symmetry

MDPI

Article

Tropical Lexicographic Optimization: Synchronizing Timed Event Graphs

Alan Mendes Marotta [1,2,*], Vinicius Mariano Gonçalves [2] and Carlos Andrey Maia [2]

[1] Departamento de Engenharia Mecatrônica, Centro Federal de Educação Tecnológica de Minas Gerais, CEFET-MG, Rua Álvares de Azevedo, 400, Bela Vista, Divinópolis MG 35503-822, Brazil

[2] Programa de pós-graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, UFMG, Av Antonio Carlos 6627, Pampulha, Belo Horizonte MG 31270-901, Brazil; marianoauto@ufmg.br (V.M.G.); maia@cpdee.ufmg.br (C.A.M.)

* Correspondence: alanmarotta@cefetmg.br; Tel.: +55-37-988336064

Received: 3 September 2020; Accepted: 21 September 2020; Published: 25 September 2020

Abstract: Tropical Algebra is used to model the dynamics of Timed Event Graphs (TEG), a particular class of Timed Discrete-Event System (TDES) in which we are interested only in synchronization and delay phenomena. Whenever this TEG has control inputs, we can use them to control the synchronization of the system to achieve some objective. Thus, this paper formulates a framework based on tropical algebra and lexicographic optimization to synchronize a TEG when dealing with many synchronization objectives that are ranked in previous priority order. We call this kind of problem the Tropical Lexicographic Synchronization Optimization (TLSO). This work develops a solution to this problem, based on Tropical Fractional Linear Programming (TFLP) and lexicographic optimization concepts. In this way, the basics of tropical algebra are determined, including essential terms to this paper, such as left and right residuations, and the following stages of the solution to the TLSO problem are explained. Therefore, this work presents a general framework based on structured algebraic models with application to TEG synchronization. By synchronization, we mean balancing and organizing events chronologically in order to achieve the desired goal. So, we are dealing with concepts closely related to symmetry ones. An illustrative numerical example is presented, which demonstrates the implementation of the proposed algorithms. The acquired results confirm the efficiency of the proposed methodology. Codes used for implementing the algorithms are listed in the appendix section of the article.

Keywords: tropical algebra; timed discrete-event systems; optimization

1. Introduction

Timed Event Graphs (TEGs) are used to model a wide range of Timed Discrete-Event Systems (TDES), which is a class of dynamic systems in which the change of state is governed by events [1]. Examples of model applications include manufacturing systems, production chains, queues, and urban traffic systems [2].

A powerful tool for analysis of TEGs is the so-called Tropical Algebra, which is a kind of idempotent algebraic structure which can be a semiring or a semifield depending on the application. French researchers gave the denomination as a tribute to Brazilian mathematician *Imre Simon*, who had a considerable influence on the development of automata and semigroup theory [3].

There are many types of tropical algebras that have applications in areas such as mathematics, engineering, and computer science. Two important examples are the Min-Plus and Max-Plus algebra. Concerning Max-Plus algebra, several results were obtained for analysis and synthesis of TDES.

Among them, we highlight remarkable developments in control synthesis for such systems in the last decades (e.g., [4–9]).

The methodology used in this work addresses the synchronization problem, giving priority to TEG transitions to be synchronized, as proposed by lexicographic optimization. Each stage of the general problem is treated as a subproblem, which is solved through Tropical Fractional Linear Programming (TFLP). This optimization employs the dual iterative method, which, in turn, uses the residency operation. The original contribution of this paper is a lexicographic formulation of a TFLP problem aimed at synchronizing multiple disconnected TEGs. As references for TFLP, the reader is invited to consult [10,11], and for lexicographic optimization, the reference is [12]. More precisely, the authors propose a framework to synchronize the firing times of the transition of a TEG, considering a priority list of the pair of transitions that must be synchronized. This framework models this problem as a Lexicographic variation of a TFLP [10,11]. The authors denote the problem as a Tropical Lexicographic Synchronization Optimization (TLSO). It is important to stress that this framework can be applied in a general context. Moreover, as far as the authors' knowledge goes, there are few works regarding linear fractional programming applications, and the authors do not find researches that deal with the insertion of lexicographic optimization in this context. Therefore, this article contributes to a formulation that unites existing concepts and relates them with a numerical example. As an illustrative application, traffic light synchronization is used as an example. In particular, this paper refers to Egmund and Olsder [13], who introduce the basics of Max-Plus algebra to deal with synchronization between phases in the traffic light system, with the purpose to generate the green waves.

This work has a theoretical impact by proposing a framework capable of synchronizing a class of models—the timed event graphs. There are few applications of this concept, for instance, manufacturing [2,14], traffic light control [13], payoff games [10], and multiprocessed systems [15,16]. The remainder of the paper is organized as follows: In Section 2, we introduce the preliminary concepts, in which the elements of tropical algebra are defined. In Section 3, the Tropical Lexicographic Synchronization Optimization is discussed, and the used tools to solve it are explained. Section 4 deals with a numerical example, which is explained while developing a study case. Finally, in the Appendix Section, we provide the listed codes implemented in ScicosLab. Note that Scicolab is a free package for mathematical computing, similar to Matlab. It can be found in http://www.scicoslab.org. The software is very convenient to work with tropical algebra since it has many of the related functions native implemented. (Version 4.4) code for the algorithm.

2. Tropical Algebra

This work is heavily based on tropical algebra. Therefore, in this section, we introduce some concepts and definitions related to this algebra. For more details on academic background, consult a historical review of Max-Plus tropical algebra [2,17,18]. "Tropical algebra" is a generic name for a family of idempotent algebraic structures, which can be a semiring or a semifield depending on the application. A special example of tropical algebra is the so-called Max-Plus Algebra. It is a particular Tropical structure act on the set $\mathbb{T}_{max} = \mathbb{R} \cup \{-\infty\}$ with two operations, named $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$. Furthermore, let $\mathbb{T}^n_{max}$ and $\mathbb{T}^{n\times m}_{max}$ be the set of column vectors with n entries in T_{max} and the set of matrices with n rows and m columns in T_{max}, respectively. From now on, we consider that "Max-Plus algebra" and "tropical algebra" can be used as synonyms, and this paper adopts the latter notation.

This algebra can be extended to matrices and vectors. If $A = (A_{ij})$, $B = (B_{ij})$, and $C = (C_{ij})$ are matrices of compatible sizes with entries in $\mathbb{T}_{max}$, we denote by $C = A \oplus B$ the tropical matrix sum, in which $C_{ij} = A_{ij} \oplus B_{ij}$. Furthermore, we define as $C = A \otimes B$ the tropical matrix product, in which $C_{ij} = \bigoplus_k A_{ik} \otimes B_{kj}$. In later notations, we omit the symbol $\otimes$ and denote the tropical product by juxtaposition, so $A \otimes B = AB$. We use the symbol ε to denote $-\infty$. Let $\varepsilon_{n\times m}$ be a matrix of ε with n rows and m columns. This article denotes the Max-Plus identity matrix of appropriate order, a matrix

with 0 in the diagonal and ε outside, by $Id_{\mathbb{T}}$. For a square matrix A, we denote by A^k the k^{th} tropical power of A, defined recursively as $A^k = AA^{k-1}$ and $A^0 = Id_{\mathbb{T}}$.

If A is a matrix (or a scalar) let $-A$ be the matrix (or the scalar) obtained by switching the sign of its entries. If a is a vector and b is a scalar $b \neq \varepsilon$, let $\frac{a}{b}$ be $(-b)a$ (all the elements of a subtracted by b). Let $\rho(A)$ denote the *spectral radius* of A, i.e., the largest eigenvalue. In Tropical Algebra, the eigenvalue defines the cycle time in a TEG and is still active research topic, with recent results, such as involving nontrivial eigenvectors [19,20]. The symbol is because $\frac{a}{b}$ is the analog of a division in Max-Plus algebra. If $\rho(A) \leq 0$, let A^* denote the *Kleene closure* of A, i.e., $\bigoplus_{i=0}^{\infty} A^i$. It is assumed from now on that the reader is familiar with the basics of this algebra, including concepts as Kleene closure [21].

Remark 1. *Unless stated otherwise, all operations in this paper are done in tropical algebra. So it is essential to be careful with the symbol that we use for the operations.*

An important operation in tropical algebra is the *residuation*. The residuation works as a kind of multiplicative inverse in the matrix tropical algebra. This work defines it below:

Definition 1. *We define $\oslash$, $\oblong$—the left and right residuation of the product, respectively—as (see more details in [22])*

$$A \oblong B \equiv \begin{array}{c} \max X \\ XB \leq A \end{array}, \tag{1}$$

$$A \oslash B \equiv \begin{array}{c} \max X \\ AX \leq B \end{array}. \tag{2}$$

Note that the residuations are implicitly defined by multiobjective optimization problems, since we want to optimize all the entries of the matrix X of appropriate dimension subject to the constraint $XB \leq A$ or $AX \leq B$. Perhaps surprisingly, there is indeed a solution X, which is optimal in all its entries (the so-called *utopical solution* in optimization), and it is found by the residuations $A \oblong B$ and $A \oslash B$.

For instance, consider problem (3) below:

$$\max_X \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}, \tag{3}$$

subject to

$$\underbrace{\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}}_{A} \underbrace{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}}_{X} \leq \underbrace{\begin{pmatrix} 5 \\ 6 \end{pmatrix}}_{B}. \tag{4}$$

Note that this is of the form of (2). Thus, notice that the previous inequality can be rewritten as

$$\begin{array}{l} 1x_1 \oplus 2x_2 \leq 5 \\ 3x_1 \oplus 4x_2 \leq 6 \end{array} \iff \begin{array}{l} 1x_1 \leq 5\,,\ 2x_2 \leq 5 \\ 3x_1 \leq 6\,,\ 4x_2 \leq 6 \end{array} \tag{5}$$

Let $A \wedge B$ be the entrywise minimum between the matrices A and B. Consequently,

$$\begin{array}{l} x_1 \leq 4\,,\ x_2 \leq 3 \\ x_1 \leq 3\,,\ x_2 \leq 2 \end{array} \iff \begin{array}{l} x_1 \leq 4 \wedge 3 = 3 \\ x_2 \leq 3 \wedge 2 = 2 \end{array}. \tag{6}$$

Therefore, we have that

$$X = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 3 \\ 2 \end{pmatrix}. \tag{7}$$

Note that all the previous operations can be reversed. This implies that $X_{max} = (3 \;\; 2)^T$ is the maximum possible value for the matrix X, optimal in all its entries x_1 and x_2. This is exactly the (left) residuation: $X_{max} = A \backslash\!\!\diamond B$.

The general formula for computing the left residuation is $A \backslash\!\!\diamond B = -(A^T(-B))$. Analogously, for the right residuation, $A \diamond\!\!/ B = -((-A)B^T)$. In both cases, in the internal tropical product, we consider that $-\varepsilon + \varepsilon = \varepsilon$.

3. Tropical Lexicographic Synchronization Programming

In general, the purpose of this research is to synchronize the transitions of the TEGs, which describes the discrete event systems. In this section, we propose a framework for establishing synchronization among the transitions of a TEG, a model for TDES. This is applicable in, for instance, traffic light systems synchronization, as is discussed further in this paper. Note that this is only one of the various possible applications of this kind, in which it is desired to have synchronization among transitions.

3.1. Motivation

As an example of application of the proposed procedure, consider a traffic light system, in which the goal is to obtain synchronization between phases at adjacent crossings, which allows green waves, a sequence of green lights in continuous crossings.

A traffic light system can be modeled as a TEG [13]. Figure 1 illustrates a TEG example in which we have two subsystems, in which each one is related to a crossing. Each transition x_i represents a change of phases in each crossing, in which a phase corresponds to a combination of enabled green lights. We also have transitions u_i, which represents the control of each phase. This means that we can delay the end of each phase.

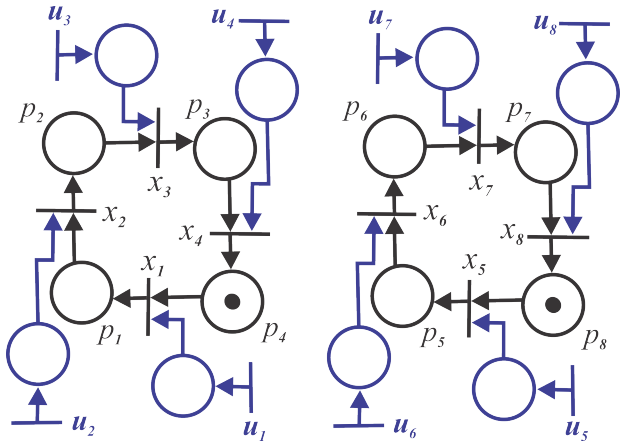

Figure 1. Timed Event Graph (TEG) of synchronization example.

A correct synchronization of the system allows the creation of green waves. The synchronization consists on having firing times x_i as close as possible. Thus, the problem involves choosing the firing times u_i, which optimize the synchronization of the system, considering a priority order. Prioritization is necessary because it may not be possible to synchronize all the transitions x_i, it is the same as synchronizing the TEG. More details are seen in the next section.

Note that the application of traffic light control is only one of the possible applications. For instance, we could consider manufacture processing and other systems that need sharing and

synchronization of resources. The generic framework used to solve these described problems is proposed in the next subsections.

3.2. Formulation of the Tropical Lexicographical Synchronization Problem

Consider the Tropical linear system below, in which we have the state $x[k] \in \mathbb{T}^n_{max}$ and input $u[k] \in \mathbb{T}^m_{max}$

$$x[k+1] = Ax[k] \oplus Bu[k+1]. \tag{8}$$

We suppose that our policy for $u[k]$ is $u[k] = \lambda^k \mu$, in which λ is a scalar and μ is a constant vector of the same dimension as $u[k]$. With this same λ and μ we can obtain a closed-loop controller $u[k] = Fx[k]$, see [23]. If $\lambda > \rho(A)$, it is known [8] that there are a finite K such that for all $k \geq K$, $x[k] = \lambda^k \chi$ for a constant vector χ of same dimension as $x[k]$ that depends on μ. Let $M = ((-\lambda)A)^* B$. More precisely,

$$\chi(\mu) = M\mu. \tag{9}$$

For a fixed choice of $\lambda > \rho(A)$, we want to choose the vector μ such that the entries of $x[k]$ in steady-state ($k \geq K$) are as close as possible, i.e., they are synchronized. Unfortunately, it is not possible to have all the entries of $x[k]$ synchronized (i.e., equal), because we may not have enough freedom in μ to achieve such an objective. So, we need to prioritize the synchronizations. So, it is necessary to define some elements:

- Let P be the number of pairs of transitions of the TEG that we want to synchronize.
- Let $\mathcal{L}$ be the list of P pairs of indexes $(i(p), j(p))$ of entries of $x[k]$ that we want to synchronize, ordered according to a decreasing defined priority. So, for instance, the pair of indexes $(i(1), j(1))$ is more important to be synchronized than the pair of indexes $(i(2), j(2))$. Furthermore, we assume that in a pair $(i(p), j(p))$, $x_{i(p)}(k) \geq x_{j(p)}(k)$, i.e., we pre-specify a temporal order on the state in those indexes (the firing time $x_{i(p)}$ comes after the firing time $x_{j(p)}$, or, in the best case, at the same time).
- Let $\delta_p = x_{i(p)}(k) - x_{j(p)}(k)$. This is the synchronization index for the p^{th} pair, which we want to minimize. Since the $x_n(k)$ represent the firing times of the transitions for the k^{th} time, δ_p represents the delay between the k^{th} firing time of transition $x_{j(p)}(k)$ and the k^{th} firing time of transition $x_{i(p)}(k)$. (So, if δ_p equals to 0, the transitions happen simultaneously).

Note that, since $x_{i(p)}(k) \geq x_{j(p)}(k)$, this is always a non-negative number. Moreover, since $x[k] = \lambda^k \chi(\mu)$ is in steady-state, one can see that $\delta_p = \chi_{i(p)}(\mu) - \chi_{j(p)}(\mu)$, and therefore is independent on both k (that is why the symbol δ_p does not depend on k, although the definition seems to depend on k) and λ, depending only on χ, which in turn depends on our decision variable μ according to (9).

We finally conclude that a higher priority pair has an utmost priority over a lower priority pair. This implies that we have a lexicographical optimization on the indexes δ_p. Let I_p^T and J_p^T be the $i(p)^{th}$ and $j(p)^{th}$ row of the matrix M. Let m_r be the r^{th} row of M, respectively. Note that $x_p(k) = \lambda^k m_p^T \mu$. So, our optimization problem is

$$\begin{aligned} &\underset{\mu}{lex\min} \left\{ \begin{array}{c} I_p^T \mu \\ \hline J_p^T \mu \end{array} \right\}_{p=1,2,\ldots,P} \\ &\text{subject to: } I_p^T \mu \geq J_p^T \mu, \ p = 1, 2, \ldots, P. \end{aligned} \tag{10}$$

It is possible that a pair of $(i(p), j(p))$ cannot be synchronized, given the previous constraints. In Section 3.5, it is explained how this can be detected. In this case, we need to skip this pair and continue to the next one.

We define problems of this type as Tropical Lexicographic Synchronization Optimization (TLSO). Since the parameters of the problem are the matrix M, the list $\mathcal{L}$, and the scalar $\lambda > \rho(A)$, we can denote the problem by $TLSO(A, B, \mathcal{L}, \lambda)$.

3.3. Lexicographic Optimization Problem

In a general form, a Lexicographic Optimization Problem consists of solving $lex\min_{\mu} f_i(\mu)$, $i = 1, 2, ..., p$, subject to a constraint $g(\mu) \leq 0$. Note that the objective functions $f_i(\mu)$ are ordered so an index i has more priority than index $i+1$. One procedure to solve this kind of problem consists of solving a sequence of traditional optimization problems for $j = 1$ to $j = p$:

$$\begin{gathered} \min_{\mu} f_j(\mu) \\ \text{subject to: } g(\mu) \leq 0\,, \\ f_i(\mu) = f_i^*,\ i = 1, 2, ..., j-1 \end{gathered} \tag{11}$$

in which f_i^* is the optimal value of $f_i(\mu)$ in the previous problem [12].

3.4. Tropical Fractional Linear Programming

We want to solve a TLSO, which is a Lexicographic Optimization Problem. We use the strategy explained in Section 3.3, which reduces lexicographic optimization problems into a sequence of traditional optimization problems. In our case, the particular traditional optimization problems are TFLP. The general form of a TFLP problem is

$$\begin{gathered} \max_{\mu} \frac{w^T \mu \oplus \alpha}{f^T \mu \oplus \beta}, \\ \text{subject to: } R\mu \oplus r = S\mu \oplus s \end{gathered} \tag{12}$$

in which w, α, f, β, R, r, S, s are parameters. This kind of problem has been studied in [10,11], and algorithms have been proposed to solve them. The algorithm proposed in [11], in particular, relies on the ability to solve a particular case of TFLP. This particular case is called *Max-Type Tropical Linear Problem (Max TLP)*:

$$\begin{gathered} \max_{v}\ g^T v, \\ \text{subject to: } Ev \oplus e = Dv \oplus d. \end{gathered} \tag{13}$$

In order to guarantee that the solution v for this problem is finite, the constraint of (13) must generate a set, which is upper bounded; in better words, there must exist a finite vector v_{max} such that $v \leq v_{max}$. From now on, we assume that this is true.

An algorithm to solve this problem is called *Dual Method* [11]. This is an iterative method that works with the following recursion:

$$v[k+1] = E \backslash (Dv[k] \oplus d) \wedge D \backslash (Dv[k] \oplus e) \wedge v[k], \tag{14}$$

with the initial condition $v(0) = v_{max}$. In each iteration k, the following conditions must be tested:

$$\begin{gathered} e \leq Dv[k] \oplus d, \text{ and} \\ d \leq Ev[k] \oplus e. \end{gathered} \tag{15}$$

If these conditions are not true, the algorithm halts, and we can conclude that the feasible set of (13) is empty. Otherwise, when the algorithm converges (new v equal to a previous v), it is able to find the optimal solution for the optimization problem. Note that the algorithm is independent of g. It turns out that, indeed, the solution μ is independent of these elements (see [11]). Any solution v which is optimal for a vector g_1 is also optimal for any other g_2. More generally, any objective function $f(v)$ which is nondecreasing implies the same behavior. This happens because the algorithm in (14) will find a solution v inside the feasible set of (13) that is maximal in any entry of v [24].

When using the discussed Tropical Dual method by solving (14), we can solve the TFLP problems by converting Max-Type TFLP into Max-Type TLP problem according to Figure 2 through

Charnes–Cooper conversion [25]. The complete TFLP problem is defined in (12). It is straightforward to adapt the Charnes–Cooper transformation to the tropical setting [11]. Set

$$\begin{aligned} \theta &= \frac{\mu}{f^T\mu \oplus \beta}, \\ \hat{\theta} &= \frac{0}{f^T\mu \oplus \beta}. \end{aligned} \tag{16}$$

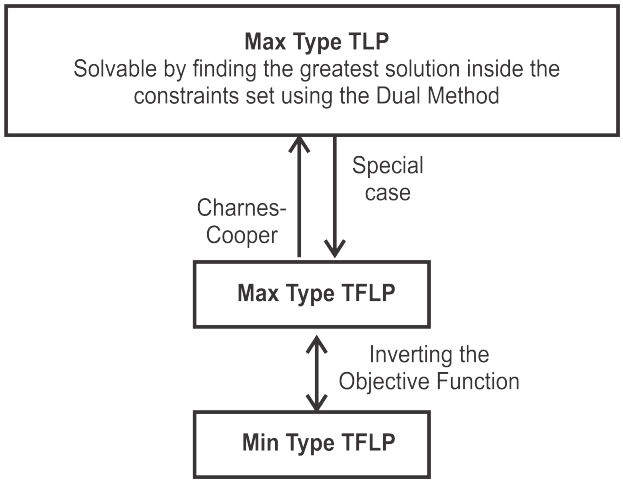

Figure 2. Using Max-Type Tropical Linear Problem (Max TLP) to solve Tropical Fractional Linear Programming (TFLP) problems (adapted from [11]).

Then, by dividing (in tropical algebra) both sides of the affine equation on (12) by $f^T\mu \oplus \beta$, it can be rewritten as

$$\begin{gathered} \max_{\theta,\hat{\theta}} w^T\theta \oplus \alpha\hat{\theta}, \\ \text{subject to:} \\ R\theta \oplus r\hat{\theta} = S\theta \oplus s\hat{\theta}; \\ f^T\theta \oplus \beta\hat{\theta} = 0 \end{gathered} \tag{17}$$

in which the additional equation $f^T\theta \oplus \beta\hat{\theta} = 0$ condenses (16).

Define the vector

$$v = \begin{bmatrix} \theta \\ \hat{\theta} \end{bmatrix}. \tag{18}$$

Let n and m be the number of rows and columns of R, respectively. With this definition, we can rewrite (17) as

$$\begin{gathered} \max_{v} \underbrace{[w^T \ \alpha]}_{g^T} v, \\ \text{subject to:} \\ \underbrace{\begin{bmatrix} R & r \\ f^T & \beta \end{bmatrix}}_{E} v \oplus \underbrace{\begin{bmatrix} \varepsilon_{n\times 1} \\ \varepsilon \end{bmatrix}}_{e} = \underbrace{\begin{bmatrix} S & s \\ \varepsilon_{1\times m} & \varepsilon \end{bmatrix}}_{D} v \oplus \underbrace{\begin{bmatrix} \varepsilon_{n\times 1} \\ 0 \end{bmatrix}}_{d}. \end{gathered} \tag{19}$$

Remember that $\varepsilon_{n\times 1}$ is a vector of ε with n rows and one column, and $\varepsilon_{1\times m}$ is a vector of ε with one row and m columns.

This problem is a Max-type TLP as (13) and can be solved by Algorithm 1. The program's code in the Scicoslab language is listed in Appendix A.1. Once θ and $\hat{\theta}$ are obtained, in order to return to the original variable, one needs to revert (16).

$$\mu = (f^T \mu \oplus \beta)\theta = \frac{\theta}{\hat{\theta}}. \tag{20}$$

Algorithm 1: Tropical Dual Function: TDual Algorithm

l

Inputs: E, D, e, d;
Output: v;
1. Initialize upperbound variable $v(k)$ equals to vector of big numbers with size of E;
2. Initialize stop flags: $converge = false$; $itok = true$; $condi = true$;
3. Initialize iteration $k = 0$;
4. **while** *not converge AND itok AND condi* **do**
   Calculates the new $v(k+1)$ according to Equation (14);
   Increasing the step $k = k + 1$;
   Execute the test of conditions (15), store *condi*;
   Verify the convergence, store *converge*;
   Execute test of maximum iterations, store *itok*;
5. **if** *not condi OR not itok* **then**
   v equals empty;
6. Return v.

The TFLP method is implemented by the function $\texttt{TFLP}(w, \alpha, f, \beta, R, r, S, s)$, in which w, α, f, β, R, r, S, and s are parameters and it returns μ and δ, in which δ is the value of the optimal objective function, according to the implementation listed in Algorithm 2. The program's code in the Scicoslab language is listed in Appendix A.2.

Algorithm 2: Tropical Fractional Linear Programming: TFLP

Inputs: $w, \alpha, f, \beta, R, r, S, s$;
Outputs: u, δ;
1. Create the matrices E, e, D, d according to (19);
2. Calls the Dual Method to solve Max TLP;
3. **if** *solution is empty* **then**
   u equals to empty and δ equals to empty;
else
   If the solution exists, convert back to the original variable according to (20);
   $\delta = (w * u + \alpha) - (f * u + \beta)$;
4. Return u, δ.

3.5. Tropical Lexicographic Synchronization Programming

We solve the TLSO (10) using the general strategy proposed in Section 3.3. This implies solving a sequence of TFLPs (12), as discussed in Section 3.4. In this subsection, we describe this procedure in more detail.

First, note that the constraints in problem (10) can be rewritten as

$$\begin{aligned} &lex\min_{\mu} \ \left\{ \frac{I_p^T \mu}{J_p^T \mu} \right\}_{p=1,2,\ldots,P} \\ &\text{subject to: } (I_p^T \oplus J_p^T)\mu = I_p^T \mu. \end{aligned} \tag{21}$$

Remember that we have a list $\mathcal{L}$ of pairs $(i(p), j(p))$

$$\mathcal{L} = \{(i(1), j(1)); (i(2), j(2))...(i(P), j(P))\} \tag{22}$$

ordered by a decreasing priority and also that $x_{i(p)}(k) \geq x_{j(p)}(k)$. So, according to the procedure described in 3.3, the first problem that we need to solve is

$$\begin{aligned} &\min_{\mu} \ \frac{I_1^T \mu}{J_1^T \mu} \\ &\text{subject to: } (I_1^T \oplus J_1^T)\mu = I_1^T \mu, \end{aligned} \tag{23}$$

a TFLP (as in (12)) which can be solved using the procedure described in Section 3.4. After this TFLP is solved, we obtain a solution μ_1^*, and then we compute the first δ_1 as

$$\delta_1 = \frac{I_1^T \mu_1^*}{J_1^T \mu_1^*}. \tag{24}$$

Now, the next step is to create the second problem in the sequence. We need to change the objective function to $I_2^T \mu - J_2^T \mu$ and also to add two new constraints:

$$(I_2^T \oplus J_2^T)\mu = I_2^T \mu. \tag{25}$$

$$\frac{I_1^T \mu}{J_1^T \mu} = \delta_1 \iff I_1^T \mu = \delta_1 J_1^T \mu. \tag{26}$$

The second problem, in addition to the previous constraints and the change of objective function, is augmented by two new constraints, one to synchronize the new pair and another constraint to guarantee the previous result δ_1:

$$\begin{aligned} &\min_{\mu} \ \frac{I_2^T \mu}{J_2^T \mu} \\ &\text{subject to:} \\ &(I_2^T \oplus J_2^T)\mu = I_2^T \mu \\ &I_1^T \mu = \delta_1 J_1^T \mu \\ &(I_1^T \oplus J_1^T)\mu = I_1^T \mu. \end{aligned} \tag{27, 28}$$

From the third problem on, up to $p = P$, we have two more constraints for each new pair:

$$\begin{aligned} &\min_{\mu} \ \frac{I_p^T \mu}{J_p^T \mu} \\ &\textit{subject to:} \end{aligned} \tag{29}$$

$$(I_p^T \oplus J_p^T)\mu = I_p^T \mu \Big\} \ p = P \tag{30}$$

$$\left.\begin{aligned} I_{p-1}^T \mu = \delta_{p-1} J_{p-1}^T \mu \\ (I_{p-1}^T \oplus J_{p-1}^T)\mu = I_{p-1}^T \mu \end{aligned}\right\} \ p = P - 1$$

$$...$$

$$\left.\begin{array}{r} I_2^T\mu = \delta_2 J_2^T\mu \\ (I_2^T \oplus J_2^T)\mu = I_2^T\mu \end{array}\right\} p = 2 \qquad (31)$$

$$\left.\begin{array}{r} I_1^T\mu = \delta_1 J_1^T\mu \\ (I_1^T \oplus J_1^T)\mu = I_1^T\mu \end{array}\right\} p = 1.$$

If the p^{th} pair cannot be synchronized, that is, if the Algorithm 2 returns μ equal to empty, the pair is deleted from the list $\mathcal{L}$, the size P of the list $\mathcal{L}$ is updated, and the iteration p is not increased.

The implementation of the TLSO is given by the function `TLSO(A,B,L,lambda)` in Algorithm 3. The program's code in the Scicoslab language is listed in Appendix A.3, in which the parameters A and B are Max-Plus matrices. In ScicosLab the Max-Plus denomination is used instead of Tropical. L is the priority pairs list $\mathcal{L}$ represented by a matrix with P rows and two columns, and λ is a scalar that is related to the period. The function `TSLP` returns the optimal μ^*.

It is important to stress that the TLSO is *tropical homogeneous*—that means, if μ is a solution, so is $\alpha\mu$ with α, a scalar. Therefore, we can shift the vector μ that we obtain after we solve the problem, so that all its elements are non-negative.

A numerical example is developed in Section 4 to clarify the TLSO method.

Algorithm 3: Tropical Lexicographic Synchronization Optimization: TLSO

Inputs: $A, B, \mathcal{L}, \lambda$;
Output: u^*;
1. Define $M = [((-\lambda)A)B]^*$;
2. Define the maximum possible iterations: P=size($\mathcal{L}$);
3. Initialize the first pair iteration $p = 1$;
4. **while** $(p < P)$ **do**
   - Obtain the p^{th} pair (i, j) of the list $\mathcal{L}$;
   - Obtain the line i and j of the matrix M;
   - Define w, f, R, and S (iteration constraints (30) and prior constraints (31));
   - Define r and s as vector of ϵ;
   - Call the function TFLP and obtain u;
   - **if** *u is empty* **then**
     - erase pair p of the list $\mathcal{L}$ and update the size P of list $\mathcal{L}$;
   - **else**
     - $u^* = u$;
     - $p = p + 1$;
5. Return u^*.

4. Numerical Example

In this section, we develop a numerical example. We work with the TEG illustrated in Figure 1. As mentioned in the motivation Section 3.1, the illustrated TEG can model a traffic light system, which has two subsystems that refers to crossings. For more about the traffic light modeling with TEGs, see [13,26].

We suppose that the state equations are known and is given as follows:

$$\begin{aligned} x_1(k+1) &= 10.6x_4(k) \\ x_2(k+1) &= 38.6x_1(k+1) \\ x_3(k+1) &= 12x_2(k+1) \\ x_4(k+1) &= 20.8x_3(k+1) \\ x_5(k+1) &= 20.8x_8(k) \\ x_6(k+1) &= 15.7x_5(k+1) \\ x_7(k+1) &= 12x_6(k+1) \\ x_8(k+1) &= 25.9x_7(k+1) \end{aligned} \tag{32}$$

We also suppose that the list $\mathcal{L}$ of pairs that have a connection is known and is given as follows:

$$\mathcal{L} = \{(3,7),(4,8),(4,7))\}. \tag{33}$$

With these equations, we can write in state-space form $x[k+1] = Ax[k] \oplus Bu[k+1]$ once we define the matrices:

$$A_0 = \begin{bmatrix} A_0^1 & \varepsilon_{4\times4} \\ \varepsilon_{4\times4} & A_0^2 \end{bmatrix}, \; A_1 = \begin{bmatrix} A_1^1 & \varepsilon_{4\times4} \\ \varepsilon_{4\times4} & A_1^2 \end{bmatrix}, \tag{34}$$

in which,

$$A_0^1 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 38.6 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 12 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 20.8 & \varepsilon \end{bmatrix}, \tag{35}$$

$$A_0^2 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ 15.7 & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 12 & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & 25.9 & \varepsilon \end{bmatrix}, \tag{36}$$

$$A_1^1 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & 10.6 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}, \tag{37}$$

$$A_1^2 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & 20.8 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \tag{38}$$

The matrices A_0 and A_1 have relation with the times $x[k+1]$ (next state) and $x[k]$ (current state), respectively. We execute the operation Kleene closure (see [21]) and obtain the matrix $A = A_0^* A_1$:

$$A = \begin{bmatrix} A^1 & \varepsilon_{4\times4} \\ \varepsilon_{4\times4} & A^2 \end{bmatrix}, \tag{39}$$

in which,

$$A^1 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & 10.6 \\ \varepsilon & \varepsilon & \varepsilon & 49.2 \\ \varepsilon & \varepsilon & \varepsilon & 61.2 \\ \varepsilon & \varepsilon & \varepsilon & 82 \end{bmatrix}, \tag{40}$$

$$A^2 = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & 20.8 \\ \varepsilon & \varepsilon & \varepsilon & 36.5 \\ \varepsilon & \varepsilon & \varepsilon & 48.5 \\ \varepsilon & \varepsilon & \varepsilon & 74.4 \end{bmatrix}. \tag{41}$$

The matrix B is associated to control transition, thus, we obtain the matrix $B = A_0^*$, as

$$B = \begin{bmatrix} B^1 & \varepsilon_{4\times4} \\ \varepsilon_{4\times4} & B^2 \end{bmatrix}, \tag{42}$$

in which,

$$B^1 = \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \\ 38.6 & 0 & \varepsilon & \varepsilon \\ 50.6 & 12 & 0 & \varepsilon \\ 71.4 & 32.8 & 20.8 & 0 \end{bmatrix}, \tag{43}$$

$$B^2 = \begin{bmatrix} 0 & \varepsilon & \varepsilon & \varepsilon \\ 15.7 & 0 & \varepsilon & \varepsilon \\ 27.7 & 12 & 0 & \varepsilon \\ 53.6 & 37.9 & 25.9 & 0 \end{bmatrix}. \tag{44}$$

The cycle time of the system can be obtained by calculating the eigenvalue, which results $\lambda = 82s$. Once the data $(A, B, \mathcal{L}, \lambda)$ is ready, as a result we get the synchronizing vector μ from the algorithm `TLSO` (Algorithm 3, listed code Appendix A.3).

Obtaining matrix M**:** The TLSO algorithm uses the parameters λ, A, and B to calculate the matrix $M = ((-\lambda)A)^* B$, as

$$M = \begin{bmatrix} M^1 & \varepsilon_{4\times4} \\ \varepsilon_{4\times4} & M^2 \end{bmatrix}, \tag{45}$$

in which,

$$M^1 = \begin{bmatrix} 0 & -38.6 & -50.6 & -71.4 \\ 38.6 & 0 & -12 & -32.8 \\ 50.6 & 12 & 0 & -20.8 \\ 71.4 & 32.8 & 20.8 & 0 \end{bmatrix}, \tag{46}$$

$$M^2 = \begin{bmatrix} 0 & -23.3 & -35.3 & -61.2 \\ 15.7 & 0 & -19.6 & -45.5 \\ 27.7 & 12 & 0 & -33.5 \\ 53.6 & 37.9 & 25.9 & 0 \end{bmatrix}. \tag{47}$$

Step one: The algorithm `TLSO` starts the search by finding an optimal synchronized vector μ by solving the first optimization problem for the first pair of indexes from the list $\mathcal{L}$.

In the first iteration ($p = 1$), the pair (3,7) from the list $\mathcal{L}$ is used, we have $i_p = 7$ and $j_p = 3$. In this way, I_1 (48) is the 7th line and J_1 (49) is the 3rd line of the matrix M:

$$I_1^T = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & 27.7 & 12 & 0 & -33.5 \end{bmatrix} \tag{48}$$

$$J_1^T = \begin{bmatrix} 50.6 & 12 & 0 & -20.8 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \tag{49}$$

In order to solve the first TFLP problem (see Section 3.4), we use $w = I_1$, $\alpha = 0$, $f = J_1$, $\beta = \varepsilon$, $R = I_1 \oplus J_1$, $r = \varepsilon$, $S = I_1$, and $s = \varepsilon$.

The lexicographic algorithm calls the function TFLP (Algorithm 2) for these defined variables $(w, \alpha, f, \beta, R, r, S, s)$. The function returns the variables μ_1^*:

$$\mu_1^{*T} = \begin{bmatrix} -50.6 & -12 & 0 & 20.8 & -27.7 & -12 & 0 & 33.5 \end{bmatrix} \tag{50}$$

and $\delta_1 = 0$ (perfect synchronization between the transitions x_7 and x_3). In that way, we have the first vector μ_1^*.

Second step: At the second iteration ($p = 2$) the pair $(3, 8)$ is used, we have $i(2) = 3$ and $j(2) = 8$, being I_2 and J_2:

$$I_2^T = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & 53.6 & 37.9 & 25.9 & 0 \end{bmatrix}, \tag{51}$$

$$J_2^T = \begin{bmatrix} 71.4 & 32.8 & 20.8 & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \tag{52}$$

From (28), we compute the R (53) and S (54) variables in the second iteration ($p = 2$):

$$R = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & 53.6 & 37.9 & 25.9 & 0 \\ 50.6 & 12 & 0 & -20.8 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 27.7 & 12 & 0 & -33.5 \end{bmatrix}, \tag{53}$$

$$S = \begin{bmatrix} 71.4 & 32.8 & 20.8 & 0 & 53.6 & 37.9 & 25.9 & 0 \\ \varepsilon & \varepsilon & \varepsilon & \varepsilon & 27.7 & 12 & 0 & -33.5 \\ 50.6 & 12 & 0 & -20.8 & 27.7 & 12 & 0 & -33.5 \end{bmatrix}. \tag{54}$$

The second iteration returns from the function TFLP the result of μ_2^{*T}:

$$\mu_2^{*T} = \begin{bmatrix} -76.5 & -37.9 & -25.9 & -5.1 & -53.6 & -37.9 & -25.9 & 0 \end{bmatrix} \tag{55}$$

and $\delta_2 = 5.1$, which is the best synchronization possible in this case for this index pair.

Third step: At the third iteration ($p = 3$), we have the pair $(4, 7)$, in which we compute I_3 and J_3 correspondent to stages $i(3) = 4$ and $j(3) = 7$:

$$I_3^T = \begin{bmatrix} \varepsilon & \varepsilon & \varepsilon & \varepsilon & 27.7 & 12 & 0 & -33.5 \end{bmatrix}, \tag{56}$$

$$J_3^T = \begin{bmatrix} 71.4 & 32.8 & 20.8 & 0 & \varepsilon & \varepsilon & \varepsilon & \varepsilon \end{bmatrix}. \tag{57}$$

Once I_3 (56) and J_3 (57) are defined, we compute R and S as

$$R^T = \begin{bmatrix} \varepsilon & 76.5 & \varepsilon & 50.6 & \varepsilon \\ \varepsilon & 37.9 & \varepsilon & 12 & \varepsilon \\ \varepsilon & 25.9 & \varepsilon & 0 & \varepsilon \\ \varepsilon & 5.1 & \varepsilon & -20.8 & \varepsilon \\ 27.7 & \varepsilon & 53.6 & \varepsilon & 27.7 \\ 12 & \varepsilon & 37.9 & \varepsilon & 12 \\ 0 & \varepsilon & 25.9 & \varepsilon & 0 \\ -33.5 & \varepsilon & 0 & \varepsilon & -33.5 \end{bmatrix}, \tag{58}$$

$$S^T = \begin{bmatrix} 71.4 & \varepsilon & 71.4 & \varepsilon & 50.6 \\ 32.8 & \varepsilon & 32.8 & \varepsilon & 12 \\ 20.8 & \varepsilon & 20.8 & \varepsilon & 0 \\ 0 & \varepsilon & 0 & \varepsilon & -20.8 \\ 27.7 & 53.6 & 53.6 & 27.7 & 27.7 \\ 12 & 37.9 & 37.9 & 12 & 12 \\ 0 & 25.9 & 25.9 & 0 & 0 \\ -33.5 & 0 & 0 & -33.5 & -33.5 \end{bmatrix}. \tag{59}$$

Using these parameters R (58) and S (59), the third problem is solved, and the variable μ returns empty, which indicates that the algorithm does not converge and the synchronization is not possible. Therefore, we have to skip this phase and keep the last vector $\mu^* = \mu_2^*$. Since there are no more pairs to synchronize, the algorithm is over.

As we explained in Section 3.5, we can shift the vector μ in a way that we have only non-negative values. Thus, we use $\mu = \lambda\mu_2^*$, obtaining

$$\mu^T = \begin{bmatrix} 5.5 & 44.1 & 56.1 & 76.9 & 29.4 & 44.1 & 56.1 & 82 \end{bmatrix}. \tag{60}$$

The proposed example in this section clarifies the implementation of the lexicographic algorithm for synchronization among stages according to the previously defined list. In the first iteration, there is the perfect synchronization ($\delta_1 = 0$). In the second iteration, the synchronization is possible with delay ($\delta_2 = 5.1$). Lastly, in the third iteration, it is not possible to synchronize (μ is empty).

5. Conclusions

In this work, we propose the TLSO (Tropical Lexicographic Synchronization Optimization) framework. This framework is based on TFLP (Tropical Fractional Linear Programming), which allows the formulation and solution of problems in which it is desirable to synchronize transitions of TEGs. As far as the authors' knowledge goes, there are few practical TLFP applications in the literature. Thus, this article's contribution is the formalization and utilization of TFLP for modeling of synchronization problems, in which we develop a numerical example of traffic light synchronization. It is important to note that if synchronization is not feasible, the method will handle it.

The research gap served by this work is the lexicographic proposition together with models resolved by TFLP for synchronization of TEGs subject to delays. The framework proposed in this research is limited to the characteristics resulting from the time events graph model, which applies to systems subjected to synchronization and delay phenomena. The study does not apply to systems where there is a dispute over resources of another nature. As future work, we propose the formulation for broader models, represented by more general TDES. Another possibility for future work is to consider an approach for closed-loop systems. The proposed lexicographic method is also suitable for other systems, such as manufacturing systems, and others in which synchronization and resource sharing are desired.

Author Contributions: Methodology development, manuscript writing and checking, A.M.M.; project supervision, methodology development and manuscript checking,V.M.G.; project supervision, methodology development and manuscript checking, C.A.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the CNPq (Brazilian Ministry of Science and Technology), the CAPES Foundation (Brazilian Ministry of Education), UFMG and CEFET-MG, which was greatly appreciated.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DES	Discrete-Event Systems
TDES	Timed Discrete-Event Systems
TEG	Timed Event Graph
TFLP	Tropical Fractional Linear Programming
TLSO	Tropical Lexicographic Synchronization Optimization

Appendix A. Codes of Algorithms

In this section, we list the implemented codes of algorithms in a practical free language Scicoslab. Note that all the listed codes are in Scicoslab's language, that implements tropical algebra in its Max-Plus mode. Therefore, the bar symbol (/) implements the right residuation operation, the backslash symbol (\) implements the left residuation operation, the plus symbol (+) implements the oplus operation ($\oplus$), and the asterisk symbol (*) implements the otimes operation ($\otimes$).

Appendix A.1. Code of Algorithm Tropical Dual

The program's code that implements the Algorithm 1 TDual, in the free language Scicoslab, is listed below:

```
// Implementation of Algorithm 1:TDual
// Tropical Dual Function
function [vnew]=TDual( E, D, e, d )
  big=1000;
  itmax=10000;
  tol=#(0.0001);
  // initialize upperbound
  vprev=#(ones(size(E,2), 1)*big);
  k=0;
  // stop flags
  converge=%F;
  itok=%T;
  condi=%T;
  while (~converge & itok & condi)
    // Recursive Equation (14)
    vnew=(E\(D*vprev+d)) &
     (D\(E*vprev+e))&vprev;
    k=k+1;
    // test conditions (15)
    condi = and(e <= (D*vnew+d)) &
            and(d <= (E*vnew+e));
    converge=sum(vprev-vnew)<tol;
    itok = (k < itmax);
    vprev=vnew;
  end;
  if ~condi | ~itok
    vnew=[];
  end
  return [vnew];
endfunction
```

Appendix A.2. Code of Algorithm TFLP

The program's code that implements the Algorithm 2 TFLP, in the free language Scicoslab, is listed below:

```
// Implementation of Algorithm 2:TFLP
// Tropical Fractional Linear Programming
function [u,delta]=TFLP(w,alfa,f,Beta,R,r,S,s)
 n = size(R,1);
 m = size(R,2);
 epsn = full(%zeros(n,1));
 epsm = full(%zeros(1,m));
 // create the matrices according to (19)
 E = [ R r ; f Beta ];
 e = [ epsn ; %0 ];
 D = [ S s ; epsm %0];
 d = [ epsn; 0 ];
// calls the Dual Method to solve Max TLP in (19)
 [v] = TDual( E, D, e, d );
 // verify if the solution is found
 if isempty(v)
   u =[];
   delta = [];
  else
   // convert back to the original variable according to (20)
   y = v(1:$-1);
   z = v($);
   u = y*z;
   delta= (w*u+alfa)-(f*u+Beta);
  end
 return [u,delta];
endfunction
```

Appendix A.3. Code of Algorithm TLSO

The program's code that implements the Algorithm 3 TLSO, in the free language Scicoslab, is listed below:

```
// Implementation of Algorithm 3:TLSO
// Tropical Lexicographic Synchronization Optimization
function [u]=TLSO(A,B,L,lambda)
 M=star(A-lambda)*B;
  // maximum possible iterations
 P=size(L,1);
 p=1; // first iteration
 while (p<P)
  i(p) = L(p,1);
  j(p) = L(p,2);
  I(p,:)  = M(i,:(p,:),:);
  J(p,:)  = M(j(p,:),:);
  // iteration constraint Equation (30)
  w = I(p,:);
  f = J(p,:);
  R = I(p,:)+J(p,:);
```

```
S = I(p,:);
// prior constraints (31)
n = p-1;
while (n>0)
 R = [ R; J(n,:); I(n,:)+J(n,:)];
 S = [ S; delta(n)*J(n,:);I(n,:)];
 n=n-1;
end;
r = full(%zeros((p-1)*2+1,1));
s = full(%zeros((p-1)*2+1,1));
[u,delta(p)]=
  TFLP(w,alpha,f,Beta,R,r,S,s);
if isempty(u)
 L(p,:)  = [];
 P = size(L,1);
else
  ustar=u;
  p=p+1;
end;
end;
return [ustar];
endfunction
```

References

1. Baccelli, F.; Cohen, G.; Olsder, G.J.; Quadrat, J.P. *Synchronization and Linearity an Algebra for Discrete Event Systems*; John Wiley and Sons: New York, NY, USA, 1992.
2. Hardouin, L.; Cottenceau, B.; Shang, Y.; Raisch, J. Control and State Estimation for Max-Plus Linear Systems. *Found. Trends® Syst. Control* **2018**, *6*, 1–116. [CrossRef]
3. Pin, J.E. The influence of Imre Simon's work in the theory of automata, languages and semigroups. *Semigroup Forum* **2019**, *98*, 1–8. [CrossRef]
4. Amari, S.; Demongodin, I.; Loiseau, J. Control of linear min-plus systems under temporal constraints. In Proceedings of the 44th IEEE Conference on Decision and Control, Seville, Spain, 12–15 December 2005; pp. 7738–7743. [CrossRef]
5. Amari, S.; Demongodin, I.; Loiseau, J.; Jacques, J.J.; Martinez, C. Max-plus control design for temporal constraints meeting in timed event graphs. *IEEE Trans. Autom. Control.* **2012**, *57*, 462–467. 10.1109/TAC.2011.2164735. [CrossRef]
6. Atto, A.M.; Martinez, C.; Amari, S. Control of discrete event systems with respect to strict duration: Supervision of an industrial manufacturing plant. *Comput. Ind. Eng.* **2011**, *61*, 1149–1159. [CrossRef]
7. Kim, C.; Lee, T.E. Feedback control of cluster tools for regulating wafer delays. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 1189–1199. [CrossRef]
8. Gonçalves, V.M.; Maia, C.A.; Hardouin, L. On the Steady-State Control of Timed Event Graphs With Firing Date Constraints. *IEEE Trans. Autom. Control* **2016**, *61*, 2187–2202. [CrossRef]
9. Majdzik, P.; Seybold, L.; Witczak, M. A max-plus algebra predictive approach to a battery assembly system control. In Proceedings of the 2014 IEEE International Symposium on Intelligent Control (ISIC), Juan Les Pins, France, 8–10 October 2014; pp. 2202–2207. [CrossRef]
10. Gaubert, S.; Katz, R.; Sergeev, S. Tropical Linear-fractional programming and parametric mean payoff games. *J. Symb. Comp.* **2012**, *47*, 1447–1478. [CrossRef]
11. Gonçalves, V.M.; Maia, C.A.; Hardouin, L. On tropical fractional linear programming. *Linear Alg. App.* **2014**, *459*, 384–396. [CrossRef]
12. Zykina, A.V. A Lexicographic Optimization Algorithm. *Autom. Remote Control* **2004**, *65*, 363–368. [CrossRef]

13. Egmund, R.J.; Olsder, G.J. The (max,+) algebra applied to synchronization of traffic light processes. *WODES* **1998**, *26*, 451–456.
14. Dias, J.R.S.; Maia, C.A.; Lucena, V.F. A Computationally Efficient Method for Optimal Input-Flow Control of Timed-Event Graphs Ensuring a Given Production Rate. *J. Control Autom. Electr. Syst.* **2015**, *26*, 348–360. [CrossRef]
15. Butkovic, P.; Aminu, A. Introduction to max-linear programming. *IMA J. Manag. Math.* **2008**, *20*, 233–249. [CrossRef]
16. Butkovic, P.; MacCaig, M. On the integer max-linear programming problem. *Discret. Appl. Math.* **2014**, *162*, 128–141. [CrossRef]
17. De Schutter, B.; van den Boom, T.; Xu, J.; Farahani, S.S. Analysis and control of max-plus linear discrete-event systems: An introduction. *Discret. Event Dyn. Syst.* **2020**, *30*, 25–54. [CrossRef]
18. Komenda, J.; Lahaye, S.; Boimondb, J.L.; Boom, T. Max-plus algebra in the history of discrete event systems. *Annu. Rev. Control* **2018**, *45*, 240–249. [CrossRef]
19. Umer, M.; Hayat, U.; Abbas, F. An Efficient Algorithm for Nontrivial Eigenvectors in Max-Plus Algebra. *Symmetry* **2019**, *11*, 738. [CrossRef]
20. Umer, M.; Hayat, U.; Abbas, F.; Agarwal, A.; Kitanov, P. An Efficient Algorithm for Eigenvalue Problem of Latin Squares in a Bipartite Min-Max-Plus System. *Symmetry* **2020**, *12*, 311. [CrossRef]
21. Cassandras, C.G.; Lafourtune, S. *Introduction to Discrete Event Systems*, 2nd ed.; Springer: New York, NY, USA, 2008; pp. 53–57.
22. Blyth, T.; Janowitz, M. *Residuation Theory*; Pergamon Press: Oxford, UK, 2008; ISBN 9781483157146.
23. Gonçalves, V.M.; Maia, C.A.; Hardouin, L. On max-plus linear dynamical system theory: The regulation problem. *Elsevier Autom.* **2017**, *75*, 202–209. [CrossRef]
24. Gonçalves, V.M.; Maia, C.A.; Hardouin, L. Weak dual residuations applied to tropical linear equations. *Linear Alg. Its App.* **2014**, *445*, 69–84. [CrossRef]
25. Charnes, A.; Cooper, W.W. Programing with linear fractional functionals. *Nav. Res. Logist. Q.* **1962**, *9*, 181–186. [CrossRef]
26. Marotta, A.M.; Maia, C.A. Modeling and Optimization of Semaphore Networks Via Linear Max-Plus Model: Application to Synchronism and Flow Control in Crossings. In Proceedings of the 47th International Conference on Computer and Industrial Engineering, Lisboa, Portugal, 11–13 October 2017. Available on-line: https://www.dropbox.com/sh/98wen5smashdq4m/AACHGFgY7Tcj6LzObobWWeJca/CIE47_paper_300.pdf (accessed on 27 August 2020).