Supported by the National Science Foundation Engineering Research Center Program (NSFD CD 8803012), Industry and the University

Event Graphs for Modeling and Evaluating Modern Production Systems

by G. Harhalakis, S. Laftit, and J.M. Proth

TR 90-73

EVENT GRAPHS FOR MODELING AND EVALUATING MODERN PRODUCTION SYSTEMS

HARHALAKIS G.(1), LAFTIT S.(2) and PROTH J.M.(3)

ABSTRACT:

Very few Mathematical Tools are available to study the dynamics of discrete manufacturing systems. Petri Nets, and in particular a special type of Petri Nets called Timed Event Graphs, seem to be of special interest for studying discrete manufacturing systems. In this paper, we define Timed Event Graphs and emphasize the properties which are of interest for our purpose.

Modeling job-shop systems as well as assembly systems using event graphs is then explained. The model obtained is a strongly connected event graph whose properties are presented in the first part of the paper. These properties are used to derive the properties of the manufacturing system.

In particular, it can be shown that the productivity of the manufacturing system is defined by the cycle time of the critical circuit in its event graph model. Blocking conditions of the system are also studied.

Finally, we show how to use the previous results to maximize the productivity with a minimal in-process inventory when the sequences of product types are fixed at the entrance of each machine.

Key words :

Petri Nets, Event Graphs, Evaluation of Manufacturing Systems, Discrete Event Systems, Graph Theory.

(1) Department of Mechanical Engineering and Systems Research Center, University of Maryland, USA

(2) INRIA - CESCOM, Technopôle Metz 2000, 57070 METZ Cedex

(3) INRIA - CESCOM, Technopôle Metz 2000, 57070 METZ Cedex and Systems Research Center, University of Maryland, USA

1. INTRODUCTION

In this paper, the performance problem of job-shop and assembly systems with deterministic manufacturing times under cyclic production process is addressed. In both types of systems, the ratios of the various product types to manufacture are given. The set of the ratios is known as the **product mix**. The control of a production system in order to meet the given ratios is obtained by associating a machine sequence to each machine. Such a sequence defines the sequencing of the product types (or the sequencing of the components of the products types, in the case of an assembly system) on a machine. The proportion of a product type (or a component) in a machine sequence tally with the related ratio. If one of the machine sequences does not reflect the corresponding ratio, it has been proved in [1] that the system blocks after a finite period of time.

The modeling of job-shop and assembly systems is based on Event Graphs ([1], [2] and [5]), a special class of Petri Nets. The model obtained is a strongly connected event graph, as shown in [4] for the job-shop systems.

The first goal of this paper is to explain how to model job-shop and assembly systems using event graphs. The second goal is to show that, for both types of production systems, it is always possible to fully utilize the bottleneck machine, and thus to reach the maximal productivity. Finally, the machine sequences being given, we show that the optimal solution to the problem (i.e. the solution which fully utilizes the bottleneck machine with a minimal work-in-process (WIP)) is the solution to an integer linear programming problem. Note that, in real-world production systems and in particular Flexible Manufacturing Systems (FMS), minimizing WIP is the same than minimizing transportation resources such as pallets or carts, which are often very expensive.

The paper is organized as follows. Section 2 is devoted to the presentation of the event graphs and their properties. In section 3, we show how to use event graphs to model job-shop as well as assembly systems. In section 4, we derive the performance evaluation of the production systems from the properties of the event graphs. Finally, we explain how to reach the optimal solution to the system by solving an integer linear programming problem.

2. PETRI NETS AND EVENT GRAPHS

A Petri net is a bipartite directed graph with two types of nodes: the **places** (represented by circles) and the **transitions** (represented by bars). These nodes are joined by **directed arcs**. These arcs can be weighted. In the following, we assume that every arc has a weight equal to 1. The two ends of a given arc are of different types. **Tokens** evolve in the Petri net and represent the dynamics of the system. They are represented by **black dots**. A

Petri net is given in figure 1. In this example, transitions are denoted by t_i ($i = 1, \dots, 6$) and places by p_j ($j = 1, \dots, 6$). Furthermore, p_1 contains two tokens, p_3 one and the other places are empty.

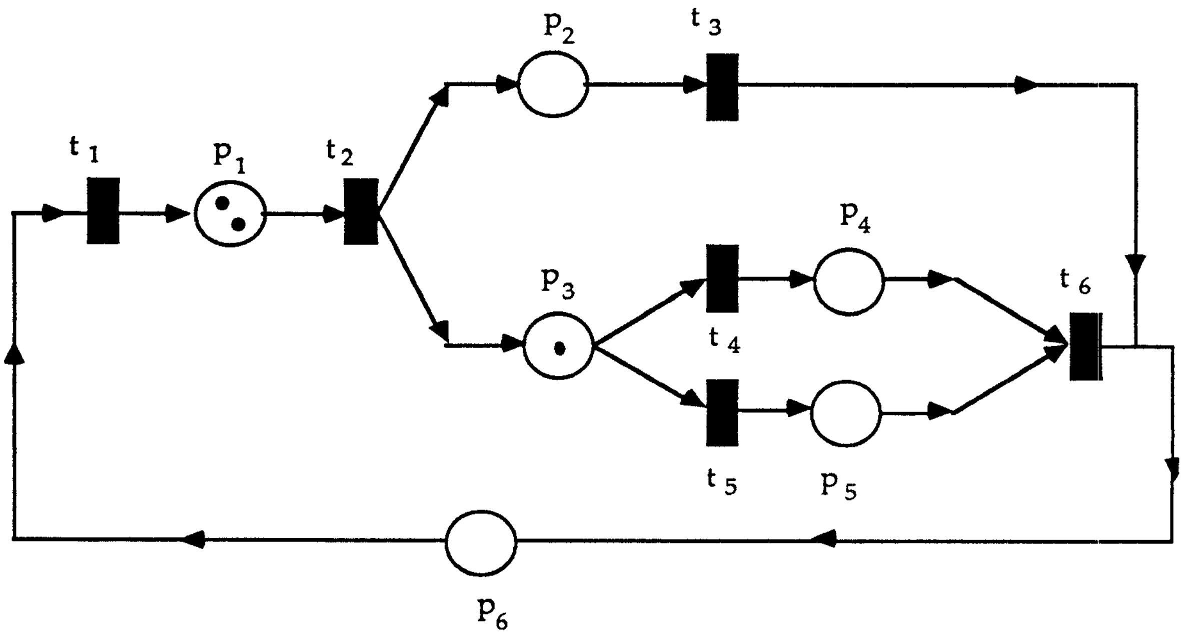

Fig. 1: A Petri net example

Let P and T be respectively the set of places and the set of transitions. A **marking** of a Petri net is a function

$$M: P \to \{0, 1, 2, \dots\}$$

which assigns a non-negative number to each place. Such a number represents the number of tokens in the place. A transition t is **enabled** by a marking M if and only if each of its input places contains at least one token (assuming that the weight of each arc is equal to 1). For instance, in figure 1, t_2, t_4 and t_5 are enabled. The **firing** of a transition consists in removing one token from each input place and adding one token to each output place. In a Petri net, it may arise that several transitions are enabled, but only some of them can be fired. For instance, in figure 1, t_4 and t_5 are enabled, but only one of them can be fired because p_3 contains only one token. In that case, the tokens can evolve in the net only if we add some decision-making processes to make this kind of decision.

It is possible to assign a time to each transition. In that case, the Petri net is called **timed Petri net**. Such a time represents the time taken by the related transition to fire. In a timed Petri net, a firing is initiated by removing one token from each of the input places.

The firing terminates after a period of time equals to the time assigned to the transition, and one token appears in each output place at the instant when the firing terminates.

An **event graph** is a Petri net such that each place has exactly one input transition and one output transition and such that every arc is 1-weighted. Figure 2 gives an example of event graph.

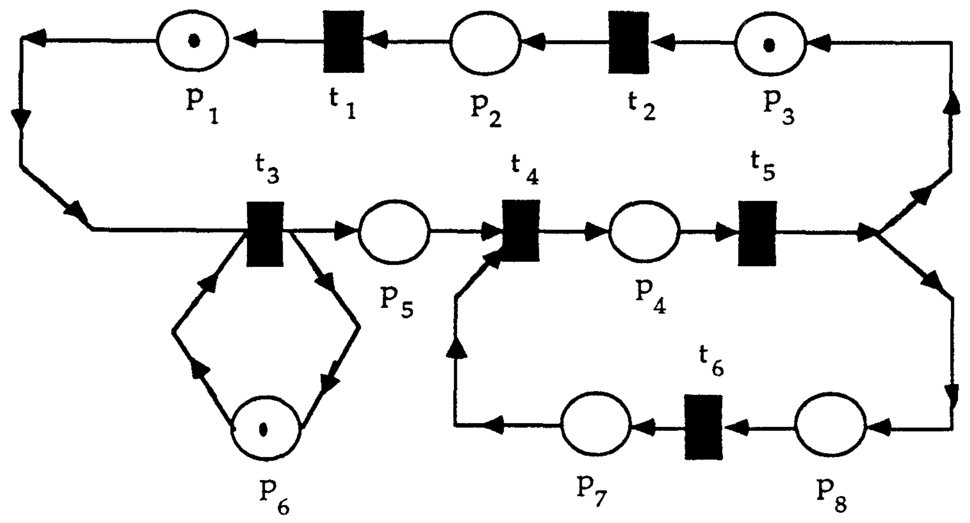

Fig. 2: An event graph

In such a Petri net, we never have to make a choice between several enabled transitions to decide which one to fire. In other words, an event graph is a **decision-free Petri net**. In the following, we consider only timed event graphs.

An **elementary circuit** is a directed path which goes from one transition back to the same transition, and which has at most one time assigned to each transition. For instance, in the event graph given in figure 2, the following circuits are elementary:

$\gamma_1 = \{t_1, p_1, t_3, p_5, t_4, p_4, t_5, p_3, t_2, p_2, t_1\}$

$\gamma_2 = \{t_3, p_6, t_3\}$

$\gamma_3 = \{t_4, p_4, t_5, p_8, t_6, p_7, t_4\}$

The following property has been demonstrated by Commoner and al. [2]:

Property 1:

The total number of tokens in any elementary circuit is invariant by transition firing.

We define the **cycle time** of an elementary circuit as follows:

$$C(\gamma) = \mu(\gamma) / M(\gamma) \qquad (1)$$

where:

$\mu(\gamma)$ is the sum of the times assigned to the transitions of γ

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

According to property 1, $C(\gamma)$ remains constant by transition firing and $C(\gamma)$ can be computed using $M^{\circ}(\gamma)$, initial marking of γ.

Γ being the set of elementary circuits of a strongly connected event graph, the maximum cycle time over all the elementary circuits is:

$$C^{*} = \mathcal{M}\text{ax}_{\gamma \in \Gamma} \; C(\gamma) \tag{2}$$

If $\gamma \in \Gamma$ is such that:

$$C(\gamma) = C^{*}$$

γ is called **critical circuit.**

The following property is due to Chretienne and al. [1]:

Property 2:

Assuming that the transitions fire as soon as they are enabled (i.e. ready to fire) and that the event graph considered is strongly connected, then the firing rate of all transitions in steady state is given by:

$$\lambda = 1 / C^{*} \tag{3}$$

Broadly speaking, λ represents the speed at which the strongly connected event graph evolves. As a consequence, it is possible to increase the "speed" of the event graph by decreasing C^{*}, i.e. by increasing the number of tokens in the critical circuits.

Finally, property 3 also holds:

Property 3:

If one of the elementary circuits does not contain any token, then the system blocks after a finite period of time.

In the next section, we show how to model job-shop and assembly systems using event graphs.

3. MODELING JOB-SHOP AND ASSEMBLY SYSTEMS

For the sake of clarity, we explain separately how to model job-shop and assembly systems using event graphs.

3.1 Modeling job-shop systems

In the following, we explain how to model a job-shop system using the example shown in figure 3.

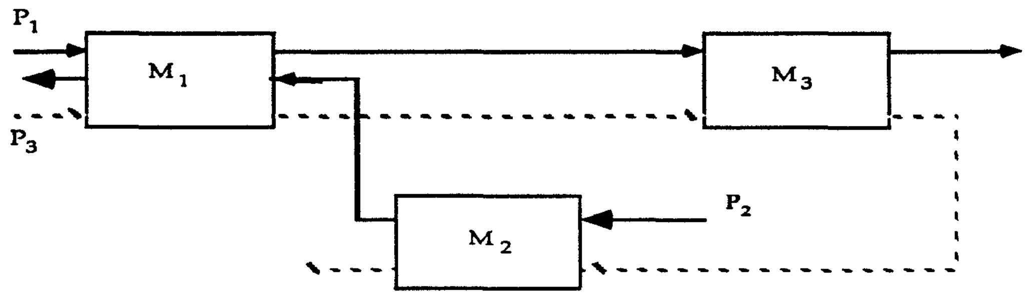

Fig. 3: A job-shop system

The job-shop under consideration is composed of three machines denoted by M_1, M_2 and M_3. Three types of products, denoted by P_1, P_2 and P_3, can be manufactured using this job-shop system. Let us assume that the product mix is (0.25 ; 0.50 ; 0.25), which means that the goal is to manufacture 25% of products of type P_1, 50% of products of type P_2 and 25% of products of type P_3. A sequence which reflects the product mix is P_1, P_2, P_2, P_3. We assign to each element of this sequence a **process circuit** as presented in figure 4.

Each process circuit is an elementary circuit and models the routing of the related product type. Each transition firing corresponds to the execution of an operation, according to the order specified by the manufacturing process. The time assigned to each transition is the time corresponding to the related operation. Each token in a process circuit is a work-in-process (WIP). A place corresponds to a storage buffer and we assume that a new part is launched in the system as soon as a part is completed. Thus, in the process circuits, tokens recirculate indefinitely.

We then model the sequencing of the part types in the machines by means of **command circuits**. A command circuit is an elementary circuit which connects all the transitions that correspond to operations performed on the same machine. In figure 5, we present the command circuit corresponding to machine M_1.

The order of the transitions in command circuits is determined by the input sequence assigned to the corresponding machines.

Note that a command circuit contains only one token, and a token in a command circuit does not represent a part.

We can see that the model is a strongly connected event graph which contains three types of elementary circuits:

- the command circuits
- the process circuits
- the mixed circuits, that include nodes of both processing and command circuits.

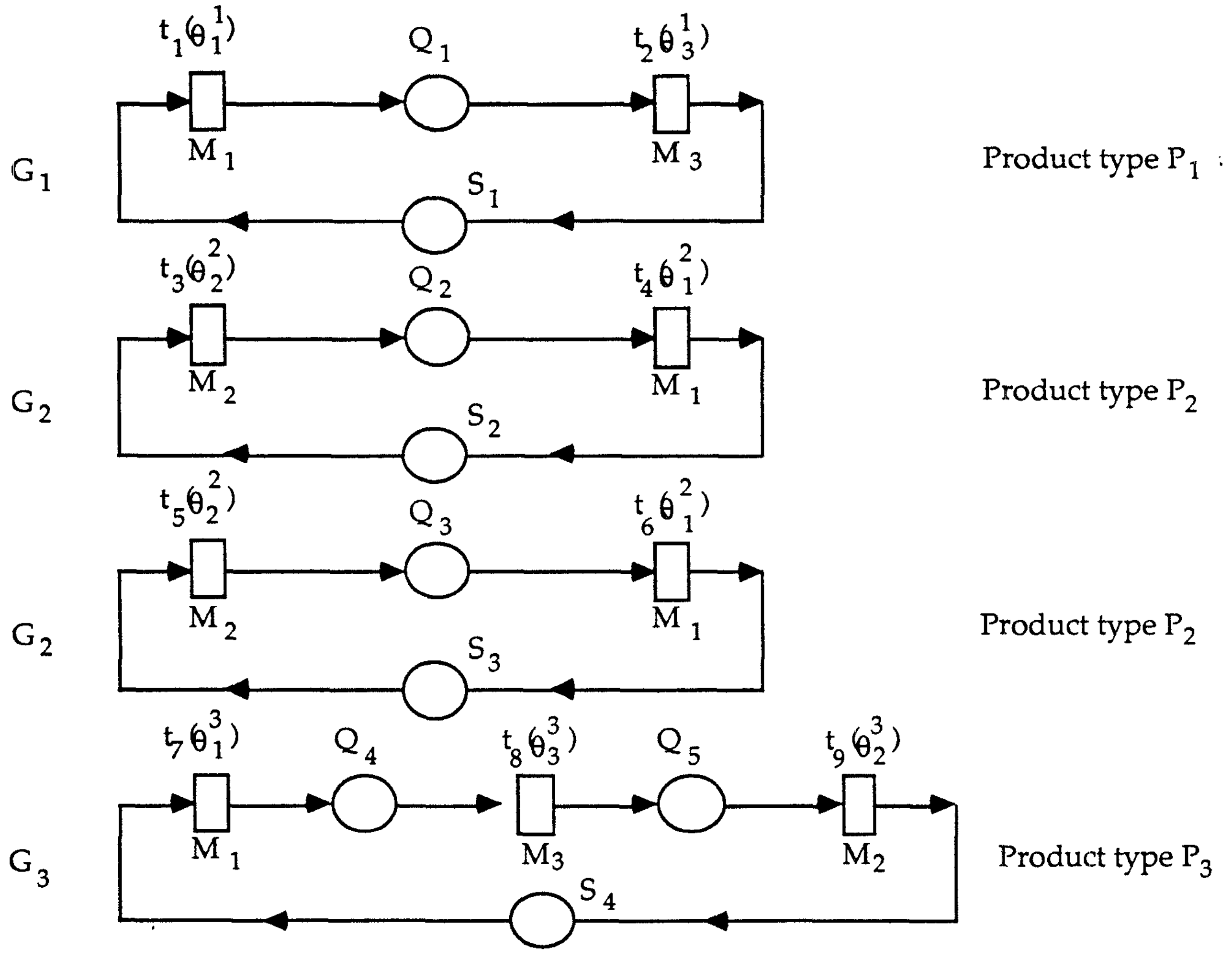

Fig. 4: The process circuits

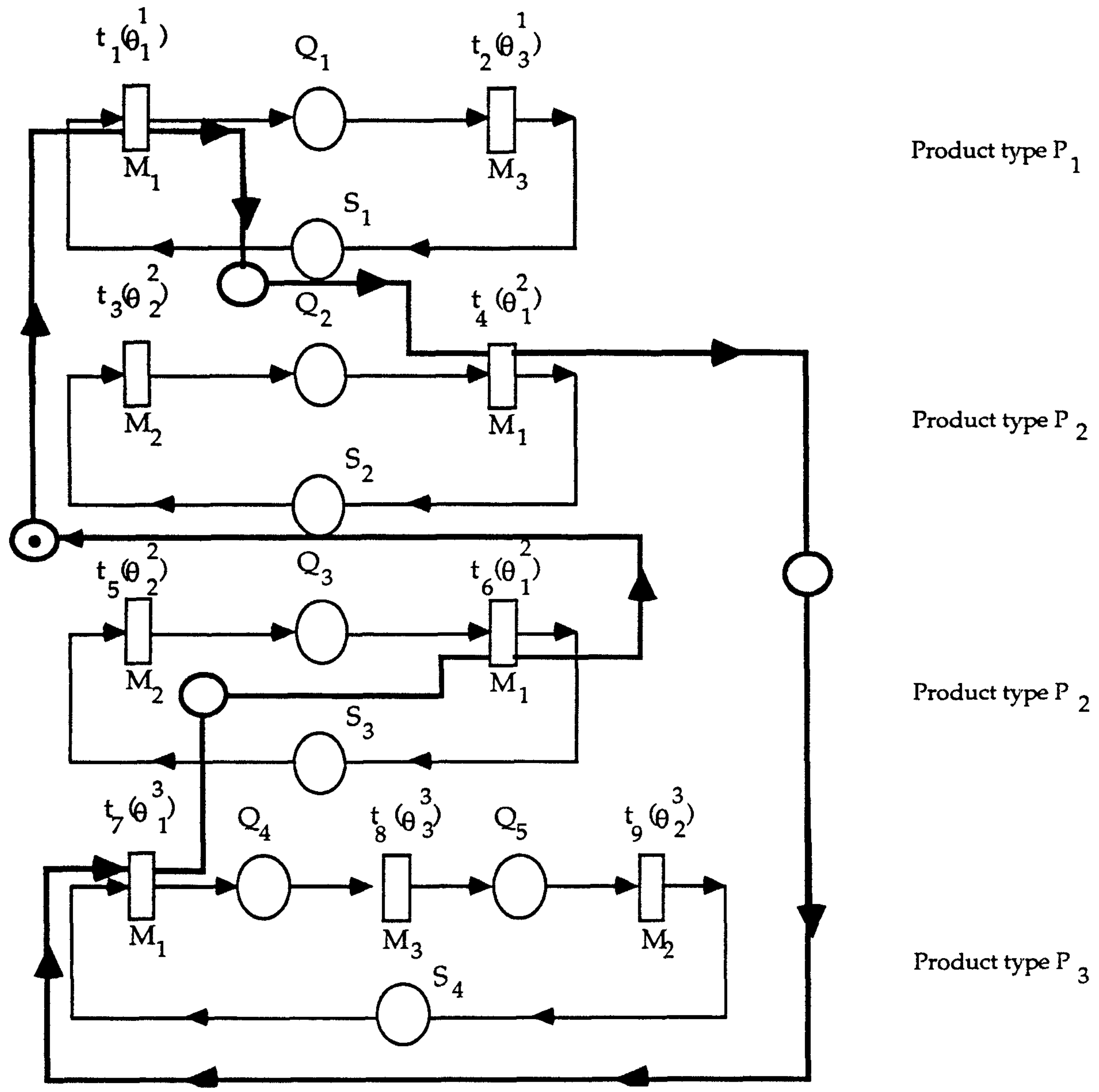

Fig. 5: The job-shop model

3.2. Modeling assembly systems

In this section, we use again a small example to explain how to model an assembly system. Let us consider for instance two product types P_1 and P_2 whose manufacturing processes are presented in figure 6.

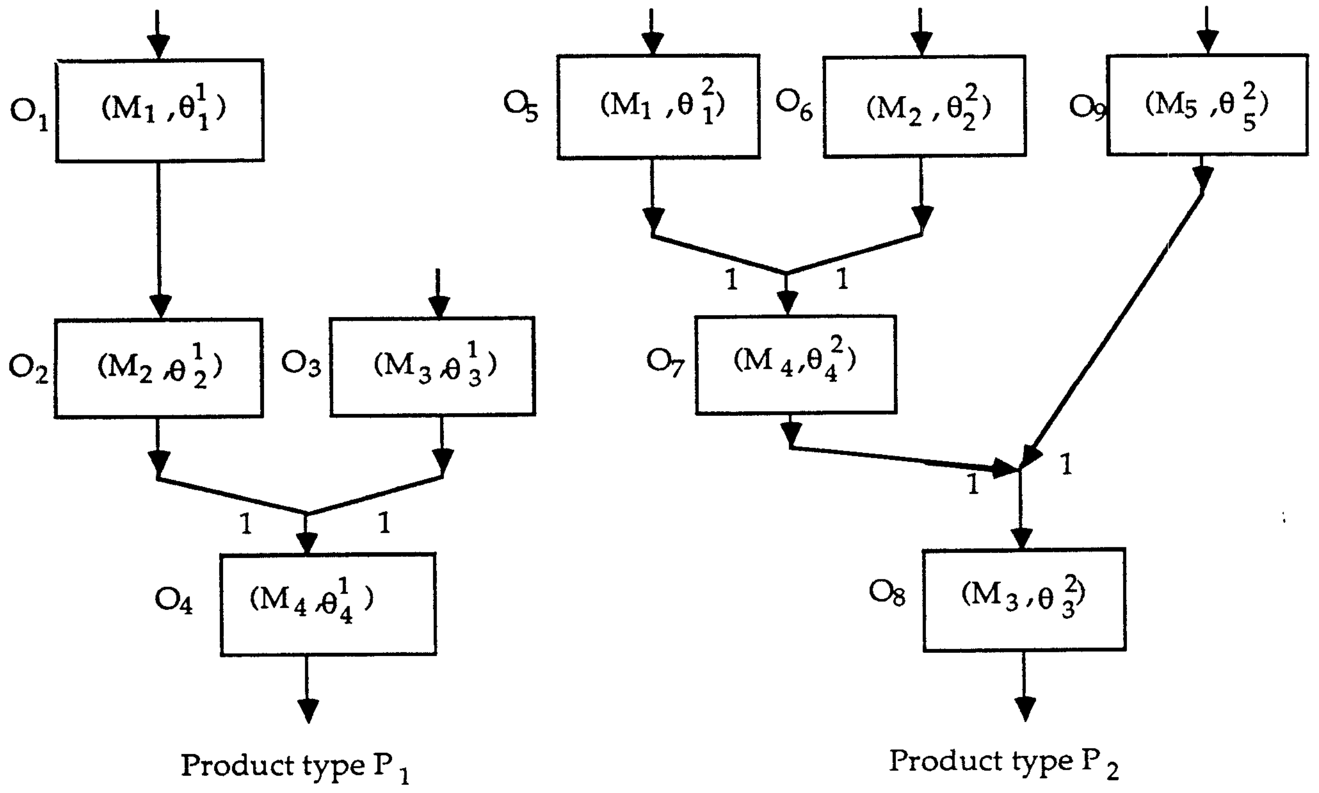

Fig. 6: Product Types P_1 and P_2

In figure 6, M_i (i = 1, 2, 3, 4, 5) are the machines, O_i (i = 1, ..., 8) represent operations and θ_i^j represent the manufacturing times. Two types of operations are involved in the manufacturing process. Operations such as O_1, O_2 just transform a component of the final product. In the following, we refer to these operations as **regular operations**. Another type of operations are the **assembly operations**. For instance, O_4, O_7 and O_8 are such types of operations. An assembly operation puts together components to obtain a more complex component of a final product or the final product itself. In figure 6, the integer values assigned to the arrows represent the number of components of each type of product required to obtain one unit of the next component.

In figure 7, we present the models related to the previous manufacturing processes.

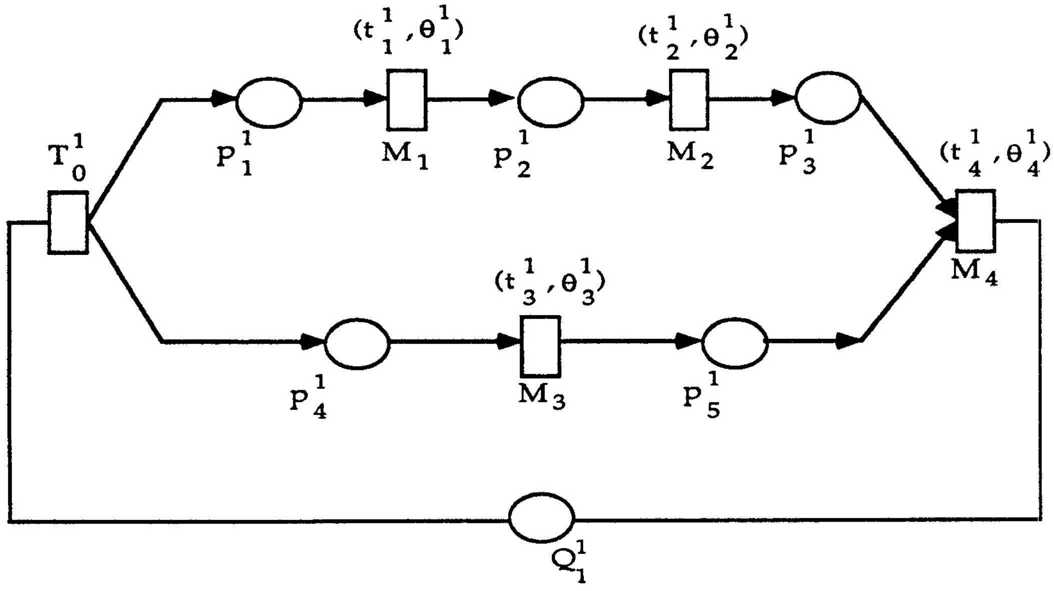

Fig. 7.1: Model related to manufacturing process P_1

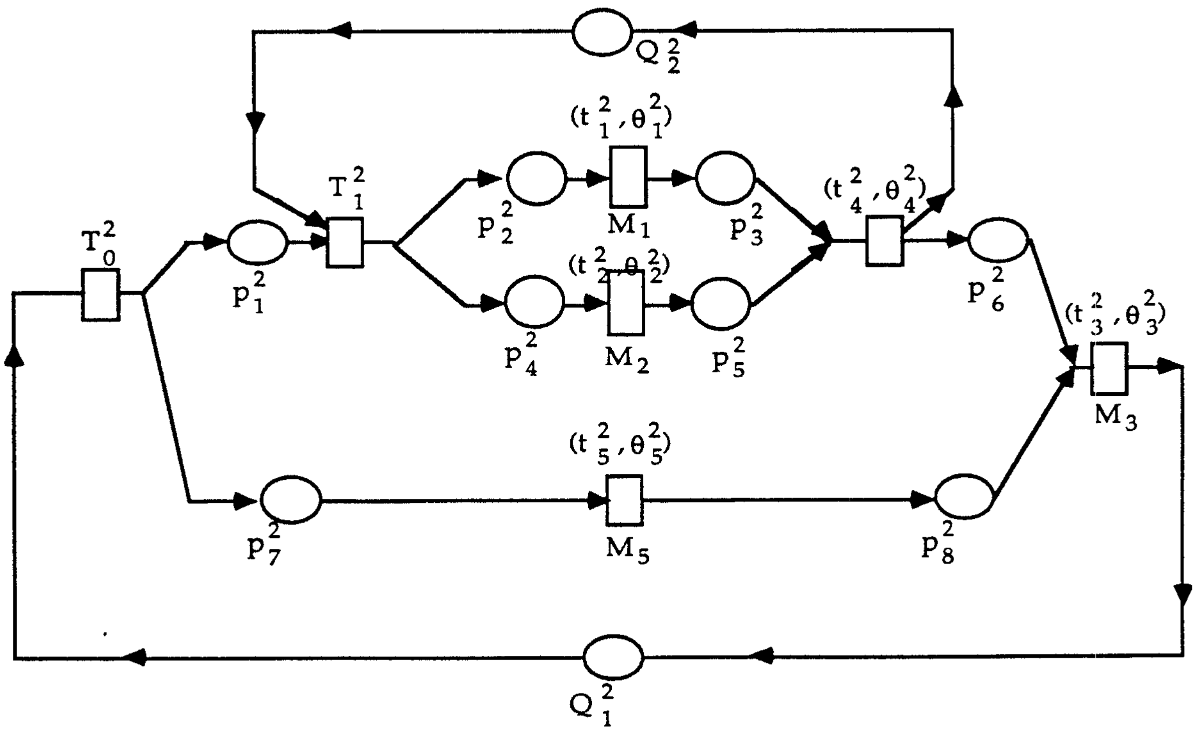

Fig. 7.2: Model related to manufacturing process P_2

Note that these models make use of transitions which do not represent operations but the beginning of the manufacturing of components which are required to assemble a more sophisticated component or the final product. In figure 7.1, it is the case of T^1_0. It is also the case of T^2_0 and T^2_1 in figure 7.2. We call these transitions **launching transitions**. A directed path connects each transition representing an assembly operation to the related launching transition. In practice, it means that the transportation resources used to carry the components at this level of complexity recirculate as soon as the assembly operation is completed. Thus, transportation resources are assigned to each level of manufacturing.

The manufacturing process models correspond to the process circuits in the case of job-shops. These models are completed by command circuits which model the sequencing of the part types in the machines. In figure 8, we represent the command circuit related to machine M_1, assuming that the sequencing related to M_1 is (P_1, P_2, P_2).

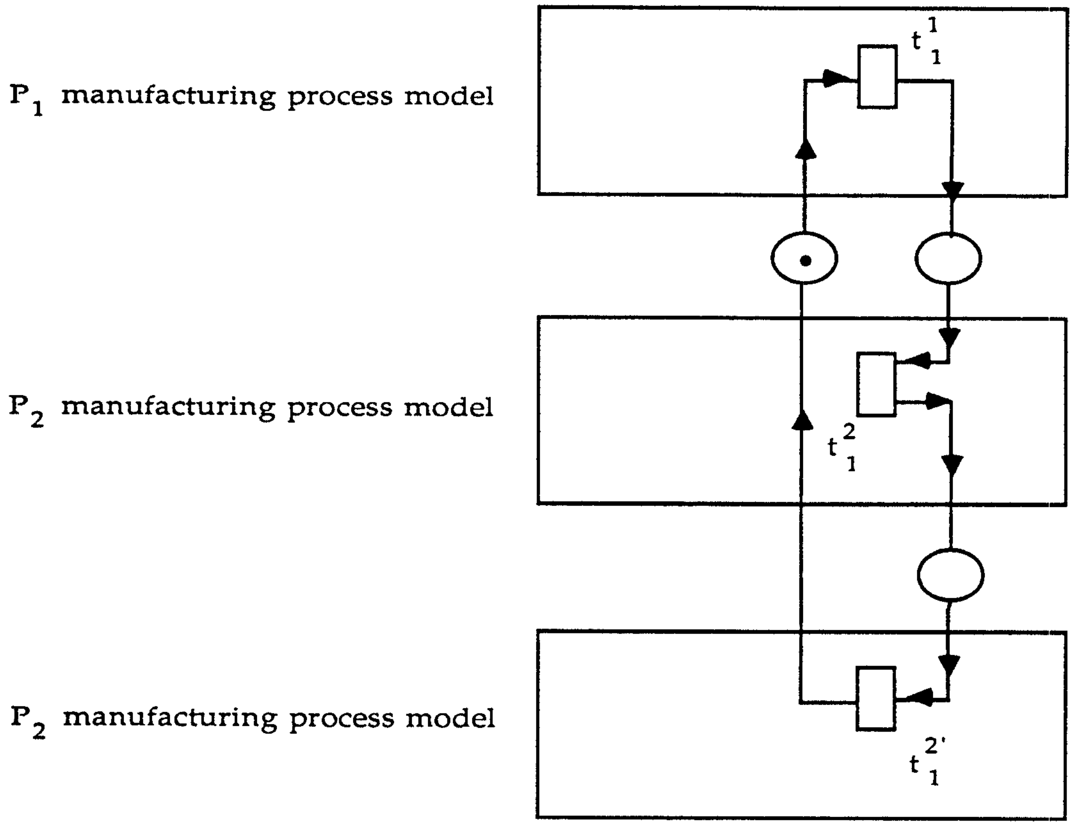
P_1 manufacturing process model
P_2 manufacturing process model
P_2 manufacturing process model
t_1^1
t_1^2
$t_1^{2'}$

Fig. 8: The command circuit related to M_1

Each command circuit contains only one token, like in the job-shop case.

Using the modeling process just described on a small example, we obtain a strongly connected event graph, like in the case of a job-shop.

Each manufacturing process model contains several elementary circuits. For instance, the model related to the manufacturing process P_1 (see figure 7.1) contains two elementary circuits:

$$\gamma_1 = \{p_4^1, t_3^1, p_5^1, t_4^1, Q_1^1, T_0^1, p_4^1\}$$

$$\gamma_2 = \{p_1^1, t_1^1, p_2^1, t_2^1, p_3^1, t_4^1, Q_1^1, T_0^1, p_1^1\}$$

The model related to the manufacturing process P_2 (see figure 7.2) contains the following elementary circuits:

$$\gamma_3 = \{T_1^2, p_2^2, t_1^2, p_3^2, t_4^2, Q_2^2, T_1^2\}$$

$$\gamma_4 = \{T_1^2, p_4^2, t_2^2, p_5^2, t_4^2, Q_2^2, T_1^2\}$$

$$\gamma_5 = \{T_0^2, p_1^2, T_1^2, p_4^2, t_2^2, p_5^2, t_4^2, p_6^2, t_3^2, Q_1^2, T_0^2\}$$

$$\gamma_6 = \{T_0^2, p_1^2, T_1^2, p_2^2, t_1^2, p_3^2, t_4^2, p_6^2, t_3^2, Q_1^2, T_0^2\}$$

$$\gamma_7 = \{T_0^2, p_7^2, t_5^2, p_8^2, t_3^2, Q_1^2, T_0^2\}$$

Others elementary circuits are the command circuits and the mixed circuits, like in the job-shop case.

4. PERFORMANCE EVALUATION OF THE PRODUCTION SYSTEMS

Let us consider an event graph model of a job-shop or an assembly system. Such a system blocks after a finite period of time if one of the elementary circuits does not contain any token (see property 3). If C* is the cycle time related to the critical circuit, $\lambda = 1 / C^*$ is a parameter defining the average productivity of the system: on the average, the system produces one sequence of products reflecting the product mix every λ–period (see property 2).

Finally, if the initial marking of the event graph is given, it is possible to compute the productivity of the related manufacturing system or to see if it will block after a finite period of time, providing that the elementary circuits have been identified. Note that the initial marking is known as soon as the initial work-in-processing and the machine sequences are known.

Algorithms to compute the elementary circuits are given in [5] and [6].

5. OPTIMIZATION OF THE PRODUCTIVITY

The productivity of a manufacturing system is maximal if one of the command circuits of the related model is a critical circuit. Thus, optimizing the productivity consists in adding as few tokens as possible in the elementary circuits (except in the command circuits which contain only one token) in order to make critical a command circuit.

Let $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ be the set of elementary circuits *except the command circuits* and $P = \{p_1, p_2, \ldots, p_q\}$ the set of places in the event graph model.

γ being an elementary circuit, we denote by $\mu(\gamma)$ the sum of the times assigned to the transitions of γ.

If Γ_c is the set of command circuits, let:

$$M = \mathcal{M}\text{ax}_{\gamma \in \Gamma_c} \; \mu(\gamma)$$

and, for i = 1, 2, ..., n:

$$n_i = \lceil \mu(\gamma_i) / M \rceil$$

where $\lceil x \rceil$ is the smallest integer greater than or equal to x.

n_i is then the minimal number of tokens to be put in the elementary circuit $\gamma_i \in \Gamma$ in order that γ_i be not critical.

We define:

- the matrix $A = [a_{ij}]$; $i = 1, \ldots, n$; $j = 1, \ldots, q$

where:

$$a_{ij} = \begin{cases} 0 \text{ if } p_j \text{ does not belong to } \gamma_i \\ 1 \text{ otherwise} \end{cases}$$

- the matrix $X = [x_1, \ldots, x_q]^t$, where x_i is the number of tokens in p_i
- the matrix $N = [n_1, n_2, \ldots, n_n]^t$

The problem to solve is as follows:

$$\text{Minimize} \sum_{j=1}^{q} x_j$$

s.t.

$$\begin{cases} A\,X \geq N \\ x_j \geq 0 \text{ and integer for } j = 1, \ldots, q \\ \text{the } x_j \text{ corresponding to places belonging to command circuits are known} \end{cases}$$

In [3] an efficient heuristic algorithm has been proposed to solve this integer LP-problem.

The optimal solution of the previous integer LP-problem is the initial marking of the strongly connected event graph. This initial marking corresponds to the initial WIP of the production system. Some of these work-in-process may not actually exists. In that case, we introduce artificial WIP, and the system reaches its steady state when all these artificial WIP are removed from the system.

The problem with the previous algorithm is that the number of elementary circuits (i.e. the value of the parameter n) grows very fast with the size of the model. As a consequence, only quite small systems can be handled using this approach.

Another approach is currently under development. This approach also leads to an integer LP program, but the number of constraints is equal to the number of places. A step by step heuristic algorithm is under development around this new approach. It will allow to handle models with thousands of places.

6. CONCLUSION

In this paper, we have explained how to model job-shop and assembly systems using event graphs. In the case when the demand is expressed as a product mix, we show that models are always strongly connected event graphs. Thus, the initial state of the system being given, we can derive the behavior of the manufacturing system from the properties of the strongly connected event graph. We also explain how to reach the maximal productivity with a minimal number of parts in process, the machine sequences being given.

The critical point in the previous work is the computation of the elementary circuits whose number increases very fast with the size of the model. A research currently in progress investigates how to avoid the computation of these elementary circuits.

BIBLIOGRAPHY

[1] CHRETIENNE P.,
"Les réseaux de Petri temporisés", Université Paris VI, Thèse d'Etat, 1983.

[2] COMMONER F., HOLT A., EVEN S. and PNUELI A.,
"Marked Directed Graphs", Journal of Computer and System Science, vol. 5, n° 5, 1971.

[3] HILLION H.P. and PROTH J.M.,
"Analyse de fabrications non linéaires et répétitives à l'aide de graphes d'événements temporisés", RAIRO, vol. 22, n° 2, Septembre 1988.

[4] HILLION H.P. and PROTH J.M.,
"Performance Evaluation of Job-Shop systems Using Timed Event Graphs", IEEE Transactions on Automatic Control, vol. 34, n° 1, Janvier 1989.

[5] RAMAMOORTY C.V. and HO G.S.,
"Performance Evaluation of asynchronous concurrent systems using Petri nets", IEEE Trans. Software Eng., vol. SE-6, n° 5, pp. 440-449, 1980.

[6] RAMCHANDANI C.,
"Analysis of asynchronous concurrent systems by timed Petri Nets", Lab. Computer Science, MIT, Cambridge, MA, Tech. Rep. 120, 1974.