Deadlock Control Methods in Automated Manufacturing Systems

Maria Pia Fanti, *Senior Member, IEEE,* and MengChu Zhou, *Fellow, IEEE*

***Abstract*—As more and more producers move to use flexible and agile manufacturing as a way to keep them with a competitive edge, the investigations on deadlock resolution in automated manufacturing have received significant attention for a decade. Deadlock and related blocking phenomena often lead to catastrophic results in automated manufacturing systems. Their efficient handling becomes a necessary condition for a system to gain high productivity. This paper intends to present a tutorial survey of state-of-the art modeling and deadlock control methods for discrete manufacturing systems. It presents the updated results in the areas of deadlock prevention, detection and recovery, and avoidance. It focuses on three modeling methods: digraphs, automata, and Petri nets. Moreover, for each approach, the main and relevant contributions are selected enlightening pros and cons. The paper concludes with the future research needs in this important area in order to bridge the gap between the academic research and industrial needs.**

***Index Terms*—Automata, automated manufacturing systems, deadlock resolution, diagraph, petri nets.**

I. Introduction

AUTOMATED manufacturing systems (AMSs) are modern production facilities exhibiting a high degree of resource sharing. An AMS consists of a set of workstations, each one capable of processing parts of different kind according to a prescribed sequence of operations. The concept of agile manufacturing has been introduced to satisfy the demand for low-volume and high-variety products and requires extending flexibility in production to system configuration and control. Hence, by integrating computer systems, hardware, and information flows, AMS provide manufactures with flexibility and efficiency. The interacting parts and shared resources are characterized as a resource allocation system that can incur deadlock conditions. Deadlock is a situation that may occur in complex computer-integrated systems, including automated production systems, transportation systems, computer operating systems, concurrent software systems, computer networks, and distributed database systems. It is a phenomenon in which a system or a part of it remains indefinitely blocked and cannot terminate its task. To demonstrate this condition, we consider a simple and intuitive example of traffic flow along the edges of a square, depicted in Fig. 1. The boxes represent the cars and the arrows the directions in which they have to move. In this situation, each car of the displayed set is blocked by another car in the same set, occupying the space that it requires. So this situation represents a deadlock.

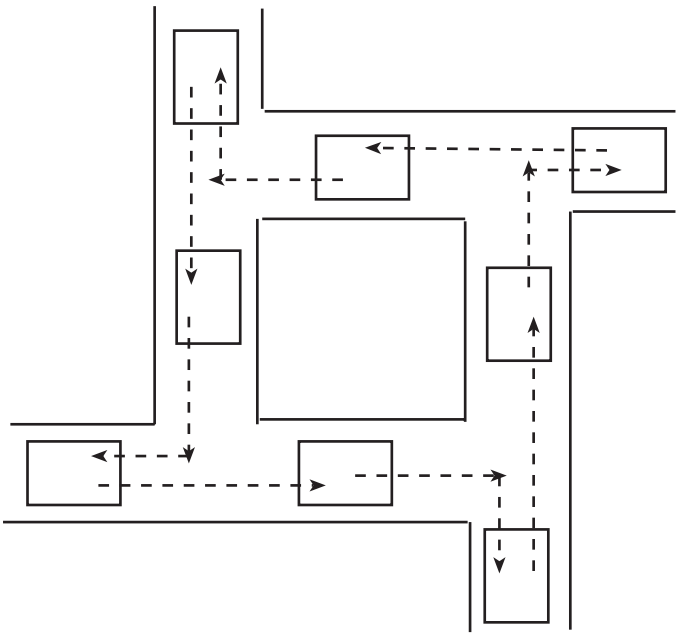

Fig. 1. Deadlock condition in traffic flow.

Deadlock is a logical condition arising in discrete event dynamic systems (DEDSs) [4], [25]. It is caused by an inappropriate allocation of resources to concurrent executing processes. Hence, to analyze deadlock phenomena, it is necessary to focus a DEDS model on the interactions between processes and resources. Here, as in [36], [55], and [56], we characterize a system of interacting processes with shared resources as a resource allocation system (RAS), i.e., a set of concurrently running processes requiring resource sharing. Moreover, we suppose that the number of resources in a RAS is limited and resources are characterized as reusable, i.e., their allocation/deallocation to a requesting process does not affect their quality and quantity. In particular, reusable resources can model physical objects (like machines, robots, drives, etc.), which are shared by processes (like parts, vehicles, programs, data, etc.). Moreover, the use of resources is supposed to be sequential, i.e., each process has to acquire resources in a predefined or partially predefined order.

Deadlock is first addressed by computer scientists developing resource allocation logic in operating systems [7], [21], [23], [29]. In particular, Coffman *et al.* [7] give four conditions that must be held for a deadlock to occur:

Manuscript received August 15, 2002; revised April 20, 2003. This paper was recommended by Guest Editor M. D. Jeng.

M. P. Fanti is with the Electrical and Electronic Engineering Department, Polytechnic of Bari, Bari, Italy (e-mail: fanti@deemail.poliba.it).

M. Zhou is with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102 USA, and also with the Institute of Automation, Chinese Academy of Sciences, Beijing 100080, China (e-mail: zhou@njit.edu).

Digital Object Identifier 10.1109/TSMCA.2003.820590

i) "mutual exclusion," i.e., tasks claim exclusive control of the resource they require;
ii) "no preemption," i.e., resources cannot be forcibly removed from the tasks holding them until the resources are used to completion;
iii) "wait for condition," i.e., processes hold resources allocated to them, while waiting for additional ones;
iv) "circular wait," i.e., a circular claim of tasks exists, such that each task holds one or more resources that are being requested by the next task in the claim.

In the past, manufacturing community has ignored the deadlock problem since human operators have solved its occurrence by aborting one or more parts involved in the deadlock and by granting the released resources to other implicated parts. However, agile manufacturing paradigm increased the focus on system flexibility and efficiency. Therefore, a manufacturing system controller that allocates resources in real time without causing deadlock plays a crucial role. Consequently, in the past ten years, much research was conducted to resolve deadlock in AMS. In such a context, the strategies for solving deadlock problems are introduced *ad hoc* and differ from those used in computer operating systems. While, for computer applications, it is normally assumed that only bounds on the total number of resources required by each process is known, in an AMS, a part requires visitation to a predictable sequence of machines for processing. More precisely, each part (job or piece) enters the system and follows a specific working procedure (or production sequence), i.e., an ordered or partially ordered succession of resources necessary to provide the scheduled service (buffer spaces, cutting tools, work stations, transportation vehicles, etc). Moreover, in AMS, the first three conditions given by Coffman *et al.* are always satisfied. Indeed, pieces use resources in an exclusive mode, they hold resources while waiting for the next resources specified by their operation sequence, and resources cannot be forcibly removed from the pieces utilizing them until operation completion. Hence, deadlock is excluded only if the fourth condition, i.e., the circular wait, fails to hold. So, in AMS deadlock is caused by parts in a set requiring access to resources held by other parts in the same set. In this way, a circular wait condition arises and can be broken only by a recovery procedure.

Three strategies addressing deadlock issues in different contexts are *prevention*, *detection/recovery*, and *avoidance* methods. In particular, prevention methods outlaw circular waits among concurring jobs at the design stage and offline. Detection and recovery approaches use a monitoring mechanism for detecting the deadlock occurrence and a resolution procedure for appropriately preempting some deadlocked resources. Finally, deadlock avoidance schemes in manufacturing processes prevent circular waits from occurring by the proper operational control of part flow.

Researchers and engineers have presented a large number of new solution strategies following *prevention*, *detection/recovery*, and *avoidance* strategies. The aim of this paper is to present a structured classification of the work about deadlock problems. A first selection of papers is made on the basis of the model used to describe the AMS and, in particular, the interaction between jobs and resources. Three modeling methods are singled out: digraphs, automata, and Petri nets. There is no conclusive argument making evident the superiority of one of these approaches to another. Each approach, indeed, has its own advantages and feeds the researches of many authors that have greatly improved the insight into deadlock phenomena. In this survey, however, the authors describe in detail only the contributions they consider the most relevant in each of the mentioned approaches. Hence, the paper is a starting point to approach deadlock in AMS and a concise summary of the significant results. Moreover, a simple example is used to help readers clarify the presented strategies. Some basic control strategies are briefly exposed and are applied to the considered system to give a clear idea of the proposed models and approaches. The paper is organized as follows: Section II describes the methods to solve deadlock in AMS, and Sections III–V discuss the main deadlock resolution strategies by using digraphs, automata, and PN, respectively. Section VI recalls some comparison studies, and finally, Section VII presents concluding remarks.

II. Deadlock Resolution Strategies

This section presents in detail the three methodologies for addressing deadlock problems in AMS. Since each procedure is a technique controlling the acquisition and release of resources in AMS, it is necessary to classify the system on the basis of the types of resource allocation. Hence, by viewing AMS as a resource allocation system (RAS), the following taxonomy has been introduced [46], [55] on the basis of the resource allocation request structure.

1) *Single Unit RAS (SU-RAS):* At each step of its processing, a part requires a single unit of a single resource for its successive execution;
2) *Single Type RAS (ST-RAS):* At each step of its processing, a part requires several units of a single resource.
3) *Conjunctive RAS (C-RAS):* At each step of its processing, a part requires an arbitrary number of units of each resource from a set of resources.
4) *Conjunctive/Disjunctive RAS (CD-RAS):* Every process stage poses a finite number of alternative conjunctive-type resource requests.

A. Deadlock Prevention

Basic strategies for handling deadlocks are to ensure violation of at least one of the four conditions necessary for deadlock. In particular, *prevention* methods guarantee that necessary conditions for deadlock cannot be simultaneously satisfied at any point of the RAS dynamic. Moreover, prevention constrains system users and imposes a number of restrictions on interactions between processes and resources so that requests leading to deadlock are never executed. Usually, prevention strategies use information about process resource requirements and demand each process to specify all required resources before transactions begin. The simplest means of preventing deadlock is to outlaw concurrency, but such a method is overly conservative, generally leading to significantly low utilization of system resources. The state transitions follow the prescribed rules ensuring that the system is prevented to enter a deadly embrace status. For example, consider the AMS in Fig. 2, where part 1

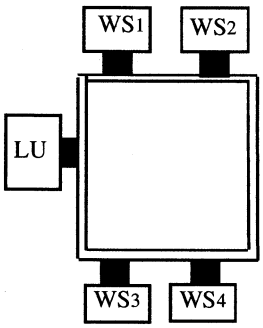

Fig. 2. Four-workstation AMS.

has to follow the routing (L/U, WS_1, WS_2, WS_3, L/U), and part 2 follows the route (L/U, WS_4, WS_2, WS_1, L/U). Since the first three conditions given by Coffman are verified, deadlock can be prevented only if the "circular wait" condition is not met. Hence, a simple prevention method can allow the production of parts 1 and 2 in succession so that two types of pieces cannot be processed simultaneously. Thus, parts flow in only one direction, and circular wait never occurs.

An advantage of the prevention mechanism is that it does not require the knowledge of the system state to realize the control, leading to a straightforward often-simple control law.

B. Deadlock Detection/Recovery

Detection/recovery approaches use a monitoring mechanism for detecting the deadlock occurrence and a resolution procedure for pre-empting some deadlock resources. When deadlock occurs, the controller detects it and then recovers by terminating some deadlocked processes or by preempting resources from processes. This strategy can in general allow higher resource utilization than that a prevention method can offer. It should be used when deadlock is rare and detection and recovery are not expensive and much faster than deadlock generation.

Consider the previous example. A deadlock detection procedure monitors the system and allows and detects a deadlock occurrence. Then, a recovery method is employed. For example, suppose that part 1 is in WS1 and requires WS2, and simultaneously, Part 2 is in WS2 and requires WS1. Such a condition is a deadlock. As a recovery procedure, a central or temporary buffer can be used to accommodate either deadlocked piece so that the system can restart its work.

C. Deadlock Avoidance

Deadlock avoidance is a strategy that falsifies one or more of necessary conditions for deadlock by keeping track of the current state and possible future conditions. This is a dynamic approach that can utilize the knowledge of the current allocation of resources and the future behavior of processes to control the release and acquisition of resources. The main characteristic of the deadlock avoidance strategies is that they work in real time and base the decision on information about the resource status. More precisely, a deadlock controller inhibits or enables events involving resources acquisition or release by using look-ahead procedures. Moreover, when a deadlock avoidance algorithm is implemented, it is necessary to consider and avoid not only deadlock states but also unsafe states, i.e., the states that are not deadlocked but lead inevitably the system to a deadlock. By definition, a state is safe if there exists a sequence of resource acquisition and release operations that allows all the processes in the system to reach the completion [3], [38], [55], [56]. Thus, even if deadlock detection can be performed in polynomial time, it is shown that in the general case, the problem of determining the safety of a RAS state is NP-complete [2], [23], [55]. An important consequence is that polynomial deadlock avoidance algorithms must sacrifice flexibility in resource allocation and system performance because they reject some safe state. In particular, Lawley *et al.* [38] single out four properties that a useful deadlock avoidance policy (DAP) must have.

i) *Correctness:* A DAP must guarantee the deadlock-free behavior of the system.
ii) *Scalability:* A DAP must be computationally tractable so that it can be executed in real-time.
iii) *Operational flexibility:* It measures how much a DAP sacrifices system operating flexibility in resource acquisition to achieve scalability.
iv) *Configurability:* A DAP is configurable if the operating constraints that it imposes can be quickly generated for system configurations, i.e., correctness and scalability must be independent of system configuration.

One basic scheme of deadlock avoidance, introduced into the operating system context, is the "banker's algorithm" [21], [23]. It manages multiple units of a single resource by requiring that the processes specify their total resource needs at the initiation time. More precisely, the worst-case scenario, in which all processes simultaneously require their maximum allowable claims for resources to proceed forward, is assumed. The algorithm refuses the requests of the process that needs resources in excess of the available ones. Such a schema is very simple, but it can be conservative from the resource allocation point of view. On the other hand, banker's algorithm does not consider the particular order in which each process acquires and releases resources. Starting from this first procedure, a large number of deadlock avoidance algorithms have been introduced in recent decades to avoid deadlock efficiently. Indeed, deadlock avoidance techniques are considered the most efficient strategies to avoid deadlock in AMS. In particular, deadlock/recovery strategies require continuous monitoring for detecting deadlocks and can lead to a reduction of production rate due to the recovery process. On the other hand, the prevention approaches limit the production flexibility by reducing the variety of the in-process part mix or by imposing the job flow in the same direction. In general, deadlock avoidance results in better resource utilization than prevention. A supervisory controller that enables or inhibits the resource allocation in real time can implement each deadlock avoidance policy. When a part is ready to have its next operation performed or to enter the system, the AMS supervisor grants the request. However, the deadlock avoidance policies are different in operational flexibility or complexity and can yield different performance, depending on AMS layout.

In any case, whatever may be the technique used to characterize deadlock in RAS, it is necessary to describe the job flow through the system and the dynamic of an allocation/deallocation resource mechanism. Hence, each researcher facing deadlock in AMS refers to a DEDS model so that the system state

describes the jobs in process, the status of each resource, the interactions between resources and jobs, and the sequence of resources necessary to process each job. The used models can be in the framework of Petri nets, automata, timed DEDS, and graph theoretic approaches. The following sections discuss the commonly used approaches to deal with deadlock problems.

III. Graph Theoretic Approaches

The graph-theoretic approach is a simple and intuitive tool suitable for describing the interactions between jobs and resources. This peculiarity allows an efficient deadlock characterization even in complex RAS and permits the derivation of detection and avoidance strategies. However, since digraphs are not able to fully model the system dynamics, a discrete event model is necessary to complete the system behavior description. In the sequel, we describe in a tutorial form only one approach based on digraphs. The reason is that the other approaches use similar tools and obtain comparable results. Hence, among the digraph representations proposed in literature, we describe the model that allows us to obtain a rigorous deadlock characterization and efficient deadlock avoidance policies.

A. Definitions and Main Results

In the following, a generic resource is indicated by $\mathrm{r_i}$, with $\mathrm{i} = 1, 2, , \mathrm{R} - 1$, while $\mathrm{C}(\mathrm{r}_i)$ stands for the capacity of r_i, i.e., the maximum number of parts that can contemporaneously hold such a resource. Moreover, we consider a further fictitious resource $\mathrm{r_R}$ to model the system output. Hence, we assume that each part acquires $\mathrm{r_R}$ as it leaves the system. In this way, the complete resource set of AMS is $R = \{\mathrm{r}_i, \mathrm{i} = 1, 2, \ldots, \mathrm{R}\}$. We also assume $\mathrm{C}(\mathrm{r_R}) = \infty$, i.e., there is no restriction on jobs leaving the system. To be processed, each job must receive service by some resources from R in a specified order. Let $\mathbf{w}$ be such sequence of resources (i.e., the *working procedure*), whereas, let $W = \{\mathbf{w}\}$ indicate the set of all the available working procedures necessary to process the jobs. Obviously, $\mathrm{r_R}$ is the last resource in each working procedure.

The considered AMS, processing the set J of jobs, satisfies the following assumptions.

i) Working procedures are deterministic, allowing no alternatives for each operation.
ii) No preemption is allowed.
iii) Each part can be processed by only one machine at any instant of time, i.e., SU-RAS.

On the basis of graph-theoretic approaches, Cho *et al.* [5], Fanti *et al.* [14], [16], Kumaran *et al.* [32], and Wysk *et al.* [61] present algorithms for deadlock detection and avoidance. The important definition and notation about digraphs are summarized in Appendix A.1.

Wysk *et al.* [61] were the first researchers who introduced a graph theoretic approach to characterize deadlock in AMS where each machine can perform only one operation at any instant of time ($\mathrm{C}(\mathrm{r}_i) = 1$ for $\mathrm{i} = 1, 2, , \mathrm{R} - 1$). In his framework, deadlock is defined as a circular wait situation between several machines and can be shown that the total number of deadlock possibilities for any given manufacturing system is

$$\sum_{i=2}^{\mathrm{R}-1} \binom{\mathrm{R}}{i} (-1).$$

They use a digraph whose nodes represent resources and edges represent all possible transitions of the parts in a system. Moreover, they state that a cycle (or circuit) formed by the part routing is a necessary condition for a deadlock. A cycle is a deadlock if it satisfies the sufficient conditions: the number of jobs occupying the nodes of the cycle must be equal to the numbers of nodes and edges of the cycle. Cho *et al.* [5], Fanti *et al.* [14], and Kumaran *et al.* [32] realize that to better characterize deadlock with necessary and sufficient conditions, one needs to use two digraphs to improve such methodology. The first digraph synthetically represents all the possible resource allocations, and the second one describes the current interactions between jobs and resources.

Before rigorously defining such digraphs, it is necessary to describe the AMS model used. From a theoretic point of view, the AMS can be modeled as a DEDS [14], where parts acquire and release resources on the occurrence of some events, such as the arrival of a new job at the system (type-1 event) and the progress of a job from one resource to another or the departure of a job from the system (type-2 event). Obviously, the DEDS state $\mathbf{q}$ (system state) must encapsulate information about the AMS operating conditions concerning the set $J_{\mathbf{q}} \subseteq J$ of jobs in process, the corresponding working procedures, and their *residual working procedures*, i.e., the sequence of resources necessary for each $j \in J_{\mathbf{q}}$ to complete its processing, including the resource currently held. In the following, symbol RWP(j) indicates the residual working procedure pertaining to a job $j \in J_{\mathbf{q}}$.

Using previous definitions and notations, we build two digraphs representing the interactions between jobs and resources in a concise and useful way. The former, which is denoted by $\mathrm{D}_W = (N, E_W)$ and named *working procedure digraph*, represents all the resources pertaining to each working procedure from W in their specific sequence. The vertex set coincides with the resource set, i.e., $N = R$. Moreover, the edge $\mathrm{e_{im}} = (\mathrm{r}_i, \mathrm{r}_m)$, which is directed from r_i to r_m, is in $E_W \subset N \times N$ iff (if and only if) r_m immediately follows r_i in some $\mathrm{w} \in W$. In this case, $\mathrm{e_{im}}$ can be labeled by the working procedures generating it. Obviously, digraph D_W only depends on the mix characteristics and does not vary as the state $\mathbf{q}$ changes. On the contrary, the second digraph $\mathrm{D_{Tr}}(\mathbf{q}) = [N, E_{\mathrm{Tr}}(\mathbf{q})]$, named *transition digraph* (similar to the *system status graph* in [5]), depends on the current DEDS state. As shown by the adopted notation, while the vertex set N still coincides with the resource set and is fixed, the edge set $E_{\mathrm{Tr}}(\mathbf{q})$ changes as $\mathbf{q}$ is updated. In particular, $E_{\mathrm{Tr}}(\mathbf{q})$ is a subset of E_W defined as follows: An edge $\mathrm{e_{im}}$ from E_W is in $E_{\mathrm{Tr}}(\mathbf{q})$ if a part $j \in J\mathbf{q}$ holds r_i in the state $\mathbf{q}$ and requires r_m as the next resource. In constrast to D_W, the transition digraph only exhibits the resources currently held by each job from $J\mathbf{q}$ and required by the same job in the next step.

An example of the working procedure and transition digraphs is demonstrated in the following.

TABLE I
ROUTES ASSOCIATED WITH TWO IN-PROCESS JOBS

Jobs	Working procedures
j_1	$\mathbf{w_1}$=(r_1, r_2, r_3, r_5)
j_2	$\mathbf{w_2}$=(r_3, r_4, r_2, r_1, r_5)

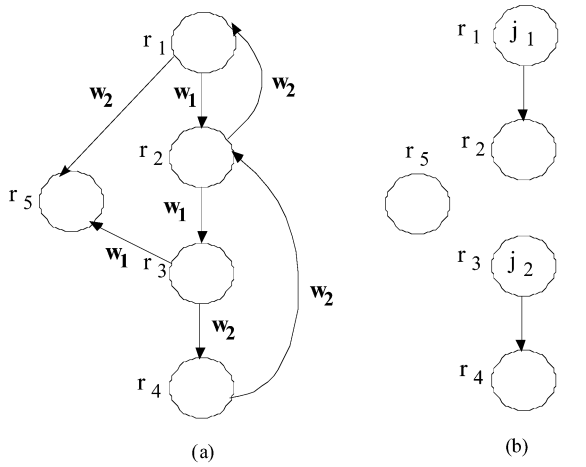

Fig. 3. (a) Working procedure digraph for Example 1. (b) Transition digraph for Example 1.

Example 1—Four Workstation AMS With Two In-Process Jobs: Consider the system shown by Fig. 2. Denote workstation i by r_i, $i = 1, \ldots, 4$ with $C(r_i) = 1$ for i = 1, 3, 4, $C(r_2) = 2$, and $C(r_5) = \infty$, where r_5 represents the system output. Table I describes the two available working procedures, and Fig. 3(a) depicts the corresponding working procedure digraph $D = (N, E_W)$, where N is the set of nodes corresponding to $R = \{r_1, r_2, r_3, r_4, r_5\}$.

Suppose that the system is in state $\mathbf{q}$ such that $J\mathbf{q} = \{j_1, j_2\}$ with RWP$(j_1) = (r_1, r_2, r_3, r_5)$ and RWP$(j_2) = (r_3, r_4, r_2, r_1, r_5)$. Fig. 3(b) illustrates the associated transition digraph $D_{Tr}(\mathbf{q})$.

The necessary and sufficient condition of deadlock in systems with resources of unit capacities is the existence of a cycle in the transition digraph [5], [14], [32]. Some more general results are proved in [16], where it is shown that a deadlock state can be detected easily in SU-RAS using a particular subdigraph of $D_{Tr}(\mathbf{q})$, which is known as the *maximal-weight zero-outdegree strong component* (MZSC).

Definition 1: Let $\mu = (N\mu, E\mu)$ be a strong component of $D_{Tr}(\mathbf{q})$. We call μ a MZSC of $D_{Tr}(\mathbf{q})$ if the following properties hold true.

D1a) *Maximal-weight*: All the resources from $N\mu$ are busy.
D1b) *Zero-outdegree*: All the edges of $D_{Tr}(\mathbf{q})$ outgoing from vertices of $N\mu$ belong to $E\mu$.

There is a simple connection between deadlocks and MZSCs, as established by the following proposition [16].

Proposition 1: $\mathbf{q}$ is a deadlock state iff there exists at least one MZSC in $D_{Tr}(\mathbf{q})$.

The following example clarifies the meaning of such a result.

Example 2—Four-Workstation AMS With Five In-Process Jobs: Let us consider the system of Example 1 in a state $\mathbf{q}$, with $J\mathbf{q} = \{j_1, j_2, j_3, j_4, j_5\}$, RWP$(j_1) = (r_1, r_2, r_3, r_5)$, RWP$(j_2) = (r_2, r_3, r_5)$, RWP$(j_3) = (r_2, r_1, r_5)$, RWP$(j_4) = (r_3, r_4, r_2, r_1, r_5)$, and RWP$(j_5) = (r_4, r_2, r_1, r_5)$. Moreover, the transition digraph contains a MZSC, with $N\mu = \{r_1, r_2, r_3, r_4\}$ and $E\mu = \{e_{12}, e_{23}, e_{34}, e_{42}, e_2\}$ (see Fig. 4). Obviously, no job from $J\mathbf{q}$ can progress to its next resource. Hence, $\mathbf{q}$ is a deadlock state.

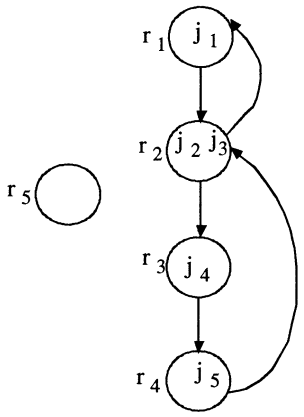

Fig. 4. Transition digraph exhibiting a deadlock.

In this framework a deadlock is a circular wait condition that blocks indefinitely the processing of a set of jobs in the system. Obviously, this definition allows for the possibility that the jobs not involved in the deadlock can end their process routes.

From the computational complexity point of view, a cycle or an MZSC in the transition digraph can be found by using a depth-first search algorithm [52] that requires $O(|E_{Tr}(\mathbf{q})|)$ operations (symbol $|X|$ stands for "cardinality of set X"). The deadlock detection and recovery schemes are described in detail in [11] and are based on the algorithms proposed in [52] to find cycles and strong components. In order to perform a deadlock detection and recovery procedure, the system controller checks if a finished part is involved in a deadlock. If this is the case, the controller moves the part to a special storage buffer and then sequentially moves the other parts when deadlock has been solved, i.e., a part leaves a resource involved in the circular wait condition, the job in the storage buffer can be moved to next machine in its route.

B. Deadlock Avoidance Policies

More efficient strategies based on digraphs are deadlock avoidance policies. In this case, a supervisory controller checks for a deadlock before a part moves to the next destination or enters the system. More precisely, when an event has to occur, the controller answers the following question by using a look-ahead procedure of one step: If it happens, will the next state be a deadlock? To answer the question, the supervisor builds the new transition digraph associated with the system state obtained after the event occurrence and checks for a MZSC. If the updated transition digraph does not contain a MZSC, the controller lets it occur. In the contrary case, the event is inhibited. This DAP is based on a look ahead of one step and inhibits an event iff it leads to a deadlock at next step. As previously pointed out, such a policy can be performed in polynomial time. However, the described DAP can lead the system to unsafe states called impending part flow deadlock

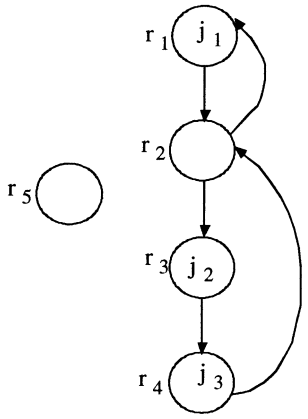

Fig. 5. Transition digraph exhibiting an impending system deadlock.

states [5]. In such states, immediate transitions of a part are possible, but the system (or a portion of the system) will inevitably terminate in a deadlock after a finite number of transitions or events. The following example illustrates such a situation.

Example 3—Four-Workstation AMS With Three In-process Jobs: Consider the system described in Examples 1 and 2 with $\mathrm{C}(\mathrm{r}_2) = 1$. Suppose that the system is in state $\mathbf{q}$ such that $J\mathbf{q} = \{j_1, j_2, j_3\}$, RWP$(j_1) = (\mathrm{r}_1, \mathrm{r}_2, \mathrm{r}_3, \mathrm{r}_5)$, RWP$(j_2) = (\mathrm{r}_3, \mathrm{r}_4, \mathrm{r}_2, \mathrm{r}_1, \mathrm{r}_5)$, and RWP$(j_3) = (\mathrm{r}_4, \mathrm{r}_2, \mathrm{r}_1, \mathrm{r}_5)$. The transition digraph shown in Fig. 5 exhibits an impending part flow deadlock. Indeed, if j_1 acquires r_2, then r_2, r_3, and r_4 will be in deadlock. On the other hand, if j_3 acquires r_2, then r_1 and r_2 will be in deadlock. In this condition, the system is not in a deadlock state, but it will evolve into a deadly embrace in the next step. Such a situation must be avoided.

Unfortunately, impending part flow deadlocks are difficult to detect because it is necessary to explore all the routes of parts in a system. Indeed, the methods to solve a deadlock avoidance problem either do not guarantee that the system will never enter deadlock and impending part flow deadlock, or if such a guarantee is provided, it is obtained at excessive computation cost (the required computation is exponential in the size of the RAS configuration). To address this problem, Cho *et al.* [5] present a heuristic procedure verifying in polynomial time at each event occurrence if there exists an impending part flow deadlock. The procedure is based on the consideration that impending part flow deadlock occurs because of interactions among cycles having only one node in common. However, the procedure performs a complete look-ahead of the assigned routes and can be tedious and computationally very complex when the number of parts and machines is large.

To overcome this difficulty, Fanti *et al.* [14], [16] characterize a particular case of impending part flow deadlock: the second level deadlock (SLD). The SLD is not a circular wait even if it necessarily evolves into a deadlock in the next step. By determining necessary and sufficient conditions that characterize a SLD occurrence, they prove that the maximally permissive deadlock avoidance policy can be performed in polynomial time by a one step look-ahead strategy if an SLD cannot occur. More precisely, such a policy is deadlock-free when the layout of the system enjoys some peculiarities guaranteeing that the second level deadlock cannot occur. For example, this happens if the capacity of each resource in an AMS is more than one or when each unit-capacity resource is equipped with an input or output buffer [14], [16], [39]. On the other hand, if the impending deadlocks can occur, they are prevented by controlling the job-input events and by limiting the number of in-process jobs. An algorithm based on the knowledge of the system layout and on the possible job routing provides the appropriate bound.

Example 4—Four-Workstation AMS With Three In-process Jobs (Continued): If we consider the system described in Examples 1 and 2 with $\mathrm{C}(\mathrm{r}_2) = 2$, the maximally permissive deadlock avoidance policy can be applied with polynomial complexity. On the contrary, if $\mathrm{C}(\mathrm{r}_2) = 1$, a second level deadlock can occur as shown by Example 3. Hence, to avoid impending system deadlocks, the policy proposed in [14] and [16] imposes a maximum of two jobs in the system. Applying this constraint, the state described in Example 3 cannot be reached.

Starting from the previous results, the deadlock avoidance strategies defined for SU-RAS are extended to more complex systems including cellular manufacturing [17] and AMS providing assembly operations [18].

In addition, Lawley [39], [40] uses a digraph named *resource allocation graph* to detect deadlock in $\mathrm{SU}^{(n)}$-RAS, which assumes that parts make disjunctive requests at each step of their processing sequences. In other words, $\mathrm{SU}^{(n)}$-RAS represents AMS with flexible routing, i.e., every operation of every part type can be performed by a machine in a set of n elements. We do not describe in detail this digraph because it is similar to the transition digraph, and the author reaches a similar characterization of deadlock. Moreover, Lawley also establishes the correctness of deadlock avoidance methods for systems with flexible routing and identifies special structures that make the optimally flexible deadlock avoidance algorithm computationally tractable [39].

The deadlock avoidance method in [68] is based on the capacity designed directed graph (CDG) representing the relationship between resources and processing sequences of jobs. In the modeling point of view, CDG is similar to the previously mentioned graphs and considers various capacities of nodes in order to include the modeling of the input/output buffer attached to each machine. A general deadlock avoidance method that can resolve possible deadlocks while preserving a high level of utilization is suggested in [31].

Finally, a deadlock problem is investigated in [20] using graphs in automated tool sharing systems where different machines use the same tool by automatically transferring them from machine to machine in AMSs. A tool allocation graph (TAG) is defined to provide a compact and precise way to represent tool allocation and request. Hence, structural subdigraphs of the TAG characterize deadlock and a polynomial algorithm is proposed for their detection. The detection approach allows a part of the system to become indefinitely blocked and requires a resolution algorithm to resume the part processes. Moreover, an alternative approach is the deadlock avoidance that prevents the allocation of available tools to idle parts and delays their processing. To this end, a banker's algorithm is configured for deadlock avoidance in systems with arbitrary tool sequences.

IV. Automata Approaches

Supervisory control of AMS is a growing area of manufacturing research. One of its essential tasks is to guarantee deadlock-free operation. As an important and efficient formal description of AMS behavior, finite state automata are adopted to establish new deadlock avoidance control techniques [33]–[38], [55], [56]. In their pioneering work [55], Reveliotis and Ferreira propose an analytical framework to design deadlock avoidance policies for a subclass of resource allocation systems (SU-RAS). They describe the AMS as a finite state automaton model with static routing. The events of the automaton are

i) loading a new job;
ii) advancing a partial process job;
iii) unloading a finished part.

The state of the automaton is defined as a vector $\mathbf{q} = [\mathbf{q}_1 \ \ldots \ \mathbf{q}_k \ \ldots \ \mathbf{q}_W]$, where the components of $\mathbf{q}_{\mathrm{k}}$ correspond to the distinct job stages of the kth working procedure, i.e., $\mathbf{q}_{\mathrm{k}}(\mathrm{h})$ is equal to the number of jobs executing the hth stage of the kth working procedure. Named $\mathrm{L_t}$, the sum of the lengths of all the working procedures $\mathbf{q}$ is a vector of $\mathrm{L_t}$ elements. Moreover, the initial and final states correspond to the state $\mathbf{q}_0$ in which the AMS is empty of jobs. The AMS state safety is characterized with the aid of the automata state transition diagram. Therefore, in order to avoid deadlock in the least restrictive way, the system operation should be confined to the maximal communicating set of states containing $\mathbf{q}_0$. The novelty of the state safety decision problem presented in [55] and in the subsequent papers [36]–[38] and [56] is that it serves as a framework for the analysis of the correctness of certain class of deadlock avoidance policies. This class of policies can be formulated as a set of inequalities, which constrain the allocation of system resources at any single instance

$$\mathrm{Aq} \leq \mathrm{f}$$

where $\mathbf{A}$ is an incidence matrix. Each row of $\mathbf{A}$ can be associated with a subset of process stages. Matrix $\mathbf{A}$ and vector $\mathbf{f}$ characterize a particular deadlock avoidance policy. These constraints are static, i.e., they consider only the present status of resource allocation and not its history. Moreover, each policy can be easily synthesized in polynomial time by a supervisory controller.

As an example, consider the avoidance policy known as the resource upstream neighborhood (RUN) policy [55]. Such a control scheme is based on the idea that there are some resources with higher capacity than others in the system from the resource set R that can function as temporary buffer for the jobs that they support. More precisely, the upstream neighborhood of the resource $\mathrm{r_k} \in R$ is indicated by $\mathrm{UN(r_k)}$, and it is defined as the union of resources that precede $\mathrm{r_k}$ in the routing steps that it performs, with the property that every resource $\mathrm{r_i}$ in $\mathrm{UN(r_k)}$ has capacity $\mathrm{C(r_i)} \leq \mathrm{C(r_k)}$. The RUN policy requires that the number of jobs in the upstream neighborhood of a resource never exceeds the capacity of that resource. Moreover, the policy is proved to be deadlock and RD-free with complexity $\mathrm{O(R \times W \times L^2)}$, where L is the maximum number of operations in each working procedure. The efficiency of the control policy has been studied in [33] and [34].

Example 5—Four-Workstation AMS (Continued): Let us consider again the system described in Example 1 with $\mathrm{C(r_i)} = 1$ for $\mathrm{i} = 1, 3, 4$ and $\mathrm{C(r_2)} = 2$. By previous definitions we obtain: $\mathrm{UN(r_1)} = \{\mathrm{r_3}, \mathrm{r_4}\}$, $\mathrm{UN(r_2)} = \{\mathrm{r_1}, \mathrm{r_3}, \mathrm{r_4}\}$, $\mathrm{UN(r_3)} = \{\mathrm{r_1}\}$, and $\mathrm{UN(r_4)} = \{\mathrm{r_3}\}$. The RUN policy imposes at most one job in $\mathrm{UN(r_1)}, \mathrm{UN(r_3)}$ and $\mathrm{UN(r_4)}$ and at the most two jobs in $\mathrm{UN(r_2)}$. Since $\mathrm{w_1}$ is composed of three operations and $\mathrm{w_2}$ is of four stages, the vector state $\mathbf{q} = [\mathbf{q_1}\mathbf{q_2}]$ has seven elements (we do not consider the stage corresponding to the fictitious resource). More precisely, $\mathbf{q_1}(\mathrm{h})$ for $\mathrm{h} = 1, 2$, and 3 is equal to the number of jobs executing the hth operation of $\mathrm{w_1}$ and $\mathbf{q_2}(\mathrm{h})$ for $h = 1, 2, 3$, and 4 is equal to the number of jobs executing the hth stage of $\mathrm{w_2}$. The RUN policy can be expressed by the following linear inequalities:

$$\mathrm{A}\mathbf{q} \leq \mathrm{f}$$

with

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{f} = \begin{bmatrix} 1 \\ 2 \\ 1 \\ 1 \end{bmatrix}.$$

Each row of the matrix $\mathbf{A}$ corresponds to a resource neighborhood and vector $\mathbf{f}$ is obtained by the RUN policy specification.

On the basis of the mentioned problem formulation, other deadlock avoidance policies are defined and applicable to different RAS types. More precisely, Reveliotis *et al.* [56] generalize the RUN policy for more complicated resource allocation systems, i.e., for the more general case of conjunctive-RAS models. Based on the same framework, Lawley *et al.*. [38] define a resource order (RO) deadlock avoidance policy. Starting from the idea that parts traveling through the same machines but in opposing directions must at some point be able to pass, the policy orders the machines and each part is categorized according to how it flows with respect to that order. Since the RO policy can be expressed as a set of linear inequalities, its application is very simple and with polynomial complexity. However, to implement RO policy, it is necessary to order the machines, and some orders generate more restrictive constraints than others do. Hence, good orders yielding less restrictive constraints are selected through mathematical integer programming. For reasonably sized systems, the problem can be solved to optimality, while for larger systems, suboptimal solutions can be obtained. Moreover, this policy can be efficiently generalized to ST-RAS [56]. Supposing that either RUN or RO avoids deadlock, in [53], the integration of deadlock avoidance control with dynamic policy reconfiguration and job rerouting is considered in order to accommodate/minimize the disruptive effects of operational contingencies like the following: i) a resource fails or ii) a job arrives that requires immediate processing. To this aim, the RUN and RO policies, expressed as a set of linear inequalities on the system state, use an available policy reconfiguration to accommodate the unexpected contingencies with minimal disruption.

In a more formal and generic framework of Ramadge–Wonham [50], Li and Wonham [43] model a DES as a controlled generator G of a formal language $\mathrm{L}(\mathrm{G})$ to characterize deadlock. Hence, they investigate the system property of deadlock, which is defined as a situation in which the system progress is blocked indefinitely, i.e., it is a *total deadlock*. Consequently, the concepts of deadlock-free supervisor and deadlock-free language are introduced and solved.

Using the same framework, Yalcin and Boucher [67] model an AMS as a finite automaton that describes the states of resources and possible part movements. The process plans for different part types are individually modeled as finite automata and are used to define the rules that govern the system. The supervisor restricts events in the system so that the remaining processing requirements of the parts in the AMS can be completed without incurring deadlock. To define the marked sequences of events of the closed-loop system, the algorithm generates the state space and determines the possible control pattern to reach the final state. When more alternative sequences exist, some network flow techniques can be applied to the directed graph representing the automaton state space for the optimal selection of the sequence. The proposed methodology to avoid deadlock is maximally permissive, but it requires the analysis of the state space that increases exponentially with the number of resources and the number of process plans. Consequently, this strategy is not suitable for application in real AMSs.

Ramirez-Serrano and Barnabib [51] consider a multiworkcell AMS that is interrelated by common shared resources and develop a strategy for the synthesis of deadlock-free supervisors to control the workcells individually. Extended Moore automata (EMA), i.e., finite automata with outputs and inputs, model the behavior of workcells, where the set of manufacturing constraints is to be imposed on the workcells and the possible sequences in which parts can be concurrently processed. The supervisors are synthesized for all workcells such that no deadlock can occur, i.e., the system cannot reach a state from which it is impossible to return to the initial state (*partial deadlock* free). They propose an algorithm to identify deadlock states when multiple devices are shared by multiple workcells and synthesize a deadlock-free supervisor for each workcell. The strategies check for deadlocks for every interrelated combination of workcells and, if deadlock states are found, reconfigures the manufacturing conditions and resynthesizes the corresponding supervisors. Considering the complexity, the proposed approach is less computationally intensive than a traditional approach that synthesizes a global supervisor.

We conclude this section recalling a recent paper [41], where authors develop the notion of robust supervisory control for AMS with unreliable resources. Moreover, they introduce the important concept of the robustness of a deadlock avoidance policy with reference to the resource failures. In addition, they propose two polynomial control policies that ensure safety of the system while unreliable resource is failed and resource is repaired.

V. Petri Net Approaches

Many researchers use Petri nets (PNs) as a formalism to describe AMSs [8], [69], [70] and to develop appropriate deadlock resolution methods [3], [6], [9], [10], [26], [45], [46], [57]–[60], [63]–[66], [69]–[71]. However, deadlock definitions are different in this framework and can lead to confusion and imprecision. Appendix A.2 gives some related notation and definitions considering ordinary PN.

In particular, the following definitions are important to clarify the connection between the deadlock in AMS and the properties of PNs. More precisely, a Petri net PN $= (P, T, F, \mathrm{M}_0)$ at the initial marking M_0 is said to be *live* if, no matter what marking has been reached from M_0, it is possible to fire ultimately any transition of the net by progressing through some further firing sequence. Consequently, *liveness* means that for every reachable state, the model can evolve in such a way that every transition can always fire in the future, and in other words, every system activity can ultimately be carried out [9]. Hence, a live PN guarantees deadlock-free operations. On the other hand, a dead marking or *total deadlock* is a marking M such that no transition is enabled, and translating this idea to the AMS domain, a dead marking represents a state where no piece can move forward.

Moreover, a PN is said to be *deadlock-free* if at least one transition is enabled at every reachable marking. This definition does not guarantee that a circular wait condition involving a part of the system cannot occur. In general, deadlock freeness in PN definition does not prevent a *partial deadlock* between a subset of resources and a subset of in-process parts.

Now, deadlocks are also linked to particular structures of PNs called siphons. More precisely, a subset of places S is called siphon if ${}^\bullet S \subseteq S^\bullet$, i.e., an input transition is also an output transition of S[44]. A siphon is said to be minimal if it does not contain other siphons. A PN is deadlock-free if no minimal siphon eventually becomes empty.

With these definitions as background, we present a review of the PN approaches to the AMS deadlock analysis.

A. Deadlock and PN Liveness

Some of the first investigations on deadlock in AMS are reported in [69] and [70]. They analyze and synthesize PN models for AMS with shared resources so that the PN is bounded, live, and reversible. As recalled before, the liveness condition implies the absence of deadlocks and is obtained by adding buffer modules and control places or limiting the number of in-process parts. Hence, in their framework, deadlock is solved using a prevention technique.

Viswanadham *et al.* [60] propose deadlock avoidance and prevention techniques. They model the system and job processes by a generalized stochastic PN. The authors define a deadlock marking as a dead marking. This definition sees a deadlock as a *total deadlock* where no part in the system can complete its route. They propose a deadlock prevention policy, which executes an exhaustive analysis of the reachability graph and considers the graph paths that do not reach a deadlock marking. The reachability analysis technique to arrive at deadlock prevention policies can become infeasible if the state space is very large. Moreover, the avoidance policy introduced in [60] consists of the construction of the reachability graph for a finite number of look-ahead steps to check if the system reaches a deadlock state. If so, the policy avoids the undesirable evolution by controlling

events or initiates appropriate recovery procedure if it is necessary.

More efficient deadlock avoidance policies are introduced in [3], [57], and [58]. Using production Petri nets (PPNs), the authors describe the AMS consisting of a set of resources $R = \{\mathrm{r_i}\}$ with assigned capacity $\mathrm{C(r_i)}$. Two places $\mathrm{a_{ri}}$ and $\mathrm{b_{ri}}$ are assigned to each resource $\mathrm{r_i} \in R$: Tokens in $\mathrm{b_{ri}}$ represent jobs detaining the resource, and tokens in $\mathrm{a_{ri}}$ indicate available resources of type $\mathrm{r_i}$. For each reachable marking $\mathrm{M} \in \mathrm{R(M_0)}$ and for each $\mathrm{r_i} \in R$, it holds that $\mathrm{M(a_{ri}) + M(b_{ri}) = C(r_i)}$. The working procedure to produce a particular product results in a sequence of resource requirements and associates a place-transition sequence with each product type. Places represent individual steps in the sequence of operations necessary to produce a particular part where each step requires a resource. A transition models releasing or acquiring resources by the parts. Denote the set of job types by Q. If processing a q-type product ($q \in Q$) requires L_q steps, we define the following place and transition sequences:

$$p_q = \{p_q(0), p_q(1), p_q(j), p_q(L_q), p_q(L_q+1)\} \quad (1)$$

$$\boldsymbol{t}_q = \{t_q(0), t_q(1), t_q(j), t_q(L_q), t_q(L_q+1)\}. \quad (2)$$

Tokens in $p_q(0)$ represent q-type jobs waiting for their processing; those in $p_q(L_q+1)$ indicate completed orders; and those in $p_q(j)$, $j \in \{1, 2, \ldots, L_q\}$ give the currently processed jobs in the jth step. $t_q(0)$ denotes the arrival of an order to execute a job for product q; firing $t_q(L_q+1)$ indicates the completion of a q-type job; and firing $t_q(j)$, $j \in \{1, 2, \ldots, L_q\}$ means that a q-type job makes transition from the $(j-1)$th production step to the jth step.

The deadlock avoidance policy is defined as a resource allocation policy [restriction policy (RP)] that selects the enabled transitions to fire. To apply this RP, it is necessary to partition the system resources into two classes: The former collects all the unshared resources, i.e., all the resources required once in only one production sequence; the latter groups the remaining resources (i.e., the shared resources). In this way, for each job-type $q \in Q$, it is possible to split up the place sequence (1) into subsequences called zones:

$$p_q = p_q(0)\boldsymbol{z}_q^1\boldsymbol{z}_q^2 \ldots \boldsymbol{z}_q^{\mathbf{k}} \ldots p_q(L_q+1). \quad (3)$$

Each zone consists of two subzones

$$\boldsymbol{z}_q^{\mathbf{k}} = \boldsymbol{s}_q^{\mathbf{k}}\boldsymbol{u}_q^{\mathbf{k}} \quad (4)$$

where $\boldsymbol{s}^{\mathbf{k}}{}_q$ represents a subsequence of production steps requiring shared resources, and $\boldsymbol{u}_q^{\mathbf{k}}$ indicates a sub-sequence of production steps requiring unshared resources.

The RP allows an enabled transition t to fire only if the following conditions hold true:

RPA) If, by firing t, a job enters a new zone $\boldsymbol{z}_q^{\mathbf{k}}$ in its production sequence, then the capacity of the unshared subzone $\boldsymbol{u}^{\mathbf{k}}q$ must exceed the number of jobs currently in the zone $\boldsymbol{z}_q^{\mathbf{k}}$.

RPB) If, by firing t, a job requests a shared resource, then all the resources in the remainder of the zone must be available, except the one held by the job, at the most.

This RP establishes a simple feedback control law. In general, applying an RP could lead to new situations, named restricted deadlocks (RD), in which an indefinite circular wait occurs partly because some transitions are not enabled and partly because some enabled transitions cannot fire owing to the RP. In this case, an RP itself helps to form a circular wait, with the same consequence as a deadlock. By using the PPN, it is shown that the RP [3], [57] prevents the system from reaching a deadlock or RD, and hence, it is *deadlock-free*. Moreover, their RP is less conservative than existing deadlock avoidance algorithms for computer operating systems because it is based only on resource requirements in the current production sequence zone for each job rather than the entire set of resource requirements. On the other hand, such a policy sometimes appears too conservative because it allows a limited number of jobs in process, resulting in poor system performance. In particular, this holds true when the number of unshared resources is small. However, it remains a simple real-time control policy that avoids deadlock and restricted deadlock with little computation.

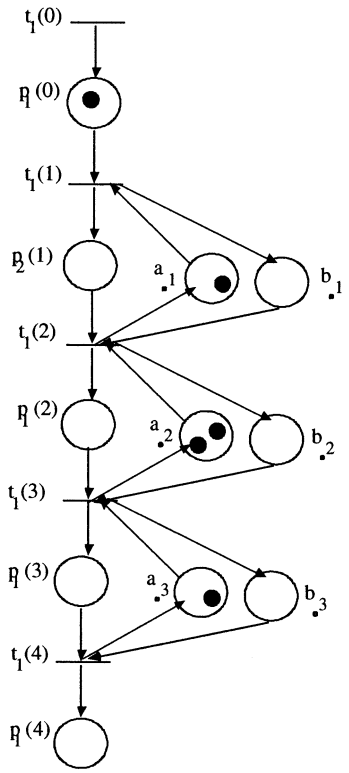

Fig. 6. PPN modeling $\boldsymbol{p}_1$ for example 6.

Example 6—Four-Workstation AMS (Continued): Consider once again the four-machine system without buffers shown described in Example 1. There are two job types ($Q = \{1, 2\}$) performed respectively, by the working procedures $\mathbf{w}_1$ and $\mathbf{w}_2$, which are shown in Table I with $L_1 = 3$ and $L_2 = 4$. As an example, Fig. 6 depicts the PPN modeling a type 1 part's processing $\boldsymbol{p}_1$.

Since only resource $\mathrm{r_4}$ is unshared, the sequence is partitioned as

$$p_1 = p_1(0)\boldsymbol{z}_1^1 p_1(4), \text{ with } \boldsymbol{z}_1^1 = \boldsymbol{s}_1^1 = \{\mathrm{r_1}, \mathrm{r_2}, \mathrm{r_3}\}$$

and

$$\boldsymbol{p}_2 = p_2(0)\boldsymbol{z}_2^1\boldsymbol{z}_2^2 p_2(5) \text{ with } \boldsymbol{z}_2^1 = \boldsymbol{s}_2^1\boldsymbol{u}_2^1 = \{\mathrm{r_3}\}\{\mathrm{r_4}\},$$
$$\boldsymbol{z}_2^2 = \boldsymbol{s}_2^2 = \{\mathrm{r_2}, \mathrm{r_1}\}.$$

Since the capacity of each resource $\mathrm{r_i}$ with $\mathrm{i} \neq 2$ is equal to 1, their RP allows a job of type 1 to acquire a resource iff each resource of its residual sequence has at least an item at its disposal. On the other hand, a job of type 2 can enter $\boldsymbol{z}_2^1$ only if $\mathrm{r_4}$ is idle and can acquire $\mathrm{r_2}(\mathrm{r_1})$ only if $\mathrm{r_1}$ ($\mathrm{r_2}$, respectively) can accommodate one more job.

Hsieh and Chang [27] use a bottom up approach for synthesizing a controlled production Petri net (CPPN) that considers a larger class of AMS where a manufacturing operation requires multiple resources. The CPPN model consists of resource subnets describing resources, job subnets describing processes, and exogenous controls. A deadlock state is defined as a state under which a set of jobs are circular, waiting for resources, and no operations can go on among them. Hence, to avoid deadlock, it is sufficient to guarantee that each operation be performed infinitely many times, i.e., to keep the system live. To this aim, they find the conditions to ensure that the CPPN is live under a control policy and an initial marking. The synthesis of the deadlock avoidance controller is based on the concept of minimal resource requirement, i.e., the minimal number of resources assuring the existence of a system evolution that allows all the jobs in the system to complete. The deadlock avoidance controller includes a sufficient validity test procedure to check whether the execution of a control action is valid to maintain the liveness of the CPPN. If the control fails the test, a valid control action is generated by the validity test. In particular, the validity test algorithm is based on the determination of a sequence of markings leading to an empty system, where every process is finished. The proposed procedure is of polynomial complexity and has good potential for real-time applications. As the restriction policy of [3], the drawback of this strategy is that it does not provide a method to apply a maximally permissive deadlock avoidance policy when it is scalable.

In the field of complex RAS, some new contributions address the problem of determining suited deadlock avoidance policies [10], [27]. In particular, [10] develops a deadlock-avoidance method applicable to systems with the following particular characteristics: Production orders are allowed to have assembly/disassembly operations and flexible part routing. The paper adopts a mixed approach based on the computation of the reachability graphs of the different production orders obtained by modeling the system by a PN belonging to the class of systems of processes with resources ($\mathrm{S^*PR}$). This class of nets described with more detail in the next section can model in a natural way both flexible routing and the use of several resources at each processing step. In this framework, a polynomial time complexity adaptation of the Banker's algorithm for deadlock avoidance is proposed.

B. Deadlock and Siphons

The basic contributions described in the previous section do not characterize deadlock states but just require the PNs liveness. Like [3], Xing *et al.* [65] model a production system by using PPN and define a set of transitions in deadlocks as a set of transitions that cannot fire any more because they are not enabled by resource places. To guarantee that all the transitions of the PPN are not in deadlock, they define a PN structure that is related to siphons that must not be empty. When the number of any kind of resources is greater than one, necessary and sufficient conditions are considered which guarantee against occurrence of any deadlock. The peculiarity of this policy with respect to the RP in [3] is that if all the resource capacities are more than one, the policy is maximally permissive because it cannot incur restricted deadlock. However, in the general case, it is of exponential complexity in the numbers of utilization times of each kind of shared resources.

Another important and used model is proposed by Ezpeleta *et al.* [9], who describe a production system using a particular class of nets called systems of simple sequential processes with resources ($\mathrm{S^3PR}$). This class of models is a generalization of the one used in [3] and [27] since their working procedures allow flexible routing. The deadlock definition in AMS is translated into the liveness definition of the net model. Since the liveness property means that every production process can always introduce in the system new products to be manufactured, the system is deadlock free if every transition can always fire in the future. In the case that the model is not live, a control policy will constrain the system behavior to a set of states so that, whichever state the system reaches, there is always a system evolution. Thus, under the control policy, the treatment of each product can reach its final state. Hence, the paper analyzes the liveness of the $\mathrm{S^3PR}$ net. First of all, they show that a marked $\mathrm{S^3PR}$ net is live if and only if, for each reachable marking from the initial marking, each siphon has at least one token. Considering that a siphon is called minimal iff it includes no other siphon as a proper set, the control policy determines the set of minimal siphons that can be emptied and introduces additional places that constrain the behavior of the system.

The intuitive meaning of the control prevention policy is as follows. Firing each transition modeling the start of a process needs a token from the added places related to the siphons that may lose all its tokens otherwise. It is proved that the controlled system is live. Moreover, the control policy establishment complexity is strongly conditioned by the complexity of computing the set of minimal siphons. Indeed, the number of siphons is exponential with the number of places in the net. However, the computational effort in order to obtain the set of minimal siphons is not critical because this computation is carried offline only once before implementing the deadlock prevention procedure.

Example 7—Four-Workstation AMS (Continued): Consider once again the four-machine system without buffers described in Example 1. Fig. 7 shows the $\mathrm{S^3PR}$ net modeling the system. The two siphons that can eventually become empty are $S_1 = \{\mathrm{r_1}, \mathrm{p_1}(2), \mathrm{r_2}, \mathrm{p_2}(4)\}$ and $S_2 = \{\mathrm{r_2}, \mathrm{p_1}(3), \mathrm{r_3}, \mathrm{r_4}, \mathrm{p_2}(3)\}$. Moreover, Fig. 7 shows that the introduced additional places V_{S1} and V_{S2} and the added flow relations (the sketched arcs) apply the obtained constraints: S_1 and S_2 cannot become empty.

By Example 7, it is clear that the prevention policy applies on line, by the additional places, the constraints determined by the offline setting of the conditions. Consequently, the resolving policy is realized by some constraints of the resource allocation, and it works in real-time on the basis of the system state knowledge. Hence, it is difficult to establish the true difference between avoidance and prevention methods. We observe that only

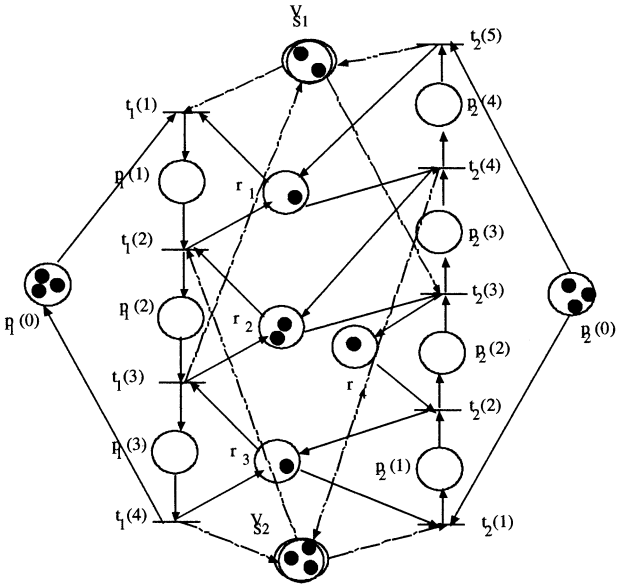

Fig. 7. Controlled system of the S^3PR net modeling the four-machine system of Example 3.

a look-ahead strategy can be properly defined as an avoidance policy because it does not prevent *a priori* any system configurations, and the decision of allowing or forbidding a state transition is taken on line.

Starting from the results of [9], Abdallah and Elmaraghy [1] model the system by using S^4R nets, which are a generalization of S^3PR nets, and PPN nets. Indeed, in such a model, an activity can use not only alternative resources, as in S^3PR nets, but it can also utilize more than one resource simultaneously. A deadlock prevention policy is proposed that augments the S^4R net by adding a local control place for every siphon so that it remains marked for all reachable marking. Moreover, a deadlock avoidance policy is introduced that controls the augmented S^4R net using a resource allocation policy based on the unsafe marking concept. The method is based on the look-ahead procedure, i.e., the controller determines one step of future evolution before making a resource allocation decision. However, the deadlock avoidance policy is not deadlock-free in any case. Indeed, the system may reach a dead marking, and if this is the case, the controller starts a recovery procedure.

Chu and Xie [6] analyze deadlock in PNs that can potentially model different discrete event systems. They state that a PN subclass known as augmented marked graphs (AMGs) is deadlock-free if no siphon is a potential deadlock where a siphon is a potential deadlock iff it can be emptied. Using a linear programming approach, it is necessary to examine all minimal siphons. Furthermore, the authors show that it is possible to check deadlock freeness without generating all the minimal siphons. Indeed, mathematical programming or mixed integer programming problems find the maximal siphon unmarked of a reachable marking and checks whether there exists a minimal siphon that eventually becomes unmarked. A consequence of these results is that liveness for some particular nets can be checked in polynomial time. Moreover, the proposed methods are applied to PN modeling of manufacturing systems and provide prevention and detection methods. The prevention algorithm is presented in detail in a successive paper [28], where the manufacturing system is modeled by an S^3PR net. The method is an iterative approach consisting of two main stages. The first stage, which is known as *siphon control*, adds, for each unmarked minimal siphon, a control place to the original net with its output arcs to the sink transitions of the minimal siphon. The second stage, which is known as *augmented siphon control*, assures that there are no new unmarked siphons generated when adding the control places of the first stage. The authors show that the resulting control policy is less restrictive than the prevention method proposed in [9].

Starting from the basic results obtained in [6], Park and Reveliotis establish in [46] that AMG, for example S^3PR nets, can effectively model the controlled behavior of a class of sequential RAS under algebraic polynomial deadlock avoidance policy control. Hence, RUN and RO policies are improved by allowing more permissive constraints. More precisely, the AMG-based framework has the capability of enhancing the flexibility of any existing algebraic DAP and enriching the space of effectively computable algebraic DAP with new policy instantiations. In [46], a novel framework using PN structural analysis is provided for analyzing the correctness of any tentative algebraic deadlock avoidance policy.

The above-mentioned approaches successfully address the deadlock avoidance problem primarily in the context of SU-RAS. Recently, Park and Reveliotis [47] have shown that the empty siphon can be the deadlock-interpreting mechanism, even in the case of CD-RAS. On the basis of the results obtained in [46], they provide a complete PN-based structural characterization of CD-RAS liveness through a new siphon construct that effectively extends the notion of an empty siphon to a non ordinary generalization of S^3PR and S^4PR nets called system of simple sequential processes with general resource requirements, which is denoted by S^3PGR^2. The paper investigates the liveness properties of S^3PGR^2 nets and establishes their strong relationship to the development of a new siphon construct, called the deadly marked siphon. The paper exploits the derived siphon-based liveness characterization of CD-RAS in order to develop a sufficiency test for the correctness of CD-RAS DAP that can be expressed as a set of invariant imposing control places, superimposed on the PN modeling the original RAS behavior.

Finally, in a subsequent work [48], the authors synthesize a liveness-enforcing supervision which is scalable and correct for CD-RAS that present uncontrollability with respect to i) the timing of some requested resource allocations, i.e., these allocations will take place as long as the requested resources are available, and ii) the routing of certain jobs that, after some processing stages, might request special treatment or rework. In addition, the paper addresses the accommodation in the liveness-enforcing supervision synthesis problem of forbidden state specifications.

On the basis of the siphon analysis approach, some mixed models are introduced to improve the controller synthesis and the model potentiality. To this purpose, we recall the paper of Fanti *et al.* [15] that compares PN and digraph models to solve deadlock in AMS. First, digraphs are more concise tools than PPN and S^3PR nets because their vertices only represent the

system resources. On the other hand, PNs are bipartite digraphs whose place-type nodes represent available or used resources and process states for the in-process jobs, while transition-type nodes describe events representing changes in resource allocations. Second, PNs can describe fully the dynamic behavior of a system, while digraphs describe just the interaction between jobs and resources. For these reasons, the paper indicates a tight connection between the two approaches to the deadlock problem, building a unitary framework that links graph-theoretic and PN models and results. In particular, it is shown that iff there is an unmarked siphon in the PN modeling the system, then the transition digraph contains a particular subdigraph structure. Finally, it shows how avoidance policies derived from digraphs can be implemented by controlled PNs.

In this context, [22] and [41] deal with deadlock avoidance for a large class of multiple reentrant flow-lines using S^3PR nets to explore the exact relationship among circular waits, siphons, and deadlock situations. A novel matrix-based model of AMS is employed which, together with the PN marking transition equation, provides a complete dynamical description for the manufacturing system. The matrix framework is very convenient for computer simulation as well as for supervisory controller design. The introduced deadlock-avoidance schema is based either on a critical siphon that becomes empty or on a wait relation graph exhibiting a circular wait condition. Based on these structures, the matrix equations of the controller are written in *(min, +)* algebra, and a minimally restrictive control policy is synthesized. The originality of this method consists in the use of the matrix-based model of discrete event manufacturing systems utilized in conjunction with the PN model and the digraphs to synthesize a job-sequencing controller with resource allocation. The approach leads to an efficient and simple design of a supervisory controller to avoid deadlock.

C. Deadlock and Circuits

A PN model, which is called the resource oriented PN (ROPN), is developed by Wu [63]. The resource-oriented model mainly describes the dynamic of the resource allocation instead of the production processes of operations in the system. In this proposed framework, just one place is necessary to model a resource. More precisely, the authors define a ROPN $= (P, T, \mathbf{I}, \mathbf{O}, \mathbf{M}, \mathrm{K})$, where the function $\mathrm{K} : P \rightarrow \{0, 1, 2, \ldots\}$ is a capacity function, where $\mathrm{K(p)}$ represents the number of tokens that place p can hold. In this framework, the transition $\mathrm{t} \in T$ in the ROPN is enabled if for all $\mathrm{p} \in P$

$$\mathrm{M(p)} \geq \mathbf{I}(\mathrm{p}, \mathrm{t}) \tag{5}$$

and

$$\mathrm{K(p)} \geq \mathrm{M(p)} - \mathbf{I}(\mathrm{p}, \mathrm{t}) + \mathbf{O}(\mathrm{p}, \mathrm{t}). \tag{6}$$

If a transition is enabled, it can fire. Firing an enabled transition t in marking $\mathbf{M}$ changes $\mathbf{M}$ to $\mathbf{M}'$ according to

$$\mathbf{M}'(\mathrm{p}) = \mathbf{M}(\mathrm{p}) - \mathbf{I}(\mathrm{p}, \mathrm{t}) + \mathbf{O}(\mathrm{p}, \mathrm{t}). \tag{7}$$

In ROPN, when a token in a place and (6) is satisfied for one of the output transitions of that place, we say that the transition is process-enabled. Moreover, if (7) is satisfied for a transition, we say that the transition is resource enabled.

We assume the following.

i) Each machine has one output buffer with finite capacity.
ii) No job visits the same machine consequently.
iii) There is a load-unload station with infinite capacity.

Assumption i) can be relaxed to a machine with or without input and/or output buffers.

Parts in the system are represented by the tokens. To distinguish different processes, colors [30] are introduced into the ROPN, resulting in colored ROPN (CROPN). The CROPN is a class of finite capacity PNs and it is structurally a strongly connected state machine PN. Because of the equivalence between deadlock in the AMS and liveness in the CROPN, the sufficient and necessary conditions for deadlock occurrence are proved. Before introducing the deadlock avoidance policy proved in [63] and [64], the following definitions are necessary.

a) Regarding transitions and places as nodes of the net, a *circuit* is a path from a node to the same node where every transition and node appears once, except for the last one.
b) Symbol $\mathrm{v_i}$ indicates a *production process circuit* (PPC) if it is a *circuit* in the CROPN that does not contain the place $\mathrm{p_0}$ representing the load/unload station.
c) In ROPN, a subnet formed by a number of PPCs is said to be an interactive PPC subnet if every PPC in the subnet has common places and transitions with at least one other PPC in the subnet, and the subnet is strongly connected. We denote with $\mathrm{v^n}$ an interactive PPC formed by n PPCs.
d) A transition t is said to be an input transition of an interactive subnet $\mathrm{v^n}$ in a CROPN if t is not in $\mathrm{v^n}$ and the output place of t is in $\mathrm{v^n}$. Symbol $\mathrm{T_I(v^n)}$ denotes the set of the input transitions of $\mathrm{v^n}$.
e) We call a transition an intercircuit input transition (IIT) if it belongs to a PPC and its output place belongs to another PPC in the same $\mathrm{v^n}$.

Now, the following policy, which is called the L-policy, is established in [64]

L-Condition: The L-condition holds for PPC v and marking M iff there is at least a nonshared free space available in v at M.

L-Policy: The L-policy is a control policy under which 1) the L-condition holds for any v in $\mathrm{v^n}$ after t's firing if $t \in \mathrm{T_I(v^n)}$ is selected to fire, and 2) the L-condition holds for any v in subnet $\mathrm{v^{n-1}}$ of $\mathrm{v^n}$ after IITs firing if it is external to $\mathrm{v^{n-1}}$ and selected to fire.

It is proved that a CROPN controlled by the *L-policy* is live.

The following example shows the application of the conditions and the obtained control policy for deadlock avoidance.

Example 8—Four-Workstation AMS With Buffers (Continued): The CROPN of Fig. 7 models the system described in Example 3. In this case, we consider each machine provided with an output buffer of capacity equal to 1. The parts of type 1 can be released into the system from $\mathrm{p_0}$ to $\mathrm{p_1}$ by going through $\mathrm{t_1(I)}$. Moreover, the piece of type 2 enters the system from $\mathrm{p_0}$ to $\mathrm{p_4}$ by the firing of $\mathrm{t_2(I)}$. When the part is completed by machine 3, it goes back to $\mathrm{p_0}$ through $\mathrm{t_1(O)}$. On the other

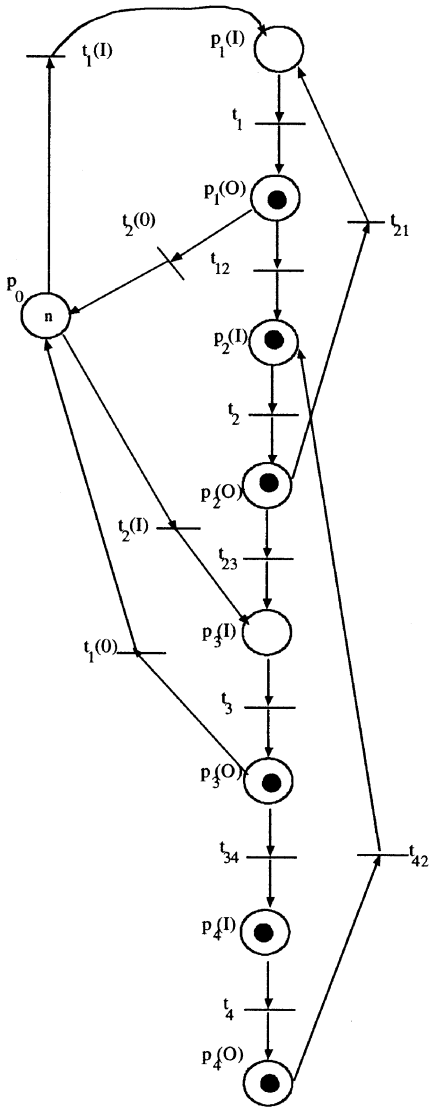

Fig. 8. Resource-oriented PN.

hand, when a part is completed by machine 1, it goes back to p_0 through $\mathrm{t}_2(\mathrm{O})$. The CROPN contains two PPCs:

$$\mathrm{v}_1 = (\mathrm{p}_1(\mathrm{I}), \mathrm{t}_1, \mathrm{p}_1(\mathrm{O}), \mathrm{t}_{12}, \mathrm{p}_2(\mathrm{I}), \mathrm{t}_2, \mathrm{p}_2(\mathrm{O}), \mathrm{t}_{21}, \mathrm{p}_1(\mathrm{I})) \text{ and}$$
$$\mathrm{v}_2 = (\mathrm{p}_2(\mathrm{I}), \mathrm{t}_2, \mathrm{p}_2(\mathrm{O}), \mathrm{t}_{23}, \mathrm{p}_3(\mathrm{I}), \mathrm{t}_3, \mathrm{p}_3(\mathrm{O}), \mathrm{t}_{34}, \mathrm{p}_4(\mathrm{I}))$$
$$\mathrm{t}_4, \mathrm{p}_4(\mathrm{O}), \mathrm{t}_{42}, \mathrm{p}_2(\mathrm{I})).$$

These two PPCs form an interactive subnet v^2 with IIT $= \{\mathrm{t}_{12}, \mathrm{t}_{42}\}$ and $\mathrm{T_I}(\mathrm{v}^2) = \{\mathrm{t}_1(I), \mathrm{t}_2(I)\}$. Suppose that the CROPN is at the marking shown by Fig. 8, where only places $\mathrm{p}_1(\mathrm{I})$ and $\mathrm{p}_2(\mathrm{I})$ are empty, and the other places in the subset are full with the token in $\mathrm{p}_1(O)$ representing job type 1, and the token in $\mathrm{p}_3(O)$ representing job type 2. In this marking, transitions $\mathrm{t}_1(\mathrm{I})$ and $\mathrm{t}_2(\mathrm{I})$ are resource and process enabled, but they are inhibited by the control. Indeed, since there is only one nonshared free space available in v_1 and v_2, the firing of $\mathrm{t}_1(\mathrm{I})$ ($\mathrm{t}_2(\mathrm{I})$, respectively) leads to a deadlock.

We remark that the benefits of this model with respect to the S^3PR nets and the PPN are its simplicity and its capacity for modeling each resource with finite capacity by using only one place. Indeed, it results in an efficient tool to model the dynamic of AMS and to describe the interaction between jobs and resources. Since, in scheduling an AMS, two things should be avoided, i.e., deadlock and starvation, this control law provides a good opportunity to prevent starvation or to improve resource utilization while avoiding deadlocks [64]. By using a conservative policy for deadlock avoidance, the number of parts allowed in the system leads the AMS to starvation and low resource utilization. On the other hand, a maximally permissive policy can introduce blocking. In [64], a novel approach is taken and a slightly more restrictive policy than the maximal one is proposed. Such a policy gives clear advantages over the maximally permissive policy in the rule-based scheduling environment.

VI. Comparison and Benchmark Studies

Tables II and III summarize the main approaches to deal with deadlock in AMS using digraphs and automata (Table II) or PN (Table III). The first column of each table lists the references, and the second one denotes the model used to describe a system and the interactions between jobs and resources. Moreover, the third column exhibits some key words to show the used approach to characterize deadlock, and the following three show whether a detection/recovery, avoidance, or prevention strategy is defined. Finally, last three columns exhibit the RAS type considered in the model, the complexity of the proposed solution algorithm, and if it allows flexible routings, respectively.

Comparing different approaches, digraphs appear to be very simple and intuitive instruments that are able to fully characterize allocations and deallocations of resources. This tool allows a researcher to characterize deadlock conditions and to propose efficient deadlock detection/recovery and deadlock-avoidance strategies. However, digraphs cannot model the complete system dynamics. Hence, it is necessary to use DEDS models (automata or PN) to describe the AMS behavior. For this reason, some efficient deadlock avoidance policies derived from digraphs are put into the PN framework [15].

Based on automata, efficient deadlock-avoidance strategies of polynomial complexity are defined [55], [56]. Since these policies are expressed by linear inequalities, they are easy to apply and suitable for supervisory control implementation. On the other hand, classical models in the Ramadge–Wonham framework provide some techniques based on the analysis of the reachability graph. For their exponential complexity, these techniques are not suited to real-time implementation in large AMSs.

Finally, PNs result in a complete and powerful instrument that models the system dynamics and allows the introduction of new deadlock resolution techniques. Moreover, PNs provide a suitable framework where policies obtained by digraphs and automata can be implemented. However, a drawback of PN approaches is that the method utilized to characterize and resolve deadlock is related to the PN describing the system. For example, the presence of an unmarked siphon is a necessary and sufficient condition for deadlock only if PNs modeling a system fall into the class of S^3PR nets.

Referring to the specific application of the various schemes, a crucial problem is what policy must be applied to a given

TABLE II
SUMMARY OF DIGRAPH/AUTOMATON-BASED DEADLOCK CONTROL METHODS, WHERE J = NUMBER OF JOBS IN PROCESS, W = NUMBER OF AVAILABLE WORKING PROCEDURES, L = MAXIMAL WORKING PROCEDURE LENGTH, L_n = NUMBER OF PROCESSING ALTERNATIVES FOR EVERY STAGE OF EVERY WORKING PROCEDURE, L_t = CUMULATIVE NUMBER OF OPERATIONS OF ALL THE WORKING PROCEDURES, R = NUMBER OF RESOURCES IN THE SYSTEM, E = NUMBER OF EDGES IN THE DEFINED DIGRAPH, N = NUMBER OF PROCESSING ALTERNATIVES FOR EVERY STAGE OF EVERY PART TYPE

References	System model	Approach to characterize deadlock	Detect ion	Preve ntion	Ave- dance	Type of RAS	Complexity	routing
[55]	Automata	State safety and reachability graph			X	SU-RAS	$O(R\times W\times L^2)$	static
[38]	Automata	State safety and reachability graph			X	SU-RAS	$O(L_t\times M^2)$	static
[56]	Automata	State safety and reachability graph			X	C-RAS ST-RAS	$O(R\times W\times L^2)$ $O(L_t\times M^2)$	static
[67]	Automata in Ramadge-Wonham framework	State safety and reachability graph			X	SU-RAS	Exponential	static
[51]	Extended Moore Automata	Fully reachable automaton			X	SU-RAS	Polynomial	static
[61, 62]	Digraphs	Cycles	X		X	SU-RAS	O(min (J, R))	static
[5, 32]	Digraphs	Cycles	X		X	SU-RAS with unit capacity	Polynomial	static
[11,14, 15, 16, 17]	Digraphs	Cycles and strong components	X		X	SU-RAS	$O(J\times E)$	static
[39]	Automata and digraphs	Digraphs and knots			X	SU-RAS	$O(L_n\times R^2)$	Flexible

TABLE III
SUMMARY OF PN-BASED DEADLOCK CONTROL METHODS

References	System model	Approach to characterize deadlock	De- tection	Pre- vention	Avoi- dance	Conside- red RAS	Comple- xity	routing
[60]	PN	Reachability graph		X	X	SU-RAS	Expo- nential	static
[3, 57, 58]	PPN (Production Petri nets)	liveness			X	SU-RAS	Polyno- mial	static
[27]	CPPN (Contr. PPN)	liveness			X	SU-RAS	Polyno- mial	static
[65]	PPN (Production Petri nets)	liveness			X	SU-RAS	Expo- nential	static
[9]	PN S^3PR	Liveness and siphons		X		SU-RAS	Expo- nential	Flexible
[1]	PN S^4R	Liveness and siphons		X	X	CD- RAS	Expo- nential	Flexible
[6, 28]	PN	Liveness, Siphons and integer programming	X	X		SU-C RAS	Polyno- mial	Static
[41]	PN	Siphons and digraphs			X	SU-RAS	Polyno- mial	static
[15]	S^3PR	Siphons and digraphs			X	SU-RAS	Polyno- mial	Flexible
[46, 47]	S^3PGR^2 net	Liveness and siphons			X	CD-RAS	Polyno- mial	Flexible
[63, 64]	Resource Oriented PN	Liveness and circuits			X	SU-RAS	O(R)	Flexible

AMS to obtain good performance indices. Indeed, the methodologies to solve deadlock affect AMS performance indices. The problem of selecting an appropriate deadlock resolution strategy for a given RAS configuration has been given very limited consideration in literature. On the other hand, it is hard to evaluate *a priori* which technique is the most suitable for global performance improvement. The first selection must be based on the type of the RAS under study. While few policies can

be applied to CD-RAS and T-RAS, a large number of strategies can manage SU-RAS. If the SU-RAS enjoys the conditions for which the maximal permissive policy to avoid deadlock has polynomial complexity, then it must be applied. If the layout of the system is complex and impending part flow deadlock can occur, then the choice can be difficult and can be based on the computational complexity of the various schemes. In addition, to evaluate the impacts of the strategies on the system performance, simulation results can be analyzed. In [62], avoidance and detection/recovery strategies are compared on the basis of performance indices evaluated through simulated experimentation on some randomly generated system configuration. Several deadlock control methods are compared for a particular distributed system using stochastic Petri nets in [71].

Moreover, Reveliotis [54] undertakes an analytical investigation of the problem of selecting optimal deadlock resolution strategy for buffer space allocation in a simple AMS. The author develops an analytical formulation for selecting the optimal deadlock resolution strategy that maximizes the system throughput and identifies the conditions under which each of the two strategies, i.e., prevention/avoidance and detection/recovery, is the optimal selection for the considered system.

In addition, the avoidance algorithms can be distinguished based on the effects of the constraints they impose on the freedom in resource allocation. Referring to this algorithm characterization, Fanti *et al.*. [12], [13] compare the performances of deadlock avoidance policies given in [3] and [14]. The analysis is carried out from two different points of view. The first is theoretical and aims at evaluating how heavy are the restrictions each policy imposes on the resource allocation. The second point of view of the analysis investigates the magnitude of the penalty that each policy imposes on the system performance indices. This subject is approached by simulation studies involving some significant production systems presented in literature concerning deadlock in AMS, and the throughput of these systems have been compared corresponding to different operating conditions. The simulation results confirm the theoretical analysis showing that the reduced flexibility in resource allocation changes the throughput for the worse.

Ferrarini *et al.* [19] evaluate and compare the performance of deadlock-avoidance control policies by means of a simulation study. Some indices are proposed to assess the structured properties of AMS with respect to deadlock occurrence and their performance. Different deadlock-avoidance control algorithms are considered among the most common in literature, e.g., one in [3], and the Banker's algorithm. They show how it is possible to design appropriate control schemes tailored to specific purposes through the definition of suitable indices and reference models and of suitable analysis of simulation results.

VII. Conclusions

This paper presents a tutorial survey of deadlock control methods suitable for automated manufacturing systems (AMS) based on digraphs, automata, and Petri net (PN) approaches. The different models and deadlock solution techniques are presented in their *historical* evolution so that the reader can follow the development of the resolution methods over the last ten years. Indeed, starting from the analysis of deadlock in operating systems, the solution methods have been specialized for AMS and for resource acquisition structures, which are increasingly complex. The development of solution strategies shows that the tractable avoidance techniques are the most promising policies to avoid deadlock while achieving the desired system performance. Hence, research studies have focused especially on these strategies.

Among many approaches, digraphs are a powerful and intuitive means to model interactions between jobs and resources. They allow us an easy deadlock characterization and the synthesis of some flexible and permissive avoidance policies. On the other hand, automata approaches provide scalable and correct deadlock avoidance policies. Finally, as derived from the large research results, PNs offer a complete and formal model able to describe fully the AMS dynamics and to introduce deadlock avoidance and prevention policies. Moreover, using particular PN models (e.g., S^3PR and S^3PGR^2 nets), the siphon characterization of deadlock allows us to obtain PN-based supervisory control theory to avoid deadlock in several types of resource allocation systems (RAS) modeling complex AMS as well. To sum up, approaching a deadlock problem by means of different tools, the past research efforts led to a deep understanding of this phenomenon in AMS and to a satisfactory number of available solving techniques.

Future research should continue to investigate efficient and high-performance deadlock-avoidance policies fit to complex RAS such as conjunctive/disjunctive RAS. Different strategies and deadlock-avoidance policies to improve performances of real AMS should be compared and recommended to industrial use. Finally, deadlock and blocking problems must be considered in systems with unreliable resources. In particular, research should analyze the robustness of relevant deadlock resolution strategies with respect to unreliable resource failures.

Appendix

A. Basic Digraph Definitions and Properties

A digraph is denoted by $\mathrm{D} = (N, E)$, where N is the set of vertices, and E is the set of edges [24]. The *outdegree of* a vertex $r \in N$ is the number of edges from E having r as their starting node. Analogously, the *indegree of* r is the number of edges having r as their ending node. A *path* is a subdigraph of D composed by an alternating sequence of distinct vertices and arcs. All the vertices (but the first and terminal ones) have unit outdegree and indegree in the path. Moreover, the first vertex in the path has zero indegree and unit outdegree, whereas the ending vertex has unit indegree and zero outdegree.

If D contains a path from $\mathrm{r_i}$ to $\mathrm{r_m}$, $\mathrm{r_m}$ is said to be *reachable* from $\mathrm{r_i}$. Moreover, if $\mathrm{r_m(r_i)}$ is reachable from $\mathrm{r_i(r_m)}$, $\mathrm{r_i}$ and $\mathrm{r_m}$ are said to be *mutually reachable*.

A *cycle* of D is a nontrivial path in which all vertices are distinct, except for the first and last ones.

A subdigraph $\mu = (N\mu, E\mu)$ of D is *strong* if every two vertices from $N\mu$ are mutually reachable in μ. Finally, a *strong component* of D is a maximal strong subdigraph, i.e., no larger strong subdigraph (with more nodes and more edges) contains

it. This implies that each strong subdigraph of D is contained in exactly one of its strong components.

B. Ordinary PNs

An ordinary (marked) PN is a bipartite digraph $\mathrm{PN} = (P, T, F, \mathrm{M})$, where P is a set of places, and T is a set of transitions [44]. The set of arcs $F \subset (P \times T) \cup (T \times P)$ is the flow relation. Given a PN and a node $x \in P \cup T$, the set $\bullet \mathrm{x} = \{\mathrm{y} \in P \cup T : (\mathrm{y}, \mathrm{x}) \in F\}$ is the preset of x, and $\mathrm{x}\bullet = \{\mathrm{y} \in P \cup T : (\mathrm{x}, \mathrm{y}) \in F\}$ is the post-set of x. Similarly, if $S \subseteq P \cup T$, ${}^\bullet S = \cup_{\mathrm{x} \in S} \mathrm{X}^\bullet$, and $S^\bullet = \cup_{\mathrm{x} \in S} \mathrm{X}^\bullet$. The set of places S is a siphon iff ${}^\bullet S \subseteq S^\bullet$, and it is a trap iff $S^\bullet \subseteq^\bullet S$

The state of a PN is given by its marking $\mathrm{M} : P \to \bar{\mathrm{N}}$, where $\bar{\mathrm{N}}$ is the set of non-negative integers. M is described by a $|P|$-vector, and $\mathrm{M}(\mathrm{p})$ represents the number of tokens pictured by dots in place $\mathrm{p} \in P$.

A PN structure can be described by two matrices $\mathbf{O}$ and $\mathbf{I}$ of non-negative integers of dimension $|P| \times |T|$. The typical entries of $\mathbf{O}$ and $\mathbf{I}$ for an ordinary PN are given by

$$\mathbf{O}(\mathrm{p}, \mathrm{t}) = 1, \text{ if } (\mathrm{t}, \mathrm{p}) \in F, \text{ else } \mathbf{O}(\mathrm{p}, \mathrm{t}) = 0$$
$$\mathbf{I}(\mathrm{p}, \mathrm{t}) = 1, \text{ if } (\mathrm{p}, \mathrm{t}) \in F, \text{ else } \mathbf{I}(\mathrm{p}, \mathrm{t}) = 0.$$

Matrix $\mathbf{O}$-$\mathbf{I}$ is called the incidence matrix. A transition $\mathrm{t} \in T$ is enabled at a marking M iff $\forall \mathrm{p} \in \bullet \mathrm{t}$, $\mathrm{m}(\mathrm{p}) > 0$. It is denoted as $\mathrm{M}[\mathrm{t}>$. Firing an enabled transition t yields a new marking M', which is denoted as $\mathrm{M}[\mathrm{t} > \mathrm{M}'$, where $\forall \mathrm{p} \in P$, $\mathrm{M}'(\mathrm{p}) = \mathrm{M}(\mathrm{p}) - \mathbf{I}(\mathrm{p}, \mathrm{t}) + \mathbf{O}(\mathrm{p}, \mathrm{t})$. M' is reachable from M iff there exists a firing sequence $\sigma = \mathrm{t}_1 \mathrm{t}_2 \ldots \mathrm{t}_\mathrm{n}$ such that $\mathrm{M}[\mathrm{t}_1 > \mathrm{M}_1[\mathrm{t}_2 > \mathrm{M}_2 \ldots \mathrm{M}_{\mathrm{n}-1}[\mathrm{t}_\mathrm{n} > \mathrm{M}'$. It is denoted as $\mathrm{M}[\sigma \mathrm{M}'$. The set of markings reachable from M in PN is denoted $\mathrm{R}(\mathrm{M})$.

Given a marked PN $\mathrm{PN} = (P, T, F, \mathrm{M}_0)$, a transition $\mathrm{t} \in T$ is live iff for each $\mathrm{M} \in \mathrm{R}(\mathrm{M}_0)$, there exists $\mathrm{M}' \in \mathrm{R}(\mathrm{M})$ s.t. $\mathrm{M}'[\mathrm{t}>$, and $\mathrm{t} \in T$ is dead at $\mathrm{M} \in \mathrm{R}(\mathrm{M}_0)$ iff there exists no $\mathrm{M}' \in \mathrm{R}(\mathrm{M})$ s.t. $\mathrm{M}[\mathrm{t}>$.

A marking $\mathrm{M} \in \mathrm{R}(\mathrm{M}_0)$ is a dead marking or a total deadlock iff for each $\mathrm{t} \in T$, t is dead.

A marked PN is weakly live or deadlock-free iff for each $\mathrm{M} \in \mathrm{R}(\mathrm{M}_0)$, there exists $\mathrm{t} \in T$ s.t. $\mathrm{M}[\mathrm{t}>$, and it is live iff for each $\mathrm{t} \in T$, t is live.

REFERENCES

[1] I. B. Abdallah and A. Elmaraghy, "Deadlock prevention and avoidance in FMS: A Petri net based approach," *Int. J. Adv. Manuf. Technol.*, vol. 14, no. 10, pp. 704–715, 1998.

[2] T. Araki, Y. Sugiyama, and Kasami, "Complexity of the deadlock avoidance problem," in *Proc. 2nd IBM Symp. Mathematical Foundations Comput. Sci.*, Tokyo, Japan, 1977, pp. 229–257.

[3] Z. A. Banaszak and B. H. Krogh, "Deadlock avoidance in flexible manufacturing systems with concurrently competing process flows," *IEEE Trans. Robotics Automat.*, vol. 8, pp. 724–734, Dec. 1990.

[4] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Boston, MA: Kluwer, 1999.

[5] H. Cho, T. K. Kumaran, and R. A. Wysk, "Graph-Theoretic deadlock detection and resolution for flexible manufacturing systems," *IEEE Trans. Robotics Automat.*, vol. 11, pp. 413–421, June 1995.

[6] F. Chu and X. Xie, "Deadlock analysis of Petri nets using siphons and mathematical programming," *IEEE Trans. Robot. Automat.*, vol. 13, pp. 793–804, Dec. 1997.

[7] E. G. Coffman, M. J. Elphick, and A. Shoshani, "System deadlocks," *ACM Comput. Surv.*, vol. 3, pp. 67–78, 1971.

[8] A. A. Desrochers and R. Y. Al-Jaar, *Applications of Petri Nets in Manufacturing Systems*. New York: IEEE, 1995.

[9] J. Ezpeleta, J. M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing system," *IEEE Trans. Robotics Automat.*, vol. 11, pp. 173–184, Apr. 1995.

[10] J. Ezpeleta and L. Recalde, "A deadlock avoidance approach for non-sequential resource allocation systems," in *Proc. IEEE Int Conf. Syst., Man, Cybern.*, Oct. 2002.

[11] M. P. Fanti, G. Maione, and B. Turchiano, "Digraph-Theoretic Approach for Deadlock Detection and Recovery in Flexible Production Systems," *Stud. Inform. Contr.*, vol. 5, no. 4, pp. 373–383, 1996.

[12] M. P. Fanti, B. Maione, S. Mascolo, and B. Turchiano, "Performance of deadlock avoidance algorithms in flexible manufacturing systems," *J. Manufact. Syst.*, vol. 15, no. 3, pp. 164–178, 1996.

[13] ——, "Low-Cost deadlock avoidance policies for flexible production systems," *Int. J. Model. Simul.*, vol. 17, no. 4, pp. 310–316, 1997.

[14] ——, "Event-based feedback control for deadlock avoidance in flexible production systems," *IEEE Trans. Robotics Automat.*, vol. 13, pp. 347–363, June 1997.

[15] M. P. Fanti, B. Maione, and T. Turchiano, "Comparing digraph and Petri net approaches to deadlock avoidance in fms modeling and performance analysis," *IEEE Trans. Syst., Man, Cybern. A*, vol. 30, pp. 783–798, Sept. 2000.

[16] P. Fanti, B. Maione, and B. Turchiano, "Event control for deadlock avoidance in production systems with multiple capacity resources," *Stud. Inform. Contr.*, vol. 7, no. 4, pp. 343–364, 1998.

[17] M. P. Fanti, G. Maione, and B. Turchiano, "Distributed event-control for deadlock avoidance in automated manufacturing systems," *Int. J. Prod. Res.*, vol. 39, no. 9, pp. 1993–2021, 2001.

[18] ——, "Design of supervisors avoiding deadlock in flexible assembly systems," *Int. J. Flexible Manufact. Syst.*, vol. 14, pp. 157–175, 2002.

[19] L. Ferrarini, L. Piroddi, and S. Allegri, "A comparative performance analysis of deadlock avoidance control algorithms for FMS," *J. Intell. Manufact.*, vol. 10, no. 6, 1999.

[20] N.Z. Gebraeel and M. A. Lawley, "Deadlock detection, prevention, and avoidance for automated tool sharing systems," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 342–356, June 2001.

[21] E. M. Gold, "Deadlock prediction: Easy and difficult cases," in *SIAM J. Comput.*, vol. 7, 1978, pp. 320–336.

[22] Gurel, S. Bogdan, and F. L. Lewis, "Matrix approach to deadlock-free dispatching in multi-class finite buffer flowlines," *IEEE Trans. Automat. Contr.*, vol. 45, pp. 2086–2090, Nov. 2000.

[23] A. N. Habermann, "Prevention of system deadlocks," *Commun. ACM*, vol. 12, no. 7, pp. 373–378, July 1969.

[24] F. Harary, R. Z. Norman, and D. Cartwright, *Structural Models: An Introduction to the Theory of Directed Graphs*. New York: Wiley, 1965.

[25] Y. C. Ho, *Discrete Event Dynamical Systems*. New York: IEEE, 1997.

[26] F. Hsieh, "Reconfigurable fault tolerant deadlock avoidance controller synthesis for assembly processes," in *Proc. IEEE Int Conf. Syst., Man, Cybern.*, 2000, pp. 3045–3050.

[27] F. Hsieh and S. Chang, "Dispatching-driven deadlock avoidance controller synthesis for flexible manufacturing systems," *IEEE Trans. Robot. Automat.*, vol. 10, pp. 196–209, Apr. 1994.

[28] Y. Huang, M. Jeng, X. Xie, and S. Chung, "Deadlock prevention policy based on Petri nets and siphons," *Int. J. Prod. Res.*, vol. 39, no. 2, pp. 283–305, 2001.

[29] S. S. Isloor and T. A. Marsland, "The deadlock problem: An overview," *Computing*, pp. 58–78, Sept. 1980.

[30] K. Jensen, *Colored Petri Nets, Basic Concepts, Analysis Methods and Practical Use*. New York: Springer Verlag, 1992, vol. I, EATS Monograph and Theoretical Computer Science.

[31] C. O. Kim and S. S. Kim, "An efficient real-time deadlock-free control algorithm for automated manufacturing systems," *Int. J. Prod. Res.*, vol. 35, no. 6, pp. 1545–1560, 1997.

[32] T.K. Kumaran, W. Chang, H. Cho, and R.A. Wysk, "A structured approach to deadlock detection, avoidance and resolution in flexible manufacturing systems," *Int. J. Prod. Res.*, vol. 32, no. 10, pp. 2361–2379, 1994.

[33] M. A. Lawley and J. Mittenthal, "Order release and deadlock avoidance interactions in counter-flow system optimization," *Int. J. Prod. Res.*, vol. 37, no. 13, pp. 3043–3062, 1999.

[34] M. A. Lawley, S. Reveliotis, and P. Ferreira, "The application and evaluation of banker's algorithm for deadlock-Free buffer space allocation in flexible manufacturing systems," *Int. J. Flexible Manufact. Syst.*, vol. 10, pp. 73–100, 1998.

[35] ——, "Design guidelines for deadlock handling strategies in flexible manufacturing systems," *Int. J. Flexible Manufact. Syst.*, vol. 9, pp. 5–30, 1997.

[36] ——, "Flexible manufacturing system structural control and the neighborhood policy Part. 1. Correctness and scalability," *IIE Trans.*, vol. 29, pp. 877–887, 1997.
[37] ——, "Flexible manufacturing system structural control and the neighborhood policy, part. 2. Generalization, optimization, and efficiency," *IIE Trans.*, vol. 29, pp. 889–899, 1997.
[38] M. A. Lawley, S. A. Reveliotis, and P. M. Ferreira, "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," *IEEE Trans. Robotics Automat.*, vol. 14, pp. 796–809, Oct. 1998.
[39] M. A. Lawley, "Deadlock avoidance for production systems with flexible routing," *IEEE Trans. Robotics Automat.*, vol. 15, pp. 497–509, June 1999.
[40] ——, "Integrating flexible routing and algebraic deadlock avoidance policies in automated manufacturing systems," *Int. J. Prod. Res.*, vol. 38, no. 13, pp. 2931–2950, 2000.
[41] M. A. Lawley and W. Sulistyono, "Robust supervisory control policies for manufacturing systems with unreliable resources," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 346–359, June 2002.
[42] F. L. Lewis, A. Gürel, S. Bogdan, A. Doganalp, and O. C. Pastravanu, "Analysis of deadlock and circular waits using matrix model for flexible manufacturing systems," *Automatica*, vol. 34, no. 9, pp. 1083–1100, 1998.
[43] Y Li and W. M. Wonham, "Deadlock issues in supervisory control of discrete- event systems," in *Proc. 22nd Annu. Conf. Inform. Sci. Syst.*, Princeton, NJ, Mar. 1988, pp. 57–63.
[44] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, pp. 541–580, Apr. 1989.
[45] Y. Narahari and N. Viswanadham, "A Petri net approach to the modeling and analysis of flexible manufacturing systems," *Ann. Oper. Res.*, vol. 3, no. 6, pp. 449–472, 1985.
[46] J. Park and S. A. Reveliotis, "Algebraic synthesis of efficient deadlock avoidance policy for flexible manufacturing systems," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 190–195, Apr. 2000.
[47] ——, "Deadlock avoidance in sequential resource allocation systems with multiple resource acquisitions and flexible routings," *IEEE Trans. Automat. Contr.*, vol. 46, pp. 1572–1583, Dec. 2001.
[48] ——, "Liveness-Enforcing supervision of resource allocation systems with uncontrollable behavior and forbidden states," *IEEE Trans. Robot. Automat.*, vol. 18, pp. 234–239, Apr. 2002.
[49] S. E. Ramaswamy and S. B. Joshi, "Deadlock-free schedules for automated manufacturing workstations," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 391–400, June 1996.
[50] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *SIAM J. Contr. Optim.*, vol. 25, no. 1, pp. 206–230, 1987.
[51] A. Ramirez-Serrano and B. Benhabib, "Supervisory control of multiworkcell manufacturing systems with shared resources ," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 668–683, Oct. 2000.
[52] E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms: Theory and Practice*. Englewood Cliffs, NJ: Prentice-Hall, 1997.
[53] S. A. Reveliotis, "Accommodating fms operational contingencies through routing flexibility," *IEEE Trans. Robot. Automat.*, vol. 15, pp. 328–339, Feb. 1999.
[54] ——, "An analitical investigation of the deadlock avoidance detection and recovery problem in buffer-space allocation of flexibility automated production systems," *IEEE Trans. Syst., Man, Cybern. B*, vol. 30, pp. 799–811, Oct. 2000.
[55] S. A. Reveliotis and P. M. Ferreira, "Deadlock avoidance policies for automated manufacturing cells," *IEEE Trans. Robot. Automat.*, vol. 12, pp. 845–857, Dec. 1996.
[56] S. A. Reveliotis, M. A. Lawely, and P. M. Ferreira, "Polynomial-complexity deadlock avoidance policies for sequential resource allocation systems," *IEEE Trans. Automat. Contr.*, vol. 42, pp. 1344–1357, Oct. 1997.
[57] E. Roszkowska, "Deadlock avoidance in concurrent compound pipeline processes," *Arch. Inform. Teoret. Stosowanej*, vol. 2, pp. 227–242, 1990.
[58] E. Roszkowska and J. Jentink, "Minimal restrictive deadlock avoidance in fmss," in *Proc. Eur. Contr. Conf.*, vol. 2, 1993, pp. 539–534.
[59] A. Taubin, A. Kondratyev, and M. Kinshinevsky, "Deadlock prevention using Petri nets and their unfoldings," *Int. J. Manufact. Technol.*, vol. 14, pp. 750–759, 1998.
[60] N. Viswanadham, Y. Narahari, and T. L. Johnson, "Deadlock prevention and deadlock avoidance in flexible manufacturing systems using Petri net models," *IEEE Trans. Robot. Automat.*, vol. 6, pp. 713–723, Dec. 1990.
[61] R. A. Wysk, N. S. Yang, and S. Joshi, "Detection of deadlocks in flexible manufacturing cells," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 853–859, Dec. 1991.
[62] ——, "Resolution of deadlocks in flexible manufacturing systems: Avoidance and recovery approaches," *J. Manufact. Syst.*, vol. 13, no. 2, pp. 128–138, 1994.
[63] N. Q. Wu, "Necessary and sufficient conditions for deadlock-free operation in flexible manufacturing systems using a colored Petri net model," *IEEE Trans. Syst., Man, Cybern. C*, vol. 29, pp. 192–204, May 1999.
[64] N. Q. Wu and M. C. Zhou, "Avoiding deadlock and reducing starvation and blocking in automated manufacturing systems," *IEEE Trans. Robot. Automat.*, vol. 17, pp. 658–669, Oct. 2001.
[65] K. Y. Xing, B. S. Hu, and H. X. Chen, "Deadlock avoidance policy for Petri net modeling of flexible manufacturing systems with shared resources," *IEEE Trans. Automat. Contr.*, vol. 41, pp. 289–295, Feb. 1996.
[66] H. H. Xiong and M. C. Zhou, "A Petri net method for deadlock-free scheduling of flexible manufacturing systems," *Int. J. Intell. Contr. Syst.*, vol. 3, no. 3, pp. 277–295, Sept. 1999.
[67] A. Yalcin and T. O. Boucher, "Deadlock avoidance in flexible manufacturing systems using finite automata," *IEEE Trans. Robot. Automat.*, vol. 16, pp. 424–429, Aug. 2000.
[68] D.-S. Yim, J.-I. Kim, and H.-S. Woo, "Avoidance of deadlocks in flexible manufacturing systems using a capacity-designated directed graph," *Int. J. Prod. Res.*, vol. 35, no. 9, pp. 2459–2475, 1997.
[69] M. C. Zhou and F. DiCesare, "Parallel and sequential mutual exclusions for Petri net modeling of manufacturing systems with shared resources," *IEEE Trans. Robot. Automat.*, vol. 7, pp. 515–527, Aug. 1991.
[70] M. C. Zhou, F. DiCesare, and A. A. Desrochers, "A hybrid methodology for synthesis of Petri nets for manufacturing systems," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 350–361, June 1992.
[71] M. C. Zhou, "Deadlock avoidance methods for a distributed robotic system: Petri net modeling and analysis," *J. Robot. Syst.*, vol. 12, no. 3, pp. 177–187, 1995.

Maria Pia Fanti (M'94–SM'02) received the "Laurea" degree in electronic engineering from the University of Pisa, Pisa, Italy, 1983.

She joined the Department of Electrical and Electronic Engineering, Polytechnic of Bari, Bari, Italy, in 1984, where she is currently Associate Professor. She was Assistant Professor from 1990 to 1998 and Visiting Researcher at the Rensselaer Polytechnic Institute, Troy, NY, in 1999. From 1994 to the present, she developed courses on automatic control for electrical and electronic engineering. Her research interests include control and modeling of automated manufacturing systems, control and modeling of computer integrated systems (i.e., automatic guided vehicle systems, automated storage and retrieval systems, railway networks, and urban traffic signals), supply chain modeling and management, discrete event systems, Petri nets and colored Petri nets, and structural properties of linear systems. She has more than 50 publications including 24 journal articles, three book chapters, and many conference proceedings papers. She is referee of *IIE Transactions*, the *International Journal of Flexible Manufacturing Systems,* and the *International Journal of Production Research*.

Prof. Fanti has organized and chaired many technical sessions for international conferences and has served as a Member of the program committees of the following international conferences: the IEEE International Conference on Systems, Man, and Cybernetics (Orlando, FL, October 12–15, 1997), the 1998 IEEE International Conference on Systems, Man, and Cybernetics (La Jolla, CA, October 11–14, 1998), the IEEE International Conference on Emerging Technologies and Factory Automation, (Barcelona, Spain, October 18–22, 1999), and the Fifth Multi-Conference on Systemics, Cybernetics, and Informatics, (Orlando, FL, July 22–25, 2001). She was a Member of the International Program Committee of the 2003 IEEE International Conference on Systems, Man, and Cybernetics (Washington, DC, October 5–8, 2003). She serves as a referee for the following journals: the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, and the IEEE TRANSACTIONS ON AUTOMATIC CONTROL. She a member of the New York Academy of Sciences.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree from Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree from Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1990.

He joined New Jersey Institute of Technology (NJIT), Newark, in 1990, where he is currently Professor of electrical and computer engineering and the Director of Discrete-Event Systems Laboratory. His research interests are in computer-integrated systems, Petri nets, semiconductor manufacturing, multilifecycle engineering, *ad hoc* networks, and system security. He has over 200 publications including four books, over 50 journal papers, and ten book chapters. He coauthored with F. DiCesare, *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems* (Norwell, MA: Kluwer, 1993), edited *Petri Nets in Flexible and Agile Automation* (Norwell, MA: Kluwer, 1995), and coauthored with K. Venkatesh, *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach* (Singapore: World Scientific, 1998). He has been an editor of *International Journal of Intelligent Control and Systems* since 1996.

Dr. Zhou has been invited to lecture in Australia, Canada, China, France, Germany, Hong Kong, Italy, Japan, Korea, Mexico, Taiwan, and the United States. He served as Associate Editor of IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION from 1997 to 2000 and is currently Associate Editor of the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS and the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He organized and chaired over 60 technical sessions and served on program committees for many conferences. He was Program Chair of the 1998 and Co-Chair of the 2001 IEEE International Conferences on Systems, Man, and Cybernetics (SMC) and the 1997 IEEE International Conference on Emerging Technologies and Factory Automation and Guest Editor for the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS and the IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING. He was General Co-Chair of the 2003 IEEE International Conference on System, Man, and Cybernetics, Washington, DC, October 5–8, 2003. He is serving as a founding Co-General Chair of the 2004 IEEE International Conference on Networking, Sensing, and Control. He has led or participated in 26 research and education projects funded by the National Science Foundation, Department of Defense, Engineering Foundation, New Jersey Science and Technology Commission, and industry. He was the recipient of the National Science Foundation's Research Initiation Award, the CIM University-LEAD Award of the Society of Manufacturing Engineers, the Perlis Research Award from NJIT, the Humboldt Research Award of the U.S. Senior Scientists, the Leadership Award and Academic Achievement Award of the Chinese Association for Science and Technology-USA, and the Asian American Achievement Award of the Asian American Heritage Council of New Jersey. He is Chair (founding) of the Discrete Event Systems Technical Committee of the IEEE Systems, Man, and Cybernetics Society and Co-Chair (founding) of the Semiconductor Factory Automation Technical Committee of the IEEE Robotics and Automation Society. He is a life member of the Chinese Association for Science and Technology-USA and served as its President in 1999.