TECHNICAL TRANSACTIONS | CZASOPISMO TECHNICZNE
FUNDAMENTAL SCIENCES | NAUKI PODSTAWOWE
2-NP/2014

JERZY RASZKA AND LECH JAMROŻ[*]

MAX-PLUS LINEAR SYSTEM IN CONTROL OF DATA PROCESSING

LINIOWY SYSTEM MAX-PLUS W STEROWANIU PROCESAMI PRZETWARZANIA DANYCH

Abstract

The increasing complexity of information processing in distributed computer systems and microprocessors requires the use of time-saving devices and extended capacities of transmission channels. Processes in computers systems need effective processing time. This article describes an application of the theory of the Max Plus Linear System (MPLS) to controlling digital information processing and transmission time in information systems. System processes are described by an MPLS state equation and an MPLS output equation. The MPLS model makes use of formal mathematical methods of max-plus algebra which include maximization and addition operations in the domain of non-negative real numbers with the addition of minus infinity. The input data and the structure of the processes under consideration are represented by the Timed Event Graph (TEG) formalism constituting a special case of Timed Petri Nets. The suggested MPLS methods are useful for investigating selected properties of network models. They may be applied, among others, to evaluate performance criteria, cycle time, predictive control etc. This article presents the theoretical considerations used to determine the input signals controlling discrete processes, which are then illustrated with examples of numerical computations.

Keywords: max-plus algebra, Petri nets, data processing, discrete processes

Streszczenie

Zwiększająca się złożoność procesów przetwarzania informacji w rozproszonych systemach komputerowych i mikroprocesorowych wymaga oszczędnego wykorzystania czasu pracy urządzeń i zwiększonej pojemności kanałów transmisyjnych. Procesy w systemach komputerowych potrzebują efektywnego czasu przetwarzania W niniejszym opracowaniu przedstawiono zastosowanie teorii max-plus liniowego systemu (MPLS) w sterowaniu czasem przetwarzania informacji i czasem transmisji informacji cyfrowej w systemach informatycznych. System procesów opisany jest w przestrzeni MPLS równaniem stanu i równaniem wyjścia. Model MPLS jest oparty na formalnych matematycznych metodach max plus algebry, które są wyposażone w operacje maksymalizacji i dodawania w dziedzinie nieujemnych liczb rzeczywistych rozszerzonych o minus nieskończoność. Dane wejściowe i struktura rozważanych procesów są określone przez formalizm czasowych sieci zdarzeń jako szczególnego przypadku czasowych sieci Petri'ego. Zaproponowane metody MPLS są użyteczne w badania wybranych właściwości sieci. Między innymi mogą być one zastosowane do oceny wydajności, czasu cyklu, sterowania predykcyjnego, itp. W artykule zastosowano teoretyczne rozważania określające wejściowe sygnały sterujące procesem dyskretnym oraz przedstawiono przykładowy wyniki z numerycznych obliczeń.

Słowa kluczowe: max-plus algebra, sieć Petri'ego, przetwarzanie danych, procesy dyskretne

[*]Institute of Computer Science, Cracow University of Technology; jraszka@pk.edu.pl, ljamroz@pk.edu.pl

1. Introduction

Technological advances in the field of computer information processing require the use of more effective methods to analyze and model these processes. The Internet evolved from a simple store-and-send network into a more complex communications infrastructure. The security and flexibility of ICT connections require extra processing (e.g. filtering, encryption and prediction) as well as resources, all of which delay signal data transmission. Furthermore, the amount of information in distributed microprocessors and computer systems is growing rapidly (as exemplified by weather information systems or communication networks). The time of both digital information processing and its transmission is increasing (computer networks might soon reach out into the interplanetary space). Consequently, processes in computer systems require effective time control. Measurements of event occurrence times are generally not as susceptible to noise as those of continuous signals, namely variables such as the temperature, speed, pressure, etc. The suggested methods are used to control and minimize the delay of output results compared to selected values by varying the starts of processing tasks and moments of data entry at different points of the studied system.

Discrete systems, and particularly discrete-event dynamic systems, often appear in the context of parallel computing, manufacturing [6] or project management systems, railway [5] or telecommunication networks etc. Recent years have seen a quantitative growth of research on discrete systems that can be modelled as max-plus linear systems. Most of the earlier literature concerning this class of systems discussed modelling, performance and properties analysis rather than control [1, 6] and so did many other authors e.g. J. Bernd Heidergott, Geert Jan Olsder, Didier Dubois, Jean-Pierre Quadrat and Jacob van der Woude. However, articles have recently been published on the control of max plus systems. The authors in [7] designed a feedback controller to guarantee that the system evolves without violating time restrictions . In [10], authors describe a bus network as a non-stationary linear state model and determine eigenvalues as well as eigenvectors used to evaluate the cycle time. Most of existing publications about controlling general discrete systems are presented by B. De Schutter [11].

The work reported in this paper is but a part of a more general study aimed at evaluating and controlling computation on multiprocessor platforms, using some classical analysis and control methods from systems theory utilizing (max, +) algebra. Mikel Cordovilla [12] has worked on the real-time execution of dependent periodic task sets on multiprocessor platforms and I. Elmahi [4] has used max-plus algebra and Petri Nets for supply chain logistics.

Petri nets (PN) [8, 14] are very popular formal mathematical methods of analyzing and representing parallel and distributed computing in concurrent systems, and they draw much attention to modelling and verifying these types of systems. P systems, also referred to as membrane systems [9], are a class of parallel and distributed computing models [6]. The interest in linking P systems with the PN computation model has produced several important results in simulation and decidability issues. Some efforts have been made to simulate P systems with Petri nets and thus to verify many useful behavioral properties such as reachability, boundedness, liveness, terminating, etc.

With regard to manufacturing or chemical engineering processes, their behavior can often be adequately represented by a discrete event model accounting for the usually discrete sensor and the actual equipment for these processes. In addition, the behavior of these processes is often adequately described by a sequence of transitions between discrete process states. This contribution focuses on a particular class of such discrete event systems in which computational processes are synchronized and controlled. This system class has attracted significant interest in recent years because sequences of event times for such processes can be described by equations that are linear in one specific algebra, namely maxplus algebra [1]. The resulting equations are structurally equivalent to system descriptions from conventional control engineering such as transfer functions or state space models.

Hence, a system theory for these max–plus linear systems has been developed, and various concepts known well from control engineering have been adapted to this system class in the control design and diagnosis.

Timed Event Graphs (TEG's) are a subclass of timed Petri nets which can be used for modelling Discrete Event Dynamic Systems (DEDS) in which synchronization phenomena occur, such as manufacturing, multiprocessor and transportation systems in particular.

In this paper, we have introduced a deterministic Petri net model of a computational system that can be considered to be a discrete event system. Moreover, as such DEDS can be easily modelled with a subclass of Petri nets for evaluation purposes, we subsequently suggest a TEG approach to the model, analyze and control the computational process from this TEG model, and formulate a mathematical model based on max-plus algebra. In this algebra, the behavior of the DEDS can be easily described by a linear system. In the second part, we introduce a computational process model. The third part presents an overview of the max algebra theory and an analysis of a specific model. The last part contains simulation results.

2. Data Processing

Let us consider a data process that allows event-driven applications to take advantage of multiprocessors by running the code for event handlers in parallel. To achieve high performance, servers must overlap computation with the I/O. Programs typically achieve this overlap by using threads or events. Threaded programs usually process every request in a separate thread; while one thread block is waiting for the I/O, another thread can run. Event-based programs are structured as a collection of call-back functions which are called by the main loop when I/O events occur. Threads provide an intuitive programming model but require coordinating the access of different threads

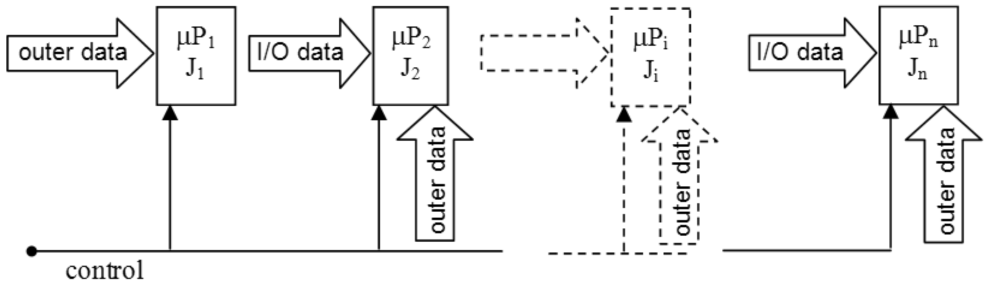

Fig. 1: The structure of the process.

to the shared state, even on a uniprocessor. Event-based programs execute call-backs sequentially so the programmer need not worry about concurrency control; however, event-based programs have so far been unable to make good use of multiprocessors. Much of the effort required to make existing event-driven programs take advantage of multiprocessors is in specifying which events can be handled in parallel.

This paper exemplifies the simple problem of designing the control of a system in which the cost is chosen so that it provides a trade-off between minimizing the delays of the end time of computational process operations (the real time to complete all the tasks in a cyclic computational process, times of final results of one cycle) and the periodicity of the desired output (the time desired or needed) to complete the process

Simple data processing consists of several tasks linked by the waiting for I/O data (Fig. 1). To illustrate our approach, let us consider a process that consists of some tasks (jobs): J_i which runs on microprocessors: $\mu\mathrm{P}_i$ for i=1..n. Each of these tasks is executed on a dedicated microprocessor. In this process, the digital information flows as input/output processing data and a control signal. Outer input data is processed as the first task on the $\mu\mathrm{P}_1$ and its output data has to be saved to memory while it waits to be processed. The other microprocessors operate in the same way, but their input data simultaneously constitutes the output result from the previous microprocessor and may need extra outer data.

The aim of this modelling exercise is to evaluate the command of the process according to pre-established criteria. For instance, to ensure continuous computation, we have to input data at a certain specified time and have enough memory to save input, output and temporary data. A good schedule will keep the costs to a minimum and optimize the size of the memory required.

3. Net Representation

Petri nets, – graph-oriented formalism – facilitate modelling and analyzing systems which feature such properties as concurrency and synchronization. The Petri net model of a dynamical system consists of two parts: the net structure and marking with tokens. The net structure is a weighed-bipartite directed graph representing the static part of the system. The tokens represent the distributed overall state of the structure. This separation allows one to reason on a net based model on two levels – structural and behavioral.

The net structure is built of two disjoint sets of objects - places and transitions - which are connected by arcs. In a graphic representation, places are drawn as circles, transitions are drawn as thin bars, and arcs as arrows. Places may contain tokens which are drawn as dots. The vector representing the number of tokens in every place is the state of the Petri net and is referred to as its marking. This marking can be changed by the firing of the transitions. Petri nets do not include any notion of time and are aimed at modelling only the logical behavior of systems. The introduction of a timing specification is essential if we want to use this model class to consider the performance problem.

More formally, timed Petri nets (TPN) are 5-tuples: TPN= $(\mathbf{P},\mathbf{T},\mathbf{F},\mathbf{M}_0,\tau)$, where $\mathbf{P}=(p_1,p_2,...,p_n)$, $|\mathbf{P}|\neq 0$; $\mathbf{T}=(t_1,t_2,...,t_m)$, $|\mathbf{T}|\neq 0$ is a finite, disjoint set of suitable places and transitions; $\mathbf{M}_0\mathbf{:}\ \mathbf{P}\rightarrow N$ is the initial marking function which defines the initial number of tokens for every place. (N={0,1,...}); τ : T$\rightarrow R$ is the firing time function; and

$\mathbf{F} \subset (\mathbf{P}\mathrm{x}\mathbf{T}) \cup (\mathbf{T}\mathrm{x}\mathbf{P})$ is the set of arcs.

Many different models of timed Petri nets are evolved from the fundamental definition. Its include models in which the parameter of time is also assigned to the places and edges ([15] and others). In this article, the type of DEDS is represented graphically using a Timed Event Graph (TEG), is a class of timed Petri nets in which all places have only one upstream and only one downstream transition and as also mentioned, time delays. The model proposed in chapter 2 has the three microprocessors and time parameters which are represented by τ in the basic definition of Petri nets.

Fig. 2 show the TEG in which time parameters have been moved to the places and practically are described as ti, - mathematically in the next chapters that will be expressed as t_i. This net has three inputs u_1, u_2, and u_3, and one output y. The firing time units u_1, u_2, and u_3 are the start times of jobs J_1, J_2, J_3, respectively. Finally, the end time of the process is represented by the firing time of transition y.

4. Max-Plus Linear System

In recent years, the concept of a max-plus linear system (MPLS) has been increasingly frequently used in the literature [11]. It is based on a mathematical formalism, namely max-plus algebra. The basic operations of max-plus algebra [1] are maximization and

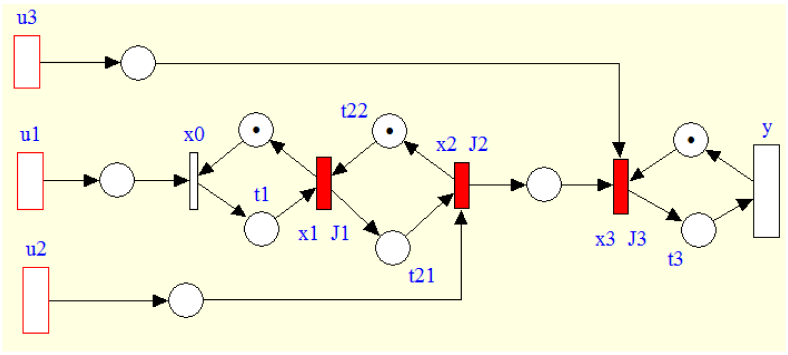

Fig. 2: Timed Event Graphs - a process model.

addition, which will be represented, respectively, by $\oplus$ and $\otimes$:$x \oplus y = \max(x, y)$ and $x \otimes y = x + y$ for

$$x, y \in R_{\varepsilon}\,, \quad R_{\varepsilon} \ =^{def} R \cup \{-\infty\}$$

The reason for using these symbols is that there is a remarkable analogy between $\oplus$ and conventional addition, and between $\otimes$ and conventional multiplication: many concepts and properties from linear algebra (such as the Cayley-Hamilton theorem, eigenvectors and eigenvalues, Cramer's rule,...) can be translated to max-plus algebra by replacing $+$ with $\oplus$ and $\times$ with $\otimes$. Hence we also call $\oplus$ the max-plus algebraic addition, and $\otimes$ the max-plus algebraic multiplication. Note, however, that a major difference between conventional algebra and max-plus algebra is that, in general, there are no inverse elements with respect to $\oplus$ in R_{ε}. The zero element for $\oplus$ is $\varepsilon =^{def} -\infty$ and we have $a \oplus \varepsilon = a = \varepsilon + a$ for all $a \in R_{\varepsilon}$. The structure $(R_{\varepsilon}, \oplus, \otimes)$ is referred to as max-plus algebra.

Let $r \in R$. The r^{th} max-plus algebraic power of $x \in R$ is denoted by $x^{\otimes r}$and corresponds to rx in conventional algebra. If $r \in R$, then $x^{\otimes 0} = 0$ and the inverse element of x w.r.t.$\otimes$ is $x^{\otimes -1} = -x$. There is no inverse element for ε since ε is absorbing for $\otimes$. If $r > 0$, then $\varepsilon^{\otimes r} = \varepsilon$, and if $r < 0$, then $\varepsilon^{\otimes r}$ is not defined. In this paper, we have $\varepsilon^{\otimes 0} = 0$ by definition.

The rules for the order of evaluation of max-plus algebraic operators correspond to those of conventional algebra. So the max-plus algebraic power has the highest priority, and max-plus algebraic multiplication has a higher priority than max-plus algebraic addition.

The basic max-plus algebraic operations are extended to matrices as follows. If $\mathbf{A}, \mathbf{B} \in R_{\varepsilon}^{m \times n}$ and $\mathbf{C} \in R_{\varepsilon}^{m \times p}$, then:

$$(\mathbf{A} \oplus \mathbf{B})_{ij} = a_{ij} \oplus b_{ij} = \max(a_{ij}, b_{ij})$$
$$(\mathbf{A} \otimes \mathbf{C})_{ij} = \bigoplus_{k=1}^{n} a_{ik} \otimes c_{kj} = \max_{k=1\ldots n} (a_{ik} \otimes c_{ki})$$

for all i, j. Note the analogy with the definitions of the matrix sum and the product in conventional linear algebra.

The matrix $\boldsymbol{\mathcal{E}}_{m \times n}$ is the $m \times n$ max-plus algebraic zero matrix: $(\boldsymbol{\mathcal{E}}_{m \times n})_{i,j} = \varepsilon$ for all i, j; and the matrix $\mathbf{E}_n$ is the $n \times n$ max-plus algebraic identity matrix: $(\mathbf{E}_n)_{i\,i} = 0$ for all i and $\mathbf{E}\,(\mathbf{E}_n)_{i\,j} = \varepsilon \quad i,\, j$ with $i \quad \neq \quad j$. If the size of the max-plus algebraic identity matrix or the max-plus algebraic zero matrix is not specified, it should be clear from the context. The max-plus algebraic matrix power of $\mathbf{A} \in R_{\varepsilon}^{n \times n}$ is defined as follows: $\mathbf{A}^{\otimes 0} = \mathbf{E}_n$ and $\mathbf{A}^{\otimes k} = \mathbf{A} \otimes \mathbf{A}^{\otimes (k-1)}$for k = 1,2, . . .

Discrete event systems with synchronization only and no concurrency can be modeled by a max-plus algebraic model. The state equation of a max-plus linear system has the following form [1]:

$$\mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1) \quad \oplus \quad \mathbf{B} \otimes \mathbf{u}(k) \tag{1}$$
$$\mathbf{y}(k) = \mathbf{C} \otimes \mathbf{x}(k) \tag{2}$$

with $\mathbf{A} \in R_{\varepsilon}^{n \times n}$, $\mathbf{B} \in R_{\varepsilon}^{n \times m}$, $\mathbf{C} \in R_{\varepsilon}^{l \times n}$ and

$\mathbf{x} \in R_{\varepsilon}^{m}$ $-$represents the state vector with the initial value $\mathbf{x}(0) = \mathbf{x}_0$,

$\mathbf{u} \in R_{\varepsilon}^{n}$ $-$the input vector,

$\mathbf{y} \in R_{\varepsilon}^{l}$ $-$the output vector,

where

m - the number of inputs,

$l-$the number of outputs.

5. Process model

5.1. Investigation

The purpose of this study is to show that the process satisfying the above assumptions can be modelled in max-plus algebra to determine the input vector $\mathbf{u}$(k) for known values of $\boldsymbol{y}$(k) and to evaluate the error between the actual and the desired output.

Let $\mathbf{u}(k) = [u_l,\, u_2,\, u_3]^T$ be the input vector, $\mathbf{x}(k) = [x_l\ ,\, x_2\ ,\, x_3]^T$ the state vector and $\mathbf{y}(k)$ the output vector coinciding with the model output $y(k)$. For each transition, x_i and u_i are associated with the indicators $x_i(k)$ and $u_i(k)$ respectively, which correspond to the steps of the k^{th} firing of transition x_i (resp. u_i) and $y(k)$ is obtained by analogy. The state system in the max plus algebra is as follows:

For example, the k^{th} firing of transition x_1(when J$_1$ starts) must wait for t_1 time units until the k^{th} input data u_1 for task J$_1$ is ready and the k^{th} input data u_2 is

$$\begin{aligned} x_1(k) &= t_1 \otimes u_1(k) \quad \oplus \quad u_2(k) \quad \oplus \quad t_{22} \otimes x_2(k-1) \quad \oplus \quad t_1 \otimes x_1(k-1) \\ x_2(k) &= t_{21} \otimes x_1(k) \\ x_3(k) &= x_2(k) \quad \oplus \quad u_3(k) \quad \oplus \quad y(k-1) \end{aligned}$$

provided. Thus, the linear equation of evolution inR_ε of this discrete event dynamic system is as follows:

$$\mathbf{x}(k) = \mathbf{A_0} \otimes \mathbf{x}(k) \quad \oplus \quad \mathbf{A_1} \otimes \mathbf{x}(k-1) \quad \oplus \quad \mathbf{B_0} \otimes \mathbf{u}(k) \quad \oplus \quad \mathbf{D}_0 \otimes y(k-1) \tag{3}$$

where

$$\mathbf{A}_0 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon \\ t_{21} & \varepsilon & \varepsilon \\ \varepsilon & e & \varepsilon \end{bmatrix}, \ \mathbf{A}_1 = \begin{bmatrix} t_1 & t_{22} & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, \ \mathbf{B}_0 = \begin{bmatrix} t_1 & e & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & e \end{bmatrix}, \ \mathbf{D}_0 = \begin{bmatrix} \varepsilon \\ \varepsilon \\ e \end{bmatrix}.$$

The solution of (3) is

$$\mathbf{x}(k) = \mathbf{A_0^*} \otimes \ (\mathbf{A_1} \otimes \mathbf{x}(k-1) \quad \oplus \quad \mathbf{B_0} \otimes \mathbf{u}(k) \quad \oplus \quad \mathbf{D_0} \otimes \mathbf{y}(k-1))$$

where

$$\mathbf{A}_0^* = \mathbf{E} \oplus \mathbf{A}_0^{\oplus} \mathbf{A}_0^2 \oplus \mathbf{A}_0^3 ...$$

$$\mathbf{A}_0^* = \begin{bmatrix} e & \varepsilon & \varepsilon \\ t_{21} & e & \varepsilon \\ t_{21} & e & e \end{bmatrix}, \qquad \mathbf{A} = \mathbf{A}_0^* \otimes \mathbf{A}_1 = \begin{bmatrix} \varepsilon & t_{22} & \varepsilon \\ \varepsilon & t_{22} \otimes t_{21} & \varepsilon \\ \varepsilon & t_{22} \otimes t_{21} & \varepsilon \end{bmatrix}.$$

$$\mathbf{B} = \mathbf{A}_0^* \otimes \mathbf{B}_0 = \begin{bmatrix} t_1 & e & \varepsilon \\ t_1 \otimes t_{21} & t_{21} & \varepsilon \\ t_1 \otimes t_{21} & t_{21} & e \end{bmatrix}, \qquad \mathbf{D} = \mathbf{A}_0^* \otimes \mathbf{D}_0 = \begin{bmatrix} \varepsilon \\ \varepsilon \\ e \end{bmatrix}.$$

Then, the model turns into:

$$\mathbf{x}(k) = \ \mathbf{A} \otimes \mathbf{x}(k-1) \quad \oplus \quad \mathbf{B} \otimes \mathbf{u}(k) \quad \oplus \quad \mathbf{D} \otimes y(k-1)) \tag{4}$$

Recurrence gives us the expression:

$$\mathbf{x}(k) = \mathbf{A}^{k-1} \otimes \ \mathbf{x}(1) \quad \oplus \quad \sum_{i=2}^{k} \mathbf{A}^{k-i} \otimes \mathbf{B} \otimes \mathbf{u}(i) \quad \oplus \quad \sum_{i=3}^{k-1} \mathbf{A}^{k-i-1} \otimes \mathbf{D} \otimes y(i) \tag{5}$$

To determine the command of the process, first we have to define the whole order-l model which describes its global behavior:

$$\begin{cases} \mathbf{x}(k) = \mathbf{A} \otimes \mathbf{x}(k-1) \ \oplus \ \ \mathbf{B} \otimes \mathbf{u}(k) \ \oplus \mathbf{D} \otimes y(k-1)\,) \\ \mathbf{y}(k) = \mathbf{C} \otimes \mathbf{x}(k) \\ \mathbf{u}(1) = \begin{bmatrix} u_1(1) & u_2(1) & u_3(1) \end{bmatrix}^{\mathbf{T}} \\ \mathbf{x}(1) = \begin{bmatrix} x_1(1) & x_2(1) & x_3(1) \end{bmatrix}^{\mathbf{T}} \end{cases} \tag{6}$$

Where $\mathbf{C} = \begin{bmatrix} \varepsilon & \varepsilon & t_3 \end{bmatrix}$, $\mathbf{u}(1)$ and $\mathbf{x}(1)$ are the initial conditions that we are going to determine below.

5.2. Initial conditions

To comply with the previous assumptions, we determine the initial values of $\mathbf{u}(1)$ and $\mathbf{x}(1)$ according to the end process $y(1)$.

$$y\,(1) = \ t_1 \otimes t_{21} \otimes t_3 \otimes u_1\,(1) = t_{21} \otimes t_3 \otimes u_2\,(1) = t_3 \otimes u_3\,(1)$$

Then, the initial control is:

$$\begin{bmatrix} u_1(1) \\ u_2(1) \\ u_3(1) \end{bmatrix} = \begin{bmatrix} y(1)\,\phi\,(t_1 \otimes t_{21} \otimes t_3) \\ y(1)\,\phi\,(t_{21} \otimes t_3) \\ y(1)\,\phi\,t_3 \end{bmatrix} \tag{7}$$

where "ϕ" represents the conventional subtraction.

Consequently, the initial value of the state vector is

$$\begin{bmatrix} x_1(1) \\ x_2(1) \\ x_3(1) \end{bmatrix} = \begin{bmatrix} t_1 \otimes u_1(1) \\ t_{21} \otimes u_2(1) \\ u_3(1) \end{bmatrix}. \tag{8}$$

These two initial vectors mean that if task J_1 begins e.g. at $t = 0$, t_1 time units later the data for task J_2 (u_2) must be prepared and that task can begin. This ensures a good start-up without the signal of the model being delayed.

5.3. Procedure

As indicated before, the network operates under a schedule defined for the final result; this schedule is used to find the suitable inputs of the model. We formulate the model output more explicitly as:

$$\mathbf{y}(k) = \mathbf{C}\otimes\mathbf{A}^{k-1}\otimes\mathbf{x}(1) \oplus \sum_{i=2}^{k}\mathbf{C}\otimes\mathbf{A}^{k-i}\otimes B\otimes\mathbf{u}(i) \oplus \sum_{i=3}^{k-1}\mathbf{C}\otimes\mathbf{A}^{k-i-1}\otimes\mathbf{D}\otimes y(i) \tag{9}$$

or

$$\mathbf{y}(k) \geqslant \max\{\,\mathbf{C}\otimes\mathbf{A}^{k-1}\otimes\,\mathbf{x}(1)\,,\ \sum_{i=2}^{k}\mathbf{C}\otimes\mathbf{A}^{k-i}\otimes\mathbf{B}\otimes\mathbf{u}(i),\ \sum_{i=3}^{k-1}\mathbf{C}\otimes\mathbf{A}^{k-i-1}\otimes \mathbf{D}\otimes y(i)\,\}$$

which is equivalent to:

$$\mathbf{y}(k) \geqslant \mathbf{C}\otimes\mathbf{A}^{k-1}\otimes\,\mathbf{x}(1) \tag{10}$$

$$\mathbf{y}(k) \geqslant \sum_{i=2}^{k}\mathbf{C}\otimes\mathbf{A}^{k-i}\otimes\mathbf{B}\otimes\mathbf{u}(i) \tag{11}$$

$$\mathbf{y}(k) \geqslant \sum_{i=3}^{k-1}\mathbf{C}\otimes\mathbf{A}^{k-i-1}\otimes\mathbf{D}\otimes\mathbf{y}(i) \tag{12}$$

We are more interested in the second equation (11) which is transformed into the form below:

$$\mathbf{y}(k) = \sum_{i=2}^{k}\mathbf{C}\otimes\mathbf{A}^{k-1}\otimes\mathbf{B}\otimes\mathbf{u}(i) \tag{13}$$

It is now straightforward that from (13), we can formulate the command $\mathbf{u}(k)$ for the process if the values of the output $\mathbf{y}(k)$ are known. For all of the rest $\mathbf{y}(k)$, there will be the need to find the final result.

For k= 2, 3, 4, ...

$$\begin{aligned}
&\mathbf{y}(2) = \mathbf{C}\otimes\mathbf{B}\otimes\mathbf{u}(2)\\
&\mathbf{y}(3) = \mathbf{C}\otimes\mathbf{A}^{1}\otimes\mathbf{B}\otimes\mathbf{u}(2) \quad\oplus\quad \mathbf{C}\otimes\mathbf{B}\otimes\mathbf{u}(3)\\
&\mathbf{y}(4) = \mathbf{C}\otimes\mathbf{A}^{2}\otimes\mathbf{B}\otimes\mathbf{u}(2) \quad\oplus\quad \mathbf{C}\otimes\mathbf{A}^{1}\otimes\mathbf{B}\otimes\mathbf{u}(3) \quad\oplus\quad \mathbf{C}\otimes\mathbf{B}\otimes\mathbf{u}(4)\\
&\cdots
\end{aligned} \tag{14}$$

Since our aim is to compute $\mathbf{u}(k)$ for the specified $\mathbf{y}(k)$, we have to solve the following equation:

$$\mathbf{y}(k) = \mathbf{C}\otimes\mathbf{B}\otimes\mathbf{u}(k) \tag{15}$$

For example, to calculate $\mathbf{u}(2)$ which is the solution of $\mathbf{y}(2) \geqslant \mathbf{C}\otimes\mathbf{B}\otimes\mathbf{u}(2)$, we solve its equation form (14) and keep its smallest solution to be sure that it is also verified in the equation. Note that we proceed by simplifying terms such as the ones in the expression of $\mathbf{y}(k)$. Indeed, these terms constitute the first condition for the desired output $y(k)$. More explicitly, if we consider that all expressions $\mathbf{y}(k)$ are equal to $\mathbf{C} \otimes \ \mathbf{B} \otimes \mathbf{u}(k)$, then :

$$\mathbf{C} \otimes \mathbf{A}^2 \otimes \mathbf{B} \otimes \mathbf{u}(2) \ \oplus \ ... \ \oplus \ \mathbf{C} \otimes \mathbf{A}^1 \otimes \mathbf{B} \otimes \mathbf{u}(k-1) \ \leqslant \ \mathbf{C} \otimes \mathbf{B} \otimes \mathbf{u}(k)$$

Which means that we must have

$$y\,(k) \geqslant \ t_1 \otimes t_{21} \otimes t_3 \otimes u_1\,(k-1) \ \oplus \ t_{21} \otimes t_3 \otimes u_2\,(k-1) \ \oplus \ t_3 \otimes u_3\,(k-1)$$

5.4. Cyclic processing

We assume that the output of the process, i.e. the final results, will be produced at regular intervals T_C. We shall later see how the network reacts depending on various values of T_C.

Let $y(k) \geqslant T_C^k \otimes y(0)$, where T_C is the periodicity of the desired output. We apply this condition in our computations in the following form:

$$T_C \geqslant \ \max(\ t_1 + t_{21} + t_3 + u_1\,(k-1)\,,\ t_{21} + t_3 + u_2\,(k-1)\ ,\ t_3 + u_3\,(k-1)\,)/k$$

We solve equation (13) and determine the control vector as follows:

$$u_j(k) = y(k)\,\phi\,(\mathbf{C} \otimes \mathbf{B})_{i,j}; \qquad k = 2,3,...; \quad i = 1 \ \text{ and } \ j = 1,2,3\,.$$

Where:

$$\mathbf{C} \otimes \mathbf{B} = \begin{bmatrix} \varepsilon & \varepsilon & t_3 \end{bmatrix} \otimes \begin{bmatrix} t_1 & e & \varepsilon \\ t_1 \otimes t_{21} & t_{21} & \varepsilon \\ t_1 \otimes t_{21} & t_{21} & e \end{bmatrix} = \begin{bmatrix} t_1 \otimes t_{21} \otimes t_3 & t_{21} \otimes t_3 & t_3 \end{bmatrix}$$

More explicitly, for every $k > 1$, the general solutions are:

$$\begin{aligned} u_1(k) &= y(k)\,\phi\,(\mathbf{C} \otimes \mathbf{B})_{1,1} \\ u_2(k) &= y(k)\,\phi\,(\mathbf{C} \otimes \mathbf{B})_{1,2} \\ u_3(k) &= y(k)\,\phi\,(\mathbf{C} \otimes \mathbf{B})_{1,3} \end{aligned} \tag{16}$$

These equations determine appropriate control of the modelled process. On the other hand, (10) and (12) contain two constraints of the desired outputs of the system. At the same time, T_C is periodic in relation to these outputs, $y(k) \geqslant T_C^{k-1} \otimes y(1) = T_C^k$. On the basis of (10) and this assumption, we conclude that:

$$y\,(k) = T_C^k \geqslant \mathbf{C} \otimes \mathbf{A}^{k-1} \otimes \mathbf{x}(1) \quad \text{or} \quad T_C^k \geqslant t_3 \otimes (t_1 \otimes t_{21})^{k-1} \otimes x_2(1)$$

In practice, $T_C \geqslant \ (\ t_3 + (t_1 + t_{21})(k-1) + x_2(1) \ )/k$ means that the periodicity must be superior at a certain value in order to have good control.

The subsequent constraint of T_C results from (14):

$y(k) = T_C^k \geqslant \sum\limits_{i=3}^{k-1} \mathbf{C} \otimes \mathbf{A}^{k-i-1} \otimes \mathbf{D} \otimes y(i)$, this constraint is continuously verified since the product $\mathbf{C} \otimes \mathbf{A}^{k-i-1} \otimes \mathbf{D}$ is null.

To conclude this section, we have introduced the following steps:

1- Determine state equations in max-plus algebra as (6).

2- Calculate the global recurrence equation of the linear evolution of the system.

3- Determine initial conditions (7,8).

4- Calculate constraints of the desired model output.

5- Calculate the control vector (16).

6. Simulation and Results

In order to simulate the process model (Fig. 2) for $t_1 = 3$, $t_{21} = 5$, $t_{22} = 5$ and $t_3 = 2$, we apply a fixed interval T_C to the desired outputs, which are: $y'\,(k) = y'\,(k-1) \otimes T_C$; $\quad k = 2, 3, \ldots$. Using (16), we can compute vectors $u(k)$ and consider the initial values. Figures 3 - 4 show the time evolution of the interpolated values of activity counts - particularly the desired $y'(t)$ and real $y(t)$ outputs represented by the signs ▼ and ▲, respectively.

All example results are obtained for various values of T_C. In the first examples, the periodicity of computations equals 10 and 15 (Fig. 3 and Fig. 4, respectively). The T_C value is large enough to contain all tasks of the process and no error occurs between the desired and actual outputs.

7. Conclusion

Engineers who build discrete event systems have to confront dynamic problems as a matter of fact. In particular, even though dynamical systems are well understood based on classical methods, they have had little mathematical support for this. This article suggests and introduces a max-plus linear system: a methodology applied to modeling and simulating discrete event processes. The control of process tasks in a simple multi-processor computational system is considered as an example. This, however, is only a small part of the studies carried out, which require additional testing and even broader-ranging experience, especially in terms of practical applications.

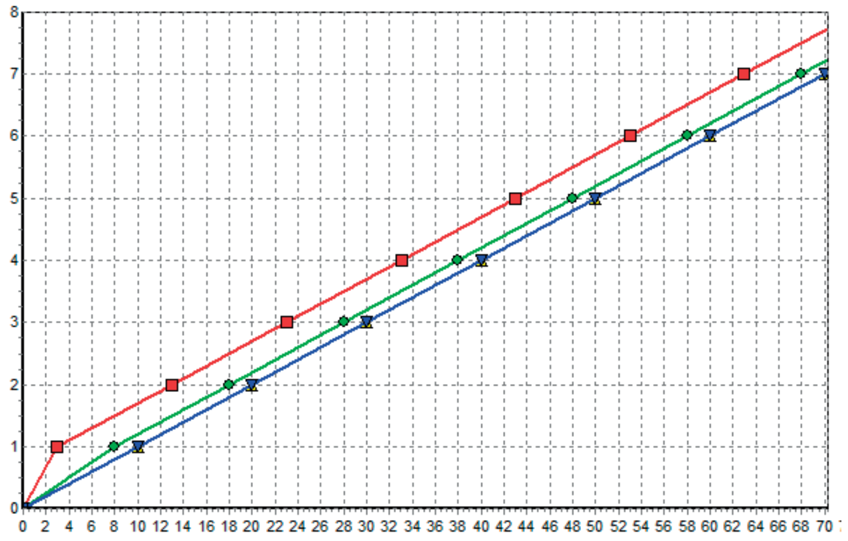

Fig. 3: Activity count of: y' ▼, y ▲, x_1 🟥 , x_2 ● for $T_C = 10$.

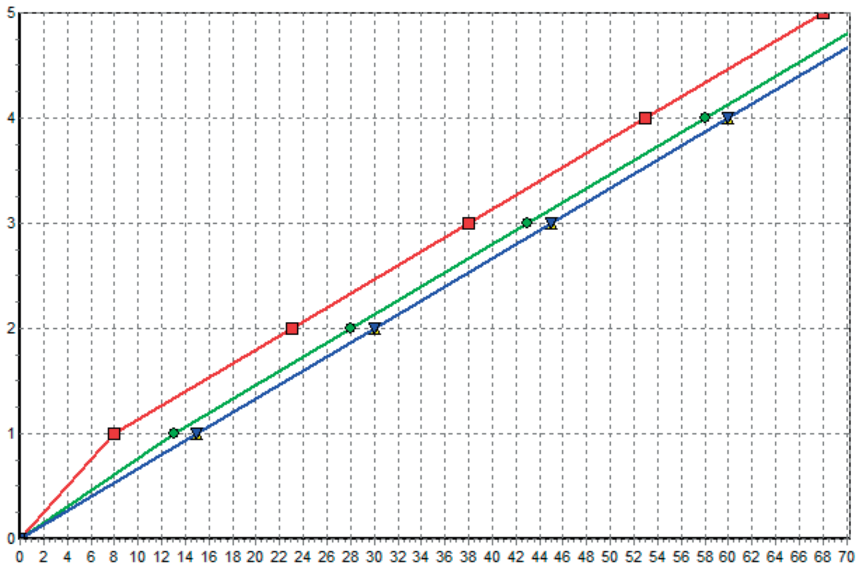

Fig. 4: Activity count of: y' ▼, y ▲, x_1 🟥 , x_2 ● for $T_C = 15$.

Furthermore, there is a need to develop a way of controlling these processes and the analyzed conditions for the periodicity of the required results. Further research will include applying the system to larger models and improving the efficiency of the optimization procedure. Future research will focus on design methods, synthesizing process control with output and/or state feedback and using models for predictive and adaptive control.

References

[1] Bacceli F., Cohen G., Olsder G., Quadrat J., *Synchronization and Linearity. An Algebra for Discrete Event Systems*, London, John Wiley & Sons Ltd, 1992.

[2] Balduzzi F., Giua Has., Menga G., *First-order hybrid Petri net: In model for optimisation and control*, IEEE Trans. On Rob. And Aut. 1 6 (4), 2000 382—399.

[3] Cassandras Ch., Lafortune St., *Introduction to Discrete Event Systems*, Springer, Kluwer Academic Publishers, 2008.

[4] Elmahi I., Grunder O., Elmoudni A., *A max plus algebra approach for modeling and control of lots delivery.* Industrial Technology, IEEE ICIT '04 Vol. 2, 8-10 Dec. 2004, 926—931.

[5] Goverde Rob M.P., *Railway timetable stability analysis using max-plus system theory*, Elsevier, Transportation Research Part B 41, 2007, p. 179–201

[6] Jamroż L., Raszka J., *Simulation method for the performance evaluation of system of discrete cyclic processes.* 16-th IASTED International Conference on Modelling, Identification and Control, Innsbruck, Austria, 17-19.02.1997, 190—193

[7] Maia C.A., Andrade C.R., Hardouin L., *On the control of max-plus linear system subject to state restriction*, Automatica, Volume 47, Issue 5, May 2011, 988—992.

[8] Murata T., *Petri nets: properties, analysis and applications.* Proceedings of the IEEE, vol. 77, no. 4, p.541-580, 1989.

[9] Guţuleac E., Balmuş I. et alt., *Descriptive Timed Membrane Petri Nets for Modelling of Parallel Computing,* International Journal of Computers, Communications & Control, Vol. I No. 3, 2006, 33—39.

[10] Nait-Sidi-Moh A., Manier M.-A., El Moudni A., *Spectral analysis for performance evaluation in a bus network*, European Journal of Operational Research Volume 193: Issue 1, 16 February 2009, 289—302.

[11] De Schutter B., van den Boom T., *Model predictive control for max-plus linear discrete event systems.* Automatica 37(7), July 2001, 1049—1056.

[12] Cordovilla M., Boniol F., et alt., *Off-line Optimal Multiprocessor Scheduling of Dependent Periodic Tasks,* Journées FAC'2011, Formalisation des Activités Concurrentes, 6 et 7 avril 2011- LAAS – CNRS, 2011.

[13] Iwaniak M., Khadzhynov W., *Usage of Petri nets for distributed transactions modeling* Studia Informatica vol. 33, no 105, Silesian University of Technology, 2012.

[14] Szpyrka M., *Sieci Petriego w modelowaniu i analizie systemów współbieżnych*, ISBN 9789788304333, WNT Warszawa, 2008.

[15] Coolahan J.E.jr, Roussopoulos N., *Timing Requirements for Time-Driven Systems Using Augmented Petri Nets*, IEEE Trans. on Software Eng. v. SE-9, no. 5, 1983, 603—616.