Division of Research
School of Business Administration

March 1988
Revised, October 1989

DYNAMIC ANALYSIS OF REPETITIVE DECISION-FREE DISCRETE EVENT PROCESSES: APPLICATIONS TO PRODUCTION SYSTEMS

Working Paper #543-b

Didier Dubois
Université Paul Sabatier

Kathryn E. Stecke
The University of Michigan

The University of Michigan
School of Business Administration
Ann Arbor Michigan 48109

ABSTRACT

This paper is the second part of a work devoted to timed marked graphs, their underlying algebra, and their applications to model the dynamic behavior of production systems in which a set of activities is performed repetitively in fixed sequences. Theoretical aspects are considered in the first part. In addition, an algorithm to properly characterize the periodic behavior of such systems is described. The relevance of this class of models and the associated solution algorithm to the performance evaluation of certain types of flexible manufacturing systems is emphasized here. Examples of various modeling capabilities are provided. The analysis of timed marked graphs is an extension of PERT/CPM-like techniques to asynchronous and repetitive processes, whose graph-theoretic models contain cycles of activities.

1. INTRODUCTION

In a companion paper (Dubois and Stecke [1989]), a model of decision-free, repetitive, discrete-event systems is proposed, using the formalism of timed marked graphs, i.e., a special class of Petri nets. A particular {max,+} algebra is described, in which the dynamical behavior of such systems can be captured by sets of linear recurrence equations. Marked graphs containing cycles, which express repeated sets of activities, have been studied. They exhibit periodic behavior in the long run and this periodic behavior can be analytically characterized by algebraic concepts such as eigenvalues and eigenvectors. Earliest and latest starting times of activities in steady state can be computed (up to a translation in time). Critical circuits of activities and sets of critical resources can be displayed.

In a marked graph representation, nodes represent activities and arcs express precedence relationships between activities. Tokens added to arcs can represent resources which are needed to perform the activities. These conventions, discussed in Cohen et al. [1985] and Dubois and Stecke [1983, 1989], are applied here to model repetitive production processes and to compute steady state periodic schedules. Applying timed Petri nets to scheduling problems is also investigated by Chretienne [1983] for a larger class of models with less algebraic structure.

In their most general case, timed Petri net models can only be evaluated using simulaton. Some applications of these general models to flexible manufacturing systems can be found in Alanche et al. [1984], Alla et al. [1984], and Wadhwa and Browne [1989]. In this paper, the timed marked graph models are comparable to a subclass of timed Petri nets, called decision-free. Analytic methods can evaluate these models.

This paper illustrates the theoretical results obtained in Cohen et al. [1985] and Dubois and Stecke [1989]. We indicate that computationally efficient techniques exist to carry out the algebraic analysis on a computer. For example, Chiola [1985] describes a

software package to analyze a different model, a generalized, stochastic Petri net model. These techniques can be useful for performance evaluation purposes.

The paper is organized as follows. §2 reviews the necessary definitions and notation. §3 presents some applications of the models and techniques described in Dubois and Stecke [1989] to some classes of flexible manufacturing systems (FMSs), namely those that exhibit a deterministic and periodic behavior. Such behavior can be due to a fixed part input sequence which is itself periodic as well as fixed part sequences on machine tools. We show that adding resources to an FMS (for instance, additional fixtures for parts) may require a modification of the topology of the corresponding marked graph model. This modification can easily be automated. Examples are provided to illustrate the concepts and models. §3.2 addresses finite buffers to show that if storage is limited in an FMS, the resulting blocking effects can be captured by the marked graph models in some cases, namely flow shop-like FMSs with sequential local buffer storage in front of machines. An example of a limited storage FMS demonstrates some potential limitations of the algebraic approach.

An extensive example in §4, based on an industrial case study, demonstrates the power of the marked graph model for performance evaluation purposes. In §5, the conclusions stress an analogy of the approaches described here with the modeling of Markovian processes. Directions for future research, both theoretical and from the point of view of FMS performance evaluation, are provided.

2. TIMED MARKED GRAPHS

A timed marked graph can be translated into a system of linear equations in the sense of an algebra where the maximum and addition operations, respectively, represent addition and product operations, respectively, in the usual linear algebra. The approach builds on previous work (Cohen et al. [1985]) in the modeling and analysis of discrete-event systems in the field of manufacturing.

2.1 A Refresher on Marked Graphs

A marked graph is a directed graph G where some arcs are assigned a number of tokens. It is a special type of Petri net (Peterson [1981]), where each place has exactly one input and one output transition. Places in such Petri nets correspond to arcs of the marked graphs, while transitions correspond to vertices. In the following, we use the terms vertex or transition interchangably, as well as the terms place and arc.

The firing of a vertex is enabled as soon as each input arc (place) of this vertex contains at least one token. Vertices without an input arc are enabled unconditionally. The firing consists of removing one token from each input arc and adding one token to each output arc of the vertex. The system evolves over time in this manner.

The vector $M = (r_1,...,r_p)$ (where r_α, $\alpha = 1,...,p$, is the number of tokens on arc number α) is called a marking. Since α may refer to arc (i,j), the terms r_α and r_{ij} are used interchangably. Firing transitions when enabled produces sequences of markings. Marked graphs are useful to describe the behavior of concurrent, asynchronous, deterministic processes. Marked graphs do capture concurrency phenomena but are decision-free: decisions on how the system should evolve have been made in advance. The consequences of these decisions can then be evaluated.

Let V be the set of vertices and Å be the set of arcs. When an enabled vertex i fires, the marking M evolves into M' such that

$$r'_\alpha = r_\alpha + 1, \quad \text{if } \alpha \text{ refers to (i,j) for some j;}$$

$$r'_\alpha = r_\alpha - 1, \quad \text{if } \alpha \text{ refers to (j,i) for some j;}$$

$$r'_\alpha = r_\alpha, \quad \text{otherwise,}$$

where (i,j) is the arc from vertex i to vertex j, if it exists. M' is said to be immediately reachable from M by i.

A chain in G is a sequence of adjacent vertices. A path in G is a chain, $i_1,...,i_n$, where $(i_j, i_{j+1}) \in$ Å, $\forall j = 1,...,n-1$. When $i_1 = i_n$, it is a circuit. When all i_j are different except for $i_1 = i_n$, the chain is called a simple circuit. G is said to be connected if there is a chain between any two vertices and strongly connected if there is a path between any two vertices. A graph containing no circuit is called acyclic.

2.2 Modeling Conventions for Timed Marked Graphs

Marked graphs can capture the time dimension by assigning durations to arcs and/or vertices and are then more useful to model a class of discrete-event dynamic systems. These systems are described by a set of activities, each of which requires a set of resources in order to be performed. Any resource that is engaged in some activity cannot be simultaneously used for another. In a manufacturing environment for example, activities include the processing operations of parts on machines as well as transportation, storage, and machine set-up activities. Resources can be parts, pallets, carts, robots, machines, cutting tools, and the like. Each activity requires a certain amount of time during which some of the resources are captured. A decision-free, discrete-event system is one that:

(i) for each resource, the set of activities in which it is involved is linearly ordered, so that when this resource is liberated by some activity, it is known which activity captures it next;

(ii) for each activity, the set of resources it captures is also ordered, so that resources do not compete.

In terms of Petri nets (i.e., see Peterson [1981]), activities can be viewed as transitions. The input places of each transition at times contain tokens, which can model resources that are required to start the activity. Condition (i) specifies that no place is an input to more than one transition. Condition (ii) means that no place is an output place of more than one transition. Hence we have the structure of a marked graph. Note that condition (i) specifies a deterministic system after some chosen systematic rule (such as first-come, first-served) has been used to solve the conflicts between resources. Condition (ii) is also necessary to obtain the useful structure of a marked graph.

3. APPLICATIONS TO THE MODELING OF PRODUCTION PROCESSES

In this section, we discuss the usefulness of the algorithms and algebraic tools described in the companion paper (Dubois and Stecke [1989]) for the analysis of production processes. Examples to demonstrate these applications are provided.

First, some results obtained in Dubois and Stecke [1983] and Cohen et al. [1985] for flexible manufacturing systems (FMSs) are reviewed in §3.1. Then in §3.2, other modeling and evaluation problems are discussed in the framework of the marked graph representation. In particular, the problem of blocking effects due to limited storage space is discussed. An extensive example is provided in §4. Finally, the limitations of the suggested model are acknowledged in §5. Open problems are mentioned. Additional scheduling applications of the $\{max,+\}$ algebra can be found in Cunninghame-Green [1979].

3.1 Modeling Flexible Manufacturing Systems That Have Periodic Behavior

Timed marked graphs and their algebraic representation are well suited to study the dynamic behavior of repetitive production processes as encountered in some automated manufacturing systems. This type of production system exists in machining, for the medium and large volume production and assembly of metal parts. Production is unit per unit, as opposed to continuous or batch processing. It concerns types of FMSs ranging from flexible transfer lines (flow shop-like structures where parts are allowed to skip one or several machines) to automated job shops (where parts come and go according to their processing sequences and are carried via a network of automated transporters). See Browne, Dubois, Rathmill, Sethi, and Stecke [1984] for more detailed descriptions of these.

Parts are fixtured on usually identical pallets in order to facilitate a precise, automatic positioning of parts on a machine tool. The number of fixtures (of different

types) allocated to each part type is limited, because fixtures are usually part type-dedicated, bulky, and expensive.

In flexible manufacturing systems, each operation requires several cutting tools which are located in limited capacity tool magazines. Then tool management for machines is organized to eliminate set-up times (associated with retooling between the production of different part types) as much as possible so that batch processing is not necessary. Bruno and Biglia [1985] suggest a timed Petri net to model tool movement between a central warehouse and the machine tools in an FMS.

Moreover, machines are able to adapt to the simultaneous production of several part types by using part recognition devices or computer communication via local networks. As a consequence, the production requirements can be expressed in terms of proportional part mix ratios (called minimal part set) to be followed over time. Alternatively, part mix ratios can be selected independent of the production requirements (see Stecke and Kim [1988]). The production rate is affected by the performance and number of available machines, by the part mix ratios, and by the number of available fixtures of different fixture types.

One way of controlling the production flow is to force a periodic input of parts in the system. This can be done by clustering the set of parts to produce into identical input sequences of multiple part types, where each sequence is devised according to the part mix ratios (see Hitz [1980], Erschler et al. [1982], Stecke [1985], and Stecke and Kim [1988, 1989]). If J is the set of part types and I is the basic input sequence to be repetitively used, I can be constructed by a concatenation of elements of J, in some multiple linear ordering. The numbers $k_1,...,k_{|J|}$ of parts of types 1,2,...,|J| define the part mix ratios. To each part type is associated a processing sequence, S_J, consisting of a sequence of machines to visit.

Operations on machines are the activities of the discrete event dynamic system describing the repetitive production process obtained by duplicating the input sequence over time. Each activity requires a machine and a fixtured part in order to be performed. The set of limited resources consists of machines, pallets, and fixtures rather than parts. Raw

castings are available in an input buffer. Each pallet and fixture combination follows the path described by the corresponding processing sequence. Then it comes back to the fixturing station where it receives a new raw casting of either the same type or of another type that can be held by the same fixture. It is then sent back into the FMS according to the order described by the input sequence.

Such a production process is easily described by a marked graph, where nodes are activities (manufacturing operations), arcs express precedence constraints due to either processing sequences or the ordering of parts on machines, and tokens are machines and fixtures. This representation requires that prescribed sequences of parts on machines are known.

This type of behavior can be contemplated for medium or large volume, simultaneous production of various part types. Sometimes it is also relevant for small volume production, where the number of available fixtures for each part type can be small (i.e., one) because of the large variety of part types. Then in order to maximize machine utilizations, one should build part input sequences containing no identical successive parts. Then there is no need to wait for fixtures before launching new parts into the system.

A detailed account of the algebraic model of a periodic input FMS with any number of fixtures per part type and a general job shop structure is available in Cohen et al. [1985]. Each node represents an operation on a part of type j carried by pallet i and performed on machine m. This operation has two predecessors which are, respectively, the previous operation on the same part in its processing sequence and the previous operation on another part on the same machine. Arcs can be weighted by set up times (when the arc links two nodes pertaining to the same machine) or travel times (when the arc links two nodes pertaining to the same part). The initial state of the FMS is described by an initial token allocation.

The marked graph accounts for one or several part input sequences according to the number of available fixtures. In particular, let k_j be the number of parts of type j in an input

sequence and let p_j be the number of fixtures for type j. For $i = 1,...,k_j$, let n_{ij} be the rank of the i^{th} part of type j in the input sequence. A proper management of the fixture set requires that fixtures be allocated to the sequence of parts of the same type in a first-come-first-served manner. More specifically, if there are two type j fixtures, then the first fixture is for parts numbered n_{1j}, n_{3j},...,$n_{2k+i,j}$, etc.... If there are three type j fixtures, then the first fixture is for parts numbered $n_{3k+1,j}$, for $k \geq 0$; the second fixture is allocated to parts numbered $n_{3k+2,j}$, for $k \geq 0$; and the third fixture is for parts numbered $n_{3k,j}$, for $k \geq 1$. In other words, two consecutive parts of the same type should not be assigned to the same fixture if possible. Otherwise the second part will have to wait for the fixture to finish. These remarks determine the topology of the marked graph model (especially for marked arcs) and its size.

In general, let $q_j = p_j:k_j$ (euclidian division). It is easy to see that the marked graph model of an FMS describes at least one part input sequence if $p_j \leq k_j$ and at least q_j+1 sequences if $0 < p_j-q_jk_j \leq k_j$. Let N_j be the number of input sequences required to allocate the fixture-tokens of type j. The total number of input sequences in the marked graph model is the max $\{N_j\}$.

The following example illustrates these modeling conventions.

EXAMPLE 1

Consider a flow shop-like FMS composed of three machines: M_1, M_2, and M_3, as shown in Figure 1. The production plan is for three part types (|J|=3) to be produced in the part mix ratios: 1/6, 1/3, and 1/2. Each part type visits some subset of the three machines, in the order indicated by Figure 1. The smallest part input sequence accounting for these mix ratios is obtained using six parts, with $k_1 = 1$, $k_2 = 2$, and $k_3 = 3$. Let $p_1 = 2$, $p_2 = 3$, and $p_3 = 2$ provide the number of available fixtures of each type. An example of a part input sequence is: (1,3,2,3,2,3). For this input sequence, $n_{11} = 1$, $n_{12} = 3$, $n_{22} = 5$, $n_{13} = 2$, $n_{23} = 4$, and $n_{33} = 6$.

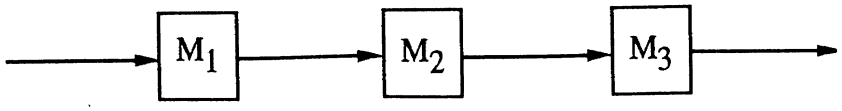


Figure 1. A Three-Machine FMS.

For part type one, $q_1 = 2$. Each of the two fixtures is assigned to a part every other sequence, i.e., $N_1 = 2$. For part type two, $q_2 = 1$, but $p_2 - q_2k_2 = 1$. If fixture one is assigned to the first part of type two in the input sequence numbered n, it will be assigned to the second part of type two in the input sequence numbered n+1, and so on. $N_2 = 2$. For part type three, $q_3 = 0$, but $p_3 < k_3$ and only one input sequence is concerned. The first pallet is for parts ranked 2 and 6 in the part input sequence number n and the second pallet is for part number 4. Their role is exchanged for input sequence number n+1.

Two input sequences are necessary to describe the behavior of the FMS. Figure 2 matches the individual parts with fixtures over two input sequences. Arrows indicate re-using the same pallet, and dotted arrows indicate fixture-part assignment for the subsequent part input sequences (the 3rd or the 4th sequences).

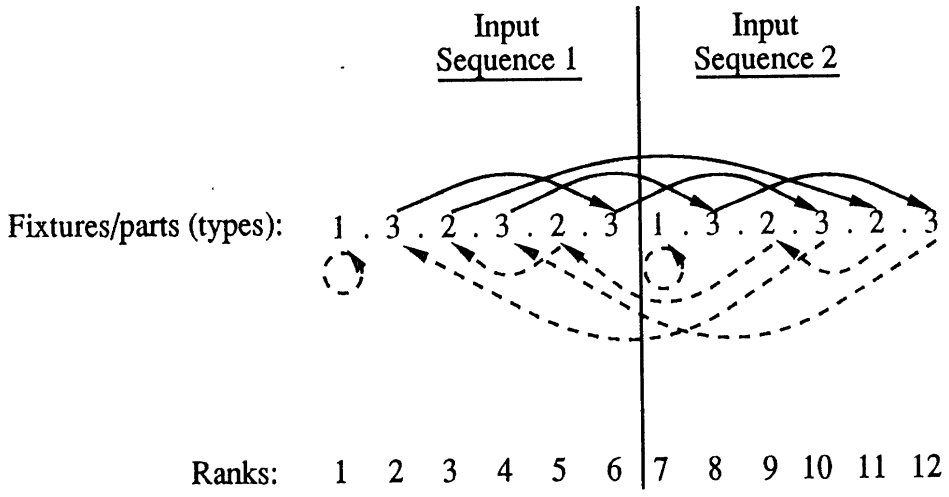


Figure 2. Matching Parts with Fixtures.

Let the processing sequences of each part type on the three machines be given as follows.

Part Type 1: $M_2\ M_3$

Part Type 2: $M_1\ M_2\ M_3$

Part Type 3: $M_1\ M_2$.

For example, part type 1 visits first machine 2, then machine 3, then exists the FMS, as indicated in Figure 1. The ordering of parts on each machine follows the part input sequence. Parts do not pass each other.

The marked graph model of the system is depicted in Figure 3.

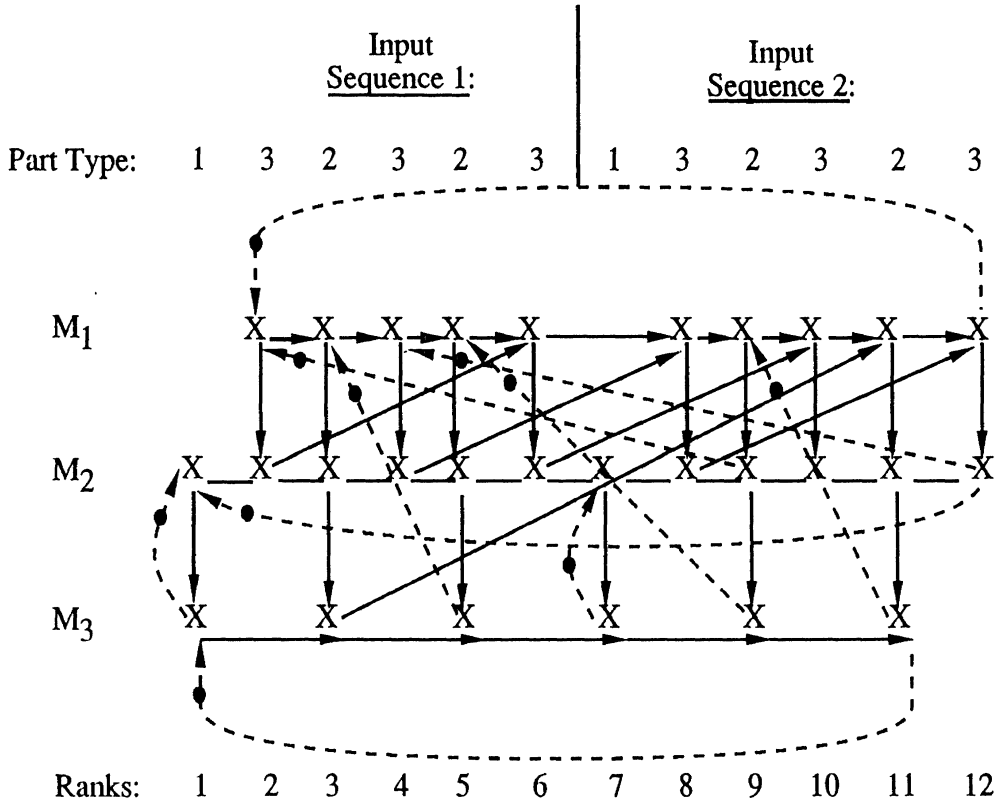


Figure 3. The Marked Graph.

Let (i,j) be the node corresponding to the processing of the part of rank j on machine M_i. Then for Figure 3,

- •Top-down vertical arcs express the processing sequence. These arcs are top-down here only because this example has a flow shop structure.
- •Right-to-left horizontal arcs express the part sequences on machines.
- •The diagonal and dotted arcs are constructed from the arrows and dotted arrows, respectively, of Figure 2.
- •Diagonal arcs express a reutilization of a fixture within the N=2 part input sequences that define the marked graph of the process (e.g., (2,2)→(1,6)).
- •Dotted arcs are marked by machine-tokens, if horizontal (e.g., (2,12)→(2,1)) and fixture-tokens, if vertical or diagonal (e.g., (3,9)→(1,5)).

The example of Figures 1, 2 and 3 can be extended to include durations associated with the activities. Then the algebraic methods of Dubois and Stecke [1989] can be used to calculate a complete characterization of the steady state behavior of the process for performance evaluation purposes. This is demonstrated in §4. These have been calculated for a similar example in Cohen et al. [1985].

Note that the initial marking for fixture-tokens is dictated by the part fixture assignment process described above. The machine-tokens are located so as to allow the system to start processing parts according to the prescribed ordering of operations. Then the first operation is performed as indicated by the part input sequence. In particular, the initial graph marking needs to represent the initial machine availability. Also note the 7 marked, dotted vertical and diagonal arcs that represent the fixture availability. These depict when each fixture is ready for use.

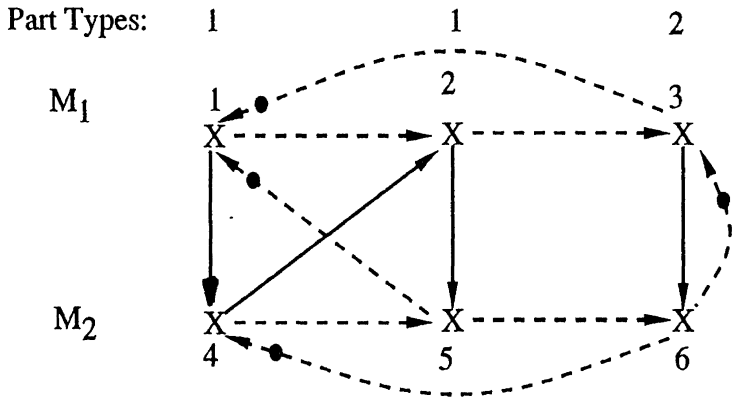
The behavior of the system depends on the choice of the initial marking. The fixture/part assignment procedure must define the most rational marking so that fixtures are as fully utilized as possible. For instance, the process of adding a fixture to the initial set of fixtures cannot usually be modeled only by adding a token to some arc in the initial marked graph. The topology of the marked graph must also usually be suitably modified so as to

insure a proper use of the additional resource. We shall see this in the following Example 2.

EXAMPLE 2

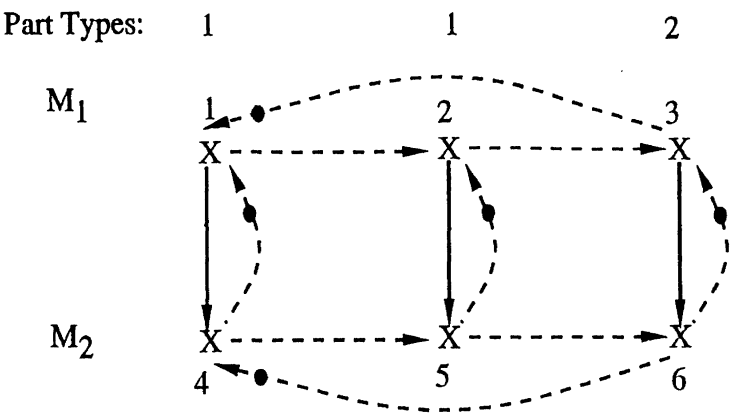
Consider the following two machine FMS producing two part types, both first on machine 1, then on machine 2, in ratios (2/3, 1/3) that are achieved by a part input sequence of (1,1,2). Note that in this example, the sequence (1, 2, 1) would be equivalent to (1, 1, 2) because of the repetitive nature of the production process. FMS performance would not improve with a different input sequence. The processing sequence is the same for all parts (first machine M_1, then M_2).

For the sake of simplicity, the notation in this example has been simplified from that of Example 1. If there is only one fixture per part type, one gets the following marked graph from the part/fixture assignment procedure.



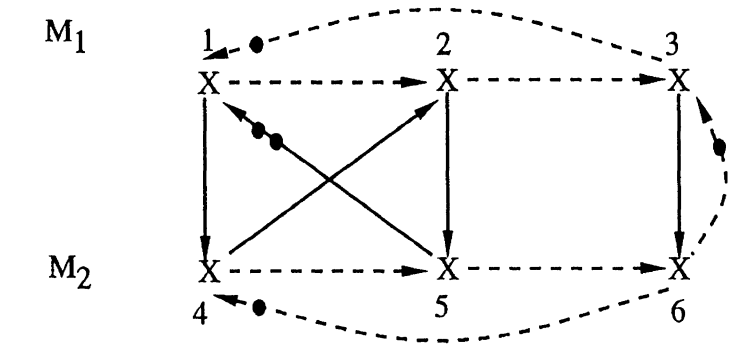
If a type one fixture is added to Configuration 1, the same assignment procedure (FCFS) yields the following Configuration 2:

Configuration 2:



Adding another token on arc (5,1) of Configuration 1 will suggest still another way of starting the system:

Configuration 3:



In Configuration 3, the behavior of the FMS is rather strange since a token is blocked forever on arc (5,1). This is because the operation expressed by node 2 has to wait for the termination of operation 4 before starting as in Configuration 1! Hence this is not a proper way of managing the set of fixtures.

Another seemingly more astute starting Configuration is to let a second token be allocated to the second operation of the sequence (node 2) without changing the topology of the graph in Configuration 1. Hence we get the graph in Configuration 1 with one additional token on arc (4,2). This is better than Configuration 3.

However, upon analyzing the corresponding fixture management policy, one finds that if fixture one carries part 1 in the n-th input sequence (while part 2 is carried by fixture

two), fixture one carries part 2 in the (n+1)-st input sequence. In other words, fixtures continuously pass each other. It would not be natural for part 1 in input sequence n+1 to wait for fixture two in order to begin processing, since fixture one will be available before fixture two at the end of the processing time for sequence n. (Indeed, parts do not pass each other because their sequencing at the machines never changes.) Only Configuration 2 allows the natural fixture management policy of using the fixtures as soon as they become available for the next input sequence. This, as shown in this Example 2, requires a modification of the topology of the graph describing the production process.

It is not difficult to mechanize the construction of the marked graph associated with a given FMS, a given part input sequence, and a given fixture allocation. The knowledge of processing times, travel times, and possibly setup times enable the marked graph to be suitably labeled. Then the techniques described in the companion paper (Dubois and Stecke [1989]) can be applied to calculate many quantities of interest, for purposes of system performance evaluation, such as the period, the order of this period, and the machine utilizations. In addition, using the token space, the set of critical resources (such as a bottleneck machine or the need for another fixture of a certain type) can be identified.

From this information, the fixture allocation can be modified to get a better performance if machines are not yet fully utilized. It is a matter of breaking critical circuits in order to improve system performance. This is demonstrated in §4. If needed, a complete characterization of the dynamic behavior of the FMS in a periodic steady state can be obtained. The critical operations can be identified as well as the slack times for noncritical operations by using the previous results.

Note that the transient period between the initial state and steady state is neglected in this analysis. However, the transient period can be omitted by starting the system in steady state, as computed by the algebraic techniques reviewed in Dubois and Stecke [1989].

Cohen et al. [1985] indicate that for a flow-shop structure, the maximum production rate attained, for given part mix ratios, does not depend on the choice of part

sequences on machines. This choice only affects the fixture allocation that is required to reach this maximum production rate (i.e., it affects the in-process inventory).

Moreover, if the bottleneck machine is unique and fully utilized, then the steady state is unique and independent of any delays which can be assigned to the release of the tokens in the initial marking that is computed by the part/fixture assignment procedure.

For a job shop-like structure, these results are no longer valid. The initial state may affect the production rate for a given fixture allocation, and the sequencing of parts on machines is more important. Cases can be constructed (see Cohen et al. [1985]) where a bad choice of part sequences on machines may prevent the bottleneck machine from being fully utilized, regardless of the fixture allocation.

The mathematical and algorithmic tools proposed in the companion paper (Dubois and Stecke [1989]) do not address the problem of finding the proper ordering of parts in the input sequence nor the sequencing on machines. Bellman, Esogbue and Nadeshima [1982] address such sequencing methods. An important criterion to industry is to determine the part sequences on machines so as to minimize in-process inventory. This can speed up fixture flow in the FMS and hence accelerate the return and reuse of fixtures. The sequencing problem here is simplified by using local decision rules at every machine. Then the queue discipline is fixed. Only the input sequence remains to be optimized. This approach has been adopted by Erschler et al. [1982], assuming a FCFS service rule at machines. This rule helps to prevent fixtured parts from remaining unprocessed for too long.

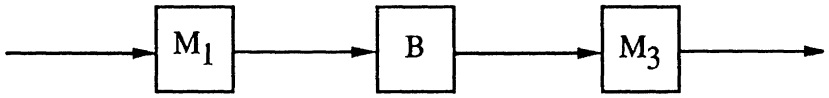
3.2 Limited Storage in Periodic Behavior FMSs

An implicit assumption in the models of §3.1 is that there is sufficient capacity of the storage areas, with direct access of parts to the machines. Arriving parts can be resequenced in little time, to satisfy the sequencing constraints at each machine. We now

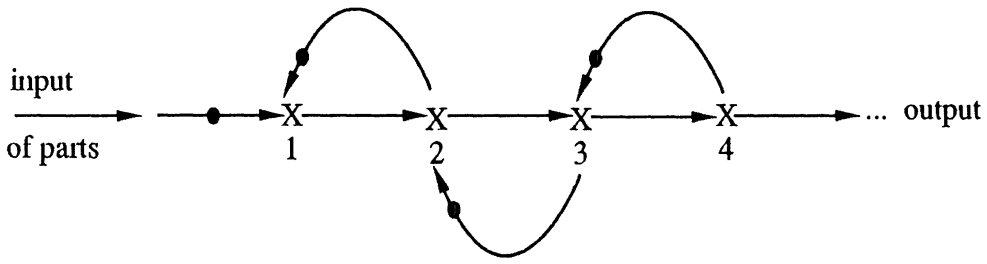
discuss the relaxation of this assumption when using the marked graph model by using the following Example 3.

EXAMPLE 3

First, consider a simple FMS with two machines and a buffer of capacity one in between.



There will be three tokens, one for each resource including the buffer. The open loop representation of this system as a marked graph is as follows, for the case of a single part type:



The four nodes correspond to the following activities:

Node 1: processing on machine Ml;

Node 2: leave Ml and go to the buffer;

Node 3: leave the buffer and go to M2;

Node 4: processing on M2.

In this model, the buffer is viewed as a machine. Nothing distinguishes it from a machine except that the processing time is zero. In this example, adding more tokens on arc (3, 2) models larger buffers. Each token models one buffer space.

For several part types, several layers are added, as in the following two part type example described in Figure 4. The part input sequence is: 1-2. There is one fixture per part type and adequate storage in front of machine one to hold all raw materials that arrive.

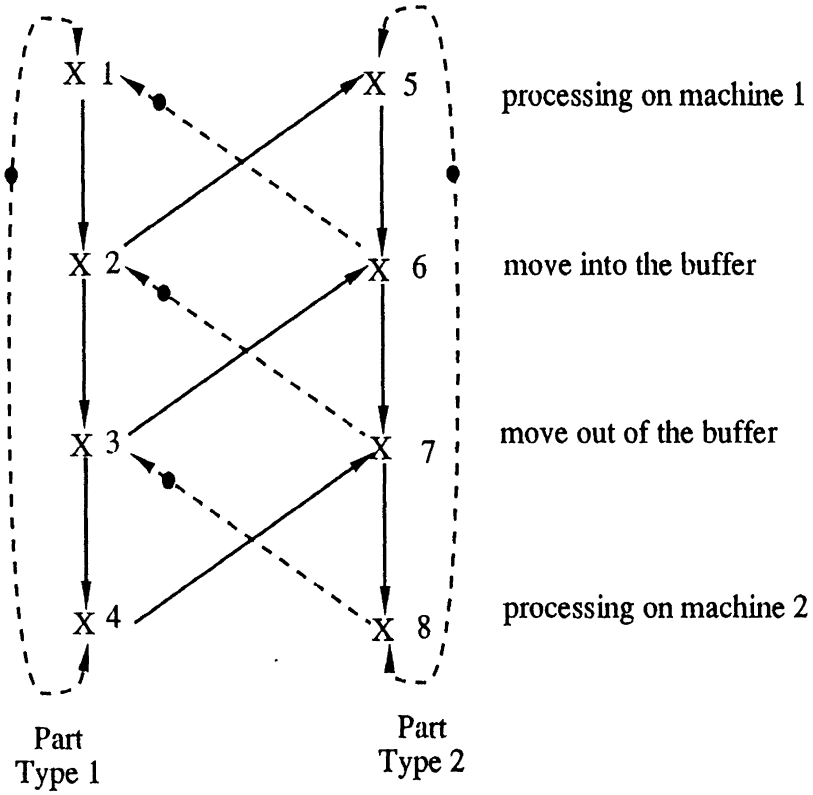


Figure 4. Marked Graph Model of Two Part Types and Two Machines.

The topology of the marked graph in Figure 4 expresses that, for instance, a part of type 1 can enter the buffer only if its processing on machine 1 is finished and the buffer does not contain a part of type 2 (i.e., there is a token on arc (7, 2)). Similarly, a part of type 2 can enter the buffer only if its processing on machine 1 is finished and the buffer does not contain a part of type 1 (i.e., there is a token on arc (3, 6)).

However, for the buffer to have more than one space, it is not enough to add tokens on the arcs corresponding to buffer space availability (e.g., (3, 6) and/or (7, 2)). The reason is that the topology of the graph forces the input of a part of type one into the buffer after the departure of a part of type 2 out of the buffer. This is expressed by arc (7, 2). In particular, a token on arc (3, 6) is useless for letting a part of type one into the buffer. If, when running the system with two tokens (one on (3, 6) and one on (7, 2)), a Configuration eventually occurs where the two tokens are on the same arc, then the difficulties are similar to those described by Configuration 3 in Example 2. The storage operations for identical parts belonging to the same input sequence are represented by distinct nodes in the marked graph.

Moreover, because a buffer is viewed as a machine, the model is equivalent to that of an FMS with no storage area. A finished operation on a machine liberates this machine only if the next machine on the processing sequence is free. Hence, for a flow shop structure with no machine passing, the part sequence must be identical on all machines. Consequently, the same holds for the same flow shop structure with limited storage.

Here, a buffer is viewed as a series of buffer spaces. Parts queue in the buffer without passing each other. The order of the parts is the same as the order of their arrival. This type of sequential local buffer storage in front of machines is appropriate for a flow shop structure where all parts visit all machines. It is then usual to have the same part sequence on each machine.

However, if parts are allowed to skip machines, the order of part arrivals in the input buffer of a machine may not be consistent with the part sequencing on this machine. This creates a problem for sequential buffers. There are two possible solutions. First, parts may be processed on a FCFS basis. However, in general, the marked graph model cannot capture this policy. Second, each machine can wait for the correct part in its sequence. However, in this case, blocking can occur while in fact downstream buffers may not be

full. This is neither rational nor efficient and in addition, could produce a deadlock in the system. A similar problem can exist with a job shop structure.

This situation may be addressed with the use of direct access buffers, modeled as parallel buffers. However, since a buffer is a resource equivalent to a machine in the marked graph model, the problem becomes one of modeling pools of identical machines. This is a topic for further research using the representation presented here.

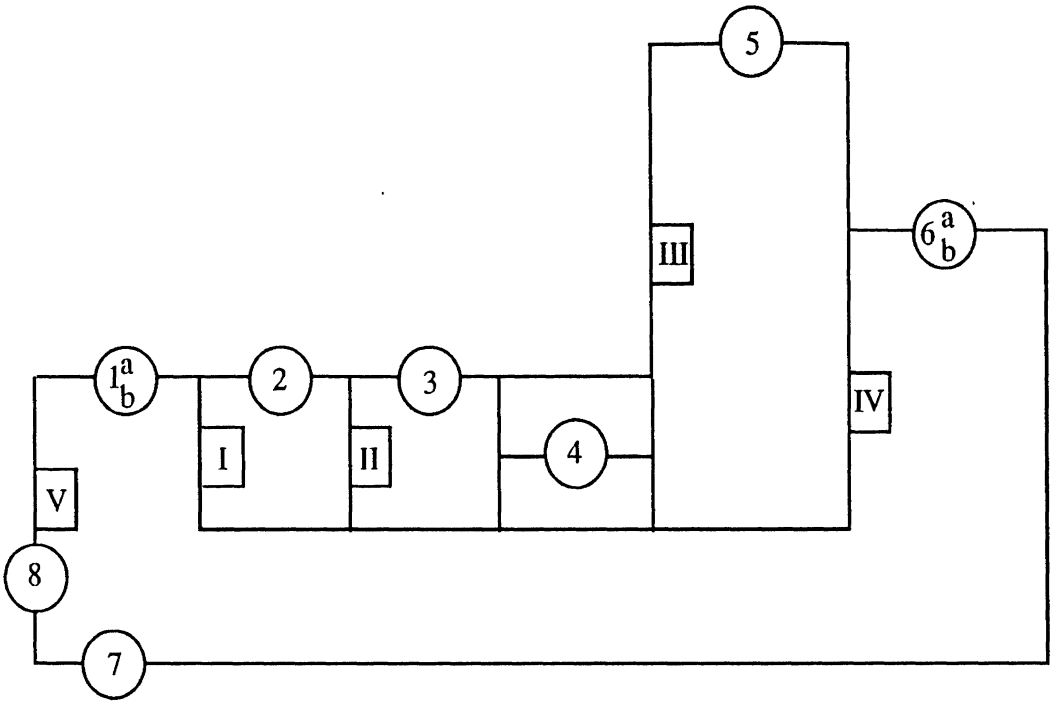
Hence a marked graph can conveniently represent blocking only for a flow shop structure with no machine skipping and finite sequential buffers. Any input sequence is allowed as is any fixture allocation. This includes systems with asynchronous transfer which do not use batch processing because of fixture limitations.

In general, product form queueing network models require an infinite buffer in front of each machine. Some recent examples of modeling finite buffer situations in flexible manufacturing can be found in Shanthikumar and Yao [1989] and So [1989].

4. APPLICATION TO THE PERFORMANCE EVALUATION OF AN FMS

Using an example of industrial origin, we now demonstrate how to model, evaluate the performance, and optimize system operation for an automated production system using the $\{\max, +\}$-based algebra. The results from this marked graph model, for several different part input sequences, are compared to results obtained using two closed queueing network models. The purpose of the comparison is to demonstrate the different types of information that be obtained through using such models. Some of this information and results can be found in Cohen et al. [1987].

The system being modeled is an automated production facility corresponding to a study done at Centre d'Etudes et de Recherches de Toulouse for a machine tool company. The facility is described in Figure 5.



1a,b - 2 drills

2,5 - 2 broaching machines

3 - lathe

4 - inspection machine

6a,b - 2 machining centers

7 - washing station

8 - L/UL area

I-V - 5 buffer areas

Figure 5. A Seven-Machine FMS.

The system of Figure 5 consists of seven machine tools of four types, an inspection station, washing area, local buffer storage, and a load/unload (L/UL) area. Dedicated pallet/fixture combinations (hereafter refered to as fixtures) transport six different part types between the machines. Table I provides the processing times (in hours, say) of the required operations of each part type on each machine.

Table I

Processing Times.

		Part Types					
		1	2	3	4	5	6
M	1	3.9	0.95	1.1	0.7	1.4	
A	2			2.	1.2		1.7
C	3	*3.7*		*2.2*		*6.4*	
H	4			2.		1.	1.
I	5	1.7	3.1	3.		1.3	
N	6	0.5	3.2	4.3	1.9	1.6	0.4
E	7	1.	1.	1.	1.	1.	1.
S	8	1.5	1.5	1.5	1.2	1.2	1.2
	Σ	16.4	12.7	16.95	6.4	13.2	6.7

The production requirements for all part types is the same. Producing all six part types on the system simultaneously implies that the part mix values for all part types are the same. Then the minimal part set (i.e., the part mix ratios) for all six part types is: (1, 1, 1, 1, 1, 1).

The bottleneck machine is machine 3: its workload (when one part of each type is produced) is 12.3 hours. Now, having only one fixture of each type is insufficient to utilize this bottleneck machine, since the duration of the longest processing sequence (for part type 3) is greater than the workload of this bottleneck machine.

In this example, the sequences of the parts on the machines is fixed and given in Table II.

The transportation times from machine to machine can be as important to consider as machining time. For this example, the transportation times are given in Table III.

For this example, the input sequence is given as (1, 2, 3, 4, 5, 6) for all machines. Some machines are bypassed.

Table II

Sequences of the Six Part Types on the Eight Machines.

		SEQUENCES					
MACHINES	1	2	3	4	5	6	
	2	3	4	6			
	3	1	3	5			
	4	3	5	6			
	5	1	2	3	5		
	6	1	2	3	4	5	6
	7	1	2	3	4	5	6
	8	1	2	3	4	5	6

Table III

Transportation Times.

		MACHINES							
		1	2	3	4	5	6	7	8
MACHINES	1		2.	4.3		6.			
	2			1.3	4.		5.3		
	3				1.4	4.2			
	4					2.1	4.		
	5						2.		
	6								
	7							5.5	
	8	1.4							1.

The system has 48 states (the number of operations to be performed) denoted by x_{ij}, i=1,...,6 and j=1,...,8. The system has six input places, u_i, i=1,...,6 and six output places, y_i, one of each for each part type. Machines are reused as consecutive input sequences are launched, until the production requirements of the six part types are completed.

Feedback equations in the {max,+}-based algebra can be developed explicitly, but this is not done here. Instead, we use Karp's algorithm [1978], which very efficiently finds the maximal average weight of a circuit in a graph G(M) of a square matrix M. The eigenvalue, λ, of matrix M can be found numerically, without requiring its algebraic interpretation. Karp shows that the eigenvalue of M can be obtained by choosing any vertex i of G(M) and calculating:

$$\lambda = \max_{j=1,\ldots m} \min_{0\le k\le m-1} \left\{ \frac{(M^m)_{ij} - (M^k)_{ij}}{m - k} \right\},$$

where $(M^k)_{ij}$ is the ij^{th} entry of the k^{th} {max,+} power of M, i.e., the maximal weight of paths of length k from i to j in G(M).

Before using Karp's algorithm, one must devise a situation where each arc (i,j) has one and only one token per arc. This can be done by increasing the number of states and solving a linear system of equations associated with a graph without tokens. Then the weight of the arc is the time that a token spends on the arc. Details of these calculations for this example are found in Cohen et al. [1987]. A simpler example showing the calculations is in Dubois and Stecke [1989]. Here we show the results and discuss the usefulness of the methods.

Using Karp's algorithm, it is possible to study the asymptotic behavior of the system, to find the critical circuits, and to optimize the production rate. This is the objective of the following numerical results.

The results are presented with the following notation:

$P = (n_1,n_2,n_3,n_4,n_5,n_6)$

n_i = number of fixtures for part type i

λ_0 = 12.3 = workload of the bottleneck machine

λ/λ_0 = maximum production rate of the system

λ_0/λ = utilization of the bottleneck machine.

The number of fixtures is eventually optimized by observing the critical circuits of a graph with a fixed number of fixtures for each part type. The critical circuits allow the identification of the "missing fixture type(s)," i.e., those fixture types for which the addition of another fixture would increase production rate and system utilization. The "missing fixture types" are identified by vertical "returns" of the arcs.

The optimization of the fixture distribution over time can be seen in Figures 6a to 6e. These five figures show the critical circuits that appear during the optimization. The procedure begins as follows.

Figure 6a begins with a reasonable fixture distribution, two fixtures for each part type plus a third fixture dedicated to the third part type, which requires the longest total processing time: P = (2,2,3,2,2,2). The critical circuits of Figure 6a show the limiting fixture types through the vertical arcs: (8,2) and (8,5). Adding another fixture each for part types 2 and 5 (not coincidentally, the part types having the next highest total processing time requirements) provides the fixture distribution of P = (2,3,3,2,3,2) of Figure 6b. This shows that the vertical arc is (8,4), implying that adding another fixture for part type 4 will again increase performance. This fixture is added to provide the fixture distribution and resulting critical circuit of Figure 6c.

The procedure of adding fixtures based on the identification of the critical circuits continues until the bottleneck machine (here, 3) is fully utilized. Here, this is achieved at the fifth iteration and is shown in Figure 6c. There are no vertical arcs remaining and the horizontal arc at machine 3 indicates that it is fully utilized.

Then with 18 fixtures total, distributed as P = (3,3,3,3,3,3), the bottleneck machine is fully utilized and the production rate is maximized. Adding additional fixtures would not improve the production rate in steady state. Adding fixtures may decrease the time necessary to reach the steady state periodic behavior. These models can also be used to determine the transient period before steady state is achieved.

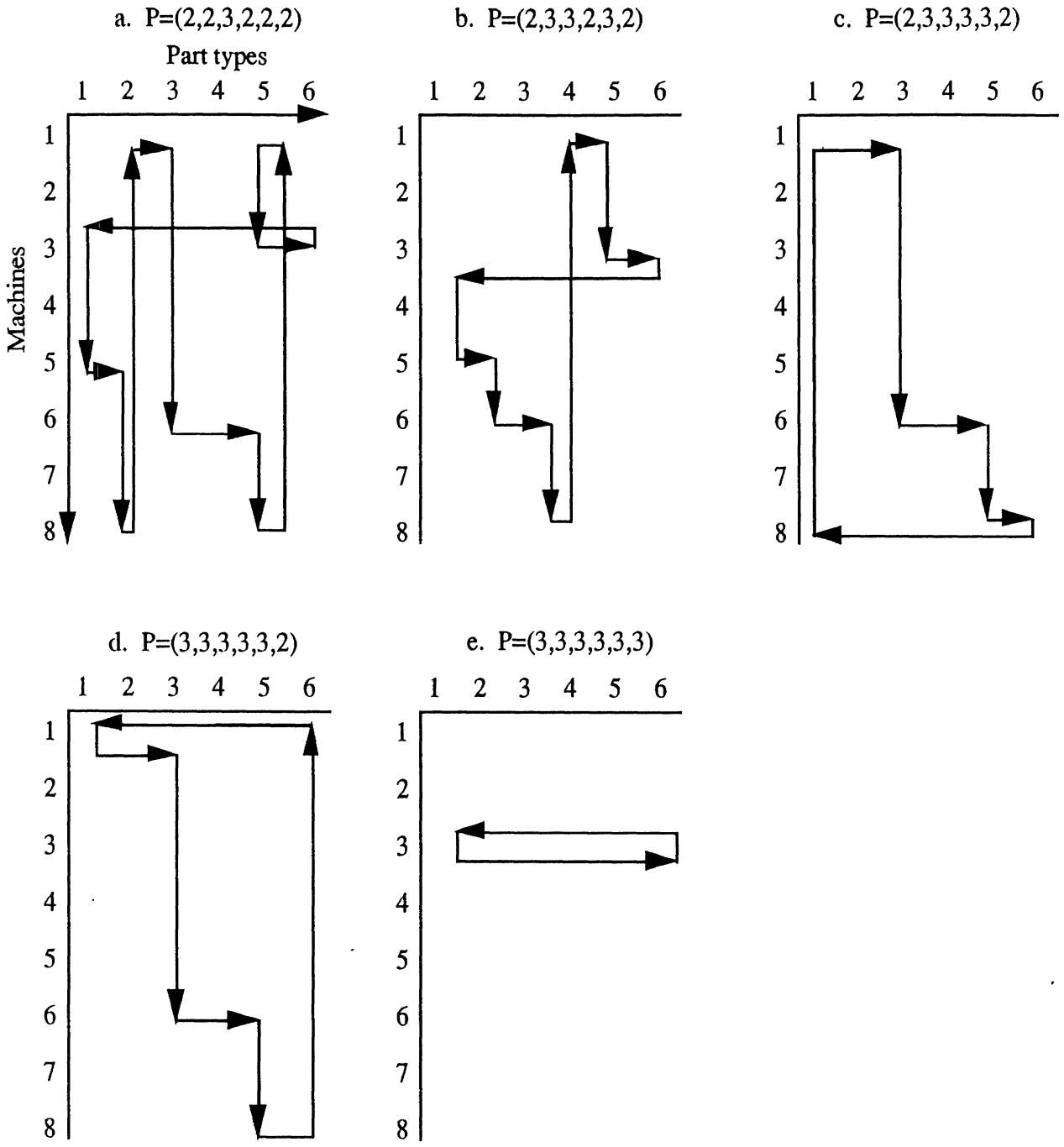


Figure 6. Critical Circuits of Graph G(M), Given Initial Fixture Distributions.

For this example, the individual machine utilizations with the fixture distribution P = (3,3,3,3,3,3) is:

Machine:	1	2	3	4	5	6
Utilization:	.817	.398	1.00	.325	.740	.967

Utilizations can only be improved by redistributing the operations among the machines from that currently given in Table I. For the given fixture distribution and input sequence, system utilization is: 70.8%.

In this example, the addition of five fixtures of different types to an original fixture distribution provided the optimum work-in-process inventory to maximize production. System and machine utilizations are provided. This demonstrates some of the uses of marked graph models.

The final demonstration of the use of the marked graph model compares the results obtained (using several part input sequences that maintain the mix ratios) to the results from using several queueing network models. The study is for the same system of Figure 5, Table I, and Table II. For this study, the transportation times of Table III are ignored, which results in fewer total numbers of pallets and fixtures required. The processing times and part mix ratios of (1,1,1,1,1,1) of before are the same. Three different part input sequences are compared, as given in Table IV.

One of the queueing network methods used is a program called CAN-Q (see Solberg [1977, 1979]), which models a closed multiserver, single class closed queueing network. With this model, the part mix ratios and expected processing times are respected. However, the individual fixture distribution cannot be modeled. The total number of fixtures in the system is modeled.

The other closed queueing network model compared here is a heuristic based on mean value analysis (see Suri and Hildebrant [1984]). This is a multiclass model and so can model the fixture distribution among the individual part types. However, modeling the fixed part mix ratios is tricky. The method assumes an integer number of fixtures and

iterates on the distribution, keeping the total number constant, until the output assures that the mix ratios are adequately satisfied.

Both queueing network models are aggregate and stochastic, requiring average input values of processing times and machine visit frequencies to result in average and steady state performance measures, such as expected production rates and average system and machine utilizations. The processing times are exponential and the queue disciplines at the machines are FCFS.

On the other hand, the marked graph model is deterministic and detailed, modeling the exact processing times of the operations. The precise input sequence and mix ratios of the system are modeled. In addition, transient behavior can be observed.

Results from applying these two closed queueing network models are compared with the results using the marked graph model for three different input sequences in Table IV. The total number of fixtures is varied from six to twelve, to see the increase in the utilization of the bottleneck machine.

Table IV

Utilizations of the Bottleneck Machine as a Function of the Total Number of Fixtures in the System.

	Total Number of Fixtures						
	6	7	8	9	10	11	12
CAN-Q	0.64	0.69	0.73	0.76	0.79	0.81	0.83
Mean Value	0.63	0.67	0.71	0.74	0.76	0.79	0.81
123456	0.73	0.81	0.82	0.86	0.86	0.87	1
321456	0.73	0.74	0.80	0.87	0.88	0.90	1
132465	0.72	0.73	0.76	0.77	0.77	0.86	1

As expected, the bottleneck machine utilization increases with the increase in the total number of fixtures available. The pessimistic nature of the two queueing network

models is seen as they constantly (but proportionally) underestimate the true machine utilization. CAN-Q provides less pessimistic results than the mean value analysis-based method. The queueing network models require a near infinite total number of parts in the system before the bottleneck machine is fully utilized (see Stecke and Solberg [1985]).

The three cases using the marked graph model provide somewhat similar results, with a total of 12 fixtures being sufficient to fully utilize the bottleneck machine. Unlike the queueing network models, the increase in utilization is not a smooth increase. With the marked graph model, the production rate seems to be a function of both the chosen input sequence and the total number of fixtures.

The performance evaluation method provided by the marked graph model is exact only in the deterministic case. The methods are useful to evaluate proposed solutions. The queueing network models are exact only when the processing times are exponential, which is usually not the case in automated manufacturing. However, the queueing network models usually provide robust relative results. The results of the marked graph model will in general be more precise and accurate.

5. DISCUSSION AND CONCLUSION

The major limitation of the marked graph model is the necessity of specifying a deterministic behavior for the system. A sequence of activities is known for each resource, for example, fixtures, machine tools, buffers, or carts. All operations are ordered in the model.

Because each resource requires a complete ordering of its events, the representation of a particular FMS should not be very detailed. The model can accomodate all details. However, the result is a large and unwieldy representation. The specification of the fixed part sequences on all machines (after using a suitable sequencing algorithm) may be reasonable for the purpose of rough performance analysis. However, it is usually not

feasible to model in all detail all possible moves of the carts, which carry the fixtured parts from machine to machine.

Moreover, the FMS has to be reliable enough so that the periodic behavior described and calculated by the {max,+} algebra techniques is a good approximation of the typical behavior of the system. For FMSs devoted to medium or large volume production, the manufacturing technology should eventually become reliable and robust over the production horizon.

Many production systems fall short of reaching such an ideal operating state. Then heavier performance evaluation tools, such as simulation or maybe queueing networks, are needed. The modeling power of simulation is useful to capture more detailed behavior.

Analytical performance evaluation models in general have limited descriptive power. This limitation is counterbalanced by the low cost of analytical evaluations, based on the marked graph models, for instance, or queueing networks. Evaluation of a marked graph model consists of computing shortest paths in a directed graph (i.e., see Karp [1978]), whose nodes are the tokens of the initial marked graph model. Hence the algorithm complexity is $0(n^3)$ at most, where n in the number of resources (machines and fixtures), regardless of the complexity of the marked graph model of the production process.

The marked graph model is complementary to other FMS analytic models based on stochastic queueing networks. (See Buzacott and Yao [1986] for a review.) Contrary to queueing network models, marked graph solution approaches produce exact performance evaluation using a periodical queue discipline at machines. Most queueing network models, such as CAN-Q (Solberg [1977, 1979]), require a product form. They require exponentially distributed processing times and produce pessimistic evaluations of production rates and machine utilizations, as was demonstrated in §4. However, some recent results have relaxed some of the modeling assumptions (e.g., limited storage and general service times, i.e., see Yao [1982]). Set-up times are usually neglected in queueing

network models, but can easily be accounted for in marked graph models. More generally, any repetitive, deterministic, asynchronous process can be analyzed by means of the {max,+} algebra.

The marked graph approach yields useful information about how to control the flow of production in an FMS, since it forces the choice of part sequences and suggests a fixture allocation that achieves best performance under this choice. Coupled with a proper sequencing algorithm, the approach could be used in a decision support system to decide, for example, how parts should be input into the system and how many fixtures are necessary to support a high system utilization.

There are some open research problems involved with this {max,+} approach. At the applied level, modeling pools of identical machines requires further research. Using a marked graph, machines and pallets (and fixtures) can be modeled in a similar manner. In particular, pools of identically-tooled machines can be modeled in a way similar to that for teams of identical fixtures. However, a deterministic policy for specifying the utilizations of the machines within each pool would need to be specified.

Also, short breakdowns in an FMS can perhaps be modeled by replacing precise durations of activities by interval-valued durations. The {max,+} algebra can be extended from the real line to the set of closed intervals, which can be useful to model short breakdowns. Many of the properties of the {max,+} algebra are preserved on intervals, as a preliminary investigation has shown (Dubois [1983]).

In conclusion, the timed marked graph models of asynchronous deterministic processes extend PERT/CPM types of analyses to the case of directed event-graphs having circuits. The existence of circuits accounts for repeated sets of activities. This view of timed marked graph models stresses the potential importance of these models for various fields of application, where timed, deterministic models of asynchronous systems are useful for performance evaluation purposes. These fields include FMSs, robotics, parallel computation, and computer system performance, for example. Perhaps the potential

usefulness of the timed marked graph analysis methods can be assessed by extrapolating the success of PERT/CPM networks. If this extrapolation is valid, the timed marked graph will be a very successful performance evaluation tool.

ACKNOWLEDGEMENTS

We gratefully acknowledge the expert evaluations of three anonymous Referees each for both this and the associated theory paper as well as the Associate Editors and the Editors, Joe Mazzola and Maurice Queyranne.

REFERENCES

ALLA, H., LADET, P., MARTINEZ, J., and SILVA, MANUEL, "Modelling and Validation of Complex Systems by Coloured Petri Nets: Application to a Flexible Manufacturing System," Lecture Notes in Computer Science 188: Advances in Petri Nets, Springer Verlag, Berlin (1984).

ALANCHE, P., BENZAKOUR, K., DOLLE, F., GILLET, P., RODRIGUES, P., and VALETTE, ROBERT, "PSI: A Petri Net Based Simulator for Flexible Manufacturing Systems," Lecture Notes in Computer Science 188: Advances in Petri Nets, Springer Verlag, Berlin (1984).

BELLMAN, RICHARD D., ESOGBUE, AUGUSTINE 0., and NADESHIMA, I., Mathematical Aspects of Scheduling and Applications, Pergamon Press, New York (1982).

BROWNE, JIM, DUBOIS, DIDIER, RATHMILL, KEITH, SETHI, SURESH P., and STECKE, KATHRYN E., "Classification of Flexible Manufacturing Systems," The FMS Magazine, Vol. 2, No. 2, pp. 114-117 (April 1984).

BRUNO, GIORGIO and BIGLIA, PAOLO, "Performance Evaluation and Validation of Tool Handling in Flexible Manufacturing Systems Using Petri Nets," IEEE International Workshop on Timed Petri Nets, Torino, Italy, pp. 64-71 (July 1-3, 1985).

BUZACOTT, JOHN A., and YAO, DAVID, D. W., "Flexible Manufacturing Systems: A Review of Analytical Models," Management Science (1986).

CHIOLA, GIOVANNI, "A Software Package for the Analysis of Generalized Stochastic Petri Net Models," IEEE International Workshop on Timed Petri Nets, Torino, Italy, pp. 136-143 (July 1-3, 1985).

CHRETIENNE, PHILIPPE, "Les Réseaux de Petri Temporisés," Thèse d'État, Université de Paris VI, France (1983).

COHEN, GUY, DUBOIS, DIDIER, QUADRAT, JEAN PIERRE, and VIOT, MICHEL, "A Linear-System-Theoretic View of Discrete-Event Processes and Its Use for Performance Evaluation in Manufacturing," IEEE Transactions on Automatic Control, Vol. AC-30, No. 3, pp. 210-220 (March 1985).

COHEN, GUY, DUBOIS, DIDIER, QUADRAT, JEAN PIERRE, and VIOT, MICHEL, "Application de la Théorie des Systèmes à Evénements Discrets a l'Evaluation des Performances d'un Atelier Flexible," Algebres Exotiques et Systèmes à Evénements Discrets, INRIA, Rocquencourt, France, pp. 167-173 (June 1987).

CUNNINGHAME-GREEN, RAYMOND A., Minimax Algebra, Springer-Verlag, New York (1979).

DUBOIS, DIDIER, "Modèles Mathematiques de l'Imprécis et de l'Incertain, en Vue d'Applications aux Techniques d'Aide à la Decision," Thèse d'État, Université de Grenoble, France (1983).

DUBOIS, DIDIER and STECKE, KATHRYN E., "Dynamic Analysis of Repetitive Decision-free Discrete Event Processes: The Algebra of Timed Marked Graphs and Algorithmic Issues," Working Paper No. 542-b, The University of Michigan, GSBA, Ann Arbor MI (Revised October 1989).

DUBOIS, DIDIER and STECKE, KATHRYN E., "Using Petri Nets to Represent Production Processes," Proceedings of the 22nd IEEE Conference on Decision and Control, San Antonio, Texas, pp. 1062-1067 (December 1983).

DUBOIS, DIDIER, STECKE, KATHRYN E., and BEL, GERARD, "Utilization des Reseaux de Pétri Déterministes et Temporisés pour la Représentation et l'Analyse de Certains Processus de Production," Proceedings of the Congress AFCET Automatique," Toulouse, France, pp. 587-596 (1985).

ERSCHLER, JACQUES, LÉVÊQUE, DIDIER, and ROUBELLAT, FRANCOIS, "Periodic Loading of Flexible Manufacturing Systems," Proceedings of the IFIP Congress "APMS-82", Bordeaux, France, pp. 327-339 (1982).

HILLION, H. P. and PROTH, JEAN-MARIE, "Performance Evaluation of Job-shop Systems Using Timed Event Graphs," IEEE Transactions on Automatic Control, Vol. 34, No. 1, pp. 3-9 (1989).

HITZ, K. L, "Scheduling of Flexible Flowshops," M.I.T., Report LIDS-R-879, Cambridge MA (March 1979).

KARP, RICHARD M. "A Characterization of the Minimum Cycle Mean in a Digraph," Discrete Mathematics, Vol. 23, pp. 309-311 (1978).

PETERSON, JAMES L., Petri Net Theory and the Modeling of Systems, Prentice Hall, Inc., Englewood Cliffs NJ (1981).

RAMAMOORTHY, C. V. and HO, GARY S., "Performance Evaluation of Asynchronous Concurrent Systems using Petri Nets," IEEE Transactions on Software Engineering, SE-6, No. 5, pp. 440-449 (September 1980).

RAMCHANDANI, CHANDER, "Analysis of Asynchronous Concurrent Systems by Timed Petri Nets," M.I.T. Report MAC TR-120, Cambridge MA (1974).

SHANTHIKUMAR, J. GEORGE and YAO, DAVID D., "Optimal Buffer Allocation in a Multicell System," International Journal of Flexible Manufacturing Systems, Vol. 1, No. 4, pp. 347-356 (September 1989).

SO, KUT C., "Allocating Buffer Storages in a Flexible Manufacturing System," International Journal of Flexible Manufacturing Systems, Vol. 1, No. 3, pp. 223-237 (June 1989).

SOLBERG, JAMES J., "A Mathematical Model of a Computerized Manufacturing System," Proceedings of the Fourth International Conference on Production Research, Tokyo, Japan (August 1977).

SOLBERG, JAMES J., "Analytical Performance Evaluation of Flexible Manufacturing Systems," Proceedings of the 18th IEEE Conference on Decision and Control, San Diego CA, pp. 640-644 (December 1979).

STECKE, KATHRYN E., "Procedures to Determine Both Appropriate Production Ratios and Minimum Inventory Requirements to Maintain These Ratios in Flexible Manufacturing Systems," Working Paper No. 448, Graduate School of Business Administration, The University of Michigan, Ann Arbor MI (October 1985).

STECKE, KATHRYN E. and KIM, ILYONG, "A Study of FMS Part Type Selection Approaches for Short-Term Production Planning," International Journal of Flexible Manufacturing Systems, Vol. 1, No. 1, pp. 7-29 (September 1988).

STECKE, KATHRYN E. and KIM, ILYONG, "Performance Evaluation for Systems of Pooled Machines of Unequal Sizes: Unbalancing Versus Balancing," European Journal of Operational Research, Vol. 42, No. 1, pp. 22-38 (September 1989).

STECKE, KATHRYN E. and SOLBERG, JAMES J., "The Optimality of Unbalancing Both Workloads and Machine Group Sizes in Closed Queueing Networks of Multiserver Queues," Operations Research, Vol. 33, No. 4, pp. 882-910 (July-August, 1985).

SURI, RAJAN and HILDEBRANT, RICHARD R., "Modelling Flexible Manufacturing Systems Using Mean-Value Analysis," Journal of Manufacturing Systems, Vol. 3, No. 1, pp. 27-38 (1984).

WADHWA, S. and BROWNE, JIM, "Modeling FMS with Decision Petri Nets," International Journal of Flexible Manufacturing Systems, Vol. 1, No. 3, pp. 255-280 (June 1989).

YAO, DAVID D. W., "Queueing Models of Flexible Manufacturing Systems," Ph.D. dissertation, University of Toronto, Department of Industrial Engineering, Toronto, Canada (1983).