Firing rate optimization of cyclic timed event graphs by token allocations

Alessandro Giua, Aldo Piccaluga, Carla Seatzu
Dept. of Electrical and Electronic Engineering, University of Cagliari
Piazza d'Armi, 09123 Cagliari, Italy
Tel: +39 (70) 675-5892. Fax: +39 (70) 675-5900. Email: {giua,aldo,seatzu}@diee.unica.it.

Abstract

In this paper we deal with the problem of allocating a given number of tokens in a cyclic timed event graph (CTEG) so as to maximize the firing rate of the net. We propose three different approaches. The first one is a "greedy" incremental procedure that is computationally very efficient. The only drawback is that the convergence to the optimum is guaranteed only when the set of places where tokens can be allocated satisfies given constraints. The other two procedures involve the solution of a mixed integer linear programming problem. The first one needs the knowledge of the elementary circuits, thus it is convenient only for those classes of CTEG whose number of elementary circuits is roughly equal to the number of places, such as some kanban–systems. On the contrary, the second one enables one to overcome this difficulty, thus providing an efficient tool for the solution of allocation problems in complex manufacturing systems like job–shop systems.

Published as:
A. Giua, A. Piccaluga, C. Seatzu, "Firing Rate Optimization of Cyclic Timed Event Graphs by token allocations," *Automatica*, Vol. 38, No. 1, pp. 91–103, January 2002.

1 Introduction

In this paper we shall consider a particular class of Petri nets called *cyclic timed event graphs (CTEG)*. The main feature of this class of nets is that each place has only one input and one output transition. Moreover, in deterministic CTEG the steady state performance can be evaluated in terms of the *cycle time* of the net. We deal with the problem of allocating a given number of tokens in a CTEG so as to maximize its *firing rate* (i.e., the inverse of the cycle time). Note that both the initial marking *and* the firing rate are decision variables in this approach. This problem has a practical relevance: as an example, in the manufacturing domain it corresponds to determining the optimal allocation of a finite set of resources so as to maximize the throughput.

We start by reviewing the approach recently proposed by Panayiotou and Cassandras [15] to allocate a given number of resources step by step to a set of concurrent processes so as to maximize a given performance index. The first algorithm proposed by these authors is called *Incremental Optimization* (IO) and may be applied when at each step the optimal solution is unique. The second algorithm is called *Generalized Incremental Optimization* (GIO) and may be applied even if at each step the optimal solution is not unique but requires to keep track of all solutions that are optimal. The same authors also provided a necessary and sufficient condition (the performance index must be *smooth* [15]) under which the IO algorithm yields an optimal allocation. The class of processes considered was not formally specified, but it was suggested that one problem that falls into this class is that of allocating a fixed number of kanbans to the different stages of a kanban–system so as to maximize its throughput. Building on their result, we define a *generalized smoothness* property and show that this is a necessary and sufficient condition for the convergence of the GIO algorithm to the optimum.

We then focus our attention to CTEG, a class of Petri net that has often been used for modeling and analyzing manufacturing systems assuming a cyclic manufacturing of the parts, since it has been shown that choice–free job–shop, kanban–systems, and assembly systems, can be modeled using event graphs. One of the advantages of framing our results in the setting of CTEG, is that we are able to formally define the class of models to which our approach can be applied.

We propose three different procedures to solve the allocation problem in the context of CTEG.

We first show that in general the firing rate of a CTEG is not a generalized smooth performance index. However, when a restriction is posed on the set of places where the tokens can be allocated — we call this "Assumption (A)" — we show that the firing rate can be guaranteed to be generalized smooth, so that the incremental procedure of Panayiotou and Cassandras can be applied. We also provide a new algorithm, denoted as *Two–index Incremental Optimization* (TIO) algorithm, that for CTEG reveals to be computationally more efficient with respect to (wrt) the GIO algorithm. In fact, at each step, we provide a criterion to select only one solution among all the optimal ones. We firstly presented such a result in [7], but in this paper we extend the class of CTEG that verify Assumption (A).

In the second part of the paper we study the same allocation problem in a more general setting,

posing no restriction on the class of allocations considered [8]. Since we have shown that in the general case the performance index is not generalized smooth, other procedures than the incremental one are required. We derive two different approaches to solve the general optimal allocation problem.

- The first procedure involves the solution of a mixed integer linear programming problem (ILPP) that requires the enumeration of all elementary circuits. Thus, it is convenient for those classes of CTEG where the number of elementary circuits is not exponential in the size of the net, such as some kanban–systems where the number of elementary circuits is limited by the number of places.
- The second procedure is inspired by a result firstly proposed by Campos *et al.* [1] to determine the cycle time of an event graph, given the initial marking. It consists in a mixed ILPP that does not require the computation of the elementary circuits and whose constraint set only involves the computation of the incidence matrix, thus resulting to be efficient for all classes of CTEG.

The paper is structured as follows. In the following section we recall the main contributions on CTEG optimization reported in the literature. In Section 3 we provide some useful background on Petri nets and CTEG. In Section 4 we recall the two incremental optimization procedures formulated by Panayiotou and Cassandras, and we extend their definition of smoothness. In Section 5 we first show how to apply all these results to CTEG. Then, we formulate a new incremental optimization algorithm that reveals to be computationally very efficient. Necessary and sufficient conditions for the convergence to the optimum are also provided. A numerical example is finally discussed. In Section 6 we consider more general allocation problems and we present two different approaches, both involving the solution of a mixed ILPP. A numerical example is widely discussed at the end of the section, together with a comparison between the two latter procedures. Conclusions are finally drawn in Section 7.

2 Relevant literature

Many different optimization problems in the setting of CTEG have been studied in the literature starting from the works of Commoner *et al.* [3], Chretienne [2] and Magott [11] who were among the first researchers to relate the firing rate of a CTEG to the analysis of the elementary circuits.

Hillion and Proth [9] developed a procedure for the minimization of the work–in–progress cost in a manufacturing system, while maximizing the system throughput. They first formulated an integer linear programming problem (ILPP). Then, they developed an efficient heuristic algorithm to obtain a near–optimal solution.

Di Febbraro *et al.* [4] developed a two-objective optimization problem aiming at maximizing the system throughput, and at minimizing a function weighting the work–in–progress along with the space in the finite buffers. They introduced a further degree of freedom compared to Hillion

and Proth [9] by allowing the tokens to represent lots whose sizes have to be determined in order to optimize some performance objective.

Campos *et al.* [1] obtained upper and lower bounds on the steady-state performance of timed and stochastic event graphs. In particular, linear programming problems (LPPs) defined on the incidence matrix of the event graphs are used to compute attainable bounds for the throughput of transitions, defined as the average number of firings per unit time. They also demonstrated that these bounds depend on the initial marking and the mean values of the delays of transitions but not on the probability distribution functions (thus including both the deterministic and the stochastic cases).

Nakamura and Silva [14] dealt with the Minimum Initial Marking problem and provided an algorithm that is able to minimize a linear function of the initial markings under the constraint that the critical time does not exceed a given value. They proposed a heuristic approach structured in two phases: the first one consists of a greedy algorithm that iteratively uses a linear relaxation, in order to compute a "reasonable" initial solution; the second one involves a refinement of the solution via a tabu–search technique.

Laftit *et al.* [10] dealt with the problem of reaching a cycle time that is smaller than a given value, while minimizing an invariant linear criterion that is a linear combination of the number of tokens in the places at the initial time. In this work the authors provided both a heuristic algorithm and an exact algorithm for solving their optimization problem.

We finally mention the work of Yamada and Kataoka [18] where the authors made a comparison between different approaches for performance evaluation of timed event graphs. In particular, they considered the works of Magott [11], Morioka and Yamada [12], and Campos *et al.* [1], where the knowledge of cycle times is not required and three different LPPs have been formulated. Yamada and Kataoka in [18] demonstrated that the first one is the dual of the second one and the third is isomorphic to the second one.

3 Background

In this section we recall the formalism used in the paper. For more details on Petri nets and CTEG we refer to [6, 9, 10, 13].

A *Place/Transition net* (P/T net) is a structure $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$, where P is a set of n places; T is a set of m transitions; $\boldsymbol{Pre} : P \times T \to \mathbb{N}$ and $\boldsymbol{Post} : P \times T \to \mathbb{N}$ are the *pre*– and *post*– incidence functions that specify the arcs; $\boldsymbol{C} = \boldsymbol{Post} - \boldsymbol{Pre}$ is the incidence matrix. A *marking* is a vector $\boldsymbol{M} : P \to \mathbb{N}$ that assigns to each place of a P/T net a non–negative integer number of tokens, represented by black dots; we denote the marking of place p as $M(p)$. A P/T *system* or *net system* $\langle N, M_0 \rangle$ is a net N with an initial marking $\boldsymbol{M}_0$. A transition t is enabled at $\boldsymbol{M}$ if $\boldsymbol{M} \geq \boldsymbol{Pre}(\cdot, t)$ and may fire yielding the marking $\boldsymbol{M'} = \boldsymbol{M} + \boldsymbol{C}(\cdot, t)$.

A P/T net is called *ordinary* when all of its arc weights are 1's. An *event graph* is an ordinary

Petri net such that each place p has exactly one input transition and exactly one output transition. A net is *strongly connected* if there exists a directed path from any node in $P \cup T$ to every other node. Let us define an *elementary circuit* (or *elementary cycle*) of a net as a directed path that goes from one node back to the same node, while any other node is not repeated. In a strongly connected net it is easy to show that each node belongs to an elementary circuit, thus the name *cyclic nets* also used to denote this class. In a cyclic event graph the total number of tokens in any elementary circuit is invariant by transition firing and the net system is live if and only if every elementary circuit contains at least one token.

A deterministic Timed P/T net is a pair (N, τ), where $N = (P, T, \boldsymbol{Pre}, \boldsymbol{Post})$ is a standard P/T net, and $\tau : T \to \mathbb{R}_0^+$, called release delay, assigns a non–negative fixed firing duration to each transition. A transition with a release delay equal to 0 is said to be immediate. We consider an infinite–server semantics, i.e., we assume that each enabled transition can fire as many times as its enabling degree.

For deterministic timed cyclic event graphs we can compute, for any elementary circuit γ, the following ratio called the *cycle time* of the circuit: $c_\gamma = \dfrac{\mu_\gamma}{x_\gamma}$ where μ_γ denotes the *circuit release delay* (it is the sum of the release delays of all transitions belonging to γ), and x_γ denotes the number of tokens circulating in γ. We assume $\mu_\gamma > 0 \ \forall \ \gamma$.

Let Γ represents the set of elementary circuits of a cyclic event graph and $\hat{c} = \max_{\gamma \in \Gamma} c_\gamma$ be the *critical time*. Any $\gamma \in \Gamma$ such that $c_\gamma = \hat{c}$ is a *critical circuit*. These circuits are the ones that actually bind the speed of the system. Under an operational mode where transitions fire as soon as they have been enabled for a time equal to the release delay, the *firing rate* of each transition in steady state is given by $\varrho = \dfrac{1}{\hat{c}}$. As a consequence, if we want to increase the speed (i.e., the firing rate of the system), we have to add tokens to the critical circuits: adding tokens to any other circuit would be useless.

4 Incremental Optimization Algorithms and Generalized Smoothness Condition

The problem we deal with in this section is that of allocating a given number K of resources to q processes so as to maximize a given performance index J.

We represent an allocation by the q–dimensional vector

$$\boldsymbol{x} = \left[\begin{array}{ccccccc} x_1 & \cdots & x_i & \cdots & x_q \end{array} \right]^T, \tag{1}$$

where x_i denotes the number of resources allocated to the i–th process.

We will make use of the following definitions. Firstly, $\boldsymbol{e}_i = [0, \cdots, 0, 1, 0, \cdots, 0]$ is a q–dimensional vector with all its elements zero except the i–th element which is equal to one. Secondly,

$$\Delta J_i(\boldsymbol{x}) = J(\boldsymbol{x} + \boldsymbol{e}_i) - J(\boldsymbol{x}) \tag{2}$$

is the change in $J(\boldsymbol{x})$ due to the allocation of an additional resource to the i–th process wrt allocation $\boldsymbol{x}$. Finally, let, for all $k = 0, 1, \cdots$,

$$\mathcal{A}_k = \left\{ \boldsymbol{x} \in \mathbb{N}^q \mid \sum_{i=1}^{q} x_i = k,\ x_i \geq 0 \right\} \tag{3}$$

be the set of all possible allocations of k *available* resources to q processes.

Using the above definitions, the optimization problem is formally stated as:

$$(P1) \qquad \max_{\boldsymbol{x} \in \mathcal{A}_K} J(\boldsymbol{x}).$$

We denote

$$\mathcal{A}_k^* = \{ \boldsymbol{x}^* \in \mathcal{A}_k \mid J(\boldsymbol{x}^*) = \max_{\boldsymbol{x} \in \mathcal{A}_k} J(\boldsymbol{x}) \} \tag{4}$$

the set of optimal allocations of k resources.

Panayiotou and Cassandras defined the following condition on $J(\boldsymbol{x})$.

Definition 1 ([15], Smoothness Condition) *The performance index J verifies the smoothness condition iff $\forall\, k \geq 0$, $\forall\, \boldsymbol{x}^* \in \mathcal{A}_k^*$, $\forall \boldsymbol{x} \in \mathcal{A}_k$,*

$$\max_{i=1,\cdots,q} J(\boldsymbol{x}^* + \boldsymbol{e}_i) \geq \max_{i=1,\cdots,q} J(\boldsymbol{x} + \boldsymbol{e}_i). \tag{5}$$

■

Corollary 1 *The performance index J verifies the smoothness condition iff $\forall k \geq 0$, and $\forall \boldsymbol{x}^{(k)} \in \mathcal{A}_k^*$, there exists an infinite sequence of optimal allocations $\boldsymbol{x}^{(k+1)}, \cdots, \boldsymbol{x}^{(k+l)}, \cdots$, where $\boldsymbol{x}^{(k+l)} \in \mathcal{A}_{k+l}^*$, $\forall l \geq 1$ and $\boldsymbol{x}^{(k+l)}$ is obtained from $\boldsymbol{x}^{(k+l-1)}$ by allocating one additional resource to just one process.*

The smoothness condition ensures that any allocation that is optimal in $\mathcal{A}_k$ will become an optimal allocation in $\mathcal{A}_{k+1}$ by allocating one additional resource to some process.

Now, let us recall the *Incremental Optimization* (IO) Algorithm proposed by Panayiotou and Cassandras in [15].

Algorithm 1 ([15] IO Algorithm)
Let $\boldsymbol{x}_0 := [0, \cdots, 0]$;
for $k = 0, \cdots, K-1$ **do**
  begin
    $i_k^* := arg\ \max_{i=1,\cdots,q}\{\Delta J_i(\boldsymbol{x}_k)\}$;
    $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \boldsymbol{e}_{i_k^*}$;
  end.

After K steps, $\boldsymbol{x}_K$ is the optimal solution of (P1) under the assumptions stated by the following theorem proved in [15].

Theorem 1 ([15]) *For any $k = 0, 1, \cdots$, if $\boldsymbol{x}_k$ computed in accordance to algorithm 1 is unique, then it is the optimal solution to problem (P1) iff the performance index J is smooth.*

Panayiotou and Cassandras in [15] also provide a straightforward extension of the IO algorithm in the case of multiple solutions, denoted as *Generalized Incremental Optimization* (GIO) Algorithm.

Algorithm 2 ([15] GIO Algorithm)
Let $\boldsymbol{x}_0 := [0, \cdots, 0]$;
let $\mathcal{U}_0 = \{\boldsymbol{x}_0\}$;
for $k = 0, \cdots, K-1$ **do**
begin
$\Delta := \max_{i=1,\cdots,q,\ \boldsymbol{x}_k \in \mathcal{U}_k} \{\Delta J_i(\boldsymbol{x}_k)\}$;
$\mathcal{U}_{k+1} := \{\boldsymbol{x}_k + \boldsymbol{e}_i \mid \Delta J_i(\boldsymbol{x}_k) = \Delta,\ \boldsymbol{x}_k \in \mathcal{U}_k\}$;
end.

The extra cost incurred by this extension involves storing additional information.

Now, to state under which assumptions any allocation in the set $\mathcal{U}_k$ computed by the GIO algorithm is an optimal solution to (P1), we introduce a variation in the definition of smoothness based on the characterization of corollary 1.

Definition 2 (Generalized Smoothness Cond.) *The performance index J verifies the generalized smoothness condition iff $\forall k \geq 0$, there exists an allocation $\boldsymbol{x}^{(k)} \in \mathcal{A}_k^*$ and an infinite sequence of optimal allocations $\boldsymbol{x}^{(k+1)}, \cdots, \boldsymbol{x}^{(k+l)}, \cdots$, where $\boldsymbol{x}^{(k+l)} \in \mathcal{A}_{k+l}^*$, $\forall l \geq 1$ and $\boldsymbol{x}^{(k+l)}$ is obtained from $\boldsymbol{x}^{(k+l-1)}$ by allocating one additional resource to just one process.* ■

The generalized smoothness condition ensures that among all allocations that are optimal at a given step, there exists at least one that originates an infinite sequence of optimal allocations. Note that if a performance index J verifies the smoothness condition, then it also verifies the generalized smoothness condition.

We now prove that the GIO algorithm provides optimal solutions at each step if and only if J verifies the generalized smoothness condition. This was implied in [15] but not formally proved.

Theorem 2 *For any $k = 0, 1, \cdots$, each $\boldsymbol{x}_k \in \mathcal{U}_k$ computed in accordance to the GIO algorithm, is an optimal solution wrt J iff J verifies the generalized smoothness condition.*

Proof. (if) We observe that $\boldsymbol{x}_0$ is the only optimal allocation in $\mathcal{A}_0^*$. If J is generalized smooth, then $\boldsymbol{x}_0$ originates an infinite sequence of optimal allocations $\boldsymbol{x}^{(l)} \in \mathcal{A}_l^*$, $\forall l \geq 1$. Since $\boldsymbol{x}^{(l)} \in \mathcal{U}_l$, we have that $\mathcal{U}_l \subseteq \mathcal{A}_l^*$.

(only if) If the generalized smoothness condition is violated, then $\boldsymbol{x}_0$ cannot originate an infinite sequence of optimal allocations. This means that there exists a k such that $\mathcal{U}_k$ is not contained in $\mathcal{A}_k^*$. □

Note that both the smoothness and the generalized smoothness conditions could also be restated

for k only varying in a finite set, i.e., $k = 0, \cdots, K$. In fact, in all real applications the number of resources to be allocated is finite. Our choice originates from the consideration that in the rest of the paper we shall deal with performance indices whose smoothness and generalized smoothness is guaranteed $\forall\, k \geq 0$.

5 Firing rate optimization for CTEG: an incremental procedure

In this section, we want to apply to CTEG the results presented in the previous section.

5.1 Problem statement

Let us consider a timed cyclic event graph with n places, m transitions and ℓ elementary circuits. We associate to each elementary circuit γ an n dimensional vector $\boldsymbol{a}_\gamma$ of zeros and ones. In particular,

$$a_\gamma(i) = \begin{cases} 1 & \text{if } p_i \in \gamma, \\ 0 & \text{otherwise,} \end{cases} \tag{6}$$

thus $\boldsymbol{a}_\gamma^T \boldsymbol{M}$ is the number of tokens in γ and $\mu_\gamma / (\boldsymbol{a}_\gamma^T \boldsymbol{M})$ is the cycle time of circuit γ.

We assume that tokens may only be allocated within a given subset of places $P_a \subseteq P$, while no token can be allocated in the places in $P_r = P \setminus P_a$. We denote as n_a the cardinality of P_a, and $n_r = n - n_a$ the cardinality of P_r. For simplicity of presentation, we assume that the place labeling is such that a marking can be written as $\boldsymbol{M} = [\boldsymbol{M}_a^T \boldsymbol{M}_r^T]^T$, where $\boldsymbol{M}_a \in \mathbb{N}^{n_a}$ and $\boldsymbol{M}_r \in \mathbb{N}^{n_r}$.

In this section we shall deal with the problem of allocating a given number K of tokens to P_a so as to maximize the firing rate of the net:

$$\begin{cases} \max\ J = \min\limits_{\gamma \in \Gamma} \dfrac{\boldsymbol{a}_\gamma \boldsymbol{M}}{\mu_\gamma} \\ \\ s.t. \quad (a) \quad \boldsymbol{M}_r = \boldsymbol{0}_{n_r} \\ \qquad\ (b) \quad \boldsymbol{1}_{n_a}^T \boldsymbol{M}_a = K \end{cases} \tag{7}$$

where $\boldsymbol{0}_{n_r}$ is the n_r–dimensional null vector and $\boldsymbol{1}_{n_a}$ is the n_a–dimensional vector of ones.

Note that in the most general cases the above performance index J, i.e., the firing rate of the net, does not verify the generalized smoothness condition. More precisely, this depends on the choice of the set of places where tokens can be allocated.

5.2 Main assumption

We now provide a condition on P_a under which the performance index J of (7), i.e., the firing rate of the net, is generalized smooth, as we shall prove in the next subsection.

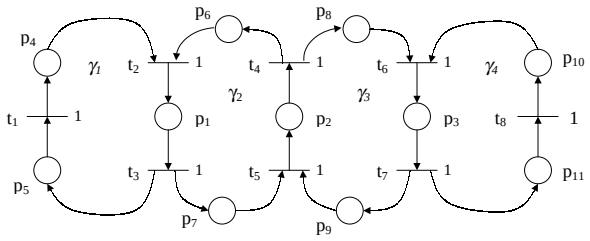

Figure 1: *Timed event graph of example 2.*

Assumption (A). If γ and γ' are two elementary circuits sharing a place in P_a, then they must have the same set of places in P_a, i.e.,

$$(\exists\, p \in P_a)\ p \in \gamma \cap \gamma' \implies \gamma \cap P_a = \gamma' \cap P_a.$$

Such an assumption implies that tokens can be allocated to places belonging to more than one elementary circuit but with the restriction that if two elementary circuits share a place in P_a, then they must have the same set of places in P_a.

In the rest of Section 5 we will consider allocation problems for which Assumption (A) hold, but let us first show, with a counterexample, that when this assumption is not verified the firing rate may fail to be a generalized smooth performance index.

Example 1 Let us consider the CTEG in figure 1 where the release delay of each transition is equal to one. There exist four elementary circuits:
$\gamma_1 = \{p_1, t_3, p_5, t_1, p_4, t_2\}$,
$\gamma_2 = \{p_1, t_3, p_7, t_5, p_2, t_4, p_6, t_2\}$,
$\gamma_3 = \{p_2, t_4, p_8, t_6, p_3, t_7, p_9, t_5\}$,
$\gamma_4 = \{p_3, t_7, p_{11}, t_8, p_{10}, t_6\}$,
and the circuit release delays are: $\mu_1 = \mu_4 = 3$, $\mu_2 = \mu_3 = 4$.

Let us assume that $P_a = \{p_1, p_2, p_3\}$. We denote as Γ_a the set of elementary circuits that contains at least one place in P_a. Thus, in this example $\Gamma_a = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$. It is immediate to show that Assumption (A) is not verified. In fact, place p_2 both belongs to γ_2 and γ_3, but $\gamma_2 \cap P_a = \{p_1, p_2\} \neq \{p_2, p_3\} = \gamma_3 \cap P_a$.

Now, let $K = 4$ be the maximum number of tokens available to be allocated. In Tab. 1 all optimal allocations for $k = 1 \cdots, 4$ are reported, together with the corresponding total number of tokens and the value of the firing rate J. From the table it can be noted that the optimal allocation for $k = 4$ cannot be obtained by simply adding one token to the allocation that is optimal at the previous step. Thus, we can conclude that J is not generalized smooth for the chosen set P_a.

On the contrary, if we choose $P_a = \{p_4, p_6, p_8, p_{10}\}$, the set Γ_a is the same as above, but there exists no place in P_a that belongs to more than one circuit in Γ_a and Assumption (A) is verified. In this case, as formally proved later, we can be sure that at least one of the solutions that are

k	$\boldsymbol{M}_k^T$	$J(\boldsymbol{M}_k)$
1	$\begin{bmatrix}1&0&0&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0&0&0\end{bmatrix}$	0
2	$\begin{bmatrix}1&0&1&0&0&0&0&0&0&0&0\end{bmatrix}$	1/4
3	$\begin{bmatrix}1&1&1&0&0&0&0&0&0&0&0\end{bmatrix}$	1/3
4	$\begin{bmatrix}2&0&2&0&0&0&0&0&0&0&0\end{bmatrix}$	1/2

Table 1: *Optimal allocations in the case of* $P_a = \{p_1, p_2, p_3\}$.

optimal when allocating k tokens ($k > 1$), can be obtained by at least one of the solutions that are optimal when the total number of tokens is $k - 1$. ■

When Assumption (A) holds, we can use a simplified notation to describe an allocation problem. We partition the set of places P_a into N subsets $P_a = P_1 \cup \cdots P_N$ and the set of ℓ elementary circuits Γ into $\Gamma = \Gamma_1 \cup \cdots \Gamma_N$. All places in the subset P_i belong to the same elementary circuits in Γ_i and there exists no place $p \in P_j \neq P_i$ belonging to a circuit in Γ_i. Note that to ensure that the net system is live we are assuming that each elementary circuit contains at least one place where tokens can be allocated.

Moreover, if we let

$$x_i = \sum_{p \in P_i} M(p), \qquad \text{and} \qquad \boldsymbol{x} = [x_1 \cdots x_N]^T,$$

we can define $\bar{c}_i = \max_{\gamma \in \Gamma_i} \left\{ \frac{\mu_\gamma}{x_i} \right\}$ as the maximum cycle time over all circuits in Γ_i, and $\bar{\mu}_i = \max_{\gamma \in \Gamma_i} \mu_\gamma$ as the maximum circuit release delay over all circuits in Γ_i.

In the following we will say that a subset of circuits Γ_i is *critical* for an allocation $\boldsymbol{x}$ if it contains at least one elementary circuit γ that is critical for $\boldsymbol{x}$.

By taking into account the above notation, the allocation problem (7) can be rewritten in the form of (P1), where $J(\boldsymbol{x})$ is the firing rate of the net, i.e., $J(\boldsymbol{x}) = 1/\hat{c}(\boldsymbol{x})$ and $\hat{c}(\boldsymbol{x})$ is the critical time for the allocation $\boldsymbol{x}$.

Note that in this case we assume that one token is initially allocated to each subset of places P_i so as to guarantee the net liveness, i.e., we assume that $\boldsymbol{x}_0 = [1, \cdots, 1]^T$ in the first step of the GIO algorithm.

5.3 Performance index smoothness

In this subsection we show that J is generalized smooth (but not necessarily smooth) if Assumption (A) holds. To prove this we first give an example showing that J is not smooth. Secondly, we define a new index $\boldsymbol{J}$ that will be proven smooth. Finally, we will show that the smoothness of $\boldsymbol{J}$ implies the generalized smoothness of J.

	Γ_1	Γ_2	Γ_3	
μ	3	4	4	$J(x)$
$x_1^{(1)}$	2	1	1	1/4
$x_1^{(2)}$	1	2	1	1/4
$x_1^{(3)}$	1	1	2	1/4

Table 2: *The results of the first step of the GIO algorithm.*

The following example shows that J is not smooth.

Example 2 Let us consider the CTEG in figure 1 and let $P_a = \{p_3, p_4, p_5, p_6\}$. It is immediate to observe that Assumption (A) is verified if we let $P_1 = \{p_4, p_5\}$, $P_2 = \{p_6\}$, $P_3 = \{p_3\}$, $\Gamma_1 = \{\gamma_1\}$, $\Gamma_2 = \{\gamma_2\}$, $\Gamma_3 = \{\gamma_3, \gamma_4\}$.

Now, to apply the GIO algorithm, we assume that one token is initially allocated to each set P_i, i.e., $\boldsymbol{x}_0 = [1\ 1\ 1]^T$. Then, we have to select the set of places P_i where to allocate an additional token, so as to maximize the performance index J. We compare the values obtained by allocating the additional token in the three sets. The result of such a comparison can be summarized in Tab. 2. As it can be noted, all allocations are characterized by the same critical time, thus the same J. Therefore, we can conclude that all solutions $\boldsymbol{x}_1^{(1)} = \boldsymbol{x}_0 + \boldsymbol{e}_1$, $\boldsymbol{x}_1^{(2)} = \boldsymbol{x}_0 + \boldsymbol{e}_2$ and $\boldsymbol{x}_1^{(3)} = \boldsymbol{x}_0 + \boldsymbol{e}_3$ are optimal wrt J. Nevertheless, we can observe that $\max_i\{J(\boldsymbol{x}_1^{(1)} + \boldsymbol{e}_i)\} = 1/4$, and $\max_i\{J(\boldsymbol{x}_1^{(2)} + \boldsymbol{e}_i)\} = \max_i\{J(\boldsymbol{x}_1^{(3)} + \boldsymbol{e}_i)\} = 1/2$. Thus we can conclude that $\boldsymbol{x}_1^{(2)}$ and $\boldsymbol{x}_1^{(3)}$ only generate optimal allocations at step 2.

Note that this follows from the fact that the first allocation $\boldsymbol{x}_1^{(1)}$ is characterized by two critical subsets of circuits (Γ_2 and Γ_3), i.e., both Γ_2 and Γ_3 contain critical circuits, so the addition of a *single* token cannot modify the critical time. ■

This example shows that if a given allocation is characterized by $\bar{n}$ critical subsets of circuits, it is necessary to add at least $\bar{n}$ tokens (one to each critical set) to improve the firing rate of the net. This conclusion reveals the requirement to distinguish among different allocations with the same firing rate but different number of critical sets. Thus, we introduce a new performance index $\boldsymbol{J} = [J'\ J'']^T$ consisting of two terms. The first one, J', is the firing rate, i.e., the performance index previously considered, while the second one is a measure of the number of Γ_i's containing critical circuits, i.e., $J'' = 1/\bar{n}(\boldsymbol{x})$, where $\bar{n}(\boldsymbol{x})$ denotes the number of critical subsets of circuits in the allocation $\boldsymbol{x}$. Note that we impose a lexicographic ordering on the performance index, i.e., $\boldsymbol{J} = \overline{\boldsymbol{J}}$ if $J' = \overline{J}'$ and $J'' = \overline{J}''$, $\boldsymbol{J} < \overline{\boldsymbol{J}}$ if $J' < \overline{J}'$ or $J' = \overline{J}'$ and $J'' < \overline{J}''$.

Example 3 Let us consider again the cyclic event graph in Fig. 1 and the allocations in Tab. 2 relative to the first step of the algorithm. The introduction of the new performance index enables us to immediately reject the allocation $\boldsymbol{x}_1^{(1)}$, being $J'(\boldsymbol{x}_1^{(1)}) = J'(\boldsymbol{x}_1^{(2)}) = J'(\boldsymbol{x}_1^{(3)}) = 1/4$ and $J''(\boldsymbol{x}_1^{(1)}) = 1/2 < J''(\boldsymbol{x}_1^{(2)}) = J''(\boldsymbol{x}_1^{(3)}) = 1$, thus $\boldsymbol{J}(\boldsymbol{x}_1^{(1)}) < \boldsymbol{J}(\boldsymbol{x}_1^{(2)}) = \boldsymbol{J}(\boldsymbol{x}_1^{(3)})$. Therefore, the only optimal solutions wrt $\boldsymbol{J}$ are $\boldsymbol{x}_1^{(2)}$ and $\boldsymbol{x}_1^{(3)}$. ■

Now, we characterize the set of optimal allocations wrt $\boldsymbol{J}$, showing that the removal of one token from any subset of circuits Γ_i makes Γ_i become a critical set. This result will be useful when proving the smoothness of $\boldsymbol{J}$.

Lemma 1 *Under Assumption (A), an allocation $\boldsymbol{x}^*$ is optimal wrt $\boldsymbol{J}$ iff:*

$$\frac{\bar{\mu}_i}{x_i^* - 1} \geq \hat{c}(\boldsymbol{x}^*) = \frac{1}{J'(\boldsymbol{x}^*)} \quad \forall\, i = 1, \cdots, N. \tag{8}$$

Proof. (if) Let us assume that the inequality (8) is verified for an allocation $\boldsymbol{x}^* \in \mathcal{A}_k$ and let $\boldsymbol{x} \in \mathcal{A}_k$ be a different allocation.

We first prove that $J'(\boldsymbol{x})$ cannot be greater than $J'(\boldsymbol{x}^*)$. Since $\boldsymbol{x}$, $\boldsymbol{x}^* \in \mathcal{A}_k$ and $\boldsymbol{x} \neq \boldsymbol{x}^*$, there exists a subset of circuits Γ_i such that $x_i < x_i^*$. This implies that

$$\frac{1}{J'(\boldsymbol{x})} \geq \bar{c}_i(\boldsymbol{x}) \geq \frac{\bar{\mu}_i}{x_i^* - 1} \geq \hat{c}(\boldsymbol{x}^*) = \frac{1}{J'(\boldsymbol{x}^*)}$$

thus $J'(\boldsymbol{x}^*) \geq J'(\boldsymbol{x})$.

Now, let us prove that if $J'(\boldsymbol{x}^*) = J'(\boldsymbol{x})$, it holds that $J''(\boldsymbol{x}^*) \geq J''(\boldsymbol{x})$. We first show that (8) implies that the number of tokens in each circuit for allocation $\boldsymbol{x}$ cannot be less than one wrt the corresponding number of tokens for allocation $\boldsymbol{x}^*$, i.e.,

$$x_i \geq x_i^* - 1 \quad \text{for all} \quad i = 1, \cdots, N. \tag{9}$$

This can be proved by contradiction. In fact, if we assume that $\exists\, \Gamma_i$ such that $x_i \leq x_i^* - 2$, then

$$\frac{1}{J'(\boldsymbol{x})} \geq \bar{c}_i(\boldsymbol{x}) \geq \frac{\bar{\mu}_i}{x_i^* - 2} > \frac{\bar{\mu}_i}{x_i^* - 1} \geq \frac{1}{J'(\boldsymbol{x}^*)}$$

and this contradicts the assumption that $J'(\boldsymbol{x}) = J'(\boldsymbol{x}^*)$.

Now, let $\mathcal{I}_0 = \{\Gamma_i \,|\, x_i = x_i^*\}$, $\mathcal{I}_- = \{\Gamma_i \,|\, x_i = x_i^* - 1\}$, $\mathcal{I}_+ = \{\Gamma_i \,|\, x_i > x_i^*\}$. By (8), in all subsets of circuits in $\mathcal{I}_-$ there is at least one critical circuit for the allocation $\boldsymbol{x}$ but not for $\boldsymbol{x}^*$, while all subsets of circuits in $\mathcal{I}_0$ that are critical for $\boldsymbol{x}^*$ are also critical for $\boldsymbol{x}$. Thus we can state that $\bar{n}(\boldsymbol{x}) = card(\mathcal{I}_0^c) + card(\mathcal{I}_-)$, and $\bar{n}(\boldsymbol{x}^*) \leq card(\mathcal{I}_0^c) + card(\mathcal{I}_+)$, where $\mathcal{I}_0^c = \{\Gamma_i \in \mathcal{I}_0 \,|\, \bar{c}_i(\boldsymbol{x}^*) = \hat{c}(\boldsymbol{x}^*)\}$. By virtue of equation (9) and by the assumption that $\sum_i x_i = \sum_i x_i^*$, it is easy to observe that $card(\mathcal{I}_+) \leq card(\mathcal{I}_-)$. We can conclude that $\bar{n}(\boldsymbol{x}) \geq \bar{n}(\boldsymbol{x}^*)$, thus $J''(\boldsymbol{x}^*) \geq J''(\boldsymbol{x})$, as we want to prove.

(only if) We prove this by contradiction. Let us assume that $\boldsymbol{x}^*$ is an optimal solution, but condition (8) is violated, i.e., there exists a subset of circuits Γ_l such that $\bar{\mu}_l/(x_l^* - 1) < \hat{c}(\boldsymbol{x}^*)$. Let Γ_m be a critical set for $\boldsymbol{x}^*$ and consider a new vector $\boldsymbol{x}$ obtained from $\boldsymbol{x}^*$ by moving one token from Γ_l to Γ_m. If Γ_m is the only critical set in $\boldsymbol{x}^*$, then $J'(\boldsymbol{x}) > J'(\boldsymbol{x}^*)$. If $\boldsymbol{x}^*$ has more than one critical set, the value of J' does not vary, i.e., $J'(\boldsymbol{x}) = J'(\boldsymbol{x}^*)$, but $J''(\boldsymbol{x}) > J''(\boldsymbol{x}^*)$. Thus, in both cases, $\boldsymbol{x}^*$ cannot be optimal. □

Theorem 3 *Under Assumption (A), the performance index* $\boldsymbol{J}$ *verifies the smoothness condition.*

Proof. Let $\boldsymbol{x}^*$ be an optimal solution for $\mathcal{A}_k$ and let $\boldsymbol{x}' \in \mathcal{A}_{k+1}$ be an allocation obtained from $\boldsymbol{x}^*$ adding a token to a subset of circuits Γ_l that is critical for $\boldsymbol{x}^*$.

We prove that $\boldsymbol{x}'$ is an optimal allocation for $\mathcal{A}_{k+1}$. Clearly, $J'(\boldsymbol{x}') \geq J'(\boldsymbol{x}^*)$ (the equality holds if $\boldsymbol{x}^*$ has more than one critical set). For all $\Gamma_i \in \Gamma \setminus \Gamma_l$ we have that

$$\frac{\bar{\mu}_i}{x'_i - 1} \equiv \frac{\bar{\mu}_i}{x^*_i - 1} \geq \frac{1}{J'(\boldsymbol{x}^*)} \geq \frac{1}{J'(\boldsymbol{x}')}. \tag{10}$$

The first inequality follows from the characterization given by (8) [Lemma 1] and the fact that $\boldsymbol{x}^*$ is optimal.

For the subset of circuits Γ_l we have that

$$\frac{\bar{\mu}_l}{x'_l - 1} \equiv \frac{\bar{\mu}_l}{x^*_l} = \frac{1}{J'(\boldsymbol{x}^*)} \geq \frac{1}{J'(\boldsymbol{x}')} \tag{11}$$

where the equality derives from the fact that Γ_l is a critical set for $\boldsymbol{x}^*$. Since $\boldsymbol{x}'$ verifies (10) and (11), by lemma 1 it follows that $\boldsymbol{x}'$ is optimal for $\mathcal{A}_{k+1}$. This shows that $\boldsymbol{J}$ verifies the smoothness condition. □

Note that even if $\boldsymbol{J}$ verifies the smoothness condition, the solution is not guaranteed to be unique, thus theorem 1 cannot be applied.

Corollary 2 *Under Assumption (A), the performance index* J'*, i.e., the firing rate, verifies the generalized smoothness condition.*

Proof. Let $\mathcal{A}^*_k$ and $\bar{\mathcal{A}}^*_k$ be the sets of allocations in $\mathcal{A}_k$ that are optimal wrt J' and $\boldsymbol{J}$, respectively.

Clearly, $\bar{\mathcal{A}}^*_k \subseteq \mathcal{A}^*_k \ \forall\, k \geq 0$. By virtue of corollary 1 all solutions in $\bar{\mathcal{A}}^*_k$ will produce an infinite sequence of optimal solutions wrt $\boldsymbol{J}$. This implies that there exist some $\boldsymbol{x}^* \in \mathcal{A}^*_k$ which originates an infinite sequence of optimal allocations wrt to J'. It follows that J' verifies the generalized smoothness condition. □

Note that from theorem 2 and corollary 2 it immediately follows that all solutions computed in accordance to the GIO algorithm are optimal wrt J'.

5.4 Two–index Optimization algorithm

In this subsection we propose a new IO algorithm, denoted *Two–index* Incremental Optimization (TIO) algorithm, which reveals to be computationally more convenient than the GIO algorithm. We will use the fact that $\boldsymbol{J}$ is smooth (as proved in the previous subsection) to show that the TIO algorithm is optimal.

The main improvement consists in the fact that while at each step the GIO algorithm must keep track of all allocations within the set $\mathcal{U}_k$, whose cardinality may greatly increase, the use

of the new performance index $\boldsymbol{J}$ allows us to neglect all those allocations that will be found non optimal at the following step(s). Moreover, as stated below, all the allocations that are optimal at a given step will converge to a common optimal one after a given number of steps: this property is exploited by the new algorithm to compute the final common allocation.

Property 1 *Let us consider a CTEG and let N be the number of subsets of places where tokens may be allocated. All the allocations that are optimal wrt to $\boldsymbol{J}$ at a given step k of the optimization algorithm, will converge to the same optimal one at the $(k + \bar{n}_k)$–th step, where $\bar{n}_k$ denotes the number of critical subsets of circuits in each optimal allocation at step k.*

Proof. Using condition 8 it is easy to show that:

- all subsets of circuits that are non critical for all optimal allocations in $\mathcal{A}_k^*$ contain the same number of tokens;
- the difference among the number of tokens in all other subsets can at most be equal to one.

Moreover, the firing rate of all allocations in $\mathcal{A}_k^*$ may only increase for the addition of $\bar{n}_k$ more tokens, one in each critical set. Thus, at the $(k + \bar{n}_k)$–th step all optimal allocations in $\mathcal{A}_k^*$ converge to the same optimal one. □

Now, let us provide the formulation of the Two–index IO algorithm.

Algorithm 3 (TIO Algorithm)
Let $\boldsymbol{x}_0 := [1, \cdots, 1]$;
let $k := 0$;
while $k < K$ **do**
  begin
    $\mathcal{I}_c := \{i \mid \Gamma_i$ *is critical for* $\boldsymbol{x}_k\}$;
    $\bar{n}_k := card\,(\mathcal{I}_c)$;
    if $k + \bar{n}_k > K$ **then** $k := K$
        else
          begin
          $\boldsymbol{x}_{k+\bar{n}_k} := \boldsymbol{x}_k + \sum_{i \in \mathcal{I}_c} \boldsymbol{e}_i$;
          $k := k + \bar{n}_k$;
          end
  end.

Note that if at a given step k we have $\bar{n}_k$ critical subsets and we still have to allocate a number $\bar{k} < \bar{n}_k$ tokens, we can be sure that no further improvement will occur in terms of firing rate and we exit without allocating the remaining tokens.

Property 2 *For any k, each $\boldsymbol{x}_k$ computed in accordance to the TIO algorithm, is an optimal solution wrt $\boldsymbol{J}$.*

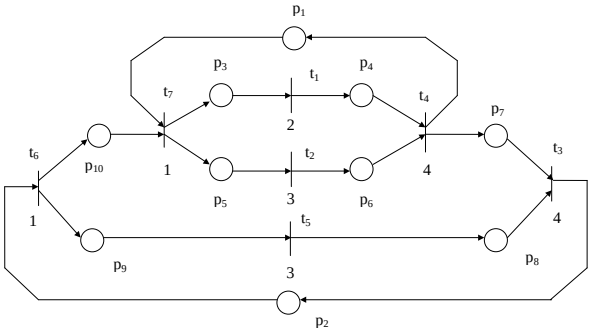

Figure 2: *Event graph model of the assembly system.*

Proof. We showed that the index $\boldsymbol{J}$ is smooth, thus (see the *if* part of the proof of Theorem 2) the GIO algorithm determines an infinite series of optimal allocations starting from $\boldsymbol{x}_0$. The modifications introduced in the new TIO Algorithm do not affect this property. In fact, when the solution is unique at each step the two algorithms provide the same allocation; in the case of multiple (say, $\bar{n}$) solutions, Property 1 ensures that arbitrarily choosing one is sufficient to recover the set of optimal solutions after $\bar{n}$ steps. □

Remark 4 We have observed that the TIO algorithm only keeps one optimal solution at each step, while the GIO algorithm keeps at each step a set of solutions $\mathcal{U}_k$. If the GIO algorithm starts from a single optimal allocation that has a number $\bar{n}$ of critical subsets of circuits, after $l < \bar{n}$ steps $\boldsymbol{J}$ will be maximized by all those allocations where a single token is assigned to l different critical subsets, and correspondingly the cardinality of $\mathcal{U}$ takes the value $\begin{pmatrix} \bar{n} \\ l \end{pmatrix}$. This value is maximal for $l = \lfloor \bar{n}/2 \rfloor$, that using Stirling approximation for large $\bar{n}$ is roughly equal to $\frac{2^{\bar{n}+1}}{\sqrt{2\pi\bar{n}}}$.

We conclude that the computational complexity of the GIO algorithm has an upper–bound given by $O(K\ 2^N\ N^{1/2})$, while the worst case computational complexity of the TIO algorithm is $O(K\ N)$ because the while loop is repeated at most K times and in each step we have to consider at most N subsets of circuits. ■

5.5 A numerical example: an assembly system

Let us consider an assembly system taken from the literature [6] whose Petri net model is sketched in figure 2.

It consists of five machines, $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$, $\mathcal{M}_4$ and $\mathcal{M}_5$ whose operational process is modeled by the firing of discrete timed transitions t_1, t_2, t_3, t_4 and t_5, respectively. Two principal types of operations are involved in this manufacturing system: *regular operations* and *assembly operations*. Regular operations (modeled by transitions t_1, t_2 and t_5) just transform a component of the intermediate product. Assembly operations (modeled by transitions t_3 and t_4) put com-

ponents together to obtain a more complex component of a final product or the final product itself. Note that this model uses transitions (t_6 and t_7) which do not represent operations but the beginning of the manufacturing of components which are required to assemble a more complex component or the final product. In this example there are two manufacturing levels, the primary one, performed by $\mathcal{M}_3$, leads to finite product, the secondary one, performed by $\mathcal{M}_4$, leads to semi–finished (in–working) product.

The markings of places p_1 and p_2 represent the number of assembly servers for t_4 and t_3 respectively. The marking of places p_3, p_5, and p_9 represent the availability of parts to be processed (raw materials), while the marking of places p_4, p_6, p_7 and p_8 represent the availability of semi–finished products.

The Petri net model in figure 2 is a strongly connected event graph with $n = |P| = 10$ and $m = |T| = 7$. There exist five elementary circuits:
$\gamma_1 = \{p_3, t_1, p_4, t_4, p_1, t_7\}$,
$\gamma_2 = \{p_5, t_2, p_6, t_4, p_1, t_7\}$,
$\gamma_3 = \{p_3, t_1, p_4, t_4, p_7, t_3, p_2, t_6, p_{10}, t_7\}$,
$\gamma_4 = \{p_5, t_2, p_6, t_4, p_7, t_3, p_2, t_6, p_{10}, t_7\}$,
$\gamma_5 = \{p_9, t_5, p_8, t_3, p_2, t_6\}$,
and the circuit release delays are: $\mu_{\gamma_1} = 6$, $\mu_{\gamma_2} = 7$, $\mu_{\gamma_3} = 13$, $\mu_{\gamma_4} = 14$ and $\mu_{\gamma_5} = 10$.

Let us assume that $P_a = \{p_1, p_2\}$. It is immediate to observe that Assumption (A) is verified if we let: $P_1 = \{p_1\}$, $\Gamma_1 = \{\gamma_1, \gamma_2\}$, $\bar{\mu}_1 = 7$, and $P_2 = \{p_2\}$, $\Gamma_2 = \{\gamma_3, \gamma_4, \gamma_5\}$, $\bar{\mu}_2 = 14$.

Now, to apply the GIO algorithm, we assume that one token is initially allocated to each set P_i, $i = 1, 2$, i.e., $\boldsymbol{x}_0 = [1\ 1]^T$. The problem we are dealing with is that of allocating $K = 9$ tokens in P_a so as to maximize the firing rate of the net. The optimal solution of this problem is $\boldsymbol{x}_7 = [3\ 6]^T$ and $J'^* = 3/7$. All the intermediate steps of the GIO algorithm are summarized in Tab. 3, where $\Delta J'_i$ denotes the increment of J' obtained adding a token to P_i.

On the contrary, the TIO algorithm allows us to keep only one solution at each step and to reduce the number of steps. Thus, in this example, only one allocation for $k = 2, 5$ needs to be taken into account and the allocations relative to $k = 1, 4, 7$ need not to be computed.

6 More general allocation problems

Now, let us consider more general allocation problems. More precisely, we remove Assumption (A) and we also assume that the allocation must satisfy a given set of s' linear inequalities each one of the form $\boldsymbol{g}^T \boldsymbol{M}_a \leq k$. Finally, we also assume that the initial marking of places where tokens cannot be allocated may be different from zero.

k	$\mathbf{x}_k^T$	$J'(\mathbf{x}_k)$	$J''(\mathbf{x}_k)$	$\Delta J'_1(\mathbf{x}_k)$	$\Delta J'_2(\mathbf{x}_k)$
0	[1 1]	1/14	1	0	5/14
1	[1 2]	1/7	1/2	0	0
2	[2 2]	1/7	1	0	1/14
	[1 3]			1/14	0
3	[2 3]	3/14	1	1/14	0
4	[2 4]	2/7	1	0	0
5	[3 4]	2/7	1/2	0	1/14
	[2 5]			1/14	0
6	[3 5]	5/14	1	0	1/14
7	[3 6]	3/7	1	0	0

Table 3: *The results of the GIO algorithm.*

Thus, in this section we shall deal with the following optimization problem:

$$\begin{cases} \max J = \min\limits_{\gamma \in \Gamma} \dfrac{\boldsymbol{a}_\gamma \boldsymbol{M}}{\mu_\gamma} \\ \\ s.t. \quad (a') \quad \boldsymbol{M}_r = \boldsymbol{M}_{r,0} \\ \qquad\quad (b') \quad \boldsymbol{G}\boldsymbol{M}_a \leq \boldsymbol{k} \end{cases} \tag{12}$$

where $\boldsymbol{M}_{r,0} \in \mathbb{N}^{n_r}$, $\boldsymbol{G} \in \mathbb{Z}^{s' \times n_a}$, and $\boldsymbol{k} \in \mathbb{Z}^{s'}$ are given, and $\boldsymbol{a}_\gamma$ is defined as in equation (6). Constraints (a') express the fact that the marking of all places in P_r is assigned. Each equation in (b') may either express an upper/lower bound on the number of tokens in a place $p \in P_a$, or an upper/lower bound on the number of tokens in a circuit $\gamma \in \Gamma_a$ or in a generic subset of places in P_a.

In the following we denote as Γ_a the set of elementary circuits that contain at least a place in P_a, i.e., $\Gamma_a = \{\gamma \in \Gamma \mid \gamma \cap P_a \neq \emptyset\}$.

Generalizing, our optimization problem can be formally written as a nonlinear integer programming problem of the form:

$$\begin{cases} \max J = \min\limits_{\gamma \in \Gamma} \dfrac{\boldsymbol{a}_\gamma \boldsymbol{M}}{\mu_\gamma} \\ \\ s.t. \quad \boldsymbol{A}\boldsymbol{M} \leq \boldsymbol{b} \end{cases} \tag{13}$$

where $\boldsymbol{M} \in \mathbb{N}^n$ is the unknown variable, $\boldsymbol{A} \in \mathbb{Z}^{s \times n}$, and $\boldsymbol{b} \in \mathbb{Z}^s$ are given.

Note that the allocation problem considered in the previous section is a special case of problem (13).

In the following we present two different solutions to the allocation problem (13) whose validity does not require Assumption (A).

6.1 First procedure

The first procedure we propose, involves the solution of a mixed ILPP and is based on the knowledge of the elementary circuits. As it is well known, such an assumption is often unrealistic, thus making it convenient only for those classes of CTEG where the number of elementary circuits does not increase exponentially with the size of the net, such as kanban–systems where the number of elementary circuits is limited by the number of places.

The mixed ILPP formulation origins from the following remark.

Remark 5 *Consider the two programming problems:*

$$\begin{cases} \max J_I = \min\limits_{i=1,\cdots,p} \{\boldsymbol{c}_i^T \boldsymbol{x}\} \\ s.t. \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b} \end{cases} \tag{14}$$

with integer variables $\boldsymbol{x} \in \mathbb{N}^N$ *and*

$$\begin{cases} \max J_{II} = \beta \\ s.t. \quad \boldsymbol{c}_i^T \boldsymbol{y} - \beta \geq 0, \quad i = 1, \cdots, p, \\ \qquad \boldsymbol{A}\boldsymbol{y} \leq \boldsymbol{b} \end{cases} \tag{15}$$

with integer variables $\boldsymbol{y} \in \mathbb{N}^N$ *and real variable* $\beta \in \mathbb{R}^+$. *Here* $\boldsymbol{c}_i \in \mathbb{R}_0^N$, $i = 1, \cdots, p$, $\boldsymbol{A} \in \mathbb{R}^{s \times N}$, *and* $\boldsymbol{b} \in \mathbb{R}^s$ *are given.*

Then $\boldsymbol{x}^*$ *is an optimal solution of (14) with performance index* J_I^* *iff* $(\boldsymbol{x}^*, J_I^*)$ *is an optimal solution of (15).*

Note that the above result also holds when $\boldsymbol{x}$ and $\boldsymbol{y}$ are real valued variables.

Proposition 1 *The optimal solution* $(\boldsymbol{M}^*, \beta^*)$ *of the mixed ILPP:*

$$\begin{cases} \max \beta \\ s.t. \quad \boldsymbol{a}_\gamma^T \boldsymbol{M}/\mu_\gamma - \beta \geq 0, \quad \gamma \in \Gamma, \\ \qquad \boldsymbol{A}\boldsymbol{M} \leq \boldsymbol{b} \end{cases} \tag{16}$$

with variables $\boldsymbol{M} \in \mathbb{N}^n$, $\beta \in \mathbb{R}^+$, *provides the optimal solution* $\boldsymbol{M}^*$ *and the corresponding optimal performance index value* $J^* = \beta^*$ *of the nonlinear integer programming problem (13).*

Proof. It immediately follows from remark 5. □

Using this new formulation, we have to solve a simpler mixed ILPP with $n+1$ variables and $\ell + s$ constraints, where ℓ denotes the number of elementary circuits. Obviously, the main drawback of the above procedure lies in the requirement of computing all elementary circuits.

6.2 Second procedure

In this subsection we propose another solution to our allocation problem (13) that still involves the solution of a mixed ILPP, but presents a significant advantage with respect to the previous

one: *it does not require the computation of the elementary circuits.*

It is inspired by a result firstly proposed by Campos *et al.* [1], where the authors dealt with the problem of determining the cycle time of an event graph, given the initial marking $\boldsymbol{M}_0$:

$$\begin{cases} \max \dfrac{\boldsymbol{a}_\gamma^T \boldsymbol{Pre}\, \boldsymbol{\theta}}{\boldsymbol{a}_\gamma^T \boldsymbol{M}_0} \\ s.t. \quad \boldsymbol{a}_\gamma^T \boldsymbol{C} = \boldsymbol{0} \\ \qquad \boldsymbol{a}_\gamma \geq \boldsymbol{0} \end{cases} \tag{17}$$

where $\boldsymbol{Pre}^T \boldsymbol{a}_\gamma$ is the characteristic vector of the set of transitions that belong to circuit γ, and $\boldsymbol{\theta} \in \mathbb{N}^m$ is the vector containing all release delays of timed transitions (recall that $m = |T|$). The two constraints in problem (17) force the vector of decision variables $\boldsymbol{a}_\gamma$ to be a P–invariant, i.e., $\boldsymbol{a}_\gamma$ represents (but for a scalar factor) the characteristic vector of the places along a circuit.

In [1] it was also shown that the same optimal solution of (17) can also be obtained by means of the following LPP:

$$\begin{cases} \max \boldsymbol{a}_\gamma^T \boldsymbol{Pre}\, \boldsymbol{\theta} \\ s.t. \quad \boldsymbol{a}_\gamma^T \boldsymbol{C} = \boldsymbol{0} \\ \qquad \boldsymbol{a}_\gamma^T \boldsymbol{M}_0 = 1 \\ \qquad \boldsymbol{a}_\gamma \geq \boldsymbol{0}. \end{cases} \tag{18}$$

whose dual problem is:

$$\begin{cases} \min v \\ s.t. \quad \boldsymbol{Cz} + v\boldsymbol{M}_0 \geq \boldsymbol{Pre}\, \boldsymbol{\theta} \end{cases} \tag{19}$$

where the decision variables are $v \in \mathbb{R}^+$ and $\boldsymbol{z} \in \mathbb{R}^m$: the optimal value of v is the cycle time and the unconstrained vector $\boldsymbol{z}$ has no physical meaning.

Now, let us consider problem (19). This problem can be easily converted into the problem of determining the optimal firing rate of the net, given the initial marking. For this purpose we only need to replace v with its inverse $\beta = 1/v$, thus obtaining:

$$\begin{cases} \max \beta \\ s.t. \quad \boldsymbol{C}(\beta \boldsymbol{z}) + \boldsymbol{M}_0 \geq \boldsymbol{Pre}\, \boldsymbol{\theta}\ \beta \end{cases} \tag{20}$$

where $\beta \in \mathbb{R}^+$, and $\beta \boldsymbol{z} \in \mathbb{R}^m$, i.e.,

$$\begin{cases} \max \beta \\ s.t. \quad \boldsymbol{Cy} - \boldsymbol{Pre}\, \boldsymbol{\theta}\ \beta \geq -\boldsymbol{M}_0 \end{cases} \tag{21}$$

where $\boldsymbol{y} \in \mathbb{R}^m$ and $\beta \in \mathbb{R}^+$ are the new decision variables.

Finally, if we assume, as in problem (13), that $\boldsymbol{M}_0$ is not known but must satisfy a set of given inequalities, we have the following result.

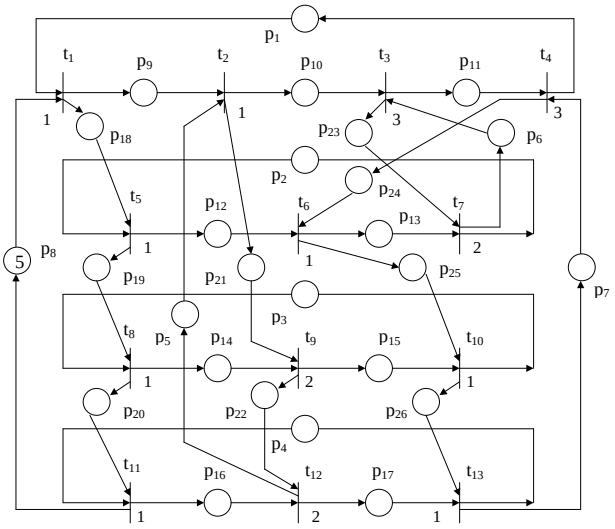

Figure 3: *Event graph model of the job–shop system.*

Proposition 2 *The optimal solution* $(\boldsymbol{M}^*, \beta^*, \boldsymbol{y}^*)$ *of the mixed ILPP:*

$$\begin{cases} \max \beta \\ s.t. \quad \boldsymbol{Cy} - \boldsymbol{Pre}\,\boldsymbol{\theta}\ \beta + \boldsymbol{M} \geq \boldsymbol{0} \\ \qquad \boldsymbol{AM} \leq \boldsymbol{b} \end{cases} \tag{22}$$

with variables $\boldsymbol{M} \in \mathbb{N}^n$, $\beta \in \mathbb{R}^+$, *and* $\boldsymbol{y} \in \mathbb{R}^m$, *provides the optimal solution* $\boldsymbol{M}^*$ *and the corresponding optimal performance index value* $J^* = \beta^*$ *of the nonlinear integer programming problem (13).*

Proof. We prove this by contradition. Let us assume that the optimal solution of problem (22) is $(\tilde{\boldsymbol{M}}, \tilde{\beta}, \tilde{\boldsymbol{y}})$ and let $(\boldsymbol{M}^*, \beta^*)$ be the optimal solution of (13) with $\beta^* > \tilde{\beta}$. Now the results of [1] ensure that if we solve problem (21) with $\boldsymbol{M}_0 = \boldsymbol{M}^*$ we get an optimal value of the firing rate that is equal to β^*, and let $\boldsymbol{y}^*$ be the corresponding optimal value of $\boldsymbol{y}$. This implies that $(\boldsymbol{M}^*, \beta^*, \boldsymbol{y}^*)$ satisfy all constraints in (22) and by assumption $\beta^* > \tilde{\beta}$. But this contradicts the hypothesis that $(\tilde{\boldsymbol{M}}, \tilde{\beta}, \tilde{\boldsymbol{y}})$ is an optimal solution of (22). □

In this way our optimization problem has been reduced to the solution of a mixed ILPP with $n + m + 1$ variables and $n + s$ constraints.

6.3 A numerical example: a job–shop system

In this subsection we deal with an example taken from the literature [16]. We consider a job–shop composed of four machines $\mathcal{M}_1$, $\mathcal{M}_2$, $\mathcal{M}_3$ and $\mathcal{M}_4$, which can manufacture three products denoted by $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$. The production mix is 25%, 25%, 50% for $\mathcal{R}_1$, $\mathcal{R}_2$ and $\mathcal{R}_3$, respectively. The production processes of the products and the corresponding circuits (the

circuit for $\mathcal{R}_3$ is repeated) are:

$$\begin{array}{lll} \mathcal{R}_1: & (\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3, \mathcal{M}_4) & \{p_1, t_1, p_9, t_2, p_{10}, t_3, p_{11}, t_4\} \\ \mathcal{R}_2: & (\mathcal{M}_1, \mathcal{M}_4, \mathcal{M}_3) & \{p_2, t_5, p_{12}, t_6, p_{13}, t_7\} \\ \mathcal{R}_3: & (\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_4) & \{p_3, t_8, p_{14}, t_9, p_{15}, t_{10}\} \\ & & \{p_4, t_{11}, p_{16}, t_{12}, p_{17}, t_{13}\}. \end{array}$$

Here the number of tokens in each product circuit represents the number of available pallets for that product.

The fixed sequencing of the part types on the machines and the corresponding circuits are:

$$\begin{array}{lll} \mathcal{M}_1: & (\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_3) & \{p_8, t_1, p_{18}, t_5, p_{19}, t_8, p_{20}, t_{11}\} \\ \mathcal{M}_2: & (\mathcal{R}_1, \mathcal{R}_3, \mathcal{R}_3) & \{p_5, t_2, p_{21}, t_9, p_{22}, t_{12}\} \\ \mathcal{M}_3: & (\mathcal{R}_1, \mathcal{R}_2) & \{p_6, t_3, p_{23}, t_7\} \\ \mathcal{M}_4: & (\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \mathcal{R}_3) & \{p_7, t_4, p_{24}, t_6, p_{25}, t_{10}, p_{26}, t_{13}\}. \end{array}$$

Here the number of tokens in each machine circuit represents the number of available servers for that machine.

The event graph modeling this system, sketched in figure 3, has $n = 26$ places, $m = 13$ transitions, and $\ell = 76$ elementary circuits. We assume that $P_a = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7\}$ where the tokens in p_1, p_2, p_3 and p_4 represent the free pallets that must be optimally allocated, while the tokens in p_5, p_6 and p_7 represent the servers that need to be optimally distributed between machines $\mathcal{M}_2$, $\mathcal{M}_3$ and $\mathcal{M}_4$. $P_r = P \setminus P_a = \{p_8, \cdots, p_{26}\}$ and we assume that: (a) the marking of place p_8 is 5 (i.e., $\mathcal{M}_1$ is the only machine that has a fixed number of servers, that is equal to 5); (b) the marking of all other places in P_r is zero (i.e., no part is initially in the job-shop). Release delays of transitions are shown in figure 3, as well as the marking of all places in P_r.

Now, let us consider the following optimization problem:

$$\begin{cases} \max J = \min\limits_{\gamma \in \Gamma} \dfrac{\boldsymbol{a}_\gamma \boldsymbol{M}}{\mu_\gamma} \\ s.t. \quad M(p_1) + M(p_2) + M(p_3) + M(p_4) \leq k_1 \\ \qquad M(p_5) + M(p_6) + M(p_7) \leq k_2 \\ \qquad M(p_8) = 5 \\ \qquad M(p_i) = 0, \quad i = 9, \cdots, 26, \end{cases}$$

and let $k_1 = 100$ and $k_2 = 20$, i.e., we want to determine the optimal token allocation when the number of pallets is equal to 100, machine $\mathcal{M}_1$ has 5 servers and the global number of servers available to machines $\mathcal{M}_2$, $\mathcal{M}_3$ and $\mathcal{M}_4$ is equal to 20.

Now, problems of the form (16) and (22) can be immediately formulated. In accordance with the previous notation we have $n_r = 19$ and $s' = 2$. Thus, in the first case the number of variables is equal to 27 and the number of constraints is equal to $76 + 19 + 2 = 97$. In the second case, there are $13 + 26 + 1 = 40$ variables and $26 + 19 + 2 = 47$ constraints. As expected, both procedures determine the same optimal firing rate $J^* = 1$, while the optimal allocations

are different. The optimal token allocation computed using the first procedure is $M(p_1) = 6$, $M(p_2) = M(p_3) = M(p_4) = 10$, $M(p_5) = 5$, $M(p_6) = 9$, $M(p_7) = 6$, while the optimal token allocation computed using the second procedure is has the same values except for $M(p_3) = 74$. In this case, the second procedure provides a solution that allocates to machine $\mathcal{M}_3$ a number of pallets significantly greater than that computed with the first procedure. This is not a drawback of the second approach but it is due to the fact that when more than one optimal solution exists, the ILPP solver just stops when the first one is found.

It may be useful to have a multicriterion to select just one among all the allocations that provide the same value of the firing rate. A possibility is that of introducing an additional term to the performance index in both problems (16) and (22) so as to select, among all the solutions characterized by the same optimal value of β, those that also minimize the total number of tokens in the net. As an example, the performance index β can be replaced by $\beta - w \cdot \boldsymbol{c}^T \boldsymbol{M}$, where w is a positive number much smaller than one, so as to maintain the maximization of β as the prior requirement and $\boldsymbol{c} : P \to \mathbb{R}_0^+$ is the cost vector of tokens. If we introduce this new performance index, choosing $w = 10^{-3}$ and $c(p) = 1$ for all $p \in P$, the two procedures find the same optimal allocation, that is the allocation previously determined by the first procedure.

Furthermore, let us observe that in the particular case examined, the bottleneck is due to the total number of servers in the machines, and cannot be eliminated by increasing the number of pallets, i.e., k_1. The only way to increase the firing rate is that of increasing the number of machine servers, i.e., k_2.

Finally, we observe that in the elementary circuit $\{p_8, t_1, p_{18}, t_5, p_{19}, t_8, p_{20}, t_{11}\}$ that corresponds to machine $\mathcal{M}_1$, all places belong to P_r. Therefore its cycle time (equal to 1.25) cannot be changed by the addition of more tokens, thus introducing an upper bound on the maximum firing rate obtainable by increasing both k_1 and k_2.

7 Conclusions

In this paper we have dealt with deterministic timed cyclic event graphs. We have discussed the problem of allocating a given number of tokens in a CTEG so as to maximize the firing rate of the net.

In the first part of the paper we have proposed an incremental algorithm that reveals to be very efficient both in terms of computational time and memory requirements (only one solution needs to be kept at each step). The only drawback of this procedure is that the set of places where tokens can be allocated, has to verify particular conditions so as to guarantee the convergence to the optimum.

More general allocation problems have been studied in the second part of the paper: the assumptions on the set of places where tokens can be allocated have been removed, and linear constraints on the marking of the net may also be taken into account. The novel contribution consists in the formulation of two mixed integer linear programming problems. The first one

needs the knowledge of the elementary circuits, thus making it unpractical for some classes of CTEG. The second one bypasses this difficulty and reveals to be efficient for analyzing complex manufacturing systems such as job–shops.

References

[1] Campos, J., G. Chiola, J.M. Colom, and M. Silva, "Properties and Performance Bounds for Timed Marked Graphs," *IEEE Trans. on Circuits and Systems*, Vol. 39, N. 5, pp. 386–401, 1992.

[2] Chretienne, P., "Timed Petri nets," *Ph.D. Thesis*, University of Paris VI, Paris, France, 1983 (in French).

[3] Commoner, F., A. Holt, S. Even and A. Pnueli, "Marked directed graphs," *J. of Computer and System Science*, Vol. 5, N. 5, 1971.

[4] Di Febbraro, A., R. Minciardi and S. Sacone, "Deterministic Timed Event Graphs for Performance Optimization of Cyclic Manufacturing Processes," *IEEE Trans. on Robotics and Automation*, Vol. 13, N. 2, pp. 169–181, 1997.

[5] Di Mascolo, M., Y. Frein, Y. Dallery and R. David, "A unified modeling of Kanban systems using Petri nets," *Int. J. of Flexible Manufacturing Systems*, Vol. 3, pp. 275–307, 1991.

[6] Di Cesare, F., G. Harhalakis, J.M. Proth, M. Silva and F.B. Vernadat, *Practice of Petri nets in manufacturing*, Chapman and Hall, 1993.

[7] Giua, A., A. Piccaluga, C. Seatzu, "Incremental optimization of timed cyclic event graphs," *Proc. 2000 IEEE Int. Conf. on Robotics and Automation* (San Francisco, California), pp. 2211-2216, April 2000.

[8] Giua, A., A. Piccaluga, C. Seatzu, "Optimal token allocation in timed cyclic event graphs," *Proc. 5th Int. Work. on Discrete Event Systems* (Ghent, Belgium), pp. 209-218, August 2000.

[9] Hillion, H.P. and J.M. Proth, "Performance evaluation of job–shop systems using timed event–graphs," *IEEE Trans. on Automatic Control*, Vol. 34, N. 1, pp. 3–9, 1989.

[10] Laftit, S., J.M. Proth and X. Xie, "Optimization of invariant criteria for event graph," *IEEE Trans. on Automatic Control*, Vol. 37, N. 5, pp. 547–555, 1992.

[11] Magott, J., "Performance evaluation of concurrent systems using Petri nets," *Inform. Proc. Lett.*, Vol. 18, pp. 7–13, 1984.

[12] Morioka, S. and T. Yamada, "Performance evaluation of marked graphs by linear programming," *Int. J. of Systems Science*, Vol. 22, pp. 1541–1552, 1991.

[13] Murata, T., "Petri nets: properties, analysis and applications," *Proc. IEEE*, Vol. Proc. 77, N. 4, pp. 541–580, April 1989.

[14] Nakamura, M. and M. Silva, "An iterative linear relaxation and tabu search approach to minimum initial marking problems of timed marked graphs," *Proc. 1999 European Control Conference*, (Karlsruhe, Germany), August–September 1999.

[15] Panayiotou, C.G. and C.G. Cassandras, "Optimization of kanban–based manufacturing systems," *Automatica*, Vol. 35, N. 9, pp. 1521–1533, September 1999.

[16] Proth J.M., N. Sauer and X. Xie, "Optimization of the number of trasportation devices in flexible manufacturing systems using event graphs," *IEEE Trans. Ind. Elect.*, Vol. 44, N. 3, pp. 298–306, 1997.

[17] Sugimori, Y., K. Kusunoki, F. Cho and S. Uchikawa, "Toyota production systems materialization of Just–in–Time and research–for–human systems," *Int. J. of Production Research*, Vol. 15, N. 6, pp. 553–564, 1977.

[18] Yamada, T. and S. Kataoka, "On Some LP Problems for Performance Evaluation of Timed Marked Graphs ," *IEEE Trans. on Automatic Control*, Vol. 39, N. 3, pp. 696–698, 1994.