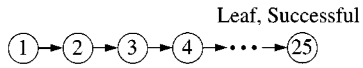

Fig. 9. The searching tree after inputting the same task command for the second time (with the initial state the same as in the first time).

The desired node picked up by the explanation-based learning engine is node 28 and the generated searching template is *superior_order([next_to(_, _), on(_, _)])*.

When the same task command was inputted for the second time (with the initial state the same as in the first time), the planned action sequence is the same as in the first time, yet the corresponding searching tree alters, as shown in Fig. 9, where nodes 1–4 and 25 simply correspond to nodes 1, 2, 16, 28, and 49 in Fig. 8, respectively. This time, however, the explanation-based learning engine fails to generate any other new searching template, because no desired node on the corresponding searching tree can be found, though there does be a successful leaf node.

It can be seen that the number of the nodes searched in the second time decreases almost a half than in the first time and no backtracking happens. The searching efficiency is greatly increased. The effect of explanation-based learning is significant.

V. Summary

The main contributions of this paper are as follows:

1) Explanation-based learning has been accurately placed in the triangle of problem solving, i.e., with the angle of searching mechanism.
2) A problem formulation has been made for robot action planning (RAP), which gives in-depth comprehensibility of RAP, especially that of means-ends analysis searching mechanism.
3) A new learning-based method has been developed for RAP, i.e., robot action planning via explanation-based learning (RAPEL), which is aiming at computer-realized recognition and acquisition of domain-specific searching heuristics.
4) The overall scheme of RAPEL has been put forward and the principle of RAPEL has been established, and terms, notations, grammars and paradigms of Prolog language are directly employed for the purpose of strictness.
5) Configuration of node has been proposed, by which node growth can be visually illustrated.
6) Logic chart has been proposed, by which processes of synthesizing action sequences can be visually illustrated.

References

[1] R. E. Fikes and N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artif. Intell.*, vol. 2, pp. 189–208, 1971.
[2] L. M. Camarinha-Matos, "Plan generation in robotics: State of the art and perspectives," *Robotics*, vol. 3, pp. 291–328, 1987.
[3] Z. X. Cai and K. S. Fu, "Robot planning expert systems," in *Proc. 1986 IEEE Int. Conf. Robot. Automat.*, vol. 3, San Francisco, CA, Apr. 1986, pp. 1973–1978.
[4] H. Eskicioglu, "An expert-system approach to the assembly planning of roller chains," *Eng. Applicat. Artif. Intell.*, vol. 3, pp. 306–312, 1990.
[5] R. E. Fikes, P. E. Hart, and N. J. Nilsson, "Learning and executing generalized robot plans," *Artif. Intell.*, vol. 3, pp. 251–288, 1972.
[6] R. S. Michalski, *Machine Learning—An Artificial Intelligence Approach*. New York: Tioga, 1983.
[7] S. Tangwongsan and K. S. Fu, "An application of learning to robotic planning," *Int. J. Comput. Inf. Sci.*, vol. 8, no. 4, pp. 303–333, 1979.
[8] H. M. Stellakis and K. P. Valavanis, "Fuzzy logic-based formulation of the organization of intelligent robotic systems," *J. Intell. Robot. Syst.*, vol. 4, pp. 1–24, 1991.
[9] K. P. Valanvanis and S. J. Carelo, "An efficient planning technique for robotic assemblies and intelligent robotic systems," *J. Intell. Robot. Syst.*, vol. 3, pp. 321–347, 1990.
[10] K. P. Valavanis and H. M. Stellakis, "A general organizer model for robotic assemblies and intelligent robotic systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 2, pp. 302–317, 1991.
[11] M. C. Moed and G. N. Saridis, "A Boltzmann machine for the organization of intelligent machines," *IEEE Trans. Syst., Man, Cybern.*, vol. 20, no. 5, pp. 1094–1102, 1990.
[12] K. M. Passino and P. J. Antsaklis, "A system and control theoretical perspective on artificial intelligence planning systems," *Appl. Artif. Intell.*, vol. 3, pp. 1–32, 1989.
[13] E. D. Sacerdoti, "Planning in a hierarchy of abstraction spaces," *Artif. Intell.*, vol. 5, pp. 115–135, 1974.
[14] B. Hayes-Roth and F. Hayes-Roth, "Modeling planning as an incremental, opportunistic process," in *Proc. 6th Int. Joint Conf. Artif. Intell.*, vol. 1, Tokyo, Japan, Aug. 1979, pp. 375–383.
[15] M. Stefik, "Planning with constraints (MOLGEN: Pt. 1), Planning and meta planning (MOLGEN: Pt. 2)," *Artif. Intell.*, vol. 16, p. 111–140, 141–170, 1981.
[16] S. Minton *et al.*, "Explanation-based learning: A problem solving perspective," *Artif. Intell.*, vol. 40, pp. 63–118, 1989.
[17] T. Mitchell, R. Keller, and S. Kedar-Cabled, "Explanation-based generalization: A unifying view," *Mach. Learn.*, vol. 1, pp. 47–80, 1986.
[18] H. Tian, "Several originative studies on intelligent machine systems," Ph.D. dissertation (in Chinese), East China Univ. Sci. Technol., Shanghai, China, Apr. 1992.
[19] H. Tian, Y. G. Xi, Z. J. Zhang, and W. S. Jiang, "Application of explanation-based learning to robot action planning," *Selected Scientific Papers of Shanghai Jiaotong University, An Annals*, pp. 25–34, 1995.
[20] H. Tian, "Robotic action planning with the application of explanation-based learning," in *Proc. 37th IEEE Conf. Decision Control (CDC'98)*, Tampa, FL, Dec. 16–18, 1998, pp. 1508–1513.

Computational Complexity of Determining Resource Loops in Reentrant Flow Lines

F. L. Lewis, B. G. Horne, and C. T. Abdallah

***Abstract*—This paper presents a comparison study of the computational complexity of the general job shop protocol and the more structured flow line protocol in a flexible manufacturing system. It is shown that the representative problem of finding resource invariants is $\mathcal{NP}$-complete in the case of the job shop, while in the flow line case it admits a closed form solution. The importance of correctly selecting part flow and job routing protocols in flexible manufacturing systems to reduce complexity is thereby conclusively demonstrated.**

I. Introduction

In a general flexible manufacturing system (FMS) where resources are shared, a key role in part routing, job selection, and resource assignment is played by the FMS controller. Given the same resources of

Manuscript received April 15, 1996; revised October 2, 1997, January 15, 1999, and August 27, 1999. This paper was recommended by Associate Editor C. Hsu.

F. L. Lewis is with the Automation and Robotics Research Institute, University of Texas at Arlington, Ft. Worth, TX 76118-7115 USA.

B. G. Horne is with AADM Consulting, Califon, NJ 07830 USA.

C. T. Abdallah is with the EECE Department, University of New Mexico, Albuquerque, NM 87131 USA.

Publisher Item Identifier S 1083-4427(00)01176-0.

machines, robots, fixtures, tooling, and so on, different structures result under different routing/assignment strategies by the controller. Unstructured strategies are generally classified as the so-called job shop organization, while structured protocols result in various sorts of flow lines, with or without assembly. The importance of *structure* in determining *complexity* has not been rigorously addressed in FMS.

The theory of $\mathcal{NP}$-completeness [5] potentially provides a comprehensive approach for analysis of computational complexity in FMS. This possibility has not been rigorously explored. Many traditional scheduling and sequencing problems have been found to be in $\mathcal{NP}$, thus it has been necessary to develop heuristics or approximate methods for analysis and solution. It has been shown, for instance that, even for the flow line with two processors, scheduling while minimizing the maximum flow time is $\mathcal{NP}$-complete for both nonpreemptive and preemptive schedules [6]. For the general job shop protocol the situation is even worse (see, for example, [5, p. 242]). Branch and bound algorithms are generally used in this case. For the flow line, the lot-sizing problem is polynomial, while for the flow line with assembly it is exponential. The complexity of many problems, including the determination of the PN p-invariants, has not yet been determined. There is currently no comprehensive theory that provides a categorization of the complexity of analysis problems for the flow line, assembly line, and job shop. There is no formal theory describing how to impose *structured flow and command protocols* on an FMS to simplify its complexity.

Petri nets (PN) [13] have been extensively used in the analysis of manufacturing systems, with quite variable results. Though, *ad hoc* applications abound, PN have a body of theoretical results on liveness, boundedness, reachability, and so on that make them very useful in studies of FMS when seriously applied. Applications of PN are found in [2], [4], and [20]. PN approaches to the design of FMS sequencing/dispatching controllers are found in [7], [8], [14], and [19].

The PN incidence matrix W can be used to study structural properties of FMS, including determination of the siphons [1] and deadlock avoidance [11]. However, matrix applications in PN had not been fully exploited. A complete matrix model for FMS is given in [17]. In many papers [2], [7], [20], the problem of finding a binary basis for the nullspace of W is important, for such a basis defines a special class of siphons known as the *p-invariants or resource loops*. The p-invariants contain important structural information about an FMS, and may be used for conflict resolution in the dispatching of shared resources in such a fashion as to avoid deadlock [11]. In this paper we show that it is possible by judicious means to reveal a special structure of the PN incidence matrix in a very general class of reentrant flow lines (RFL) that can include assembly operations. This class includes the multipart flow lines discussed for instance in [9] and [12]. To reveal the importance of *structure in the study of complexity* for RFL, we select the representative problem of determining the p-invariants. It is shown that for unstructured job shop protocols this problem is $\mathcal{NP}$-complete, while for a general class of reentrant flow line protocols it is polynomial. For this class, an explicit matrix formula is given to compute the p-invariants. The importance of selecting suitable controller sequencing protocols to reduce complexity in FMS is thereby shown.

II. Complexity Theory Overview

Until recently, it was felt that decidable problems are practically solved and thus not very interesting. The introduction of computational complexity theory has since changed this misconception. Computational complexity theory is often used to establish the tractability or intractability of computational problems, and is concerned with the determination of the intrinsic computational difficulty of these problems [5].

The complexity class $\mathcal{P}$ consists of all decision problems that can be decided in polynomial-time. In practice, such problems can be feasibly implemented on a real computer. The class $\mathcal{EXP}$ consists of those that can be decided in exponential-time. Such problems can only be run on a real computer if they are of very small dimension. The complexity class $\mathcal{NP}$ lies in between, consisting of all decision problems that can be decided algorithmically in *nondeterministic* polynomial-time. An algorithm is nondeterministic if it is able to choose or guess a sequence of choices that will lead to a solution, without having to systematically explore all possibilities. This model of computation is not realizable, but it is of theoretical importance. In practice, problems in $\mathcal{NP}$ are those for which a candidate solution can be verified to be a valid solution in polynomial-time, but the best known algorithms to find such a solution run in exponential time.

Many practical problems belong to $\mathcal{NP}$ and it is as of yet unknown whether $\mathcal{P} = \mathcal{NP}$. In other words, these two complexity classes form an important boundary between the tractable and intractable problems. A problem is said to be $\mathcal{NP}$-hard if it is as hard as any problem in $\mathcal{NP}$. Thus, if $\mathcal{P} \neq \mathcal{NP}$, the $\mathcal{NP}$-hard problems can only admit deterministic solutions that take an unreasonable (i.e., exponential) amount of time, and they require (unattainable) nondeterminism in order to achieve reasonable (i.e., polynomial) running times.

The central idea used to demonstrate $\mathcal{NP}$-hardness evolves around the $\mathcal{NP}$-complete problems. A problem is said to be $\mathcal{NP}$-complete if every decision problem in $\mathcal{NP}$ is polynomial-time reducible to it. This means that the $\mathcal{NP}$-complete problems are as hard as any decision problem in $\mathcal{NP}$. Given two decision problems Π_1 and Π_2, Π_1 is said to be polynomial-time reducible to Π_2 (written as $\Pi_1 \leq_p \Pi_2$), if there exists a polynomial time algorithm R which transforms every input x for Π_1 into an equivalent input $R(x)$ for Π_2. By equivalent we mean that the answer produced by Π_2 on input $R(x)$ is always the same as the answer Π_1 produces on input x. Thus, any algorithm which solves Π_2 in polynomial time can be used to solve Π_1 on input x in polynomial time by simply computing $R(x)$, and then running Π_2. In order to show that a particular decision problem Π_2 is $\mathcal{NP}$-complete, one starts with a problem Π_1 which is known to be $\mathcal{NP}$-complete, and shows that $\Pi_1 \leq_p \Pi_2$. This proves that Π_2 is $\mathcal{NP}$-hard. To complete the proof that Π_2 is $\mathcal{NP}$-complete, it must be demonstrated that a candidate solution can be verified in polynomial time.

In this paper, we use the One-In-3Sat problem which is known to be $\mathcal{NP}$-complete [5] in order to show that solving a certain problem for the general job shop is $\mathcal{NP}$-complete. We then use the special structure of the reentrant flow line problem to show that the same problem can be efficiently obtained for the flow line. This highlights the importance of *structure* in flexible manufacturing systems. The One-In-3Sat problem is as follows:

One-In-3Sat:

Instance: Given a set U of variables, a collection C of clauses over U such that each $c \in C$ has $|c| = 3$.

Question: Is there a truth assignment for U such that each clause in C has exactly one true literal?

Example 1: Let $U = \{a, b, c, d\}$ and $C = \{a\bar{b}c, \bar{a}bd, \bar{b}c\bar{d}\}$. Then a solution is $a = b$ = true and $c = d$ = false. □

III. Structure and Modeling of Reentrant Flow Lines (RFL)

In this section we discuss flexible manufacturing systems with several sorts of structures, including the reentrant flow line (RFL), the assembly line, and the job shop. The importance of *structure and protocol* in flexible manufacturing systems is highlighted. Some Petri net modeling techniques are introduced.

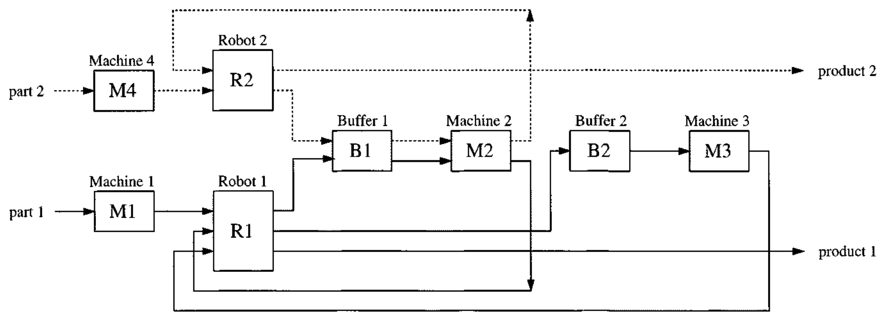

Fig. 1. RFL with four machines and two parts.

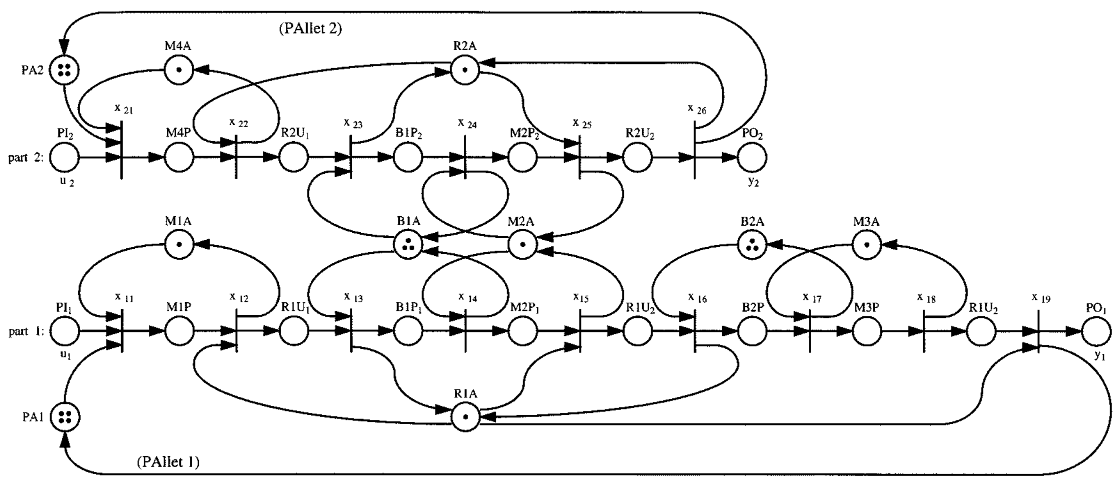

Fig. 2. PN representation of the reentrant flow line.

A. Manufacturing System Structures

The physical portion of a flexible manufacturing system (FMS) is comprised of its *resources:* the set of machines or work stations, the automated material handling system, and the distributed buffers. Given *the same* resource facilities in the FMS, different sequencing algorithms by the FMS controller produce *different flow/protocol structures,* including the reentrant flow line, the assembly line, and the general job shop protocol. A major issue is that *the structure imposed by the controller should avoid or reduce $\mathcal{NP}$ complexity problems*.

Formally, a manufacturing facility is a set $\mathcal{R} = \{r_i\}$ of resources (e.g., machines, tools, fixtures, robots, transport devices, etc.) each of which has a distinct function. Each r_i can denote a *pool* of more than one machine that performs the same function. The resources operate on parts; parts of the jth type are denoted π_j. A *job sequence for part type* π_j is a sequence of P_j jobs $\mathcal{J}_j = \{J_{1j}, J_{2j}, \cdots, J_{P_j j}\}$ required to produce a finished product. The sequence of jobs may be determined from a task decomposition, bill of materials, assembly tree, or precedence matrix [16]. If each job is performed on a single part and delivers a single part there is said to be *no assembly*.

If a single resource is needed for each job, for instance, this corresponds to a pairing (J_{kj}, r_i) of the kth job for part π_j with a resource r_i. The ordering of the jobs for a given part type can be either fixed or variable. Likewise, the resources assigned to each job can be either fixed or variable.

In the general *job shop* the sequence of jobs is not fixed, or the assignment of resources to the jobs is not fixed. The effect is that *part routing decisions* must be made during processing. In the *flow line* the sequence of jobs for each part type is fixed and the assignment of resources to the jobs is fixed. The result is that each part type visits the resources in the same sequence, though different part types may have different sequences. The sequence in which part type π_j visits the resources in a flow line will be called the jth *part path*. Once the resources have been assigned to jobs, this resource sequence is defined by the job sequence $\mathcal{J}_j$, which is therefore used to denote the jth part path.

A flow line is said to *reentrant* if any part type revisits the same resource more than once in its job sequence [9], [12]. This occurs if the same resource is assigned to different jobs in the part's sequence. A sample reentrant flow line is given in Fig. 1. In this figure, $R1$ and $R2$ could be transport robots, for instance, that move the parts between certain jobs; $B1, B2$ could be buffers; and $M1$–$M4$ could be machines. Thus, the resources may include machines, robots, buffers, transport devices, fixtures, tools, and so on.

B. Petri Net Representation of RFL

A Petri net (PN) is a bipartite (e.g., having two sorts of nodes) digraph described by $(\mathcal{P}, \mathcal{T}, I, O)$, where $\mathcal{P}$ is a set of *places*, $\mathcal{T}$ is a set of *transitions*, I is a set of (input) arcs from places to transitions, and O is a set of (output) arcs from transitions to places. In our application, the PN places represent manufacturing resources and jobs, and the transitions represent decisions or rules for resource assignment/release and starting jobs. The PN representation for the reentrant flow line in Fig. 1 is shown in Fig. 2, where the places are drawn as circles and the transitions as bars. The flow line structure is evident in the parallel *part type paths,* interconnected by *shared resource places* (e.g., $B1, M2$) that service jobs for several part types. Note that along one part path, some resources (e.g., $R1, R2$) are used more than once, so that this flow line is reentrant. Each part path in the figure has a set of *pallets* denoted by $PA1, PA2$; one pallet is needed to hold each part entering the cell. Places ending in P, all on the job paths, correspond to jobs in progress. Places ending in A correspond to the availability

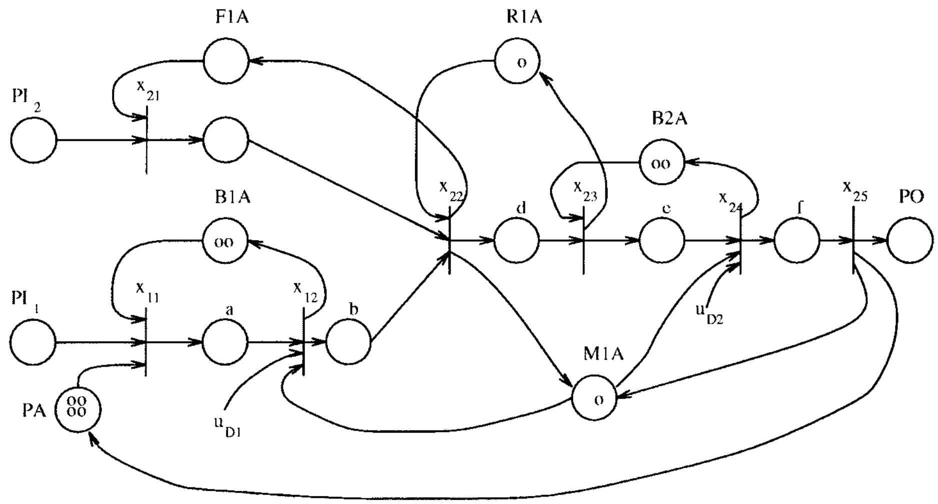

Fig. 3. Petri net of reentrant flow line with assembly.

of resources. In the reentrant flow line, note that all transitions occur along part paths, and exactly one transition feeds into its succeeding job place.

It is common in PN theory to represent the sets of arcs I and O in the PN description $(\mathcal{P}, \mathcal{T}, I, O)$ as matrices. Thus, element I_{ij} of matrix I is equal to 1 if place j is an input to transition i. Element O_{ij} of matrix O is equal to 1 if place j is an output of transition i. Otherwise the elements of I, O are set to 0. Matrix I is called the *input incidence matrix,* and O the *output incidence matrix*. Both matrices are considered as maps from $\mathcal{P}$ to $\mathcal{T}$. Then, the *PN incidence matrix* is defined as

$$W = O - I. \tag{1}$$

A column vector $\mathbf{p}$ indexed by the set of all places $\mathcal{P}$ is called the PN p-vector (place vector). The *PN marking vector* is the marking vector $m(\mathbf{p})$ defined as follows.

Definition 1—Marking and Support: Given a PN, the PN marking is the number of tokens in each place in the net. Given a place $p \in \mathcal{P}$, the marking of p, $m(p)$, is the number of tokens in p. Given a vector of places $\mathbf{p} = [p_1 \quad p_2 \quad \cdots \quad p_q]^T$, the marking $m(\mathbf{p})$ is the vector $m(\mathbf{p}) = [m(p_1) \;\; m(p_2) \;\; \cdots \;\; m(p_q)]^T$ of markings of the individual places. The support of a vector is the set of its elements having nonzero values.

It is common to simplify the notation so that $m(t)$ denotes the marking vector $m(\mathbf{p})$ at time t. Then, in terms of the PN incidence matrix, one can write the *PN marking transition equation*

$$m(t_2) = m(t_1) + W^T \tau = m(t_1) + (O - I)^T \tau \tag{2}$$

where $m(t)$ is the PN marking vector at time t, $t_1 < t_2$, and τ is a vector denoting which transitions have fired between times t_1 and t_2; element $\tau_i = n_i$ if the ith transition has fired n_i times in the interval.

Central to the study of resource allocation in RFL are the following notions.

Definition 2—p-Invariant and Resource Loop: A p-invariant is a place vector $\mathbf{p}$ having elements of zeros and ones that is in the nullspace of W, that is

$$W\mathbf{p} = 0. \tag{3}$$

The set of places corresponding to the support of $\mathbf{p}$ is known as a resource loop, also loosely called a p-invariant.

The importance of p-invariants may be understood by noting that, beginning with (2), for any p-invariant $\mathbf{p}$ one has

$$\mathbf{p}^T m(t_2) = \mathbf{p}^T m(t_1) + \mathbf{p}^T W^T \tau = \mathbf{p}^T m(t_1). \tag{4}$$

Noting that premultiplication by $\mathbf{p}^T$ simply sums up the tokens in the positions of $m(\cdot)$ corresponding to the support of $\mathbf{p}$, this is seen to be a statement that the total number of tokens in positions of $m(\cdot)$ corresponding to the support of $\mathbf{p}$ is conserved. That is the p-invariants define those loops in the PN within which the numbers of tokens are conserved. These conservative loops defined by the p-invariants are the resource loops.

The complete set of p-invariants of a PN, which defines the resource loops, gives a great deal of structural information in a RFL. They have been extensively studied in work by Desrochers [2], DiCesare *et al.* [7], Zhou *et al.* [20], and elsewhere. A common requirement in "well-defined" PN is that each job should be contained in a resource loop, i.e., the PN should be *covered* by p-invariants. They provide the basis for several FMS control techniques that involve dispatching of shared resources. In [11] it is shown that they provide the basis for deadlock avoidance algorithms. In [1] is given a complex algorithm for determining p-invariants. In Section V we shall give an explicit matrix formula for p-invariants for a large class of reentrant flow lines.

IV. Computational Complexity of Finding the p-Invariants in the Job Shop

To find the p-invariants it is necessary to solve (3), determining a basis for the nullspace of W that has only ones and zeros. In this section, we show that finding such a binary basis is an $\mathcal{NP}$-complete problem for the general job shop structure. Then, in Section V, it is shown that for the reentrant flow line, with or without assembly, an analytic solution can be given for the problem.

Theorem 1: The problem of finding a binary basis for W in the general job shop is $\mathcal{NP}$-complete.

Proof: In order to solve the general job shop problem, we need to find a basis of the nullspace of the incidence matrix W. Since W contains coefficients $w_{ij} \in \{-1, +1, 0\}$ and since a meaningful basis of its nullspace will have vectors $\mathbf{p}$ whose entries p_i also belong to $\{0, +1\}$, the problem is equivalent to finding p_i such that $\sum_i w_{ij} p_i = 0;\ \forall j$. Note however, that the zero vector $p_i = 0,\ \forall i$ should be excluded. We shall then define the following problem:

Matrix Basis:

Instance: An $n \times 2n$ matrix $A \neq 0$ with entries in $\{-1, 0, 1\}$.

Question: Does there exist a vector $x \neq 0$ with entries in $\{0, 1\}$ such that $Ax = 0$?and prove that MATRIX BASIS is $\mathcal{NP}$-complete by transformation from ONE-IN-3SAT.

We begin with a proof for A of size $n \times m$ and then later show how to augment the matrix to make it of size $n \times 2n$.

Let $n = |U| + |C|$ and $m = 2|U| + 1$, where U and C are the sets of variables and clauses in the instance of ONE-IN-3SAT. The columns of A (and thus the components of the vector x) will correspond to complemented and uncomplemented assignments of the $|U|$ literals and an auxiliary variable z, i.e.,

$$x = [x_1\ \overline{x}_1\ x_2\ \overline{x}_2\ \cdots x_n\ \overline{x}_n\ z]'.$$

A valid solution vector will correspond to each component of x being equal to 0 or 1 depending on whether the corresponding literal is true or false. All nontrivial solutions will have $z = 1$.

The first $|U|$ rows of A are used to insure that the solution vector is a valid truth assignment to the literals, i.e., so that value assigned to x_i will be the logical complement of the value assigned to $\overline{x}_i$. Specifically, the first $|U|$ rows are configured as,

$$a_{i,j} = \begin{cases} 1, & j \in 2i - 1,\ 2i \\ -1, & j = 2|U| + 1 \\ 0, & \text{otherwise.} \end{cases}$$

The remaining $|C|$ rows are used to satisfy the requirement that exactly one literal in each clause is true. Specifically, denote a literal by $\tilde{x}_i$ (i.e., $\tilde{x}_i \in \{x_i, \overline{x}_i\}$), and denote the ith clause by $c_i = \tilde{x}_{p_i}\tilde{x}_{q_i}\tilde{x}_{r_i}$. Then set

$$a_{|U|+i,j} = \begin{cases} 1, & j = 2s - 1 \quad \tilde{x}_s = x_s \quad s \in \{p_i,\ q_i,\ r_i\} \\ 1, & j = 2s \quad \tilde{x}_s = \tilde{x}_s \quad s \in \{p_i,\ q_i,\ r_i\} \\ -1, & j = 2|U| + 1 \\ 0, & \text{otherwise.} \end{cases}$$

Every solution besides the trivial solution must have $z = 1$ since if $z = 0$ then the first $|U|$ rows of A will guarantee that every other entry will also be equal to zero. The same rows will guarantee that for nontrivial solutions exactly one of x_i and $\overline{x}_i$ will be equal to one. The last $|C|$ rows of A will only be satisfied by nontrivial solutions such that exactly one literal of each clause is true.

The first part of the proof shows that the theorem holds for a variety of values of n and m. However, it is not directly applicable to the case where $2n = m$ since this would imply that $2(|U| + |C|) = 2|U| = 1$, or $2|C| = 1$. Since this can never be achieved by direct transformation from ONE-IN-3SAT, we modify A by adding one additional row and $2|C| + 1$ additional columns, i.e., construct the augmented matrix

$$A' = \left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right]$$

where B and C are matrices of zeros of sizes $(|U| + |C|) \times (2|C| + 1)$ and $1 \times (2|U| + 1)$, respectively, and D is a matrix of ones of size $1 \times (2|C| + 1)$. The last row insures that the last $2|C| + 1$ components of the solution vector must be equal to zero, but these variables in no way interfere with the construction above. The augmented matrix is of size $n \times 2n$ where $n = |U| + |C| + 1$.

The transformation is easily done in time linear in the size of the matrix, which is quadratic in $|U|$ and $|C|$. Therefore, we have shown that MATRIX BASIS is $\mathcal{NP}$-hard. On the other hand, one can easily verify the existence of p_i as a member of the nullspace of W which then proves that the problem is $\mathcal{NP}$-complete.

V. COMPUTATIONAL COMPLEXITY OF FINDING THE p-INVARIANTS IN THE FLOW LINE

Though finding the p-invariants in a general job shop protocol is $\mathcal{NP}$-complete, in this section a special job flow protocol is imposed that allows one to give an analytical solution to this problem, so that the complexity is polynomial. This protocol corresponds to a large class of reentrant flow lines with or without assembly, including those with multiple part types. Included particularly are all the reentrant flow lines without assembly treated in references such as [9] and [12]. The importance of structure in an FMS is thereby shown in regards to computational complexity, so that care should be taken in selecting job sequencing and routing strategies in FMS.

A. Structure of the Reentrant Flow Line

In the reentrant flow line with or without assembly, e.g., Fig. 2, denote the set of jobs for part type j as $\mathcal{J}_j$ and the set of all the jobs as $\mathcal{J} = \bigcup_j \mathcal{J}_j$. The set $\mathcal{J}_j$ will also be used to denote the jth part path. It is noted that the part input places PI and part output places PO are not included as jobs (they are not important for determining the structure of RFL). Places that occur off the part paths represent the availability of resources; denote by $\mathcal{R}$ the set of all such places. The set of PN places is given by $\mathcal{P} = \mathcal{J} \cup \mathcal{R}$, the set of resources plus the set of jobs. Note that all transitions occur along the part paths.

Partition the PN marking vector p as

$$p = \begin{bmatrix} v \\ r \end{bmatrix} \tag{5}$$

where v is the vector of places corresponding to the jobs $\mathcal{J}$ and r is the vector of places corresponding to the resources $\mathcal{R}$ [2], [17]. Then, referring to the p-invariant definition (3), the PN incidence matrix has the compatible structure

$$W = [W_v\ W_r] \equiv S^T - F = [S_v^T - F_v\ S_r^T - F_r] \tag{6}$$

where $S^T \equiv [S_v^T\ S_r^T]^T$ and $F \equiv [F_v\ F_r]$. Comparing this equation with (1) one sees that the output incidence matrix is $O = [S_v^T\ S_r^T]$ and the input incidence matrix is $I = [F_v\ F_r]$. Matrices S_v^T, S_r^T are the output incidence matrices of the jobs and resources, respectively, and F_v, F_r are the input incidence matrices of the jobs and resources, respectively.

In the RFL, matrices F_v, F_r have rows corresponding to the transitions that are inputs to the succeeding job. Matrix F_v has columns corresponding to jobs while matrix F_r has columns corresponding to resources. Therefore, Matrix F_v is the well-known Steward sequencing matrix [16], assembly tree, or the bill of materials (BOM) [3] in manufacturing; it has element $(i, j) = 1$ if job j is an immediate prerequisite for job i. Matrix F_r is the resource requirements matrix used in [10]; it has element $(i, j) = 1$ if resource j is required for job i.

An example of these constructions is provided by the reentrant flow line in Fig. 3. This flowline has an assembly operation as two part paths join to form one at transition x_4, corresponding to the assembly of parts b and c to form subassembly d. Define the job vector as $v = [abcdef]^T$, the resource vector as $r = [R1A\ F1A\ B1A\ B2A\ PA\ M1A]^T$, and the PN place vector as (5). Define the vector of transitions x as having components of x_i; $i = 1, 7$. Then, by inspection one determines the following matrices. The partitioning shown corresponds to the two part paths, a partial path with two transitions and a complete path with five transitions

$$S_v = \left[\begin{array}{cc|ccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right],$$

$$S_r = \left[\begin{array}{cc|ccccc} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \tag{7}$$

$$F_v = \left[\begin{array}{cc|cccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right],$$

$$F_r = \left[\begin{array}{cc|cccc} 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right]. \tag{8}$$

Using(6) one now has the PN incidence matrix in (9), shown at the bottom of the page, where the partitioning now distinguishes the job places from the resource places.

1) Definition of a General Class of Reentrant Flow Lines: The subsequent analysis deals with the broad class of reentrant flow lines now defined. This class is more general than the one in [4] as it allows assembly operations as well as the use of more than one resource per job (e.g., tool, fixture, and machine) as in [7]. Included also are all the reentrant flow lines without assembly treated in references such as [9] and [12]. Some preliminary definitions are needed.

Definition 3—Complete and Partial Part Paths: Given a reentrant flow line with assembly, define a complete part path as one that terminates in an output product (e.g., a PO place in the PN), and a partial part path as one that merges with another part path in an assembly operation.

Note that each complete part path terminates in an extra transition that feeds the part output place and is required to release the pallets, if any are used in that corresponding part path.

It is important to order the job places correctly to obtain a lower triangular matrix F_v [18], for then the sequencing of the jobs is causal. A causal ordering is also important in taking advantage of the special structure of matrices $F_v,\ F_r,\ S_v,\ S_r$ to reduce complexity in the proof of Theorem 2. To obtain a causal ordering of the jobs, number the job places sequentially from left to right along each single part path. Suppose part path $\mathcal{J}_1$ is a complete path, with a partial part path $\mathcal{J}_2$ merging into path $\mathcal{J}_1$ at the assembly point, represented by a transition on that path. In this situation, one may number the jobs of partial path $\mathcal{J}_2$ from left to right, stopping at the last job prior to the assembly transition. Then, return to the beginning of path $\mathcal{J}_1$, picking up the place ordering by numbering the job places of path $\mathcal{J}_1$ from left to right. The transitions should be numbered corresponding to the job places they feed into.

Definition 4—Dot Notation for Input and Output Sets of a Node: Given a transition $t \in \mathcal{T}$, define by $\bullet t$ the set of places that are inputs to t, and by $t\bullet$ the set of places that are outputs of t. Given a place $p \in \mathcal{P}$, define by $\bullet p$ the set of transitions that are inputs to p, and by $p\bullet$ the set of transitions that are outputs of p. Given a set of nodes $\mathcal{S} = \{v_i\}$ (either places or transitions), define $\bullet S = \{\bullet v_i\}$ and $S\bullet = \{v_i \bullet\}$.

Definition 5—Pallet Places: Let the set of transitions along the jth part path be $x_{j1},\ x_{j2},\ \cdots,\ x_{jL_j}$. Then, if part path $\mathcal{J}_j$ is complete, it may have a pallet place p_{j0}. If so, it should be selected such that $p_{j0} \in \bullet x_{j1},\ p_{j0} \notin \bullet x_{j\ell},\ \ell \neq 1$, and $p_{j0} \in x_{jL_j}\bullet,\ p_{j0} \notin x_{j\ell}\bullet,\ \ell \neq L_j$. That is, if present, pallets are used for all jobs on a complete part path.

Definition 6—Set of Jobs of a Given Resource: Given a reentrant flow line with jobs $\mathcal{J}$ and resources $\mathcal{R}$, define the jobs associated with resource $r \in \mathcal{R}$ as

$$J(r) = r \bullet \bullet \cap \mathcal{J}. \tag{10}$$

In terms of these constructions, the class of RFL studied here is given as follows. Denote the set of resources minus the pallets as $\mathcal{R}_{-0} = \mathcal{R} - \{p_{j0}\}$.

Definition 7—Definition of a Class of Reentrant Flow Lines: Define the class of reentrant flow lines with or without assembly as those satisfying the following properties.

1) For all places $p \in \mathcal{P}$, one has $\bullet p \cap p\bullet = \phi$ the empty set. (No self-loops.)
2) For each part path $\mathcal{J}_j$, the first transition satisfies $x_{j1} \bullet \cap \mathcal{R} = \phi$ and, if the path is complete the last transition satisfies $\bullet x_{jL_j} \cap \mathcal{R} = \phi$. (Each part path has a well-defined beginning and end.)
3) For each resource $r \in \mathcal{R}_{-0}$, one has $r \in p \bullet \bullet \cap \mathcal{R}$ for all $p \in J(r) = r \bullet\bullet \cap \mathcal{J}$. (Unity job duration—each job is described by only one job place along the part path.)
4) For all places $p \in \mathcal{J}$, one has $\bullet \bullet p \cap \mathcal{R} \neq \phi$. (Every job requires at least one resource.)

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 & 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{9}$$

2) Special Form of the Incidence Matrices: The reentrant flow line definition and lemma mean that the PN matrices in (6) have a particular form. Refer to (7)–(9) in the following discussion. Matrices $F_v,\ S_v^T$ consist of possibly nonsquare diagonal blocks, one per part path. In S_v^T these are identity matrices with, in the case of complete part paths, an appended bottom row of zeros. In F_v these are identity matrices with, in the case of complete paths, an appended top row of zeros. If there is assembly there will be some 1's in F_v below the diagonal blocks, where a 1 in element $(i,\ j)$ means that place j is the last place in a partial part path and joins transition i in another part path.

Matrices $F_r,\ S_r^T$ are related as follows. If the ith transition is not the last transition in a partial part path, and there is an entry of 1 in position $(i,\ j)$ of F_r, meaning resource j is committed at transition i, then there is an entry of 1 in position $(i+1,\ j)$ of S_r^T, meaning that the resource is released at the next transition. If the ith transition is the last transition in a partial part path, and there is an entry of 1 in position $(i,\ j)$ of F_r, then there is an entry of 1 in position $(k,\ j)$ of S_r^T, meaning that the resource is released at the assembly transition k.

This structure results in a particularly convenient form of the PN incidence matrix $W = [S_v^T - F_v \quad S_r^T - F_r] \equiv [W_v \quad W_r]$. Block W_v has diagonal blocks having 1's on the diagonal and –1's on the subdiagonal, with some –1's below these blocks in the case of assembly operations. In each column, matrix W_r has a –1 immediately followed by a 1, except in the case of assembly where the occurrence of the following 1 is shifted down to the assembly transition. In the case of shared resources, there is more than one –1, 1 pair in the column. In columns corresponding to pallets, the 1 occurs at the beginning of the associated diagonal block of W_v and the –1 at its end.

B. Algorithm for Computation of the p-Invariants

For the reentrant flow line, an algorithm for determining all the p-invariants in a polynomial number of operations is given by the following theorem.

Theorem 2—Computation of a Set of Independent p-Invariants: Let there be given the PN matrices (6) for a flow line satisfying Definition 7, with places in the job vector v ordered in the causal ordering specified in Section V-A. Form matrices $\hat{F}_v,\ \hat{F}_r$ by deleting the rows of $F_v,\ F_r$ corresponding to the extra terminating transitions in each complete part path. Form matrices $\hat{S}_v,\ \hat{S}_r$ by deleting the columns of $S_v,\ S_r$ corresponding to the extra terminating transitions in each complete part path. Then, the complete set of p-invariants (resource loops) is given by the columns of the matrix

$$P = \begin{bmatrix} -(\hat{S}_v^T - \hat{F}_v)^{-1}(\hat{S}_r^T - \hat{F}_r) \\ \mathcal{I} \end{bmatrix} \tag{11}$$

where $\mathcal{I}$ is the identity matrix.

Proof: The p-invariants are defined using (3) where W is given by (6) and, for the reentrant flow line, $W_v,\ W_r$ have the special form noted in Section V-A2. This shows that the p-invariants are defined by

$$[W_v\ W_r]\begin{bmatrix} v \\ r \end{bmatrix} = 0,$$

with v a vector of job places and r a vector of resource places, or $W_v v = -W_r r$.

To construct a special left inverse of W_v to solve this equation for v, delete the extra last transitions in the complete part paths to define $\hat{W} = \hat{S}^T - \hat{F} = [\hat{S}_v^T - \hat{F}_v\ \ \hat{S}_r^T - \hat{F}_r] \equiv [\hat{W}_v\ \ \hat{W}_r]$. This makes matrix $\hat{W}_v$ square. This is allowed as the deleted rows of W_v are in the row space of the remaining rows. Then, the p-invariants are defined by $\hat{W}_v v = -\hat{W}_r r$ so that $v = -\hat{W}_v^{-1}\hat{W}_r r$ for any r. To obtain a basis for nullspace W, set $r = \mathcal{I}$, the identity, resulting in (11).

It is required now to show that the resulting v is binary. According to the discussion in Section V-A2 on the special structure of the DE matrices, $\hat{W}_v$ is lower block triangular with blocks on the diagonal corresponding to each part path and having the form

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}.$$

The inverse of such a block is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix} \tag{a}$$

which appears as the corresponding diagonal block of $\hat{W}_v^{-1}$. In the case of assembly, there are some entries in $\hat{W}_v^{-1}$ below these diagonal blocks. Specifically, if there is a subdiagonal entry of –1 in position $(i,\ j)$ of W_v the meaning is that there is a partial part path $\mathcal{J}_1$ whose last place j feeds into an assembly transition i in a part path $\mathcal{J}_2$. In this event, the lower off-diagonal block corresponding to the diagonal blocks $\mathcal{J}_1$ and $\mathcal{J}_2$ [e.g., block $(\mathcal{J}_2,\ \mathcal{J}_1)$] is zero, but filled with 1's on rows i and below.

Now one must turn to the structure of $-\hat{W}_r$. Since resources are always committed prior to their release, and all jobs have unity duration, the entries in any column of $-\hat{W}_r$ consist in the case of no assembly of 1's followed immediately by –1's. It is easy to see that such entries multiplied by blocks such as (a) always result in elements of 0 or 1 in v. In the case of an assembly with partial part path $\mathcal{J}_1$ feeding into part path $\mathcal{J}_2$, an entry of 1 on the row corresponding to the last transition of partial path $\mathcal{J}_1$ is followed in any column j by $a - 1$ in row i, where transition i is the assembly transition in path $\mathcal{J}_2$. However, this corresponds to the beginning of the fill of 1's in block $(\mathcal{J}_2,\ \mathcal{J}_1)$ of $\hat{W}_v^{-1}$, and hence $\hat{W}_v^{-1}\hat{W}_r$ can be seen to yield only entries of 0 or 1 in v.

Using the formula in the theorem allows one to compute mathematically the resource loops for very large flow lines where it is very difficult to obtain any results by inspection.

VI. Conclusion

The resource loops or p-invariants of a reentrant flow line yield important information useful in job sequencing control and in assignment of shared resources to avoid deadlock. They are determined by finding a binary basis for the nullspace of a certain matrix. We have shown by reduction from the ONE-IN-3SAT problem that finding a binary basis for the nullspace of the p-invariant matrix is $\mathcal{NP}$-complete in the general job shop problem. For a large class of reentrant flow lines with assembly, however, we exhibited a closed-form solution for a binary basis. The importance of correctly selecting part flow and job routing protocols in flexible manufacturing systems is thereby conclusively demonstrated.

References

[1] E. R. Boer and T. Murata, "Generating basis siphons and traps of petri nets using the sign incidence matrix," *IEEE Trans. Circuits Syst.*, vol. 41, pp. 266–271, Apr. 1994.

[2] A. A. Desrochers, *Modeling and Control of Automated Manufacturing Systems*. Los Alamitos, CA: IEEE Computer Society Press, 1990.

[3] E. A. Elsayed and T. O. Boucher, *Analysis and Control of Production Systems*, 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1994.

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

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of $\mathcal{NP}$-Completeness*. San Francisco, CA: Freeman, 1979.

[6] S. C. Graves, "A review of production scheduling," *Oper. Res.*, vol. 29, pp. 646–675, Aug. 1981.

[7] M. D. Jeng and F. DiCesare, "Synthesis using resource control nets for modeling shared-resource systems," *IEEE Trans. Robot. Automat.*, vol. 11, pp. 317–327, June 1995.

[8] B. H. Krogh and L. E. Holloway, "Synthesis of feedback control logic for discrete manufacturing systems," *Automatica*, vol. 27, no. 4, pp. 641–651, July 1991.

[9] P. R. Kumar and S. P. Meyn, "Stability of queueing networks and scheduling policies," in *Proc. IEEE Conf. Decision Contr.*, Dec. 1993, pp. 2730–2735.

[10] A. Kusiak, "Intelligent scheduling of automated machining systems," in *Intelligent Design and Manufacturing*, A. Kusiak, Ed. New York: Wiley, 1992.

[11] F. L. Lewis, A. Gürel, S. Bogdan, A. Doganalp, and O. C. Pastravanu, "Analysis of deadlock and circular waits using a matrix model for flexible manufacturing systems," Automatica, to be published.

[12] S. H. Lu and P. R. Kumar, "Distributed scheduling based on due dates and buffer priorities," *IEEE Trans. Automat. Control*, vol. 36, no. 12, pp. 1406–1416, Dec. 1991.

[13] T. Murata, "Petri nets: Properties, analysis and applications," *Proc. IEEE*, vol. 77, no. 4, pp. 541–580, Apr. 1989.

[14] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna, "A Petri net-based controller for flexible and maintainable sequence control and its applications in factory automation," *IEEE Trans. Ind. Electron.*, vol. IE-33, pp. 1–8, Feb. 1986.

[15] R. J. G. Ramadge and W. M. Wonham, "The control of discrete event systems," *Proc. IEEE*, vol. 77, no. 1, pp. 81–98, 1989.

[16] D. V. Steward, "On an approach to techniques for the analysis of the structure of large systems of equations," *SIAM Rev.*, vol. 4, no. 4, pp. 321–342, Oct. 1962.

[17] D. Tacconi and F. L. Lewis, "A new matrix model for discrete event systems," *IEEE Contr. Syst. Mag.*, vol. 17, pp. 62–71, Oct. 1997.

[18] J. N. Warfield, "Binary matrices in system modeling," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-3, pp. 441–449, Sept. 1973.

[19] K. Y. Xing, B. S. Hu, and X. C. Chen, "Deadlock avoidance policy for Petri net modeling of flexible manufacturing systems with shared resources," *IEEE Trans. Robot. Automat.*, vol. RA-41, pp. 289–295, 1996.

[20] M.-C. Zhou, F. DiCesare, and A. D. Desrochers, "A hybrid methodology for synthesis of petri net models for manufacturing systems," *IEEE Trans. Robot. Automat.*, vol. 8, pp. 350–361, June 1992.