Proceedings of the 2000 IEEE
International Conference on Robotics & Automation
San Francisco, CA • April 2000

Incremental Optimization of Cyclic Timed Event Graphs

Alessandro Giua, Aldo Piccaluga, Carla Seatzu
Department 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, so as to maximize the firing rate of a cyclic event graph with deterministic transition firing delays. We propose a new incremental algorithm that is inspired by the algorithm formulated by Panayiotou and Cassandras for the case of kanban systems. The algorithm can be applied to a special class of nets in which tokens are allocated to places that belong to only one circuit: this class is powerful enough to model kanban systems. We provide necessary and sufficient conditions for the convergence to the optimum also in the case of multiple solutions.

1 Introduction

Cyclic timed event-graphs (CTEG) are a special class of timed ordinary Petri nets. They are often 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.

Marked graphs may be either stochastic or deterministic. The most important results are in the deterministic cyclic production case. In fact, in this case it is possible to evaluate the steady state performance of the net in terms of its critical time. This means that in steady state conditions, at any point of the net, a number of tokens (equal to the inverse of the critical time), pass through that point during each unit of period of time (on the average). Note that, such a performance evaluation can be done assuming that all the elementary circuits are known. Some simple algorithms have been described by Laftit *et al.* in [4] to obtain all the elementary circuits of a cyclic event graph.

In this paper we deal with the problem of allocating a given number of tokens to a CTEG, so as to maximize the firing rate of the net. We propose a new algorithm that is inspired by the previous ones recently formulated by Panayiotou and Cassandras in [6] and denoted as *Incremental Optimization* (IO) algorithm and *Generalized Incremental Optimization* (GIO) algorithm, in the case of unique and multiple solutions, respectively. In [6] the authors focused their attention on kanban systems and formulated efficient algorithms to allocate a fixed number of kanban to a number of stages so as to maximize the system throughput. Moreover, in the case of uniqueness of the solution, they also provided a necessary and sufficient condition under which their optimization scheme yields an optimal allocation.

In this paper we first extend their algorithms to cyclic event graphs whose structure satisfy the following restriction: in each elementary circuit there exists at least one place not shared among the other circuits. Kanban systems belong to this class.

Then, we provide necessary and sufficient conditions for the convergence to the optimum also in the case of multiple solutions. In particular, the latter result requires an extension of the smoothness condition given by Panayiotou and Cassandras in [6].

We also provide a new algorithm, denoted as *Two-index Incremental Optimization* (TIO) algorithm, that results to be equivalent to the IO algorithm in the case of uniqueness of the solution, while it reveals to be computationally more efficient with respect to (wrt) the GIO algorithm in the case of multiple solutions. In fact, at each step, we provide a criterion to select only one solution among all the optimal ones. The main improvements wrt the GIO algorithm have origin by the following considerations: (a) among all the solutions that are optimal at a given step, only those characterized by the minimal number of critical circuits, will reveal to be optimal at the following step(s); (b) all the optimal solutions with the minimal number of critical circuits will converge to the same optimal one after a number of steps that is equal to the number of critical circuits; (c) if a solution that is optimal at a given step is characterized by n critical circuits, then its firing rate will only improve after n further steps.

By taking into account the three items above, three main improvements have been introduced wrt the GIO algorithm: (a) we first define a new performance index $\boldsymbol{J}$ consisting of two terms with a lexicographic ordering. The first one is the firing rate, while the second one is the inverse of the number of critical circuits. Thus, at each step, $\boldsymbol{J}$ enables us to select, among the solutions characterized by the maximal firing rate, those whose number of critical circuits is the minimum; (b) at each step, we choose arbitrarily one among all those solutions that are optimal wrt $\boldsymbol{J}$; (c) if the solution that is optimal at a given step is characterized by n critical circuits, n tokens are allocated in a single iteration. In particular, they will be added in those circuits whose cycle time is equal to the critical time. In such a way we also reduce the number of iterations required to allocate the total number of available resources.

2 Background

In this section we recall some important formalism used in the following of the paper. In particular we provide some important features on Petri nets, cyclic event graphs and Kanban-based manufacturing systems. For more details on these subjects we address to [2, 3, 4, 5, 6].

2.1 Petri nets and cyclic event graphs

A *Place/Transition net* (P/T net) is a structure $N = (P, T, Pre, Post)$, where P is a set of places; T is a set of transitions; $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre*– and *post*– incidence functions that specify the arcs.

A *marking* is a vector $M : P \to \mathbb{N}$ that assigns to each place of a P/T net a non–negative integer number of tokens, represented by black dots. A *P/T system* or *net system* $\langle N, M_0 \rangle$ is a net N with an initial marking M_0.

A transition t is enabled at M if $M \geq Pre(\cdot, t)$ and may fire yielding the marking $M' = M + Post(\cdot, t) - Pre(\cdot, t)$.

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.

An event graph (like a Petri net in general) is said to be *strongly connected* if there exists a directed path from any node in $P \cup T$ to every other node. We also define an *elementary circuit* in a strongly connected event graph as a directed path that goes from one node back to the same node, while any other node is not repeated. A strongly connected event graph is also called *cyclic* because each node belongs to a circuit.

It can be proved that in a cyclic event graph the total number of tokens in any elementary circuit is invariant by transition firing [1].

A deterministic Timed P/T net is a pair (N, τ), where $N = (P, T, Pre, Post)$ 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.

For deterministic timed cyclic event graphs we can compute, for any elementary circuit γ_i, the following ratio called the *cycle time* of the circuit:

$$c_i = \frac{\mu_i}{x_i}$$

where μ_i denotes the sum of the firing times related to the transitions belonging to γ_i, and x_i the number of tokens circulating in γ_i.

Let Γ represents the set of elementary circuits of a cyclic event graph and:

$$\hat{c} = \max_{\gamma_i \in \Gamma} c_i.$$

Any $\gamma_i \in \Gamma$ such that $c_i = \hat{c}$ 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 are enabled, the *firing rate* of each transition in steady state is given by:

$$\varrho = \frac{1}{\hat{c}}.$$

This result means that, if we observe what happens at any point of the event graph, we can see that ϱ tokens pass through that point during each unit period of time (on the average). As a consequence, if we want to increase the speed (i.e., the firing rate of the system), we have to add one (or several) token(s) to the critical circuit(s). Adding tokens in other circuits would be worthless.

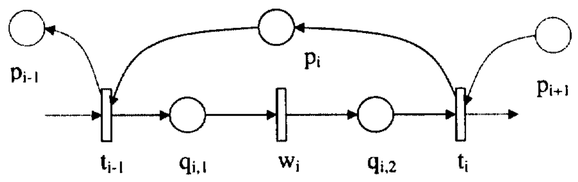

Figure 1: A stage of a kanban system.

A further important feature of cyclic event graphs concerns liveness: a cyclic event graph is guaranteed to be live if and only if every elementary circuit contains at least one token.

2.2 Kanban–based manufacturing systems

Kanban systems have been used by many researchers [2, 6, 8] to model various types of manufacturing processes. The main idea behind the kanban method is the following. A production line is divided into several stages and at every stage there is a fixed number of tickets called *kanban*. An arriving job receives a kanban at the entrance of the stage and maintains possession of it until it exits the stage. If an arriving job does not find an available kanban at the entrance, it is not allowed to enter that stage until a kanban is freed; in this case, the job is forced to wait in the previous stage.

As shown by Di Mascolo *et al.* in [2], an event graph can be used to model a kanban system. In figure 1, we show how to model the i–th stage of such a system. Transition w_i represents the operation performed at stage i, p_i contains as many tokens as free kanbans, $q_{i,1}$ contains as many tokens as parts waiting to be manufactured, and $q_{i,2}$ contains as many tokens as parts ready for the next operation. These parts will be transferred to the next operation if some kanbans are available, i.e., if there are some tokens in p_{i+1}.

Note that for CTEG models of kanban systems the following properties hold: (a) the firing rate coincides with the throughput; (b) to each stage is associated an elementary circuit $\gamma_i = \{p_i,\ t_{i-1},\ q_{i,1},\ w_i,\ q_{i,2},\ t_i,\ p_i\}$; (c) because each place p_i only belongs to circuit γ_i, the number of elementary circuits is limited by the number of places.

3 Incremental Optimization Algorithms and generalized smoothness condition

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

We assume that each process requires at least one resource to be live, thus only $K = K' - N_c$ resources are *available* to be allocated. (The relaxation of this assumption does not modify the results we present in this paper.)

We represent an allocation by the N_c–dimensional vector

$$\boldsymbol{x} = [\ x_1 \ \cdots \ x_i \ \cdots \ x_{N_c}\ ]^T, \tag{1}$$

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

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

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

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

$$\mathcal{A}_k = \left\{ \boldsymbol{x} \ : \ \sum_{i=1}^{N_c} x_i = k + N_c, \ x_i \geq 1 \right\}, \quad k = 0, 1, \cdots \qquad (3)$$

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

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

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

Let

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

be the set of solutions of (P1).

Panayiotou and Cassandras [6] defined the following conditions on $J(\boldsymbol{x})$.

Definition 1 ([6], Smoothness Condition). *The performance index J verifies the smoothness condition if $\forall\, \boldsymbol{x}^* \in \mathcal{A}_k^*$, $\boldsymbol{x} \in \mathcal{A}_k$, $k = 1, \cdots, K$,*

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

■

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

Proof. Follows from the definition. □

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 [6].

Algorithm 3 ([6] IO Algorithm).

```
Let $x_0 := [1, \cdots, 1]$;
for $k = 0, \cdots, K-1$ do
  begin
    $i_k^* := arg \max_{i=1,\cdots,N_c} \{\Delta J_i(x_k)\}$;
    $x_{k+1} := x_k + e_{i_k^*}$;
  end.
```

After K steps, $\boldsymbol{x}_K$ is the optimal solution of (P1) under the assumptions stated by the following theorem proved in [6].

Theorem 4 ([6]). *For any $k = 0, 1, \cdots$, if $\boldsymbol{x}_k$ computed in accordance to algorithm 3 is unique, then it is the optimal solution to problem (P1) iff the performance index J is smooth.*

Panayiotou and Cassandras in [6] also provide a straightforward extension of the IO algorithm in the case of multiple solutions, denoted as *Generalized Incremental Optimization* (GIO) Algorithm.

Algorithm 5 ([6] GIO Algorithm).

```
Let $x_0 := [1, \cdots, 1]$;
let $\mathcal{U}_0 = \{x_0\}$;
for $k = 0, \cdots, K-1$ do
  begin
    $\Delta := \max_{i=1,\cdots,N_c,\ x_k \in \mathcal{U}_k} \{\Delta J_i(x_k)\}$;
    $\mathcal{U}_{k+1} := \{x_k + e_i \mid \Delta J_i(x_k) = \Delta,\ x_k \in \mathcal{U}_k,$
                                   $i = 1, \cdots, N_c,\ \}$;
  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 given by Panayiotou and Cassandras in [6].

Definition 6 (Generalized Smoothness Cond.). *The performance index J verifies the generalized smoothness condition if $\forall k \geq 0$, there exists an allocation $\boldsymbol{x}_{(0)} \in \mathcal{A}_k^*$ and an infinite sequence of optimal allocations $\boldsymbol{x}_{(1)}, \cdots, \boldsymbol{x}_{(l)}$ where $\boldsymbol{x}_{(l)} \in \mathcal{A}_{k+l}^*$, $\forall l \geq 1$ and $\boldsymbol{x}_{(l)}$ is obtained from $\boldsymbol{x}_{(l-1)}$ 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 satisfies the smoothness condition, then it also satisfies the generalized smoothness condition.

We now prove that the GIO algorithm provides optimal solutions at each step if and only if J satisfies the generalized smoothness condition. This was implied in [6] but not formally proved.

Theorem 7. *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 satisfies the generalized smoothness condition.*

Proof. (if) We observe that $\boldsymbol{x}_0 = [1 \ \cdots \ 1]^T$ 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 $\mathcal{A}_k^*$. □

4 Firing rate optimization for CTEG

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

Let us consider a cyclic event graph with N_c elementary circuits and let P_i be the set of places that belong only to circuit γ_i. Clearly, $P_i \cap P_j = \emptyset\ \forall i \neq j$. We make the following assumption

Assumption $P_i \neq \emptyset, \forall\, i = 1, \cdots, N_c,$

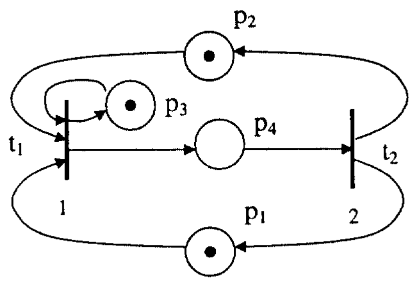

Figure 2: The cyclic event graph in example 8.

i.e., we assume that in each elementary circuit there exists at least one place not shared among the other circuits. This assumption implies that $N_c \leq card(P)$, i.e., in the restricted class of nets we are considering, the number of elementary circuits cannot be greater than the number of places (in a general CTEG the number of circuits can be exponential in the number of places). Note that CTEG models of kanban system, as discussed in Section 2.2, satisfy this assumption.

The allocation problem now becomes that of allocating $K = K' - N_c$ available tokens to N_c elementary circuits. We assume that the optimization procedure may only allocate tokens to the places in P_i, thus all places belonging to more than one circuit are empty, i.e.,

$$M(p) = 0,\ \forall p \in P \setminus \cup_{i=1}^{N_c} P_i.$$

while the i–th component x_i of the allocation vector denotes the sum of the markings of all places belonging only to the i–th circuit, i.e.,

$$x_i = \sum_{p \in P_i} M(p), \qquad (6)$$

We also define the N_c–dimensional vector

$$\boldsymbol{\mu} = [\ \mu_1 \ \cdots \ \mu_i \ \cdots \ \mu_{N_c}\ ]^T$$

whose i–th entry represents the cycle time of the i–th circuit.

In this section we consider a particular performance index J: the firing rate. To increase the firing rate of the system, we have to reduce the critical time, so we choose $J(\boldsymbol{x}) = 1/\hat{c}(\boldsymbol{x})$ as the objective function, where $\hat{c}(\boldsymbol{x})$ is the critical time for the allocation $\boldsymbol{x}$.

We first present an example showing that J is not smooth. On the contrary, we shall prove that it satisfies the generalized smoothness condition, thus ensuring that the GIO algorithm provides the set of optimal solutions. The following example also enables us to make some important considerations that will be the basis for the formulation of a new and computationally more convenient optimization algorithm.

Example 8. Let us consider the cyclic event graph in Fig. 2. It contains three elementary circuits: $\gamma_1 = \{p_1, t_1, p_4, t_2, p_1\}$, $\gamma_2 = \{p_2, t_1, p_4, t_2, p_2\}$, $\gamma_3 = \{p_3, t_1, p_3\}$. Furthermore, in accordance with the above notation: $P_1 = \{p_1\}$, $P_2 = \{p_2\}$, $P_3 = \{p_3\}$. Now, to apply the GIO algorithm, we assume that one token is initially allocated to each of circuits, i.e., $\boldsymbol{x}_0 = [1\ 1\ 1]^T$.

	γ_1	γ_2	γ_3	$J(x)$
μ	3	3	1	
$x_1^{(1)}$	2	1	1	1/3
$x_1^{(2)}$	1	2	1	1/3
$x_1^{(3)}$	1	1	2	1/3

Table 1: The results of the first step of the IO algorithm.

Then, we have to select the set 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. 1. As it can be noted, all circuits are characterized by the same cycle 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

$$\max_i\{J(\boldsymbol{x}_1^{(1)} + \boldsymbol{e}_i)\} = \max_i\{J(\boldsymbol{x}_1^{(2)} + \boldsymbol{e}_i)\} = \frac{2}{3}$$
$$\max_i\{J(\boldsymbol{x}_1^{(3)} + \boldsymbol{e}_i)\} = \frac{1}{3}.$$

Thus we can conclude that $\boldsymbol{x}_1^{(1)}$ and $\boldsymbol{x}_1^{(2)}$ only generate optimal allocations at step 2.

Note that this follows from the fact that the third allocation $\boldsymbol{x}_1^{(3)}$ is characterized by two 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 n critical circuits, it is necessary to add at least n tokens (one to each critical circuit) to improve the firing rate. This conclusion reveals the requirement to distinguish among different allocations with the same firing rate but different number of critical circuits. Thus, we introduce a new performance index $\boldsymbol{J} = [J_1\ J_2]^T$ consisting of two terms. The first one, J_1 is the firing rate, i.e., the performance index previously considered, while the second one is a measure of the number of critical circuits, i.e., $J_2 = 1/\hat{n}(\boldsymbol{x})$, where $\hat{n}(\boldsymbol{x})$ denotes the number of critical circuits in the allocation $\boldsymbol{x}$. Note that we impose a lexicographic ordering on the performance index, i.e., $\boldsymbol{J} = \boldsymbol{J}'$ if $J_1 = J_1'$ and $J_2 = J_2'$, $\boldsymbol{J} < \boldsymbol{J}'$ if $J_1 < J_1'$ or $J_1 = J_1'$ and $J_2 < J_2'$.

Example 9. Let us consider again the cyclic event graph in Fig. 2 and the allocations in Tab. 1 relative to the first step of the algorithm. The introduction of the new performance index enables us to immediately reject the allocation $\boldsymbol{x}_1^{(3)}$, being $J_1(\boldsymbol{x}_1^{(1)}) = J_1(\boldsymbol{x}_1^{(2)}) = J_1(\boldsymbol{x}_1^{(3)}) = 1/3$ and $J_2(\boldsymbol{x}_1^{(3)}) = 1/2 < J_2(\boldsymbol{x}_1^{(1)}) = J_2(\boldsymbol{x}_1^{(2)}) = 1$, thus $\boldsymbol{J}(\boldsymbol{x}_1^{(3)}) < \boldsymbol{J}(\boldsymbol{x}_1^{(1)}) = \boldsymbol{J}(\boldsymbol{x}_1^{(2)})$. Therefore, the only optimal solutions wrt $\boldsymbol{J}$ are $\boldsymbol{x}_1^{(1)}$ and $\boldsymbol{x}_1^{(2)}$. ■

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

Lemma 10. *An allocation x^* is optimal wrt J iff:*

$$\frac{\mu_i}{x_i^* - 1} \geq \hat{c}(x^*) \equiv \frac{1}{J_1(x^*)} \qquad \forall i = 1, \cdots, N_c. \qquad (7)$$

Proof. (if) Let us assume that the inequality (7) is verified for an allocation $x^* \in \mathcal{A}_k$ and let $x \in \mathcal{A}_k$ be a different allocation.

We first prove that $J_1(x)$ cannot be greater than $J_1(x^*)$. Since $x, x^* \in \mathcal{A}_k$ and $x \neq x^*$, there exists a circuit γ_i such that $x_i < x_i^*$. This implies that

$$\frac{1}{J_1(x)} \geq c_i(x) \geq \frac{\mu_i}{x_i^* - 1} \geq \hat{c}(x^*) = \frac{1}{J_1(x^*)} \qquad (8)$$

thus

$$J_1(x^*) \geq J_1(x). \qquad (9)$$

Now, let us prove that if $J_1(x^*) = J_1(x)$, it holds that $J_2(x^*) \geq J_2(x)$.

We first show that assumption (7) implies that the number of tokens in each circuit for allocation x cannot be less than one wrt the corresponding number of tokens for allocation x^*, i.e.,

$$x_i \geq x_i^* - 1, \qquad i = 1, \cdots, N_c. \qquad (10)$$

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

$$\frac{1}{J_1(x)} \geq c_i(x) \geq \frac{\mu_i}{x_i^* - 2} > \frac{\mu_i}{x_i^* - 1} \geq \frac{1}{J_1(x^*)}$$

and this contradicts the assumption that $J_1(x) = J_1(x^*)$.

Now, let Γ be the set of elementary circuits in the cyclic event graph. The set Γ can be partitioned into three disjoint sets, i.e., $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$ where

$$\begin{aligned} \Gamma_1 &= \{\gamma_i \,|\, x_i = x_i^*\}, \\ \Gamma_2 &= \{\gamma_i \,|\, x_i = x_i^* - 1\}, \\ \Gamma_3 &= \{\gamma_i \,|\, x_i > x_i^*\}. \end{aligned}$$

By assumption (7), all circuits in Γ_2 are critical for the allocation x but not for x^*, while all circuits in Γ_1 that are critical for x^* are also critical for x. Thus we can state that

$$\begin{aligned} \hat{n}(x) &= card(\Gamma_1^c) + card(\Gamma_2), \\ \hat{n}(x^*) &\leq card(\Gamma_1^c) + card(\Gamma_3), \end{aligned}$$

where $\Gamma_1^c = \{\gamma_i \in \Gamma_1 \,|\, c_i(x^*) = \hat{c}(x^*)\}$.
By virtue of equation (10) and by the assumption that $\sum_i x_i = \sum_i x_i^*$, it is easy to observe that $card(\Gamma_3) \leq card(\Gamma_2)$. We can conclude that $\hat{n}(x) \geq \hat{n}(x^*)$, thus $J_2(x^*) \geq J_2(x)$, as we want to prove.

(only if) We prove this by contradiction. Let us assume that x^* is an optimal solution, but condition (7) is violated, i.e., there exists a circuit γ_l such that $\mu_l/(x_l^* - 1) < \hat{c}(x^*)$.
Let γ_m be a critical circuit in x^* and consider a new vector x obtained from x^* by moving one token from γ_l to γ_m. If γ_m is the only critical circuit in x^*, then $J_1(x) > J_1(x^*)$. If x^* has more than one critical circuit, the value of J_1 does not vary, i.e., $J_1(x) = J_1(x^*)$, but $J_2(x) > J_2(x^*)$. Thus, in both cases, x^* cannot be optimal. □

Theorem 11. *The performance index J satisfies the smoothness condition.*

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

We prove that x' is an optimal allocation for $\mathcal{A}_{k+1}$. Clearly, $J_1(x') \geq J_1(x^*)$ (the equality holds if x^* has more than one critical circuit). For all $\gamma_i \in \Gamma \setminus \{\gamma_l\}$ we have that

$$\frac{\mu_i}{x_i' - 1} \equiv \frac{\mu_i}{x_i^* - 1} \geq \frac{1}{J_1(x^*)} \geq \frac{1}{J_1(x')}. \qquad (11)$$

The first inequality follows from the characterization given by (7) [Lemma 10] and the fact that x^* is optimal.

For circuit γ_l we have that

$$\frac{\mu_l}{x_l' - 1} \equiv \frac{\mu_l}{x_l^*} = \frac{1}{J_1(x^*)} \geq \frac{1}{J_1(x')} \qquad (12)$$

where the equality derives from the fact that γ_l is a critical circuit for x^*.

Since x' satisfies (11) and (12), by lemma 10 it follows that x' is optimal for $\mathcal{A}_{k+1}$.

This shows that J satisfies the smoothness condition. □

Note that even if J satisfies the smoothness condition, it does not necessarily give a unique optimal allocation at each step, hence it is necessary to use the GIO Algorithm (as opposed to the IO Algorithm) to compute an optimal allocation. Furthermore, the following corollary can be easily proved.

Corollary 12. *The performance index J_1, 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_1 and J, respectively.

Clearly, $\bar{\mathcal{A}}_k^* \subseteqq \mathcal{A}_k^*$ $\forall\, k \geq 0$. By virtue of corollary 2 all solutions in $cal\bar{A}_k^*$ will produce an infinite sequence of optimal solutions wrt J. This implies that there exist some $x^* \in \mathcal{A}_k^*$ which origin an infinite sequence of optimal allocations wrt to J_1. It follows that J_1 satisfies the generalized smoothness condition. □

Note that from theorem 7 and corollary 12 it immediately follows that all solutions computed in accordance to the GIO algorithm are optimal wrt J_1.

5 Two–index Optimization algorithm

In this section we propose a new IO algorithm, denoted as *Two–index* Incremental Optimization (TIO) algorithm, which reveals to be computationally more convenient wrt the generalized one.

The main improvement consists in the fact that while at each step the GIO algorithm considers as optimal all those allocations within the set $\mathcal{U}_k$, whose cardinality may greatly increase, the use of the new performance index J enables us to neglect all those allocations that will be found non optimal at the following step(s). Moreover, the new algorithm allows one to consider only one solution at each step. In fact, by virtue of the property below, it follows that all the allocations that are optimal at a given step will converge to the same optimal one after a given number of steps.

Thus no error occurs if we do not examine all the intermediate steps and directly compute the final common allocation.

Property 13. *Let us consider a cyclic event graph with N_c elementary circuits. Let us assume that at a given step k there exist multiple optimal allocations wrt J each of them containing n critical circuits. All these allocations will converge to the same optimal one at the $(k+n)$-th step of the optimization algorithm.*

Proof. By condition 7 it follows that:

- all 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 circuits 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 n more tokens, one in each critical circuit. Thus, at the $(k+n)$-th all optimal allocations in $\mathcal{A}_k^*$ converge to the same optimal allocation in $\mathcal{A}_{k+n}^*$. □

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

Algorithm 14 (TIO Algorithm).

```
Let $x_0 := [1, \cdots, 1]$;
let $k := 0$;
while $k < K$ do
  begin
    $\Gamma_c := \{\gamma_i \mid \gamma_i$ is critical for $x_k\}$;
    $n := card(\Gamma_c)$;
    if $k + n > K$ then $k := K$
        else
          begin
          $x_{k+n} := x_k + \sum_{\gamma_i \in \Gamma_c} e_i$;
          $k := k + n$;
          end
  end.
```

Note that if at a given step k we have n critical circuits and we still have to allocate a number $\bar{k} < n$ 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 15. *For any k, each x_k computed in accordance to the TIO algorithm, is an optimal solution wrt J.*

Proof. We showed that the index J is smooth, thus (see the *if* part of the proof of Theorem 7) the GIO algorithm determines an infinite series of optimal allocations starting from 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 provides the same allocation; in the case of multiple (say, n) solutions, Property 13 ensures that arbitrarily choosing one is sufficient to recover the set of optimal solutions after n steps. □

Remark 16. *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 starts from a single optimal allocation that has an (even) number n of critical circuits, after $n/2$ steps the cardinality of $\mathcal{U}_k$ may reach the value $\binom{n}{n/2}$ that using Stirling's formula can be approximated for large n by $\frac{2^{n+1}}{\sqrt{2\pi n}}$.*

Thus it is easy to observe that the worst case computationally 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 circuits, while for the GIO algorithm it has an upper-bound given by $O(K\ 2^N\ N^{1/2})$.

6 Conclusions

We have proposed an incremental algorithm to allocate a given number of tokens, so as to maximize the firing rate of a cyclic event graph with deterministic transition firing delays. The algorithm is very efficient both in terms of computational time and in terms of memory requirements (only one solution needs to be kept at each step). The algorithm can be applied to a special class of nets in which tokens are allocated to places that belong to only one circuit: this class is powerful enough to model kanban systems.

We envisage two possible extensions of this work. Firstly, it may be possible to extend this approach to cyclic event graphs with stochastic transition firing rates. Secondly, we plan to investigate the possibility of removing the assumption that tokens are allocated to places that belong to only one circuit.

References

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

[8] 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.