University of New Mexico
UNM Digital Repository

Electrical & Computer Engineering Technical Reports

Engineering Publications

4-6-1996

On the Computational Complexity of the Manufacturing Job Shop and Renetrant Flow Line

F.L. Lewis

Follow this and additional works at: https://digitalrepository.unm.edu/ece_rpts

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

SCHOOL OF ENGINEERING

UNIVERSITY OF NEW MEXICO

ON THE COMPUTATIONAL COMPLEXITY OF THE MANUFACTURING JOB SHOP AND REENTRANT FLOW LINE

F.L. Lewis
Automation and Robotics Research Institute
The University of Texas at Arlington
7300 Jack Newell Blvd. S,
Ft. Worth, Texas 76118-7115
and
B.G. Horne
NEC Research Institute
4 Independence Way
Princeton, NJ 08540
and
C.T. Abdallah
EECE Department
University of New Mexico
Albuquerque, NM 87131, USA

Contents

List of Tables

List of Figures

Abstract

This paper presents a comparison study of the computational complexity of the general job shop protocol and the flow line protocol in a flexible manufacturing system. It is shown that a certain 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.

Key Words

Job Shop problem, Flow Line, Complexity Theory, Petri Net

Chapter 1

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 machines, robots, fixtures, tooling, and so on, different structures result under different routing/assignment strategies by the controller. Unstructured strategies are generally clasified 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 [7] potentially provides a comprehensive approach for analysis of computational complexity in FMS. This possiblility has not been rigorously explored. There are many distinct *analysis and design problems* to be solved in FMS, including scheduling with optimality, computation of the Petri net (PN) p-invariants to determine resource loops, analysis of deadlocks and circular waits, design and implementation of deadlock avoidance strategies, and design/selection of dispatching and routing algorithms. These problems have varying degrees of complexity, and complexity varies as well depending on whether one has a flow line, assembly line, or job shop protocol structure.

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 2 processors, scheduling while minimizing the maximum flow time is $\mathcal{NP}$-complete for both nonpreemptive and preemptive schedules [8]. For the general job shop protocol the situation is even worse (see for example page 242 of [7]). 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. On the other hand, determining circuits in a graph, as required, e.g., to find the wait relations in an FMS, is polynomial. 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) [16, 18] 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 [3, 6, 9, 23]. PN approaches to the design of FMS sequencing/dispatching controllers are found in [10, 11, 17].

The PN incidence matrix has been used for analysis applications in FMS. It has been shown that it can be used to study structural properties of FMS, including determination of the siphons [1] and deadlock avoidance [14]. In these papers, 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*, which must be known for effective deadlock avoidance. However, matrix applications in PN have not been fully exploited. 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 that can include assembly operations. To reveal the importance of *structure in the study of complexity* for FMS, 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 reentrant flow line protocols it is polynomial. The importance of selecting suitable controller sequencing protocols to reduce complexity in FMS is thereby shown.

This paper is organized as follows. Chapter 2 presents an overview of computational complexity theory and of the reduction approach. Chapter 3 presents various manufacturing structures including the job shop, assembly line, and flow line, and introduces some PN notions. The computational complexity of finding the p-invariants is shown to be $\mathcal{NP}$-hard for the job shop in Chapter 4 and polynomial for a general class of reentrant flow lines in Chapter 5.

Chapter 2

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 [7].

In order to discuss the complexity of an algorithm, one must begin with a model of computation, for which the Turing Machine is the most commonly used. The simplicity of the Turing machine model appears to make it of little practical value; however, the Church-Turing Thesis holds that the class of problems that are tractable on a Turing machine are also tractable on any other reasonable model of computation (including the computers we use).

One important concept in this theory is that of a *polynomial-time algorithm*, i.e. an algorithm whose running time can be bounded by a polynomial in the size of the description of the problem. In practice, such an algorithm can be feasibly implemented on a real computer. This is in contrast to an *exponential-time algorithm*, which is only feasible if the problem being solved is extremely small.

The complexity class $\mathcal{P}$ consists of all decision problems that can be decided in polynomial-time, while the class $\mathcal{EXP}$ consists of those that can be decided in exponential-time. The complexity class $\mathcal{NP}$ lies inbetween 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 [7] 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 $c = d = false$.
$\square$

Chapter 3

Structure and Modeling of Flexible Manufacturing Systems

In this chapter we discuss flexible manufacturing systems with several sorts of structures, including the reentrant flow line, 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.

3.1   Flexible Manufacturing Systems (FMS)

To meet competition in a global marketplace and provide flexible manufacturing in today's high-mix low-volume manufacturing environment, manufacturing systems have gone away from old-style fixed-hardware sequential assembly lines with dedicated workstations. The trend for several years has been towards *flexible manufacturing systems (FMS)*, which have four major components [2]: a set of machines or work stations, an automated material handling system that allows flexible job routing, distributed buffer storage sites, and a computer-based supervisory controller for monitoring the status of jobs and directing part routing and machine job selections. With this change in style, the emphasis has shifted towards the design of sophisticated decision-making controllers that include functions of job sequencing and dispatching, parts routing, job release, deadlock avoidance, etc.

Unfortunately, rigorous approaches to FMS in problems such as dispatching and routing, steady-state analysis, queueing stability, bottleneck studies, and so on have focused on simple types of systems including single-server, flow line without assembly, serial forms, etc. Systems with finite buffers and nonserial systems (e.g. systems with assembly, etc.) have few results, with fewer still for general job shop structures and large-scale interconnected systems. It is by now known that many manufacturing problems are in $\mathcal{NP}$ so that significant increases in computing power do not significantly improve computational capabilities. There is no general approach for taking advantage of the FMS structure to reduce computational complexity.

3.2   Manufacturing System Structures

The physical portion of an FMS is comprised of its *resources*: the set of machines or work stations, the automated material handling system, and the distributed buffers. We call these the *manufacturing facility*. Given *the same* resource facilities in the FMS, different sequencing algorithms by the controller produce *different flow/protocol structures*, including the reentrant flow line, the assembly line, and the general job shop protocol. Not only should the controller provide guaranteed performance, but it should *impose a suitable structural protocol* to achieve prescribed performance specifications, and it should be *easily reconfigurable* to
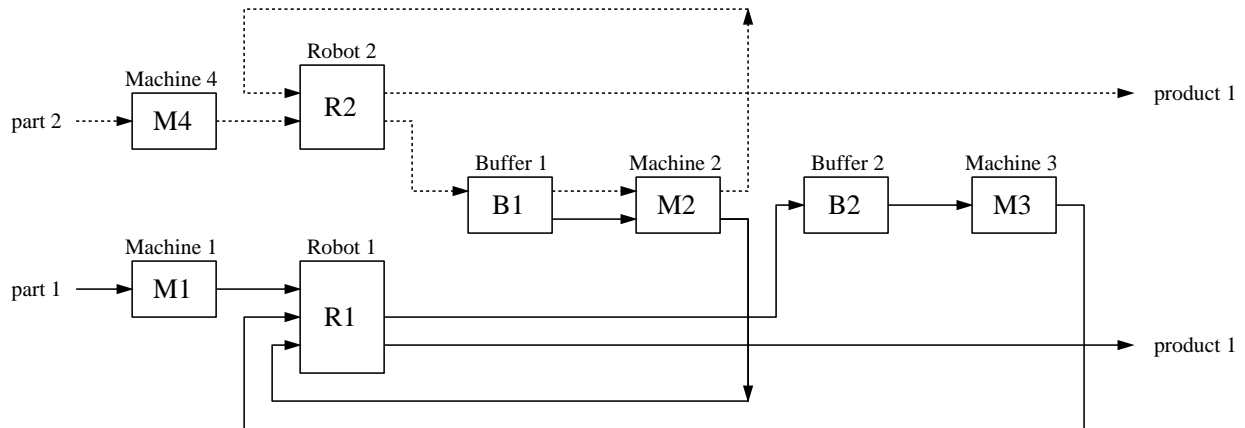
Figure 3.1: Reentrant flow line with 4 machines and 2 parts.

change the FMS structure, dispatching rules, routing algorithms, etc. as products or performance requirements change. Disciplines such as *discrete event (DE) systems* are emerging to confront such problems [19]. 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 j-th type are denoted p_j. A *job sequence for part type p_j* is a sequence of P_j jobs $\mathcal{J}_j = \{J_{1j}, J_{2j}, \dots J_{P_j j}\}$ required to produce a finished product. We distinguish between jobs in the part sequence even if, for instance J_{2j} and J_{5j} are both drilling operations. The sequence of jobs may be determined from a task decomposition, bill of materials, assembly tree, or precedence matrix (c.f. Steward's sequencing matrix [20]). If each job is performed on a single part and delivers a single part there is said to be *no assembly*.

Once the sequence of jobs for a part type has been assigned, *resources must be assigned* to perform the jobs. This is performed by a manufacturing engineer based on the facilities available. If a single resource is needed for each job, for instance, this corresponds to a pairing (J_{kj}, r_i) of the k-th job for part p_j with a resource r_i. The ordering of the jobs for a given part type can be either fixed or variable. For instance, in an application it may be allowable to either drill then machine a part, or to machine and then drill the part. Likewise, the resources assigned to each job can be either fixed or variable. For instance, either of two machines of different types (e.g. from different resource pools) might be capable of performing a given drilling job.

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 flow line is also known as the "job shop with fixed part routing". The sequence in which part type p_j visits the resources in a flow line will be called the j-th *part path*. A flow line is said to *reentrant* if any part type revisits the same resource more than once in its job sequence [12, 15]. 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. 3.1. 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 $M1, M2, M3, M4$ could be machines. Thus, the resources may include machines, robots, buffers, transport devices, fixtures, tools, and so on.
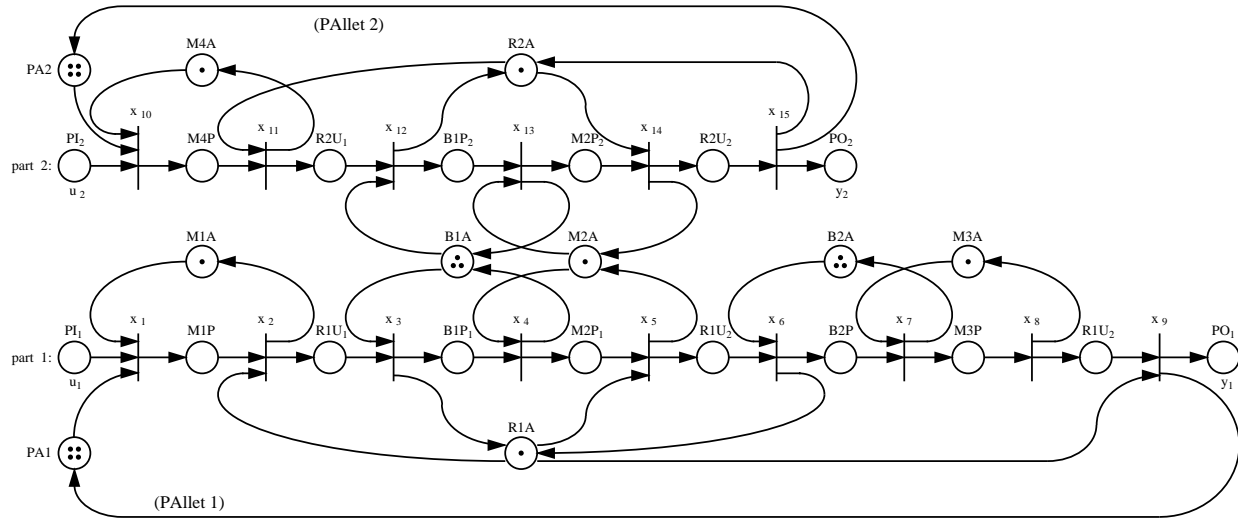
Figure 3.2: PN representation of the reentrant flow line.

3.3   Petri Net Representation of FMS

Some knowledge of Petri nets is assumed. 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. Operation duration times and resource setup times are captured in *timed places*, as opposed to the timed transition approach. For instance, a standard representation for a reentrant flow line is given in Fig. 3.1. The PN representation for the same system is shown in Fig. 3.2, 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 of resources.

3.3.1   Incidence Matrix and Marking Transition Equation

It is common in PN theory [18] 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

$$W = O - I. \tag{3.1}$$

A column vector p indexed by the set of places $\mathcal{P}$ is called the PN p-vector (place vector). The *PN marking vector* is the marking vector $m(p)$ 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 $p = [p_1 \quad p_2 \ldots p_q]^T$, the marking $m(p)$ is the vector $m(p) = [m(p_1) \quad m(p_2) \ldots m(p_q)]^T$ 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 $m(p)$ at time t. Then, in terms of the PN incidence matrix, one can write the *PN marking transition equation*

$$m(t_2) = m(t_1) + W^T \tau = m(t_1) + (O - I)^T \tau, \tag{3.2}$$

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 $i - th$ transition has fired n_i times in the interval.

3.3.2   Resource Loops and p-Invariants

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

Definition 2 (p-Invariant and Resource Loop) *A p-invariant is a place vector p having elements of zeros and ones that is in the nullspace of W, that is*

$$Wp = 0. \tag{3.3}$$

The set of places corresponding to the support of p is known as a resource loop, *also loosely called a p-invariant.*

The complete set of p-invariants of a PN gives a great deal of information. In [14] it is shown that they provide the basis for deadlock avoidance algorithms. The importance of p-invariants may be understood by noting that, beginning with (3.2), for any p-invariant p one has

$$p^T m(t_2) = p^T m(t_1) + p^T W^T \tau = p^T m(t_1). \tag{3.4}$$

Noting that premultiplication by p^T simply sums up the tokens in the positions of $m(\cdot)$ corresponding to the support of p, this is seen to be a statement that *the total number of tokens in positions of $m(\cdot)$ corresponding to the support of 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.

Chapter 4

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

The resource loops of an FMS contain information of great value in deadlock avoidance, shared resource conflict resolution using dispatching techniques, and so on. Unfortunately, to find the p-invariants it is necessary solve (3.3), determining a basis for the nullspace of W that has only ones and zeros. In this chapter, we show that finding such a binary basis is an $\mathcal{NP}$-complete problem for the general job shop structure. Then, in Section 5 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 if 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 meanigful basis of its nullspace will have vectors p whose entries p_i also belong to $\{0, +1\}$, the problem is equivalent to finding p_i such that $\sum_{i=1}^{n} w_{ij} p_i = 0; \quad \forall j = 1, \cdots n$. 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.*

$$x = \left[\begin{array}{ccccccccc} x_1 & \bar{x}_1 & x_2 & \bar{x}_2 & \ldots & x_n & \bar{x}_n & z \end{array} \right]' .$$

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 $\bar{x}_i$. Specifically, the

first $|U|$ rows are configured as,

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

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. $\tilde{x}_i \in \{x_i, \bar{x}_i\}$), and denote the ith clause by $c_i = \tilde{x}_{p_i}\tilde{x}_{q_i}\tilde{x}_{r_i}$. Then set

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

Example 2 *Let $U = \{x_1, x_2, x_3, x_4\}$ and let $C = \{x_1\bar{x}_2 x_3, x_2 x_3\bar{x}_4, \bar{x}_1 x_2 x_4\}$. Then the matrix A is given by*

$$A = \begin{array}{c} \begin{array}{ccccccccc} x_1 & \bar{x}_1 & x_2 & \bar{x}_2 & x_3 & \bar{x}_3 & x_4 & \bar{x}_4 & z \end{array} \\ \left[\begin{array}{ccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & -1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & -1 \end{array} \right] \end{array}$$

□

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 $\bar{x}_i$ 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.

We can easily make A of size $n \times 2n$ 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. ■

Chapter 5

Computational Complexity of Finding the p-Invariants in the Flow Line

Like many other problems, finding the p-invariants in a general job shop protocol is $\mathcal{NP}$-complete, as seen in the previous chapter. In this chapter, 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. 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. The flow line structure allows one to model and analyze large-scale interconnected FMS in a polynomial number of operations using *block matrices*.

5.1   Structure of the Reentrant Flow Line

In the reentrant flow line with or without assembly, e.g. Fig. 3.2, denote the set of jobs for part type j as $\mathcal{J}_j$ and the set of all the jobs as $\mathcal{J} = \bigcup_j \mathcal{J}_j$. It is noted that the *part input places PI* and *part output places PO* are not included as jobs. 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 resources may be partioned as $\mathcal{R} = \mathcal{R}_{ns} \cup \mathcal{R}_s$, with $\mathcal{R}_{ns}$ the *nonshared resources* and $\mathcal{R}_s$ the *shared resources*. 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

$$p = \left[\begin{array}{c} v \\ r \end{array} \right], \tag{5.1}$$

where v is the vector of places corresponding to the jobs $\mathcal{J}$ and r is the vector of places corresponding to the resources $\mathcal{R}$. Then, the PN incidence matrix has the structure

$$W = [W_v \quad W_r] \equiv S^T - F = [S_v^T - F_v \quad S_r^T - F_r] \tag{5.2}$$

where S_v^T, S_r^T are the output incidence matrices of the jobs and resources respectively , and F_v^T, F_r^T are the input incidence matrices of the jobs and resources respectively. This formalizes some discussions in [3] concerning places of type A, B, C. Matrix F_v is called the *Steward sequencing matrix* [20] or the *Bill of Materials (BOM)* [4] 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 [13]; it has element $(i, j) = 1$ if resource j is required for job i.

It is important to order the job places correctly to obtain a *lower triangular matrix* F_v [5, 21], for then the sequencing of the jobs is *causal*. A causal ordering is also important in taking advantage of structure to

reduce complexity. The special structure of matrices F_v, F_r, S_v, S_r for a general class of reentrant flow lines is revealed in terms of the following constructions.

5.1.1 Definition of a General Class of Reentrant Flow Lines

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 is required to produce the product output equations and to release the pallets, if any are used in that corresponding part path. 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 j_1 is a complete path, with a partial part path j_2 merging into path j_1 at the assembly point, represented by a transition on that path. In this situation, one may number the jobs of partial path j_2 from left to right, stopping at the last job prior to the assembly transition. Then, return to the beginning of path j_1, picking up the place ordering by numbering the the job places of path j_1 from left to right. The transitions should be numbered corresponding to the job places they feed into. This procedure corresponds to numbering the jobs from bottom to top as is standard in a manufacturing *assembly tree* [22].

The subsequent analysis deals with the class of reentrant flow lines now defined. This class is more general than the one in [6] 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 [10].

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 $\bullet \mathcal{S} = \{\bullet v_i\}$ and $\mathcal{S}\bullet = \{v_i\bullet\}$.*

Definition 5 (Pallet Places) *Let the set of transitions along the j-th part path be $x_{j1}, x_{j2}, \ldots, x_{jL_j}$. Then, if part path 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

$$J(r) = r\bullet\bullet \cap \mathcal{J}. \tag{5.3}$$

In terms of these constructions, the class of FMS 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 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.)*

This definition results in the following facts, easily derivable using definition.

Lemma 2 (Properties of the Class of Reentrant Flow Lines) *The class of reentrant flow lines considered satisfies the following properties:*

1. *The job set of r is given by $J(r) = r \bullet\bullet \cap \mathcal{J} = \bullet\bullet\, r \cap \mathcal{J}$ for all resources $r \in \mathcal{R}_{-0}$.*

2. *$p \bullet\bullet \cap \mathcal{R} = \bullet\bullet\, p \cap \mathcal{R}$ for all jobs $p \in \mathcal{J}$.*

3. *If there are pallets for part path j, then $p_{j0} \bullet\bullet \cap \mathcal{R} = \phi$, $\bullet\bullet\, p_{j0} \cap \mathcal{R} = \phi$. (This follows directly from Definition 5 and Definition 7 item 2).*

4. *Let $p \in \mathcal{P} = \mathcal{R} \cup \mathcal{J}$. Then $p \in p_\ell$ for some p-invariant p_ℓ. That is, the flowline is covered by p-invariants.*

5.1.2 Special Form of the Incidence Matrices.

The reentrant flow line Definition and Lemma mean that the PN matrices in (5.2) have a particular form. Matrices F_v, S_v^T consist of *diagonal blocks*, one per part path, which in S_v^T are identity matrices, and in F_v have a subdiagonal of 1's. 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 i-th 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 i-th 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 $W = [S_v^T - F_v \ \ S_r^T - F_r] \equiv [W_v \ \ W_r]$. 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.

5.2 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 (5.2) for a flow line satisfying Definition 7, with places in the job vector v ordered in the causal ordering specified in Section 5.1. 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*

$$P = \left[\begin{array}{c} -(\hat{S}_v^T - \hat{F}_v)^{-1}(\hat{S}_r^T - \hat{F}_r) \\ I \end{array} \right] \tag{5.4}$$

where I is the identity matrix.

<u>*proof:*</u>

The p-invariants are defined using (3.3) where W is given by (5.2) and, for the reentrant flow line, W_v, W_r have the special form noted in Section 5.1.2. This shows that the p-invariants are defined by

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

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

$$\hat{W} = \hat{S}^T - \hat{F} = [\hat{S}_v^T - \hat{F}_v \quad \hat{S}_r^T - \hat{F}_r] \equiv [\hat{W}_v \quad \hat{W}_r].$$

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 $r = I$, the identity, resulting in (5.4).

It is required now to show that the resulting v is binary. According to the discussion in Section 5.1.2 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

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \qquad (a)$$

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 j_1 whose last place j feeds into an assembly transition i in a part path j_2. In this event, the lower off-diagonal block corresponding to the diagonal blocks j_1 and j_2 (e.g. block (j_2, 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 j_1 feeding into part path j_2, an entry of 1 on the row corresponding to the last transition of partial path j_1 is followed in any column j by a -1 in row i, where transition i is the assembly transition in path j_2. However, this corresponds to the beginning of the fill of 1's in block (j_2, 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.

Chapter 6

Conclusion

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. This implies that the job shop deadlock analysis problem will be at least as hard as the subproblem of finding the p-invariants. In the case of the reentrant flow line 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. The job routings and resource allocations should follow the structural protocols developed Section 5.1 to simplify the complexity of shared resource dispatching analysis of the FMS.

Bibliography

[1] E.R. Boer and T. Murata, "Generating basis siphons and traps of Petri nets using the sign incidence matrix," *IEEE Trans. Circuits and Systems*, vol. 41, no. 4, pp. 266-271, April 1994.

[2] J.A. Buzacott and D.D. Yao, "Flexible manufacturing systems: a review of analytical models," *Management Sci.*, vol. 32, no. 7, pp. 890-905, July 1986.

[3] A.A. Desrochers, *Modeling and Control of Automated Manufacturing Systems*, IEEE Computer Society Press, 1990.

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

[5] S.D. Eppinger, D.E. Whitney, and R.P. Smith, "Organizing the tasks in complex design projects," *Proc. ASME Int. Conf. Design Theory and Methodology*, pp. 39-46, Sep. 1990.

[6] J. Ezpeleta, J.M. Colom, and J. Martinez, "A Petri net based deadlock prevention policy for flexible manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 11, no. 2, pp. 173-184, Apr. 1995.

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

[8] S.C. Graves, "A review of production scheduling," *Operations Research*, vol. 29, pp. 646-675, Aug. 1981.

[9] M.D. Jeng and F. DiCesare, "A synthesis method for Petri net modeling of automated manufacturing systems with shared resources," *Proc. IEEE Conf. Decision and Control*, pp. 1184- 1189, Dec. 1992.

[10] M.D. Jeng and F. DiCesare, "Synthesis using resource control nets for modeling shared-resource systems," *IEEE Trans. Robotics and Automation*, vol. 11, no 3, pp. 317-327, June 1995.

[11] 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.

[12] P.R. Kumar and S.P. Meyn, "Stability of queueing networks and scheduling policies," *Proc. IEEE Conf. Decision and Control*, pp. 2730-2735, Dec. 1993.

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

[14] F. L. Lewis, H.-H. Huang, A. Gürel, O.C. Pastravanu, and D. Tacconi, "Analysis of deadlocks and circular waits using a matrix-based approach," submitted, 1995.

[15] 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.

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

[17] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna, "A Petri net-based controller for flexible and maintanable sequence control and its applications in factory automation," *IEEE Trans. Ind. Electronics*, vol. IE-33, no. 1, pp. 1-8, Feb. 1986.

[18] Peterson, J.L. (1981), *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, Englewood Cliffs, NJ.

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

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

[21] J.N. Warfield, "Binary matrices in system modeling," *IEEE Trans. Systems, Man, Cybern.*, vol. SMC-3, no. 5, pp. 441-449, Sept. 1973.

[22] J. Wolter, S. Chakrabarty, and J. Tsao, "Methods of knowledge representation for assembly planning," *Proc. NSF Design and Manuf. Sys. Conf.*, pp. 463-468, Jan. 1992.

[23] M.-C. Zhou, F. DiCesare, A.D. Desrochers, "A hybrid methodology for synthesis of Petri net models for manufacturing systems," *IEEE Trans. Robotics and Automation*, vol. 8, no. 3, pp. 350-361, Jun. 1992.