mathbf{U} = (U_0, ..., U_{N-1})$ and $\mathbf{Y} = (Y_0, ..., Y_{N-1})$. Evidently, if all runs are normal, $Y_i = HU_i$ for every run i in the log. According to the properties of residuation presented in section II-C and the definition of right division presented in Example 3, the residual $Y_i \not/ U_i$ constitutes an upper bound of the unknown transfer matrix: $Y_i \not/ U_i \succeq H \; \forall i \in \{0, ..., N-1\}$. The next proposition naturally follows.

Proposition 2: Let us consider a log containing N runs of a system with both normal and delayed runs. The infimum of the right residuals of each output vector Y_i over its respective input vector U_i constitutes an upper bound of the normal transfer matrix of the system. Formally,

$$\bigwedge_{i=0}^{N-1} (Y_i \not/ U_i) \succeq H. \tag{5}$$

Proof: As mentioned, for a normal run i, $Y_i = HU_i$, then $Y_i \not/ U_i = (HU_i) \not/ U_i \succeq H$. For a delayed run j, there exists H' such that $Y_j = H'U_j$ (Proposition 1), then $Y_j \not/ U_j = (H'U_j) \not/ U_j \succeq H'$ and since $H' \succeq H$ and the order relation is transitive, then $Y_j \not/ U_j \succeq H$. Finally, the two previous order relations regarding H imply that for any two given runs j and i, $(Y_i \not/ U_i) \wedge H = H$ and $(Y_j \not/ U_j) \wedge H = H$; since the operator $\wedge$ is associative and commutative, $(Y_i \not/ U_i) \wedge (Y_j \not/ U_j) \wedge H = H \Leftrightarrow (Y_i \not/ U_i) \wedge (Y_j \not/ U_j) \succeq H$. ■

It is therefore possible to summarize all the information conveyed in the log in a single upper bound of H. The better the quality of the runs (see Section IV-F), the closer this upper bound is to H. This will be exploited by the method presented in the following sections.

D. General objectives

There are two main goals for the method introduced in this paper. Starting from a log described as N-tuples $\mathbf{U}$ and $\mathbf{Y}$, the first goal is to estimate the transfer matrix describing the normal behavior of the system, H. The estimated transfer matrix will be denoted $\tilde{H}_0$. The second goal is to identify delayed behavior matrices to *explain* all the runs in the log, that is, to obtain a set of matrices $\{\tilde{H}_1, ..., \tilde{H}_F\}$ such that $\forall i, \exists k \in \{0, ..., F\}$ s.t. $Y_i = H_k U_i$.

In order to achieve these goals, the proposed method is divided in two steps. The first step is to compute a set of matrices $\mathcal{S} = \{M_0, ..., M_F\}$ to explain all the runs in the log. Algorithm 1 does this by returning matrices whose entries are finite or *truncated* series. The final step is to obtain the estimated transfer matrix $\tilde{H}_0$ and the delayed behavior matrices $\{\tilde{H}_1, ..., \tilde{H}_F\}$ whose entries are infinite series by extending the finite series in the matrices in $\mathcal{S}$ (see Section IV-F).

E. Computing the finite behavior matrices

Algorithm 1 takes advantage of (5) by using the left term of the inequality to propose a finite estimation of the transfer matrix and not only an upper bound, under the assumption that enough information on the dynamics of the system is conveyed in the log:

$$M_0 = \bigwedge_{i=0}^{N-1} (Y_i \not/ U_i). \tag{6}$$

This algorithm also aims at identifying the behavior of the system for the delayed runs produced by detectable failures. By Definition 7, for such failures, $Y \neq HU$. This means that if the estimated matrix M_0 is accurate enough, it is possible to discern delayed behavior from normal behavior by calculating M_0U for a given run and comparing it to Y. An estimation of the delayed behavior matrix explaining abnormal runs can also be calculated via (6) using the delayed subset of runs. Furthermore, if more than one set of failures produced the runs in the log, a single estimation may not explain them all; in order to identify different behaviors, $Y_i = M_k U_i$ must be checked before including the information conveyed by run i in the estimation M_k. When done recursively until all the runs are explained, one obtains a set containing F delayed behavior matrices $M_1, ..., M_F$.

The inputs of Algorithm 1 are $\mathbf{U}$ and $\mathbf{Y}$. Its output is the set $\mathcal{S} = \{M_0, ..., M_F\}$ containing finite matrices describing the behaviors of the system, with M_0 estimating normal behavior. The algorithm starts by initializing the necessary variables. $\mathcal{I}_e$ (resp. $\mathcal{I}_{ne}$) is the set containing the indices of the runs explained (resp. not explained) by the current matrix M. The algorithm will perform until the set of unexplained run indices is empty (loop starting in line 5).

Algorithm 1

Input: $\mathbf{U}, \mathbf{Y}$
Output: $\mathcal{S}$

1: Let $\mathbf{U} = (U_0, ..., U_{N-1})$ ▷ N-tuple of input vectors
2: Let $\mathbf{Y} = (Y_0, ..., Y_{N-1})$ ▷ N-tuple of output vectors
3: $\mathcal{S} \leftarrow \emptyset$ ▷ Set of behavior matrices
4: $\mathcal{I}_{ne} \leftarrow \{0, ..., N-1\}$ ▷ Indices of unexplained runs
5: **while** $\mathcal{I}_{ne} \neq \emptyset$ **do**
6: $M \leftarrow (\top)_{i,j}$ ▷ Behavior matrix
7: $\mathcal{I}_e \leftarrow \emptyset$ ▷ Indices of explained runs
8: **for** $i \in \mathcal{I}_{ne}$ **do**
9: $M' \leftarrow M \wedge (Y_i \not\backslash U_i)$
10: **if** $M'U_i = Y_i$ **then**
11: $M \leftarrow M'$
12: $\mathcal{I}_e \leftarrow \mathcal{I}_e \cup \{i\}$
13: $\mathcal{I}_{ne} \leftarrow \mathcal{I}_{ne} \setminus \{i\}$
14: **end if**
15: **end for**
16: **for** $i \in \mathcal{I}_e$ **do**
17: **if** $MU_i \neq Y_i$ **then**
18: $\mathcal{I}_{ne} \leftarrow \mathcal{I}_{ne} \cup \{i\}$
19: **end if**
20: **end for**
21: $\mathcal{S} \leftarrow \mathcal{S} \cup \{M\}$
22: **end while**
23: **return** $\mathcal{S}$

We go on to perform the computations explained in the previous paragraphs. Line 10 checks whether the current matrix explains the current run in order to update the matrix; the loop on the explained runs (line 16) updates this set if there are runs contained in it that are no longer explained. Finally, the finite estimated matrix is added to the resulting set (line 21) and a new iteration of the main loop starts.

As stated, since all the runs are finite, the entries in the matrices in $\mathcal{S}$ are not periodic even if the underlying system is a periodic system. The extra step needed to obtain a satisfactory estimation of the transfer matrix and the behavior matrices will be explained in section IV-F.

Regarding complexity, Algorithm 1 is in $O(N^2)$ as it has a loop iterating on the N runs of the log, which in turn contains two separate loops on the sets of indices. The complexity of the residuation, the infimum and the product between two series with n monomials each are each in $O(n^2 \log(n))$. Overall, $M \wedge (Y \not\backslash U)$ and MU are in $O(n_u^2 n_y^2 n^2 \log(n))$.

F. Obtaining infinite series

Every matrix $M_k \in \mathcal{S}$ produced by Algorithm 1 is composed of finite series only and is not yet the transfer matrix of an underlying TEG. In some cases, the residuation can result in series lesser than or non-comparable to $\gamma^0\delta^0$, in which case the respective entry will be assumed to be equal to ε because the systems are assumed to be in phase (see Definition 4). However, in most cases the behavior of a TEG is periodic (with a production rate) and so are the entries in the transfer matrix $\tilde{H}_k$, so to finally get an estimate of this matrix, every finite series $s = \bigoplus_{i=1}^{n} \gamma^{c_i}\delta^{t_i}$ from M_k must be extended as a periodic series $s_\infty = p \oplus qm^* = p \oplus q \oplus qm \oplus qm^2 \oplus \dots$ (p expresses the transient behavior, q is the pattern, and monomial m encapsulates this pattern [1]). To perform this extension, it is required that the finite series s embeds at least at q and qm ($s = p \oplus \bigoplus_{i=0}^{I} qm^i \oplus r$, with $I > 1$, and r a finite series) which depends on the quality of the log. For this, in some runs, input events must be both frequent enough and delayed enough to exploit the resources of the system at the intended production rate. Also, some runs must contain enough events for the system to express its periodic pattern at least twice. These properties are similar to those required in the method for estimating periodic series found in [5].

The algorithm used in this step analyzes the set of n monomials of series s and computes a series $s' = p' \oplus q' \oplus q'm'$ based on the strategy composed of a sequence of *stages* $[\mathrm{S}_0, \mathrm{S}_1, \dots]$. Stage $\mathrm{S}_i, i \geq 0$ is summarized as follows. We first consider that $p' = \bigoplus_{j=1}^{i} \gamma^{c_j}\delta^{t_j}$ (for $i = 0$, $p' = \varepsilon$). Then, we assume that the pattern q' relies on the $\frac{n-i}{2}$ monomials $\bigoplus_{j=i}^{i+\frac{n-i}{2}} \gamma^{c_j}\delta^{t_j}$ of s, that is checking whether s can be rewritten as $s' = p' \oplus q' \oplus q'm'$ (two occurrences of the pattern). If it is true, the search ends and S_i is a success. If not, and if the transient behavior of s_∞ is really p', then the pattern that we seek involves fewer monomials, so we check for a pattern involving $\frac{n-i}{2} - 1$ monomials, and so on[1]. Finally, if no pattern is found, S_{i+1} is applied.

Following this algorithm, the estimation of s_∞ leads to an expression written $p' \oplus q'm'^*$. By construction, if the quality of the log is satisfactory and the series is periodic, this algorithm necessarily ends as p' necessarily converges to p. The periodic part q' might not be the shortest but computing the canonical expression of $p \oplus q'm'^*$ [6],[2] will lead to the shortest equivalent q. If a pattern is not found, then the underlying series is assumed to be non-periodic and is flattened by removing its last monomial ($\gamma^n\delta^{+\infty}$).

Finally, the result of the two-step method detailed in Section IV-E and the previous paragraphs is the set of matrices $\{\tilde{H}_0, \tilde{H}_1, \dots, \tilde{H}_F\}$ with the following properties.

Property 1: $\forall i \in \{0, ..., N-1\}, \exists k \in \{0, ..., F\}$ s.t. $Y_i = \tilde{H}_k U_i$. This means that for every run there is at least one behavior matrix that explains it in the resulting set.

Property 2: $\forall k \in \{1, ..., F\}, \tilde{H}_k \succeq \tilde{H}_0$. The estimated delayed behavior matrices are greater than the estimated transfer matrix.

Property 3: $\forall k \in \{0, ..., F\}, \exists i \in \{0, ..., N-1\}$ s.t. $Y_i = \tilde{H}_k U_i$. Every matrix in the set of behavior matrices explains at least one run.

G. Example

The TEG in Fig. 1 serves as a model for a system divided in two tasks. The first task is activated by input event u_0 and

[1]In practice, before looking for such a pattern, the algorithm checks whether it has previously searched for a pattern involving kl events where l is the number of events involved in the $\frac{n-i}{2} - 1$ monomials. If this is the case, there is no solution with $\frac{n-i}{2} - 1$ monomials and the algorithm directly checks with $\frac{n-i}{2} - 2$ monomials, and so on.

includes places p_0, p_1 and p_3. This task has a production rate of two tokens per time unit, paced by the circuit between x_0 and x_1, and output y_0 signals its completion. The second task, modeled by places p_2, p_4 and p_6, is activated by input event u_1 and takes two time units. Its completion depends on the completion of the first task and results in output y_1. The initial token in p_4 translates the fact that the second task is one resource ahead of the first one, so it can be initiated by event u_1 before the completion of the first task.

Three sets of failures were simulated for this example. The first one is a time failure of one time unit on place p_6, the second one is composed of two resource failures of one token each on places p_3 and p_4, and the last one is the union of the three previous failures.

$N = 3000$ runs of the system were simulated, 750 for each set of failures and 750 for the normal system. These runs had a number of events ranging from 4 to 8 and three scenarios were chosen (with 250 runs for each): the events of u_1 all taking place after those of u_0, then vice versa, and then both inputs having events in no particular order.

The algorithm was implemented as part of the C++ library *MaxPlusDiag*, relying on *MinMaxGD* [2]. It returns a set containing five matrices. The first two follow (truncated at event 6 for conciseness):

$$M_0=\begin{pmatrix} \gamma^0\delta^1 \oplus \gamma^2\delta^2 \oplus \gamma^4\delta^3 \oplus \gamma^6\delta^{+\infty} & \gamma^6\delta^{+\infty} \\ \gamma^0\delta^1 \oplus \gamma^1\delta^3 \oplus \gamma^3\delta^4 \oplus \gamma^5\delta^5 \oplus \gamma^6\delta^{+\infty} & \gamma^0\delta^2 \oplus \gamma^6\delta^{+\infty} \end{pmatrix}$$

$$M_1=\begin{pmatrix} \gamma^0\delta^1 \oplus \gamma^2\delta^2 \oplus \gamma^4\delta^3 \oplus \gamma^6\delta^{+\infty} & \gamma^6\delta^{+\infty} \\ \gamma^0\delta^1 \oplus \gamma^1\delta^4 \oplus \gamma^3\delta^5 \oplus \gamma^5\delta^6 \oplus \gamma^6\delta^{+\infty} & \gamma^0\delta^2 \oplus \gamma^6\delta^{+\infty} \end{pmatrix}$$

M_0 is the least of all matrices and describes the normal behavior of the system. The remaining four matrices describe delayed behavior. The algorithm to obtain periodic series described in the previous section was applied and returned the following matrices:

$$\tilde{H}_0 = \begin{pmatrix} (\gamma^0\delta^1)(\gamma^2\delta^1)^* & . \\ \gamma^0\delta^1 \oplus (\gamma^1\delta^3)(\gamma^2\delta^1)^* & \gamma^0\delta^2 \end{pmatrix},$$

$$\tilde{H}_1 = \begin{pmatrix} (\gamma^0\delta^1)(\gamma^2\delta^1)^* & . \\ \gamma^0\delta^1 \oplus (\gamma^1\delta^4)(\gamma^2\delta^1)^* & \gamma^0\delta^2 \end{pmatrix},$$

$$\tilde{H}_2 = \begin{pmatrix} (\gamma^0\delta^1)(\gamma^2\delta^1)^* & . \\ \gamma^0\delta^1 \oplus (\gamma^1\delta^4)(\gamma^2\delta^1)^* & \gamma^0\delta^3 \end{pmatrix},$$

$$\tilde{H}_3 = \begin{pmatrix} (\gamma^0\delta^1)(\gamma^1\delta^1)^* & . \\ (\gamma^0\delta^3)(\gamma^1\delta^1)^* & \gamma^0\delta^2 \end{pmatrix},$$

$$\tilde{H}_4 = \begin{pmatrix} (\gamma^0\delta^1)(\gamma^1\delta^1)^* & . \\ (\gamma^0\delta^4)(\gamma^1\delta^1)^* & \gamma^0\delta^3 \end{pmatrix}.$$

The estimated transfer matrix $\tilde{H}_0$ is equal to the real normal transfer matrix of the system in (4) and, as mentioned, describes the fastest behavior: $\tilde{H}_k \succeq \tilde{H}_0 \ \forall k \in \{1, ..., 4\}$. The matrix describing the slowest behavior is $\tilde{H}_4$: $\tilde{H}_4 \succeq \tilde{H}_k \ \forall k \in \{0, ..., 3\}$. Matrices $\tilde{H}_2$ and $\tilde{H}_3$ are non-comparable among themselves, but both are greater than $\tilde{H}_1$.

In matrix $\tilde{H}_1$, only entry $(1, 0)$ is different from the respective entry in $\tilde{H}_0$ (starting on the second event). This suggests failures occurring on places found exclusively between input u_0 and output y_1, namely p_2 and p_4. In $\tilde{H}_2$, the same logic suggests failures upstream of y_1 (places p_2, p_4 and p_6). $\tilde{H}_3$ suggests failures downstream of u_0 (places p_0, ..., p_4, p_7). In $\tilde{H}_4$, all the entries that are different from ε show delayed behavior, so the failures could potentially be in any place. On that matter, note that the entry that is equal to ε for the normal matrix was kept as such for the other matrices. All of these results are coherent with the failures that were injected in the simulations.

After completion of the algorithm, we evaluated which runs were explained by which matrices: 763 out of the 3000 runs in the log are explained by $\tilde{H}_0$, 518 are explained by $\tilde{H}_1$, 763 by $\tilde{H}_2$, 768 by $\tilde{H}_3$ and 761 by $\tilde{H}_4$. Some runs are explained by several matrices, which was expected because outputs also depend on inputs (and not only on the behavior of the system) and some inputs may not exploit the dynamics of the system.

V. Conclusions

We have introduced the subject of learning max-plus linear systems from logs containing input and output event dates for normal and abnormal behavior of a system, and proposed a method for obtaining these models relying on the residuation of matrices in $\mathcal{M}_{in}^{ax}[\![\gamma, \delta]\!]$. The core of this method is synthesized in Algorithm 1, which results in a set of matrices explaining all the information conveyed in a log. These matrices are then extended to finally obtain the potentially periodic behavior of the underlying system.

Several future work possibilities arise from our method. Since the abnormal behavior of the system is generated by failures, one application would be obtaining models that can be used to perform model-based diagnosis [8], [10]. Other interesting prospects include covering more types of failures, like the suppression of arcs in the TEG, or directly performing a diagnosis process by classifying the runs.

References

[1] F. Baccelli, G. Cohen, G.J Olsder, and J.-P Quadrat. *Synchronization and linearity: an algebra for discrete event systems*. Wiley and sons, UK, 1992.

[2] B. Cottenceau, M. Lhommeau, L. Hardouin, and J.-L Boimond. Data processing tool for calculation in dioid. In *5th International Workshop on Discrete Event Systems*, 2000.

[3] B. De Schutter, T.J.J. van den Boom, and V. Verdult. State space identification of max-plus-linear discrete event systems from input-output data. In *Proceedings of the 41st IEEE Conference on Decision and Control*, volume 4, pages 4024–4029, 2002.

[4] S. Farahani, T.J.J. van den Boom, and B. De Schutter. Exact and approximate approaches to the identification of stochastic max-plus-linear systems. *Discrete Event Dynamic Systems*, 24:447–471, 2013.

[5] F. Gallot, J.-L. Boimond, and L. Hardouin. Identification of Linear Systems using MA and ARMA Model in Dioid. In *IFAC Conference on System Structure and Control*, pages 593–599, 1998.

[6] S. Gaubert. *Théorie des systèmes linéaires dans les dioïdes*. PhD thesis, École des Mines de Paris, 1992.

[7] J. Hook. Max-plus linear inverse problems: 2-norm regression and system identification of max-plus linear dynamical systems with gaussian noise. *Linear Algebra and its Applications*, 2019.

[8] E. Le Corronc, Y. Pencolé, A. Sahuguède, and C. Paya. Failure detection and localization for timed event graphs in (max,+)-algebra. *Discrete Event Dynamic Systems*, 31:513–552, 2021.

[9] T.J.J. van den Boom, B. De Schutter, and V. Verdult. Identification of stochastic max-plus-linear systems. In *Proceedings of the 2003 European Control Conference (ECC'03)*, Cambridge, UK, 2003.

[10] I. Velasquez, Y. Pencolé, and E. Le Corronc. Analysis and control of timed event graphs in (max,+) algebra for the active localization of time failures. *Discrete Event Dynamic Systems*, 34:53–93, 2024.