TECHNICAL RESEARCH REPORT

COMPLEXITY OF PRODUCTION MANAGEMENT IN A PETRI NET ENVIRONMENT

by J-M. Proth and I. Minis

T.R. 94-19

Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

COMPLEXITY OF PRODUCTION MANAGEMENT IN A PETRI NET ENVIRONMENT

Jean-Marie PROTH[1-2] and Ioannis MINIS[2]

[1] INRIA-Lorraine, Technopôle Metz 2000, 4 rue Marconi, F-57070 Metz, FRANCE
[2] Systems Research Center, University of Maryland, College Park, MD 20742, USA

Abstract:

The objective of this paper is to show that Petri nets facilitate a comprehensive approach to production management and reduce the complexity of the problems involved at the expense of some constraints imposed on the decision making system.

The first part of the paper focuses on cyclic manufacturing systems. For this type of systems, it is always possible to propose an event graph model which represents both the physical and the decision making systems. We use such a model to propose a near-optimal scheduling algorithm that maximizes productivity while minimizing the work-in-process (WIP) in the deterministic case.

The approach used for non-cyclic manufacturing systems is different in the sense that only the manufacturing processes (i.e. the physical part of the system) and the related constraints are modelled using Petri nets. We use such a Petri net model to propose a short-term planning process which results in a trade-off between the computation burden and the level of resource utilization. The short-term planning model is then enhanced to obtain the scheduling model. The latter is used to develop an efficient scheduling algorithm that is able to satisfy the requirements imposed by short-term planning.

Key Word:

Complexity, Event graphs, Petri nets, Planning, Scheduling.

I - Introduction

During the 70s and the early 80s, Petri nets were considered mainly as modelling and simulation tools. This is the reason why research studies were first oriented toward small-sized models based on high-level nets such as the predicate/transitions nets [5], the colored Petri nets [9], and the Petri nets with individual tokens [17]. More recently, researchers became interested in the analytical properties of elementary nets, also called black-and-white nets. Very important results were proposed by COMMONER et al. [2], MURATA [13], CHRETIENNE [1], ZHOU [19], and many others. As far as we know, the application of those results to cyclic manufacturing systems, and their enhancement in order to model, evaluate and manage manufacturing systems were initiated by HILLION et al. [8]. Several studies are currently in progress and deal with both cyclic and non-cyclic manufacturing

systems. The purpose of this paper is to present the most important results in this field which can reduce the complexity of the management problems in cyclic and non-cyclic manufacturing systems.

The second section of the paper is devoted to the definitions, concepts and properties to be used in the remainder of the paper. We introduce the definition of elementary nets, the state equation, the definitions of p-invariants and t-invariants, and the qualitative properties which are desirable in manufacturing. We also introduce the event graphs (also called marked graphs), a special type of Petri nets, which are used to model cyclic manufacturing systems. The second section concludes by presenting the decomposable nets, which are the basic tools in the planning and scheduling of non-cyclic manufacturing systems.

In section three, we show that it is possible to model both the physical and the decision making systems (DMSs) of cyclic manufacturing systems with an event graph. Using the properties of event graphs and a given cyclic control, we show how to maximize the productivity of a manufacturing system (i.e. to minimize the cycle time) while minimizing the work-in-process. The properties of event graphs are also used to develop a near-optimal scheduling algorithm.

Section four focuses on non-cyclic manufacturing systems. We present a short-term planning problem and we develop a solution approach which takes advantage of the fact that the system can be modeled as a decomposable Petri net. We then propose a scheduling model and develop a scheduling algorithm which allows us to satisfy the requirements imposed by short-term planning.

Section five presents the concluding remarks.

II - Petri nets (PNs): definitions, concepts and properties

1. Elementary Petri nets

a. Definitions

A Petri net is a 5-tuple PN = (P, T, A, W, M_0) where:

$P = \{ p_1, p_2, ..., p_q \}$ is the set of places,

$T = \{ t_1, t_2, ..., t_n \}$ is the set of transitions,

$A \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,

$W: A \rightarrow \{ 1, 2, \dots \}$ is a weight function,

$M_0: P \rightarrow \{ 1, 2, \dots \}$ is the initial marking.

In Figure 1, we present a Petri net with marking:

$M_0 = < 2,0,1,3,0,4 >$

When no weight is mentioned, an arc is supposed to be weighted by 1.

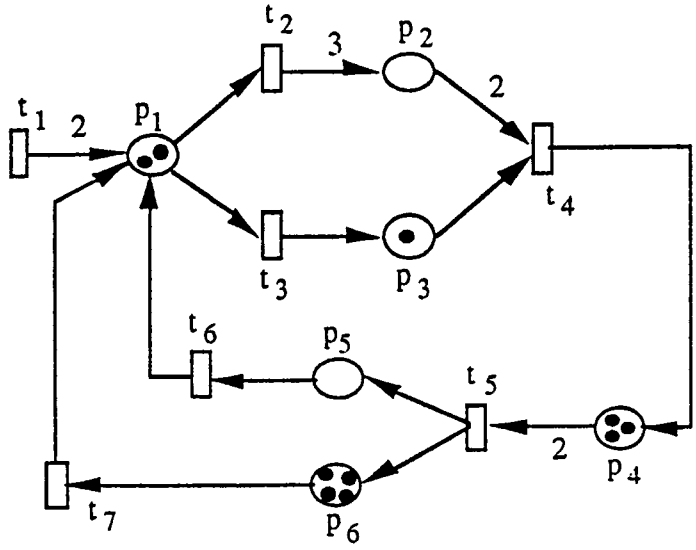


Fig. 1: A Petri net

Places are represented by circles, transitions by bars, and each place contains a number of tokens (represented by dots) equal to its marking.

Given a marking M, a transition is said to be **enabled** if and only if:

$$M(p) \geq W(p,t), \forall\ p \in {}^{o}t$$

where ${}^{o}t$ represents the set of input places of transition t.

Firing a transition consists in:

(i) removing W(p,t) tokens from each $p \in {}^{o}t$,

(ii) adding W(t,p) tokens in each $p \in t^{o}$, where t^{o} is the set of output places of t.

For instance, the marking of the net in Figure 1 becomes M = < 1,3,1,1,1,5 > after firing t_2 and t_5.

An enabled transition may, or may not, be fired.

A timed Petri net is a Petri net for which a duration, or firing time, (which is deterministic or stochastic) is associated to each transition. In this paper, only deterministic firing times are considered.

The duration associated with a transition represents the time between the instant the tokens disappear from the input places and the instant the tokens appear in the output places. Usually, we consider that tokens continue to belong to the input places of a transition t until the firing of t ends.

The **incidence matrix** of a Petri net, say U = $[u_{ij}]$, i = 1,2,...,q; j = 1,2,...,n, is defined as follows:

$$u_{ij} = \begin{cases} W(t_j, p_i) & \text{if } t_j \in {}^{o}p_i \\ -W(p_i, t_j) & \text{if } t_j \in p_i^{o} \\ 0 & \text{otherwise} \end{cases}$$

where ${}^{o}p$ (resp. p^{o}) is the set of input (resp. output) transitions of p.

Note that the incidence matrix reflects the structure of the Petri net provided that the net does not contain self-loops (i.e. loops composed by only one place and one transition).

Let M_0 be an initial marking and σ a sequence of transitions fired starting from M_0. We denote by M the marking obtained after firing the last transition of σ. We define the **firing count vector** V_σ related to the sequence σ as:

$$V_\sigma = (v_1, v_2, \ldots, v_n)$$

where v_i (i = 1,2,...,n) is the number of times transition t_i appears in σ, and n the total number of transitions.

The **state equation** is as follows:

$$M^t = M_0^t + U.V_\sigma^t \qquad (1)$$

where A^t denotes the transpose of A.

Note that if a sequence σ of transitions verifies (1), it is not guaranteed that σ is feasible (i.e. that it is possible to fire the sequence of transitions σ).

A vector Z is a **p-invariant** if:

(i) $Z.U = 0$, (2)

(ii) Z is a q-vector whose components are non-negative integers,

(iii) at least one of the components of Z is strictly positive.

Let $R(M_0)$ be the set of markings reachable from M_0. It can be easily shown that, for any $M \in R(M_0)$:

$$Z.M_0^t = Z.M^t \qquad (3)$$

The proof is made by left-multiplying both sides of (1) by Z, and by using (2). Thus, $Z.M^t$ is an **invariant**, i.e. a linear combination of the place markings that remains constant under any sequence of transition firings.

A vector H is a **t-invariant** if:

(i) $U.H^t = 0$, (4)

(ii) H is a n-vector whose components are non-negative integers,

(iii) at least one of the components of H is strictly positive.

Let σ be a firing sequence and V_σ the related firing count vector. If V_σ is a t-invariant, M_0 the initial marking and M the marking obtained after firing the sequence σ of transitions, then:

$$M = M_0$$

b. Qualitative properties

The following properties are important when using Petri nets to model manufacturing systems.

b_1. Structural liveness

Definition 1: A PN N = (P, T, A, W) is structurally live if their exists an initial marking M_0 such that, for any $t \in T$ and $M \in R(M_0)$ (i.e. M reachable from M_0) there exists a firing sequence which leads from M to a marking which enables t.

In manufacturing systems, structural liveness implies that it is always possible to perform any operation for which the system was designed, assuming that the initial state of the system is properly chosen. In other words, any operation which can be performed by the system will remain possible in the future, irrespective of the past sequence of decisions.

b_2. Reversibility and home state

Definition 2: A PN N = (P, T, A, W) is reversible for a marking M_0 if, for any $M \in R(M_0)$, there exists a firing sequence σ_M which leads to M_0 from M.

This property is also very important for manufacturing systems. It guarantees that it is always possible to return to the initial state, no matter what the current state is. This is often necessary for maintenance, tool adjustments or changes in production. The next definition generalizes definition 2.

Definition 3: A marking M_k of a Petri net N = (P, T, A, W) is a home state for the marking M_0 if it can be reached from any marking reachable from M_0, i.e. $M_k \in R(M)$ for any $M \in R(M_0)$.

The use of a home state is the same as the one of M_0 in definition 2.

According to these definitions, any marking reachable from the initial marking in a reversible PN is a home state, but a PN with a home state may be not reversible.

b_3. Boundedness

Definition 4

(i) A marked PN N = (P, T, A, W, M_0) is said to be k-bounded if $M(p) \leq k$ for any $p \in P$ and $M \in R(M_0)$.

(ii) A marked PN N = (P, T, A, W, M_0) is said to be bounded if it is k-bounded for some integer $k > 0$.

(iii) A PN N = (P, T, A, W) is structurally bounded if the marked PN (N, M_0) is bounded for any initial marking M_0.

Boundedness is not necessary in manufacturing, but may be desirable when fully automated systems are concerned. Nevertheless, it is always necessary to be able to keep the PN model of a manufacturing system bounded; for example, an unbounded model may result in WIP that increases to infinity.

2. Event graphs (or marked graphs)

a. General properties

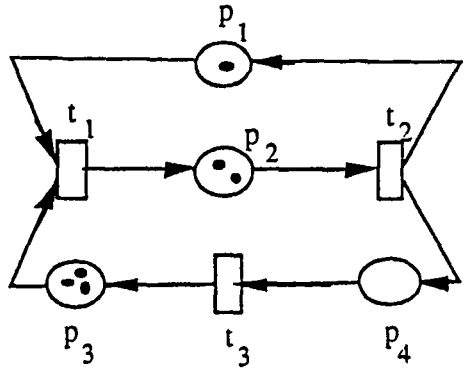


Fig. 2: An event graph

An event graph is a Petri net such that each place has exactly one input and one output transition. Furthermore, the weight associated to each transition is 1. Such a Petri net is represented in Figure 2. Note that event graphs may contain elementary circuits. For example, the event graph of Figure 2 contains two elementary circuits: namely $\gamma_1 = <p_1, t_1, p_2, t_2, p_1>$ and $\gamma_2 = <p_2, t_2, p_4, t_3, p_3, t_1, p_2>$.

It is easy to prove that:

(i) If we assign 1 to each place belonging to one of the elementary circuits and 0 to the other places, then the vector whose elements are these values is a p-invariant –more precisely a minimal p-invariant. Another way to express the foregoing is to say that the number of tokens is any elementary circuit is invariant by any transition firing. We owe this result to COMMONER et al. [2].

(ii) The n-vector whose components are equal to 1 is a t-invariant. It means that the marking returns to the initial marking after firing each transition exactly once.

The following result is due to COMMONER et al. [2].

Result 1: A strongly connected event graph is guaranteed to be deadlock-free if and only if every elementary circuit contains at least one token.

b. Deterministic event graphs

In this section, we consider timed event graphs where the times associated to the transitions are deterministic. For any elementary circuit γ, we define the **cycle time** as:

$$C(\gamma) = \mu(\gamma) / M(\gamma) \qquad (5)$$

where:

$\mu(\gamma)$ is the sum of the firing times of the transitions belonging to γ,

$M(\gamma)$ is the number of tokens circulating in γ.

If M_0 is the initial marking, we know that $M(\gamma) = M_0(\gamma)$, $\forall\ M \in R(M_0)$. Thus $C(\gamma)$ is an invariant. Let C^* be the maximal cycle time among all elementary circuits, i.e.:

$$C^* = \max_{\gamma \in \Gamma} C(\gamma) \qquad (6)$$

where Γ is the set of elementary circuits of a strongly connected event graph. Any $\gamma \in \Gamma$ such that $C(\gamma) = C^*$ is called a **critical circuit.**

In order to guarantee that only a single firing of a certain transition may occur at time t, we introduce a self-loop to each transition with initially one token in each self-loop place.

From now on, we consider only the firing policy called "Earliest Operating Mode" (EOM). It is the policy where transitions fire as soon as they are enabled. This corresponds to the common policy applied to fully automated systems as we will see in section III. CHRETIENNE [1] showed that, under an EOM, the operation of the system becomes periodic after a finite time. Periodicity means that there exist two integers n_0 and K such that:

$$S_t(n+K) = S_t(n) + KC^*, \quad \forall n \geq n_0, \quad \forall t \in T$$

where $S_t(k)$ is the starting time of the k-th firing of transition t, and K is the period. The following result holds.

Result 2: Provided that the event graph is strongly connected and that the EOM policy applies, the cycle time of the model is C^* in a steady state. In other words, the firing rate of all transitions in steady state is $\lambda = 1 / C^*$.

This result provides the productivity of a manufacturing system which is modeled by a strongly connected event graph, assuming that the WIP is known.

Several algorithms have been proposed to reach a given cycle time while minimizing a linear combination of the markings, the coefficients of which are the components of a p-invariant. They can be found in LAFTIT et al. [11]. In particular, an adjustment heuristic algorithm which can be used in practice for models of any size, is proposed in this paper.

c. Qualitative properties of event graphs

It is easy to verify that a strongly connected event graph is:

* Structurally live for any marking M_0 such that each elementary circuit contains at least one token.
* Reversible for any marking M_0 such that each elementary circuit contains at least one token. Furthermore, under the same condition, any $M \in R(M_0)$ is a home state.
* Structurally bounded, assuming that the initial marking M_0 is finite.

3. Decomposable nets

a. Definitions

Let $X = [x_1, ..., x_n]$ be a t-invariant of a Petri net $N = (P, T, A, W)$.

Definition 5: The set of transitions $t_i \in T$ such that $x_i > 0$ is called the support of the t-invariant X and is denoted by $\|X\|$.

In a non-cyclic manufacturing system, parts enter and exit the system; thus, its PN model contains source transitions (also called input transitions), which are used to model the entrance of parts into the system, and sink transitions (or output transitions), which are used to model the exit from the system.

Definition 6: Let X be a t-invariant of the PN model $N = (P, T, A, W)$ of a manufacturing system. Let $N_X \subset N$ be a PN such that:

$$N_X = (P_X, \|X\|, A_X, W_X)$$

where $P_X = \{p / p \in P \text{ and } \exists\, t_1, t_2 \in \|X\| \text{ s.t. } p \in {}^\circ t_1 \text{ and } p \in t_2^\circ \text{ in } N\}$, $A_X = A \cap \{(P_X \times \|X\|) \cup (\|X\| \times P_X)\}$ and W_X is the restriction of W to A_X.

N_X is referred to as **X-related subnet of N.**

If, in addition:

(i) the cardinality of p° is 1 for every $p \in P_X$,

(ii) there exists at least one $t_1 \in \|X\|$ and one $t_2 \in \|X\|$ such that ${}^\circ t_1 = \varnothing$ and $t_2^\circ = \varnothing$,

(iii) N_X is acyclic,

then N_X is referred to as **X-CFIO of N**, where CFIO stands for Conflict Free net with Input and Output transitions.

b. Decomposability of a net

Decomposability is the key concept in the planning and scheduling approaches for non-cyclic manufacturing systems we are proposing in this paper.

Definition 7: Let N be a PN model of a manufacturing system with input and output transitions. Let $\{X_1, ..., X_r\}$ be a set of t-invariants of N such that:

$N = \bigcup_{i=1}^{r} N_{X_i}$, where N_{X_i} is the X_i-CFIO of N (we will say that the set of X_i-CFIOs N_{X_i} covers N).

Then N is said to be decomposable.

c. Qualitative properties of a decomposable net

Result 3: A decomposable net N is structurally live for any initial marking M_0.

Proof:

α. Let $M \in R(M_0)$ and σ a firing sequence which leads to M starting from M_0. Since N is a decomposable PN, there exists a set $\{X_1, \ldots, X_r\}$ of t-invariants of N such that the corresponding set of X-CFIOs N_{X_i} covers N. Thus, it is possible to assign each element of σ to one N_{X_i}, designing firable sequences which maintain the order of the elements in σ. Let us call σ_{X_i} (i = 1,2,...,r) the sequence related to N_{X_i}. Note that some of these sequences may be empty.

β. Let us now consider $t \in T$ and let N_{X_k} be a X-CFIO containing t. Due to the definition of X-CFIOs, there exists $\tilde{\sigma}_{X_k}$ such that firing $\sigma_{X_k} \circ \tilde{\sigma}_{X_k}$ from M_0^k (restriction of M_0 to N_{X_k}) leads to M_0^k again, and this sequence contains each transition at least once ($\circ$ represents the concatenation). As a consequence, $\sigma_k^* = \tilde{\sigma}_{X_k} \circ \sigma_{X_k} \circ \tilde{\sigma}_{X_k}$ is a firing sequence which applies to M and contains t.

This proof holds for any M_0.

Q.E.D.

Result 4: A decomposable PN is reversible for any initial marking M_0.

Proof:

α. This first part of the proof is identical to the first part of the proof of result 3.

β. Due to the definition of X-CFIOs, there exists $\tilde{\sigma}_{X_i}$ such that $\sigma_{X_i} \circ \tilde{\sigma}_{X_i}$ leads to M_0^i from M_0^i for i = 1,2,...,r. Thus if $\tilde{\sigma} = \mathop{()}_{i=1,2,\ldots,r} \tilde{\sigma}_{X_i}$ then $\sigma \circ \tilde{\sigma}$ leads to M_0 from M_0.

Q.E.D.

A decomposable net is not structurally bounded; for instance, the number of tokens in such a system increases indefinitely if we keep firing only the input transitions. Nevertheless, the following result holds.

Result 5: A decomposable net N = (P, T, A, W) can be kept bounded by appropriate firings of its input transitions, no matter how many times the output transitions are fired, provided that the number of the output firings is finite.

Proof:

Let $k_1, \ldots, k_s$ be the minimal numbers of times the output transitions $t_{0,1}, \ldots, t_{0,s}$, must be fired. Let $X_1, \ldots, X_r$ be a set of t-invariants such that the corresponding X-CFIOs N_{X_i} cover N. Let n_j^i be the component of the t-invariant X_i which corresponds to $t_{0,j}$. At least one of the n_j^i, i = 1,2,...,r, is strictly positive for any $j \in \{1,\ldots,s\}$, and at least one of the n_j^i, j = 1,2,...,s, is strictly positive for any $i \in \{1,\ldots,r\}$. Thus the integer linear programming problem

$$\mathcal{M}in \sum_{i=1}^{r} y_i$$

s.t.

$$\sum_{i=1}^{r} y_i . n_j^i \geq k_j, \quad j = 1,2,\ldots,s$$

$$y_i \in \{0,1,\ldots,\}, \quad i = 1,2,\ldots,r$$

has a finite solution. The integer y_i is the number of times the set $\|X_i\|$ of transitions must be fired according to X_i in order to fire the output transitions for the required number of times. Thus, the marking M(p) of $p \in t_v^\circ$ is bounded from above by:

$$M_0(p) + \sum_{i=1}^{r} y_i x_v^i, \quad \text{where } x_v^i \text{ is the } v-\text{th componant of } X_i$$

This relation holds for any $p \in P$.

Q.E.D.

III - Cyclic manufacturing systems

1. Problem formulation

In this section, we emphasize the usefulness of event graphs to model cyclic manufacturing systems, also called off-line or ratio-driven manufacturing systems. Such a system manufactures a given set of part types at given ratios. The objective of the control problem of such a system is to maximize its productivity while minimizing the WIP.

In a cyclic system, there exists an optimal control which is periodic. From the management point of view, a periodic control can be expressed as a sequence of part types associated to each machine, cell or transportation system. Such a sequence is referred to as the input sequence of the resource. It is emphasized that the same part type can appear several times in the same input sequence in order to satisfy the production ratios.

Let us consider, for instance, a set of three machines denoted by M_1, M_2 and M_3, which manufacture three types of parts, say P_1, P_2 and P_3. Assume that the manufacturing routings of these part types are as follows (the number in parenthesis provides the manufacturing time of each operation):

P_1: $M_1(2)$, $M_2(1)$, $M_3(4)$

P_2: $M_3(2)$, $M_1(3)$

P_3: $M_2(4)$, $M_1(1)$

Assume also that the production ratios are 0.25, 0.25 and 0.5 for P_1, P_2 and P_3, respectively. In this case, a set of input sequences associated to M_1, M_2 and M_3 could be:

$\sigma(M_1) = <P_1, P_2, P_3, P_3>$, $\sigma(M_2) = <P_1, P_3, P_3>$, $\sigma(M_3) = <P_1, P_2>$.

We do not claim that this periodic control is optimal, but that there exists an optimal control with respect to the objective defined above ($\sigma^*(M_1)$, $\sigma^*(M_2)$, $\sigma^*(M_3)$) where $\sigma^*(M_i)$ is obtained by applying a cyclic permutation to $\sigma(M_i)$, i = 1,2,3.

The next subsection is devoted to the modelling of cyclic manufacturing systems. In subsection 3, we show that it is possible to maximize the productivity of the system for any cyclic control. Finally, subsection 4 proposes a heuristic algorithm to reach a near-optimal cyclic control.

2. Modelling of cyclic manufacturing systems

The event graph model of a cyclic manufacturing system is developed as follows:

(i) Model the manufacturing process of each part type. In such a model, a transition represents an operation and a place a buffer. Each transition firing corresponds to the execution of an operation and the firing time of the transition is the time required to perform this operation. Each manufacturing process is reproduced as many times as necessary to yield the given production ratios. For instance, in the example introduced in the previous subsection, the manufacturing processes of P_1 and P_2 are represented once, and the manufacturing process of P_3 is represented twice.

(ii) Model the cyclic operation mode of the system by assuming that a new part is launched in the system as soon as a part of the same type is completed. This is modelled by adding an output place to the last transition of each manufacturing process model, and by connecting this place to the first transition of the same model. Thus, we obtain elementary circuits called **process circuits**.

(iii) The final step is to model the sequencing of the part types for each machine. This is done by connecting in a unique circuit all the transitions corresponding to operations performed by the same machine. The order of transitions in these new elementary circuits (called **command circuits**) is determined by the sequencing of the jobs on the corresponding machines.

Tokens circulating in the process circuits represent parts, whereas tokens in the command circuits represent machine status information. Note that there is exactly one token in each command circuit, since a machine is assumed to manufacture at most one part at a time, and since an elementary circuit without a token blocks after a finite number of transition firings.

a. Model of a cyclic job-shop

The above steps were employed to model the job-shop introduced in subsection III-1 with the control ($\sigma(M_1)$, $\sigma(M_2)$, $\sigma(M_3)$). The resulting model is shown in Figure 3.

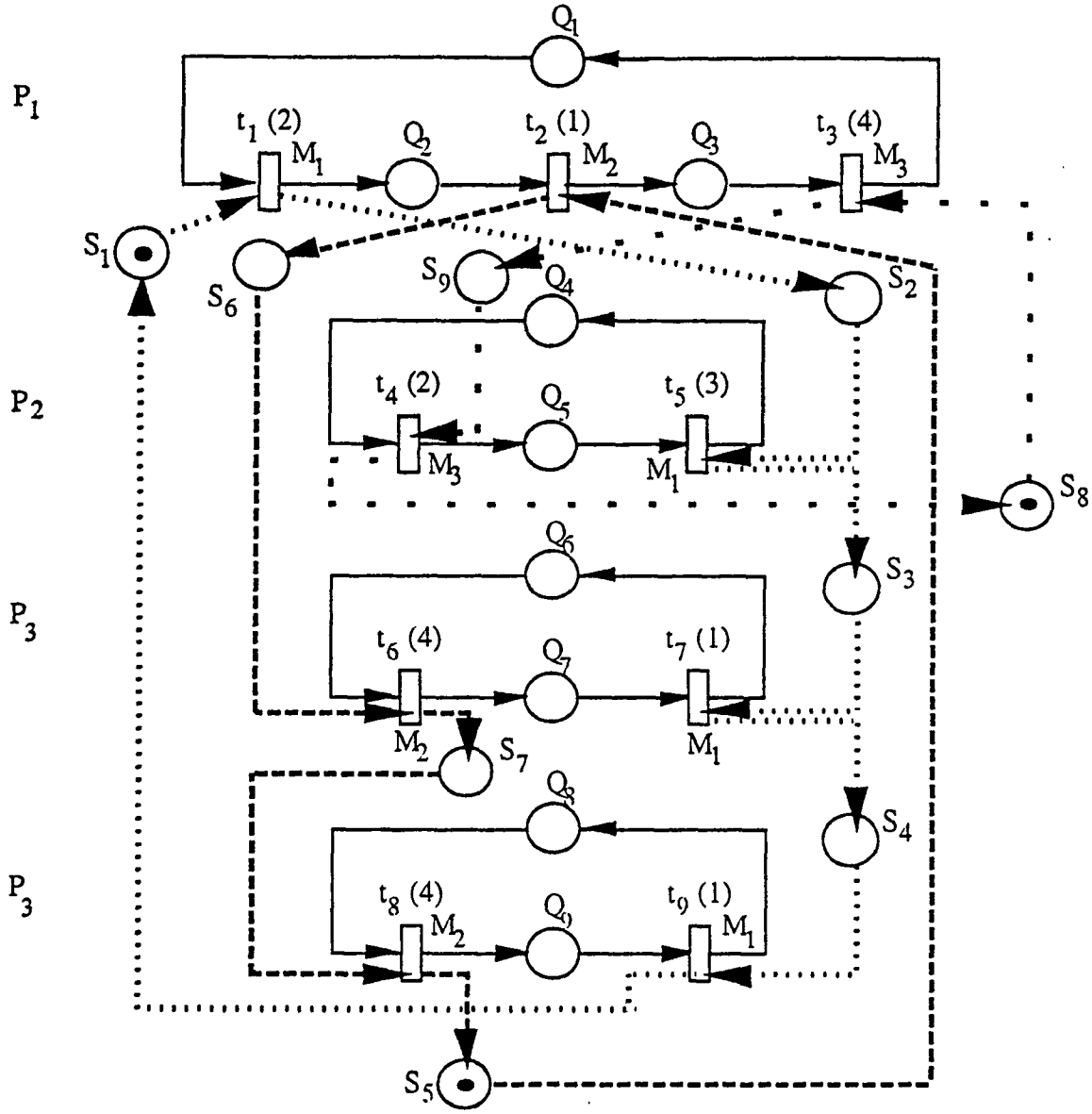


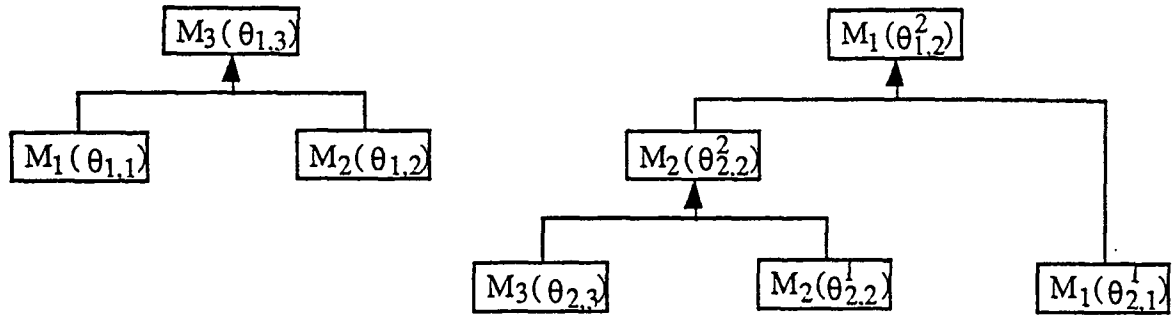
Fig. 3: A job-shop model

b. Model of a cyclic assembly system

As a second example, let us consider two product types, the bills of materials and manufacturing processes of which are represented in Figure 4. Each box in the figure corresponds to a make item (part). In addition, each box represents an operation and contains the machine which performs this operation. The parameters in parentheses denote operation times.

We assume that the production ratios are 0.5 and 0.5. Following the general approach presented in subsection 1 of this section, we obtain the model shown in Figure 5.

In this model, we only show the command circuit related to machine M_2 for simplicity.



a. Part type P_1 b. Part type P_2

Fig. 4: Two assembly manufacturing processes

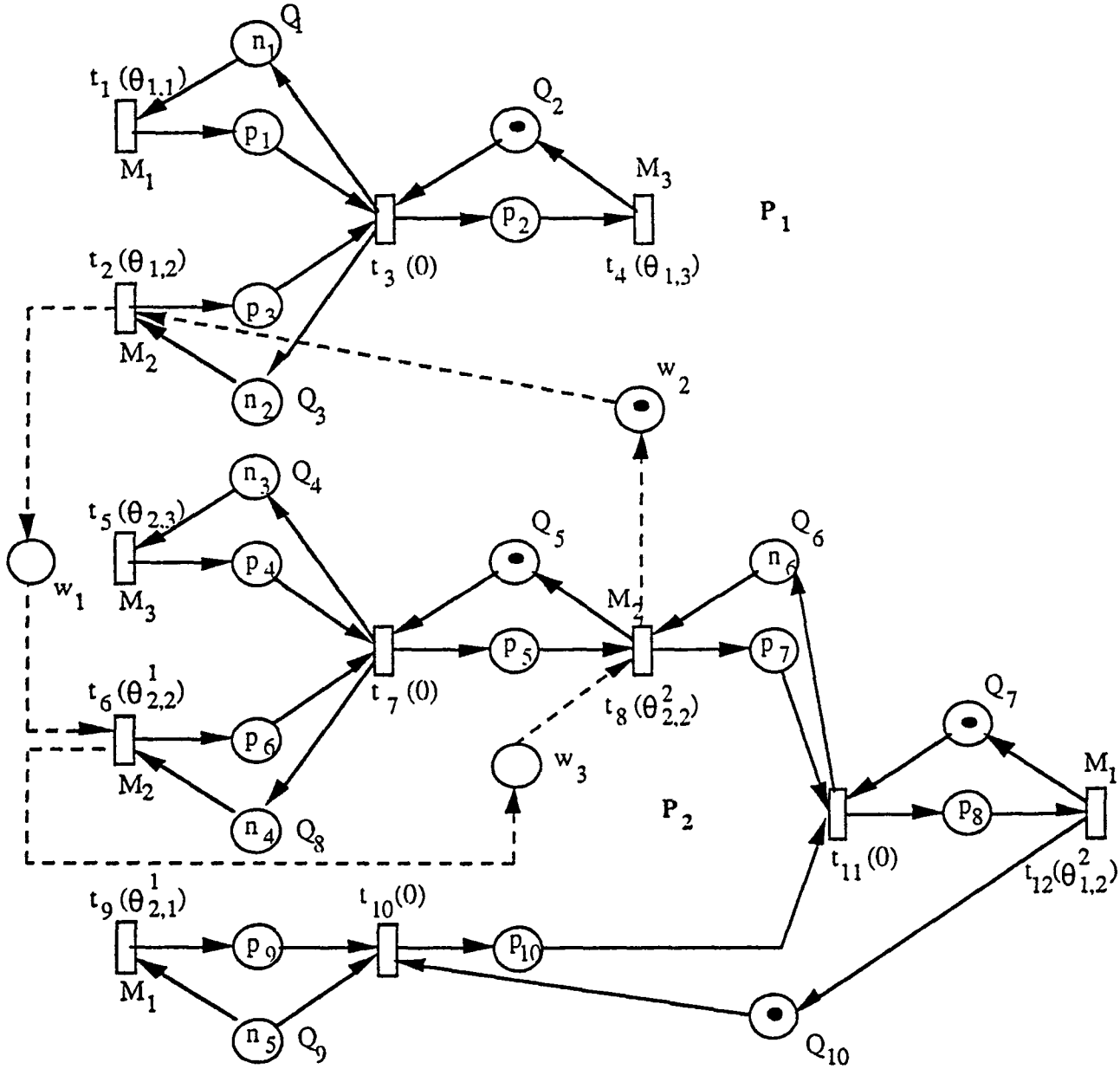


Fig. 5: The assembly system model

Places P_i (i = 1,2,...,10) represent buffers. Places Q_i (i = 1,2,...,10) are used to control the number of components of each type in the system. For instance, Q_1 contains initially n_1 tokens, which guarantees that at most n_1 pieces of the component of p_1 that is manufactured by M_1 can be found simultaneously in the system.

As shown in Figure 5, some transitions which do not represent operations have been introduced in the model. These transitions are t_3, t_7, t_{10} and t_{11}. The time associated to these transitions is 0. They have been introduced to allow for WIP control.

The following result holds for cyclic manufacturing systems:

Result 6:

Assuming that:

(i) a new part is launched in the manufacturing system as soon as a part of the same type is completed,

(ii) the control applied to the system is cyclic and can be expressed in terms of input sequences which satisfy the production ratios,

(iii) one manufacturing process is associated to each part type,

we claim that the corresponding Petri net model encapsulates both the physical and the decision making systems. Furthermore, this model is an event graph.

Remarks

1 - If more than one manufacturing process alternatives are available to manufacture some part types, then it is possible to split up each part type into as many sub-part types as the number of manufacturing processes available, and to distribute the production ratio of the part type among the sub-part types in order to optimize a criterion of interest, such as work load balancing among the resources. This leads to a problem that satisfies condition (iii) of result 6.

2 - If the model is not strongly connected, then it is always possible to introduce transitions with zero duration which represent the entrance of the parts in the system. These transitions are then connected by means of a command circuit, which transforms the model into a strongly connected event graph.

3. Evaluation of a cyclic manufacturing system

Since a Petri net model of a cyclic manufacturing system is, or can always be transformed into, a strongly connected event graph and the firing times are deterministic, the results presented in section II-2 apply. In such a model, three types of elementary circuits exist, namely:

(i) the process circuits, which can contain as many tokens as desirable,

(ii) the command circuits, which contain one token each, the initial position of which is defined by the input sequence of the related resource,

(iii) the hybrid circuits, which are composed of portions of command and process circuits; for instance, in Figure 3, the circuit $\gamma = <t_1, S_2, t_5, Q_4, t_4, S_8, t_3, Q_1, t_1>$ is a hybrid circuit.

Since a command circuit contains exactly one token and since it is possible to put as many tokens as desirable in the places which belong to process circuits, it is possible to find infinite initial markings such that the critical circuit is a command circuit, and more precisely the command circuit γ^* such that:

$$C(\gamma^*) = \underset{\gamma \in \Gamma_C}{Max} C(\gamma) \tag{7}$$

where Γ_C is the set of command circuits and $C(\gamma)$ is the cycle time of γ.

In other words, it is always possible to fully utilize the bottleneck machine if the initial WIP is large enough, and thus to reach the maximal productivity of the system. However, the objective is to reach the maximal productivity while minimizing the WIP. This problem can be written as follows for a given cyclic control:

$$\textit{Minimize} \sum_{i/P_i \in P_{\overline{C}}} x_i \tag{8}$$

s.t.

$$\sum_{i/P_i \in \gamma \setminus P_{\overline{C}}} x_i \geq \mu(\gamma) \Big/ C_0 - \sum_{i/P_i \in \gamma \cap P_C} x_i \quad , \quad \forall \gamma \in \Gamma_{\overline{C}},$$

$$x_i \in \{0,1,\ldots\}, \forall i \text{ such that } P_i \in P_{\overline{C}}.$$

where $x_i = M_0(P_i)$, P_C is the set of places belonging to the command circuits, $P_{\overline{C}}$ is the set of places which do not belong to the command circuits, $\Gamma_{\overline{C}}$ is the set of elementary circuits which are not command circuits, and $C_0 = C(\gamma^*)$ is defined by relationship (7).

Note that $x_i = M_0(P_i)$ is known for $P_i \in P_C$; this is derived from the input sequences of the resources. The difficulty in solving problem (8) is to compute all elementary circuits of the model, the number of which increases often exponentially with the size of the model. For this reason, a heuristic algorithm, called adjustment algorithm, was proposed in [11]. In practice, this heuristic algorithm is not limited by the size of the model.

So far, we have shown that it is possible to reach the maximal productivity of the system while minimizing the WIP when the periodic control is known. However the problem of finding the periodic control which results in maximal productivity with a WIP which is the minimum among all WIP values corresponding to all periodic controls remains open. This problem is considered in the next sub-section.

4. Optimization of the cyclic control

The problem to be considered is a scheduling problem which is NP-hard. Thus, only heuristic algorithms can be used to solve large-sized problems. Two algorithms have been shown to provide near-optimal solutions within reasonable times.

a. The simulated annealing algorithm

This algorithm has been presented in numerous papers, and in particular in [10] and [12]. The part of the algorithm which is specific to the problem at hand is the generation of alternatives by perturbation of a given schedule, i.e. a given set of input sequences. We generate a new schedule in the neighborhood of a previous one by permuting two elements in each input sequence of the resources.

The results obtained by applying simulated annealing to this problem have been found to be better, on the average, than those obtained from Tabou and genetic search approaches.

b. Construction approach

The basic idea of this algorithm is to (i) determine, for each process circuit, the minimal number of tokens (≥ 1) required such that a command circuit becomes the critical circuit; (ii) assign to each process circuit as many periods of duration C_0 [see Eq. (8)] as the number found in (i); and (iii) fit the firing periods of the transitions within these C_0 periods, taking into account the scheduling constraints, i.e. the fact that two firing periods of transitions corresponding to the same machine do not overlap, and the fact that the partial order imposed by the manufacturing processes is verified.

A sufficient optimality condition is to find a schedule with the appropriate number of C_0 periods initially assigned to the manufacturing process circuits.

Algorithm

1. Initialization
   - 1.1. For each process circuit i (i = 1,...,n)
     - 1.1.1. Set $k_i = 0$

       k_i is a counter equal to the number of firing periods already placed in the C_0-period related to manufacturing process i.
     - 1.1.2. Set $N_i(0) = 1$

       $N_i(k_i)$ is the number of C_0-periods available for the next firing period.
     - 1.1.3. Set $\varphi_i(0) = 0$

       $\varphi_i(k_i)$ is the ending time of the first firing of the k_i-th transition of i.
     - 1.1.4. Compute $d_i(0) = 0$

       $d_i(k_i)$ is the degree of freedom corresponding to i given the schedule of the k_i first firing periods.

$$d_i(k) = 1 - \sum_{t \in T_i - T_i(k)} \mu(t) \Big/ \left(C_0 - \varphi_i(k) + \left[N_i^*(k) - N_i(k) \right] C_0 \right)$$

where: T_i *is the set of transitions of i,*

$T_i(k)$ *is the set of the first k transitions of i,*

$N_i^*(k) = \mathrm{Max}[N_i(k), \alpha_i]$, *where* $\alpha_i = \lceil \mu(i) / C_0 \rceil$,

$\mu(i)$ *is the sum of the firing times of the transitions of i,*

$\mu(t)$ *is the firing time of t.*

1.2. Set m = 1

m is the counter for iterations.

1.3. Set $E_m = \{1,2,\ldots,n\}$

E_m is the set of process circuits for which some firing periods have not been placed.

2. Scheduling

2.1. Compute j such that $d_j(k_j) = \mathrm{Min}_{i \in E_m} d_i(k_i)$

j is the elementary circuit to be considered next.

2.2. If $d_j(k_j) < 0$

Add $g_j(k_j)$ C_0-periods for the process circuit j: $N_j(k_j) = N_j(k_j) + g_j(k_j)$ where $g_j(k_j)$ is the smallest integer such that

$$\left(\sum_{t \in T_j - T_j(k_j)} \mu(t) \right) \Big/ \left(C_0 - \varphi_j(k_j) + \left[N_j^*(k_j) - N_j(k_j) + g_j(k_j) \right] C_0 \right) \leq 1$$

2.3. Choose the transition to be fired

If $k_j > 0$, t_j is the next transition in the order derived from the manufacturing process. Otherwise, we choose the transition which yields the minimal value of $d_j(1)$.

2.4. Place the firing period of t_j at the earliest time, taking into account the manufacturing constraints.

2.5. Set $T_j(k_j + 1) = T_j(k_j) \cup \{t_j\}$.

2.6. Compute $\varphi_j(k_j + 1)$ and $d_j(k_j + 1)$.

2.7. Set $k_j = k + 1$ and $m = m + 1$.

2.8. Update E_m.

2.9. If $E_m \neq \varnothing$ go to 2.1, otherwise stop.

To illustrate this algorithm, let us consider the model represented in Figure 3. The bottleneck machine is M_2 and $C_0 = 9$. To ensure that the critical circuit of the model is the command circuit corresponding to machine M_2, we need a minimum of:

* one token in the process circuit corresponding to P_1,
* one token in the process circuit corresponding to P_2,

* one token in each of the process circuits corresponding to P_3.

Applying the previous algorithm leads to the schedule represented in Figure 6.

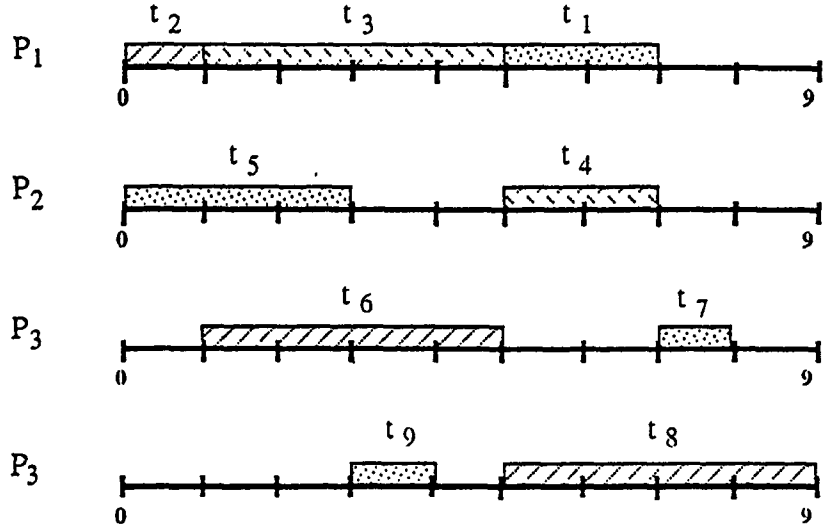


Fig. 6: An optimal schedule

Since no C_0-period was added to the initial ones, the schedule provided by the algorithm is optimal. This indicates that the optimal input sequences are:

$\sigma(M_1) = <P_2, P_3, P_1>$, $\sigma(M_2) = <P_1, P_3, P_3>$, $\sigma(M_3) = <P_1, P_2>$.

Furthermore, the initial marking is:

$M_0(Q_2) = M_0(Q_5) = M_0(Q_6) = M_0(Q_9) = 1$

$M_0(Q_1) = M_0(Q_3) = M_0(Q_4) = M_0(Q_7) = M_0(Q_8) = 1$

Finally, the command circuit and the initial locations of its tokens are fixed by the optimal input sequences.

IV - Non-cyclic manufacturing systems

1. Problem formulation

We are interested in a job-shop comprising n machines $M_1, M_2, \ldots, M_n$ that manufacture q types of parts denoted by $P_1, P_2, \ldots, P_q$. The demand for each part type is known at the end of each of R consecutive elementary periods. For instance, an elementary period could be a day and R = 5; in this case, we are interested in managing the system over a working week on a day-to-day basis. In the remainder of the paper, the union of the R elementary periods is referred to as the sub-period. Let us denote by d_i^j, i = 1,2,..,q, j = 1,2,...,R, the demand for part type Pi at the end of the j-th elementary period.

s_i^0 is the inventory level of part type i at the beginning of the first elementary period. We also define $\mathcal{M}(p_i)$ as the manufacturing process (routing) of part type P_i. $\mathcal{M}(p_i)$ provides:

(i) the sequence of operations to be performed on a part of type P_i;

(ii) the type of each operation, which can be either "assembly" or "regular"; disassembly operations are not considered;

(iii) the list of machines on which each operation can be performed;

(iv) the time required to perform each operation on each alternative machine.

The first problem to be solved is the short-term planning (STP) problem. Knowing the capacity of the system (i.e. the time available within each elementary period), we seek to determine the number of parts of each type to be manufactured during each elementary period in order to optimize a given criterion. Commonly used criteria include the number of delayed products, the maximal delay, the weighted sum of delays, or the sum of the inventory and backlogging costs. In the remainder of this paper, we seek to minimize the sum of the inventory and backlogging costs to illustrate the proposed approach.

Note that the time assigned to manufacturing tasks within each elementary period is bounded above by the duration of the period. The difference between the duration of an elementary period and the time assigned to tasks within this period is the maximal idle time of the machines during the period. It represents the flexibility of the system: the smaller the time assigned to tasks within an elementary period, the more likely it is that a feasible schedule exists. However, the productivity of the system is lower.

Starting from the number of parts of each type to be manufactured during the first elementary period (provided by the solution of the STP problem), the scheduling process (S) consists of assigning operations to their alternative machines and computing the beginning time of each operation in order to satisfy the usual manufacturing constraints, namely:

(i) operations should be performed according to the partial order specified by the manufacturing processes (routings): two operations belonging to the same manufacturing process should be performed according to the required order;

(ii) a given machine performs at most one operation at a time.

We do not try to optimize some criterion, but simply to find a feasible schedule, i.e. a schedule which meets the requirements of the STP for the first elementary period. If such a feasible schedule does not exist, the only solution is to reduce the time assigned to tasks within the elementary periods, and to re-compute the STP: since such a change reduces the amount of parts to be manufactured during each elementary period, it is more likely that a feasible schedule exists.

In Figure 7, we summarize the procedure to obtain a feasible schedule. The STP problem can usually be solved using classical optimization software tools the choice of which depends upon the

type of criterion to be optimized. However, the SP is NP-hard and, therefore, only heuristic problems are practical.

In the next section, we introduce basic concepts used to re-visit the previous approach in the light of PNs.

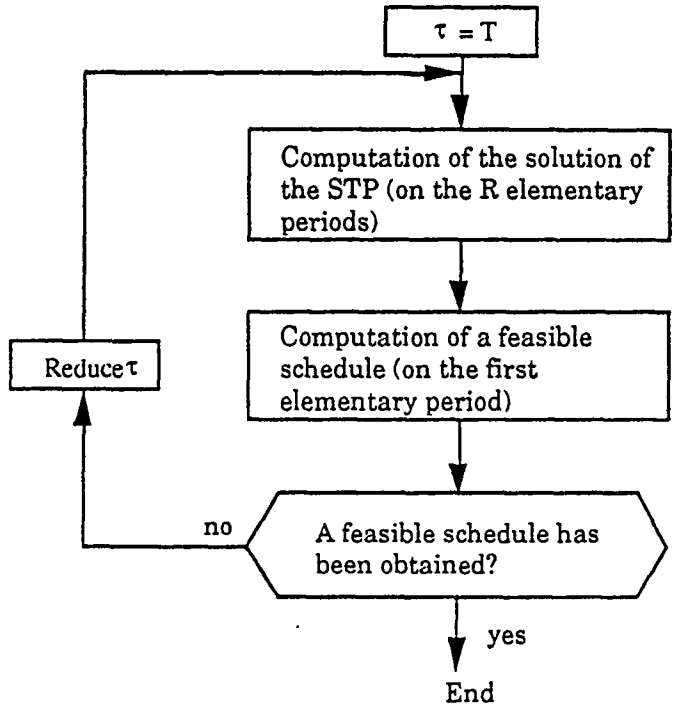


Fig. 7: General flow-chart for production management

2. Short-term planning

a. Modelling examples

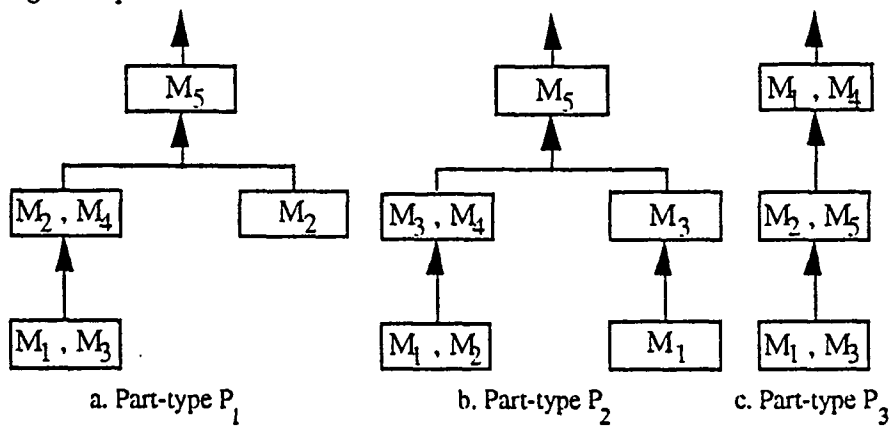


Fig. 8: Sample manufacturing processes

Figure 8 shows the manufacturing processes (routings) for three part types P_1, P_2 and P_3. Alternative machines for each operation are separated by commas in the corresponding boxes. The

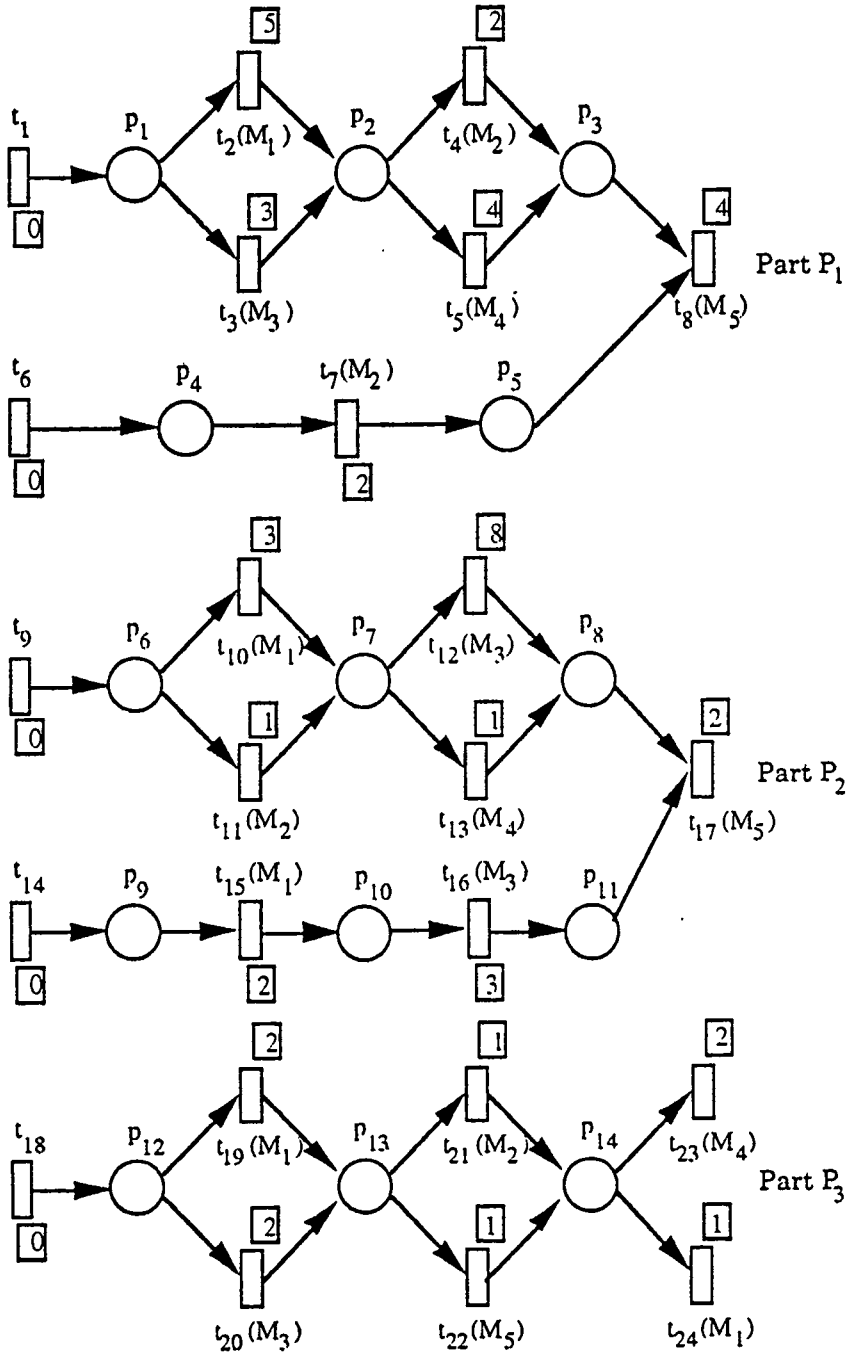


Fig. 9: The decomposable PN model of the system from the short-term planning point of view

numbers in squares are the manufacturing times of the operations (i.e. the firing times of the corresponding transitions). The duration of all input transitions of each of these models is zero. These transitions represent the launching of components to production. Each of the remaining transitions represents an operation on a machine. The machines corresponding to these transitions are included in parentheses. At most one firing is in progress at each transition at a time, which

implies that a self-loop, with one token in the corresponding place, is associated to each transition. These self-loops are not represented in Figure 9 for simplicity.

The PN model N presented in Figure 9 is obviously a decomposable net. It is possible to find several sets of t-invariants for which the corresponding CFIOs cover N. We can, for instance, choose the minimal t-invariants of the PN, which are obtained by combining the minimal t-invariants of each of the manufacturing process models. Since the models corresponding to P_1, P_2 and P_3 have respectively 4, 4 and 8 minimal t-invariants, we would obtain $4 \times 4 \times 8 = 128$ minimal t-invariants for the complete model. We can also choose some linear combinations of the minimal t-invariants provided that the derived CFIOs cover N.

b. Planning process

Let $\{X_1, \ldots, X_r\}$ be a set of t-invariants such that:

$$N = \bigcup_{i=1}^{r} N_{X_i}$$

Let O_k be the set of output transitions of the manufacturing process model of part type P_k, $k = 1,\ldots,q$. The demands for parts of type P_k are known at the end of each of R consecutive elementary periods. We denote, as in section IV-1, by d_k^j the number of parts of type P_k required at the end of the j-th elementary period; d_k^j is also the total number of times the transitions of O_k must be fired to satisfy the demand by the end of the j-th elementary period. T is the duration of an elementary period and τ is the time assigned to an elementary period (see section IV-1).

If y_i^j is the number of times the transitions of $\|X_i\|$ fire according to the components of X_i during the j-th elementary period, the following relations hold.

$$s_k^{j+1} = s_k^j - d_k^j + \sum_{i=1}^{r} \left[y_i^j \sum_{t \in O_k} x_t^i \right] \tag{9}$$

where x_t^i denotes the component of X_i corresponding to t in X_i.

This state equation holds for $k = 1,\ldots,q$ and $j = 0,\ldots,R-1$. s_k^0 is the initial inventory of parts of type P_k.

Furthermore, if Z(t) is the firing time of t:

$$\sum_{i=1}^{r} y_i^j x_t^i Z(t) \le \tau, \quad j = 1,\ldots,R, \; t \in T \tag{10}$$

are the capacity constraints.

If the criterion to be minimized is the sum of the backlogging costs and inventory costs, it can be expressed as:

$$\mathcal{M}in \sum_{k=1}^{q} \sum_{j=1}^{R} \left[b_k (-s_k^j)^+ + i_k (s_k^j)^+ \right] \tag{11}$$

where b_k (resp. i_k) is the backlogging cost (resp. the inventory cost) of one unit of part type P_k during one elementary period. Note that the problem which consists of minimizing (11) under constraints (9) and (10) can be re-written as a linear programming problem.

c. Remarks

The key to this approach is the choice of the set of t-invariants $\{X_1, \ldots, X_r\}$. Depending on this choice, N_{X_i} may be the model of one of the manufacturing processes or the model of a set of manufacturing processes corresponding to different part types. In the first case, the short-term planning procedure provides a better result, but requires extensive computation. In the second case, the number of y_i^j variables may be very small, and thus the computation required may be very limited; however, the result will be certainly inferior to that of the previous case. In general, selecting a smaller number of t-invariants (assuming that the CFIOs derived from them cover the PN model) offers the potential for reducing the computation burden at the expense of productivity. In is noted that the short-term planning process introduced in this section guarantees that the qualitative properties presented in the previous section hold.

3. Scheduling

a. Problem setting and definitions

The goal of the scheduling process is to assign operations to resources (in the case that several resources are available) and to define the starting time of each operation for the first elementary period in order to meet the firing requirements of short-term planning.

The model used in the scheduling process is obtained by adding resource sharing places to the model used for short-term planning. Such a place:

(i) initially contains one token,

(ii) forms a self-loop with each transition corresponding to the same machine.

These places guarantee non-reentrance; i.e. the fact that machines can perform at most one operation at any given time. For instance, we should have five resource sharing places in the model given in Figure 9. The place corresponding to machine M_1 would be linked in both ways to t_2, t_{10}, t_{15} and t_{19}.

Thus, there are two types of decisions that have to be made, namely:

(i) The decisions related to the selection of a resource, when several resources are available to perform the same task. We define such a decision as an RU decision, where RU stands

for Resource Use. An RU decision must be made in the model of Figure 10, in which the next operation on the part represented by the token can be performed by M_1 or M_2 or M_3.

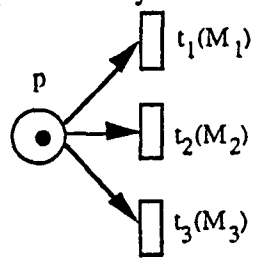


Fig. 10: Modelling of a RU type of decision

In Figure 9, this situation can be observed in p_1, p_2, p_6, p_7, p_{12}, p_{13} and p_{14}.

(ii) The decisions related to the sequencing of part types on resources, called PS decisions, where PS stands for Product Sequencing. Figure 11 represents such a situation: we can fire either t_1, which represents the manufacturing of a part of type R_1 on M_1, or t_2, which represents the manufacturing of a part of type R_2 on M_1.

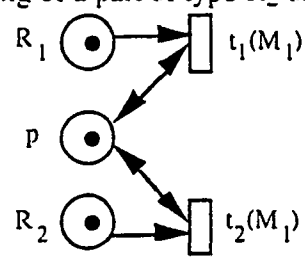


Fig. 11: Modelling of a PS type of decision

Hereafter, places related to the RU (resp. PS) type of decisions will be called RU (resp. PS) places.

We assume that transitions are fired as soon as they are enabled. Consequently, a schedule is defined as soon as a sequence of transitions is assigned to each RU and each PS place. The transitions belonging to such a sequence are the output transitions of the place, and they appear in the sequence as many times as specified by the short-term planning solution for the first elementary period. Let us for instance consider the RU place p in Figure 10. If, according to the short-term planning solution, t_1, t_2 and t_3 have to be fired twice, once and three times, respectively, a valid sequence could be $\sigma_p = < t_1, t_3, t_3, t_2, t_1, t_3 >$.

As we can see, the PN modelling of the scheduling problem represents the RU and PS decisions, thus illustrating the decisions to be made explicitly. Nevertheless, it should be noted that these decisions are not independent and that improper sequences may lead to blocking. Such a case is shown in Figure 12 which contains two PS places, q_1 and q_2: if t_1, t_2, t_3 and t_4 are fired once, then $\sigma_{q_2} = < t_2, t_3 >$ leads to a blocking situation.

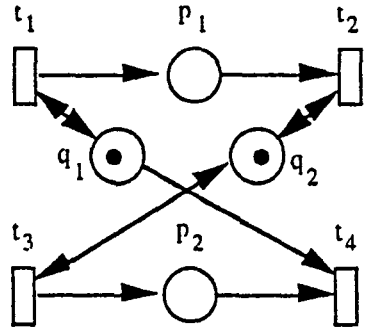


Fig. 12: A situation where blocking may occur

Having established the appropriate system model, the objective of the scheduling problem can be expressed as follows: determine the sequences to be assigned to each RU and PS places that result in a makespan which is less than the duration of an elementary period, knowing that:

(i) the transitions belonging to a sequence are the output transitions of the related RU or PS place,

(ii) a transition appears in a sequence as many times as specified by the short-term planning solution for the first elementary period.

b. Scheduling algorithm

It is well-known that the scheduling problem of the general job-shop is NP-hard. As a consequence, only heuristic algorithms can be considered for solving large-sized problems. We have developed two heuristics: a simulated annealing approach and an approach based on the improvement of a critical circuit. For the sake of brevity, only the second algorithm is presented in this section.

The algorithm starts with a feasible set of sequences, i.e. with a set of valid sequences which do not lead to blocking. An easy way to build such a feasible set of sequences consists of assigning any valid sequence to each PS place, simulate the system according to these sequences by firing the transitions as soon as they are enabled, and assign to the RU places the sequences resulting from the simulation.

Let T be the duration of the elementary period, n_t the number of times t should be fired during the first elementary period, $S_t(k)$ the instant when the k-th firing of t starts and $F_t(k) = S_t(k) + Z(t)$. This notation refers to the initial feasible set of sequences, assuming that the starting time of the first transition firing is 0 and that a transition fires as soon as it is enabled. A critical path is a sequence of pairs:

$$C = < (t_{i_1}, k_{\alpha_1}), \ldots, (t_{i_r}, k_{\alpha_r}) >$$

such that:

(i) $S_{t_{i_1}}(k_{\alpha_1}) = 0$;

(ii) $F_{t_{i_r}}(k_{\alpha_r}) = \underset{(t,k)}{\mathrm{Max}}\{F_t(k)\}$ (this value is the makespan);

(iii) $F_{t_{i_j}}(k_{\alpha_j}) = S_{t_{i_{j+1}}}(k_{\alpha_{j+1}})$, for j = 1,...,r-1.

A necessary condition to reduce the makespan is to reduce $F_{t_{i_r}}(k_{\alpha_r})$, and thus to bring forward the finishing time of one of the transition firings that belongs to the critical path. To do this, we will have to delay some transition firings which do not belong to the critical path. Note that it is not allowed to violate the machine precedence constraints (i.e. the constraints related to the manufacturing process, taking into account the initial marking). Furthermore, in order to reduce the computational burden, we introduce the **float time**, which is the maximal time a transition can be delayed without increasing the makespan. The calculation of the float time takes into account the schedule and machine precedence dependencies of the operations. The objective is to move earlier a transition firing which belongs to the critical path only if this delays another transition firing for less than its float time. The two transition firings which are involved in the scheduling perturbation are not taken into account in the computation of the float times. The following algorithm is derived from the above remarks.

Algorithm

1. Compute a feasible set of sequences and the initiation times of the transition firings.
2. Compute the critical path and the makespan.
3. Select (t_u, k_u) and (t_c, k_c) such that:
   * $t_u \neq t_c$,
   * (t_u, k_u) belongs to the critical path,
   * the k_c-th transition t_c just preceding the k_u-th transition t_u in one of the sequences belonging to the feasible set of sequences,
   * swapping the k_u-th transition t_u and the k_c-th transition t_c in the sequence does not violate the machine precedence constraints,
   * the float time associated to (t_c, k_c) is greater than or equal to the delay resulting from the swapping.

   If several pairs (t_u, k_u) and (t_c, k_c) are candidates for swapping, select the one for which the delay is the closest to the float time.
4. If a pair has been selected, go to 2, else stop.

V - Concluding remarks

Manufacturing systems can take advantage of the properties of Petri nets to cope with the complexity of scheduling problems.

In cyclic manufacturing systems, the properties of event graphs may be used to propose fast and powerful algorithms which maximize the productivity while minimizing WIP, when a cyclic control is

known. An algorithm which proceeds by construction is also available to provide a near-optimal control (i.e. schedule). This algorithm is closely related to the Petri net model.

Non-cyclic manufacturing systems are modelled using decomposable nets at the short-term planning level. The computation of a short-term plan is based on a set of t-invariants, the choice of which provides for adjustment of the computational burden. A system model applicable to scheduling is obtained by completing the planning model. It was used in this work to develop a new scheduling heuristic.

A major problem which remains open in Petri net related work for management of systems is the integration of cyclic or/and non-cyclic manufacturing systems.

Bibliography

[1] P. CHRETIENNE, "Les Réseaux de Petri Temporisés", Univ. Paris VI, Paris, France, Thèse d'Etat, 1983.

[2] F. COMMONER, A. HOLT, S. EVEN and A. PNUELI, "Marked directed graphs", *J. of Comp. and Syst. Sci.*, vol. 5, No. 5, pp. 511-523, 1971.

[3] F. DI CESARE et al., *Practice of Petri Nets in Manufacturing*, ISBN 0 412 41230 6, CHAPMAN and HALL, 1993.

[4] M. DI MASCOLO, Y. FREIN, Y. DALLERY and R. DAVID, "A Unified Modeling of Kanban Systems Using Petri Nets", Technical Report No. 89-06, LAG. Grenoble, France, September, 1989.

[5] H.J. GENRICH and K. LAUTENBACH, "System modelling with high-level Petri nets", *Theoret. Comput. Sci.*, vol. 13, pp. 109-136, 1981.

[6] G. HARHALAKIS, M. LEVENTOPOULOS, C.P. LIN, R. NAGI and J.M. PROTH, "A class of conflict free Petri nets used for controlling manufacturing systems", Technical Research Report No. TR 92-90, Institute for Systems Research, The University of Maryland at College Park, October 1992.

[8] H. HILLION and J.M. PROTH, "Performance Evaluation of Job-Shop Systems Using Timed Event-Graphs", *IEEE Trans. Automat. Contr.*, vol. 34, No. 1, pp. 3-9, January 1989.

[9] K. JENSEN, "Coloured Petri nets and the invariant method", *Theoret. Comput. Sci.*, vol. 14, pp. 317-336, 1981.

[10] S. KIRKPATRICK, C.D. GELATT and M.P. VECCHI, "Optimization by simulated annealing", *Science*, vol. 220, 13 May 1983.

[11] S. LAFTIT, J.M. PROTH and X.L. XIE, "Optimization of invariant criteria for event graphs", *IEEE Trans. on Aut. Control*, vol. 37, No. 5, pp. 547-555, 1992.

[12] M. LUNDY and A. MEES, "Convergence of an annealing algorithm", *Mathematical Programming*, vol. 34, pp. 111-124, 1986.

[13] T. MURATA, "Petri Nets: Properties, Analysis and Applications", Proceedings of the IEEE, vol. 77, No. 4, pp. 541-580, April 1989.

[14] J.M. PROTH and X.L. XIE, "Performance evaluation and optimization of stochastic timed event graphs", to appear in *IEEE Trans. on Aut. Control*, 1993.

[15] C. RAMCHANDANI, "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets", Lab. Comput. Sci., Mass. Inst. Technol. Cambridge, MA, Tech. Rep. 120, 1974.

[16] C.V. RAMAMOORTHY and G.S. HO, "Performance Evaluation of Asynchronous Concurrent Systems using Petri Nets", *IEEE Trans. Software Eng.*, vol. SE-6, No. 5, pp. 440-449, 1980.

[17] W. REISIG, "Petri nets with individual tokens", *Informatik-Fachberichte*, vol. 66, No. 21, pp. 229-249, 1983.

[18] J. SIFAKIS, "A Unified Approach for Studying the properties of Transition Systems", *Theoret. Comput. Sci.*, vol. 18, pp. 227-258, 1982.

[19] M. ZHOU and F. DI CESARE, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*, Kluwer Academic Publisher, Boston, MA, 1993