Scheduling Flexible Manufacturing Systems Using Petri Nets and Heuristic Search

Doo Yong Lee, *Member, IEEE,* and Frank DiCesare, *Member, IEEE*

***Abstract*—Petri net modeling combined with heuristic search provides a new scheduling method for flexible manufacturing systems. The method formulates a scheduling problem with a Petri net model. Then, it generates and searches a partial reachability graph to find an optimal or near optimal feasible schedule in terms of the firing sequence of the transitions of the Petri net model. The method can handle features such as routing flexibility, shared resources, lot sizes and concurrency. By following the generated schedule, potential deadlocks in the Petri net model and the system can be avoided. Hence the analytical overhead to guarantee the liveness of the model and the system is eliminated. Some heuristic functions for efficient search are explored and the experimental results are presented.**

I. Introduction

FLEXIBLE manufacturing systems (FMS's) can produce multiple types of products using various resources such as robots, multipurpose machines, etc. While the increased flexibility of an FMS provides a greater number of choices of resources and routings, and allows greater productivity, it imposes a challenging problem; i.e., the allocation of given resources to different processes required in making each product and the scheduling of the sequence of activities to accomplish the best efficiency. This paper presents a new method to schedule FMS's, which uses a Petri net formulation and heuristic search.

Scheduling problems are known to be complex even for simple formulations [4], [5], [9], [19], [32] and are NP-hard in many cases [11], [24]. Various scheduling methods have been developed to tackle the ever increasing complexity and flexibility of manufacturing [12], [33], [40], [41], [42], [49].

Traditionally, the routing of a part to complete the sequence of required processes is considered planning and the assignment of resources according to the determined routing is considered scheduling. Planning and scheduling are often separately carried out with little interaction. But in an FMS, the planning and the scheduling should be collectively carried out to take full advantage of the flexibility of the system. Since a machine in the FMS can be used for multiple jobs and there are choices of resources to be used, the assignment of resources, or scheduling, must be considered when the routing of a part is planned.

Manuscript received November 3, 1992; revised July 30, 1993.

D. Y. Lee is with Information Technology Services, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

F. DiCesare is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

IEEE Log Number 9215779.

Because of the complexity of generative scheduling methods, evaluative methods such as simulation combined with heuristic dispatch rules have recently been given much attention. However, comprehensive simulation models that effectively treat flexible and automated manufacturing environments are expensive and difficult to develop. Simulation models are often specific to particular applications so that the models and the simulation results are difficult to generalize. Moreover, instead of systematic handling of the constraints of FMS's, present heuristic dispatch rules often rely on empirical experience. This again hinders the generalization of the simulation methods.

A complete and general scheduling method must be able to formulate explicitly and concisely a scheduling problem, and provide an efficient and general technique to solve the formulated problem. Current scheduling methods in the literature have difficulties with the formulation and/or the solution techniques. Methods such as integer programming [14], [15], [16], [31], [37], search [10], [20], [29], [30], [48], [52], and network models (CPM/PERT, queuing networks, graphs) [1], [6], [8], [21], [22], [43] can provide efficient solution techniques but have formulation difficulties in accounting for shared resources, concurrency, routing flexibility, lot sizes, etc. Methods such as algebraic models [2], [42], and control theoretic approaches [12], [18], [41] have difficulties in providing efficient solution techniques.

Petri nets are well suited to model the dynamics of flexible manufacturing systems. Petri nets can concisely represent the activities, resources and constraints of the system in a single coherent formulation. And since Petri nets are a graphical tool, designers can also better understand and formulate scheduling problems. Larson and Odoni [23] discuss the merits of graphical models. Petri nets have been widely used for performance analysis of given schedules [3], [7], [13], [17], [39], [46], [47], [50].

Although Petri nets showed promise as an effective tool to formulate and solve the scheduling problems of FMS's, the actual generation of schedules has not been given much attention in the Petri net community. Recently, there have been some independent efforts to use Petri nets to generate schedules for FMS's. Shih and Sekiguchi [45] present an FMS scheduling system which simulates the evolution of the FMS as modeled by Petri nets. The scheduling system calls for a beam search routine whenever there is a conflict. The beam search routine then constructs partial schedules within the beam-depth and evaluates them to choose the best one. The cycle is repeated until a complete schedule

is achieved. This method based on partial schedules does not guarantee global optimization. Onaga *et al.* [36] present a linear programming approach for periodic scheduling of systems modeled by Petri nets. Shen *et al.* [44] present a scheme which starts with an arbitrary schedule and applies branch and bound search to find an optimal schedule. Zhang [51] presents a method which translates rules of a rule based planning system into a timed Predicate/Transition net model and applies the A^* algorithm to find an optimal schedule.

The scheduling method presented in this paper formulates a scheduling problem using a Petri net model, and employs global search and limits the search space by the use of heuristic functions [25], [26]. The methods generates an optimal or near optimal feasible schedule in terms of the firing sequence of the transitions of the Petri net model. This method is also event driven as opposed to time driven, i.e., the schedule is provided as an order of the initiations of the activities. Most of the current scheduling approaches can be considered as time driven, i.e., the schedule is a list of time instants when certain activities are to happen [35]. This approach may not always be best for the scheduling of flexible manufacturing systems that are, by nature, discrete event driven [40]. Event driven scheduling focuses on the precedence constraints of the activities and is robust to disturbances.

There are many targets for optimization in manufacturing. For example, the minimization of makespan and/or tardiness is one of the frequently adapted goals. The maximum utilization of critical machines is also often considered. Generating a schedule with the minimum or near minimum makespan is the focus in this paper.

In timed Petri nets, time can be associated with either places (timed place Petri nets), or transitions (timed transition Petri nets). Generally, a timed transition removes tokens from its input places and takes some time before it introduces tokens to its output places. Therefore, between the initiation and the termination of firing, the marking (the state of the system) is uncertain. Depending on whether timed transitions or timed places are used, activities are associated with transitions or places, respectively. In the case of timed transitions, multiple initiation or firing of transitions must be allowed to represent concurrency of activities. Therefore, the time associated with each initiated transition must be tracked in order to correctly update the marking, or state. Since the initiated transitions may not be tracked in applications of Petri net modeling, an additional tracking method is required.

But when timed places are used, multiple marked places naturally represent the concurrency of activities associated with the places and it is only necessary to keep track of time for the marked places. And at any given instant, there is no ambiguity in the marking. In this paper, time is associated only with places, and all transitions are immediate. Transitions represent the initiation or the termination of the involved activities. Places, therefore, represent activities or resources [18], [45]. Each token in the places represents either the availability of a resource, the readiness of a part for the next process, or a part being processed.

II. Petri Net Modeling for Scheduling

For the discussion of this paper, every part or partially finished product which enters and moves inside an FMS is called a part. A finished part that leaves the FMS is called a product. Each product is the result of a sequence of *processes* according to its technological requirements. Resource requirements are not considered in the processes. A sequence of processes defines a *product type* or a *job type*. The FMS can produce multiple products of the same product type, i.e., a lot of the same job type. A process can be carried out more than one way. For instance, consider the following scenario of two machines M_1 and M_2, a robot R, and a part. The part can be treated at M_1 or at M_2 with R, accomplishing the same result. Treating the part at M_1 or at M_2 is an *operation*. Hence resource requirements are considered in an operation. An operation $O_{i,j,k}$ represents the jth process of the ith job type being performed at the kth machine. All the operations are assumed to be non-preemptive. The job, therefore, can be completed by carrying out different sequences of operations although the generic sequence of processes remains the same. These different sequences of operations are vital to the flexibility of the FMS; i.e., capability to produce multiple types of product and share resources. The requirements of a job can be given by the description of alternative operations and the necessary resources as shown in the following example.

Example 1 An FMS has three machines M_1, M_2, M_3 and a robot R. Four job types J_1, J_2, J_3, J_4 are to be carried out and Table I shows the requirements for each job.

The first process of J_1 can be carried out at either M_1 with R or M_2 with R. The second process of J_1 can be carried out only at M_3 but does not require the use of R. This table also gives the technological precedence constraints among the processes. For instance, the second process of J_1 can be carried out only after the first process of J_1 is complete. No technological precedence constraints are assumed among different job types. Notice that J_3 has only two processes to be completed. Fig. 1 shows the Petri net model of J_1 with an initial marking.

The transitions are immediate. $t^b_{i,j,k}$ and $t^e_{i,j,k}$ represent the beginning and the end of an operation $O_{i,j,k}$, respectively. $p^i_{j,k}$ is the kth *initial place* of J_j representing the beginning of the job type J_j, i.e., a part for J_j is ready. The number of tokens in an initial place represents the lot size of the corresponding job. $p^f_{j,k}$ is the kth *final place* of J_j representing the end of J_j, i.e., a product J_j is completed. $p_{i,j,k}$ is an operation place representing $O_{i,j,k}$. A token in an operation place represents the specific operation being performed.

$p^n_{i,j,k}$ is the kth *intermediate place* of J_i after the jth process. A token in an intermediate place represents the readiness of the part for the next process, i.e., an intermediate place represents a buffer. p^r_i and p^m_i are *resource places* for the ith robot and the ith machine, respectively. A token in a resource place represents the availability of the corresponding resource. p^r_1 is drawn twice just for clear presentation and is actually a single place.

As indicated, a job representation may have more than one initial and/or final place. For example, assume that the

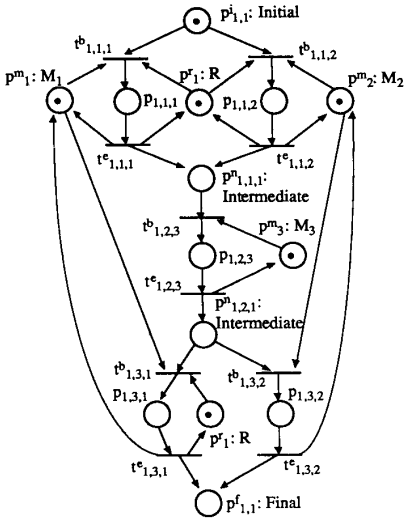


Fig. 1. The Petri net model of J_1.

TABLE I
JOB REQUIREMENTS OF EXAMPLE 1

	Job			
Process	J_1	J_2	J_3	J_4
1	M_1R/M_2R	M_1	M_3	M_2R
2	M_3	M_2R/M_3	M_1/M_2	M_3
3	M_1R/M_2	M_3	N/A	M_1R/M_2R

TABLE II
THE OPERATION TIMES OF J_1 OF EXAMPLE 1

Operation	Operation Time
$O_{1,1,1}$	10
$O_{1,1,2}$	12
$O_{1,2,3}$	7
$O_{1,3,1}$	5
$O_{1,3,2}$	8

operation represented by $p_{1,2,3}$ requires an additional partially finished part since the operation is an assembly. Then the corresponding Petri net model of J_1 can be modified as shown in Fig. 2. $p^i_{1,2}$ is an additional initial place.

In addition, if a new job type J_5 shares the same generic sequence of processes with J_1 up to the operation represented by $p_{1,2,3}$ and does not require further processes, the corresponding part of the Petri net model can be modified as shown in Fig. 3. $p^f_{5,1}$ is an additional final place for J_5. This situation can be found in chemical processes where a process generates a byproduct.

Time delays are associated with places. A token in an intermediate place or a resource place is readily available. A token at an operation place becomes available after a delay corresponding to the duration time of the operation. The duration times of operations are given as shown in Table II.

A potential deadlock is implicitly avoided by using the intermediate places. The potential deadlock happens when

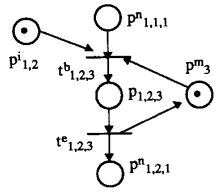


Fig. 2. The modified part of the model with an additional initial place.

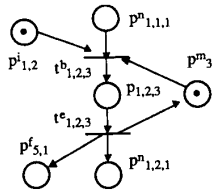


Fig. 3. The modified part of the model with an additional final place.

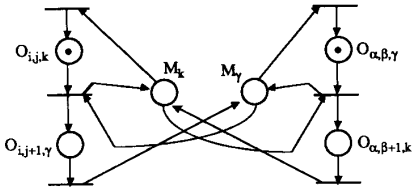


Fig. 4. A potential deadlock.

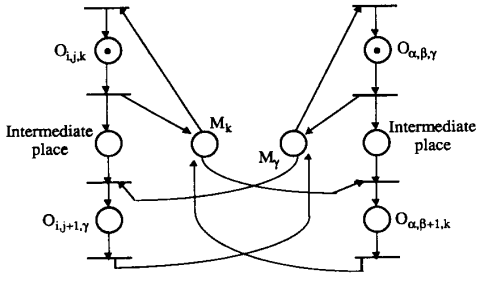


Fig. 5. The solution to the potential deadlock.

the operation $O_{i,j,k}$ is completed at the machine M_k and $O_{i,j+1,\gamma}$ requires the machine M_γ, and the operation $O_{\alpha,\beta,\gamma}$ is completed at the machine M_γ and $O_{\alpha,\beta+1,k}$ requires the machine M_k. This is illustrated in Fig. 4.

By simply swapping the resources, i.e., M_k and M_γ, the operations $O_{i,j+1,\gamma}$ and $O_{\alpha,\beta+1,k}$ can be initiated. But the model of Fig. 4 fails to achieve this. Putting intermediate places can solve this problem as shown in Fig. 5. The model of Fig. 5 corresponds to having a buffer at the machine. By unloading (by either a robot or a human operator) the part from the machine to the buffer after the operation, the machine becomes available and the deadlock does not occur. At each machine, the loading (unloading) of the part onto (from) the machine before (after) the operation is regarded as part of the operation.

This eliminates a particular type of potential deadlocks. The scheduling algorithm presented in Section III may encounter a

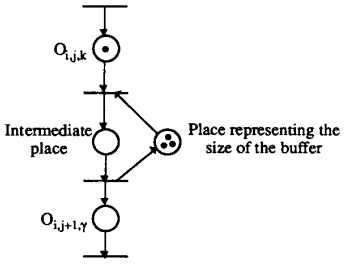


Fig. 6. A model for a buffer with a finite size.

deadlock during the search. The algorithm keeps on looking for alternative ways to get around the deadlock and chooses those paths. If there is absolutely no way to get around the deadlock, then, the desired final marking is unreachable from the initial marking. In this case, the algorithm halts and reports it.

In Fig. 5, the intermediate places, i.e., buffers, have infinite sizes. When the buffer size is finite, the model can be modified as shown in Fig. 6. The buffer size is represented by the number of tokens, e.g., the buffer size of the model shown in Fig. 6 is three.

The Petri net models of J_2, J_3 and J_4 of Example 1 can be similarly constructed using the method discussed in this section. Decomposing the entire model according to jobs and building the Petri net by subnets for jobs make it easy. Since J_1, J_2, J_3 and J_4 share common resources, i.e., places, the subnets for J_1, J_2, J_3 and J_4 are connected with each other through the places.

Modeling the job requirements of an FMS using the method described in this section gives a Petri net model which incorporates routing flexibility, shared resources and lot sizes. Since the Petri net modeling also incorporates concurrency and precedence constraints among activities, it facilitates maintaining the coherency of the problem model.

III. Scheduling Algorithm

Once the Petri net model of the system is constructed, the evolution of the system can be described by changes in the marking of the net. In other words, all possible behaviors of the system can be completely tracked by the reachability graph of the net. Theoretically, therefore, an optimal schedule can be obtained by generating the reachability graph and finding the optimal path from the initial marking to the final marking. The path is a firing sequence of the transitions of the Petri net model. But even for a simple Petri net, the reachability graph may be too large to generate in its entirety. Instead of generating the entire reachability graph, a heuristic search algorithm is employed. Depending on the heuristic functions used, this algorithm generates only the necessary portion of the reachability graph to find an optimal or near optimal path. In this way, a reasonably good schedule can be generated in a reasonable amount of time.

The scheduling algorithm $L1$ that finds the optimal path is as follows. The algorithm $L1$ is adapted from the well known graph search algorithm A^* [34], [38]. Given a Petri net model, $L1$ expands the reachability graph from the initial marking until the generated portion of the reachability graph touches the final marking. Once the final marking is found, the optimal path is constructed by tracing the pointers that denote the parenthood of the markings, from the final marking to the initial marking. Then, the transition sequence of the path provides the order of the initiations of the activities, i.e., the schedule.

Algorithm L1:
Step 1: Put the initial marking m_0 on the list OPEN.
Step 2: If OPEN is empty, terminate with failure.
Step 3: Remove the first marking m from OPEN and put m on the list CLOSED.
Step 4: If m is the final marking, construct the optimal path from the initial marking to the final marking and terminate.
Step 5: Find the enabled transitions of the marking m.
Step 6: Generate the next marking, or successor, for each enabled transition, and set pointers from the next markings to m. Compute $g(m')$ for every successor m'.
Step 7: For every successor m' of m, do the following.
a: If m' is already on OPEN, direct its pointer along the path yielding the smallest $g(m')$.
b: If m' is already on CLOSED, direct its pointer along the path yielding the smallest $g(m')$. If m' requires pointer redirection, move m' to OPEN.
c: Calculate $h(m')$ and $f(m')$, and put m' on OPEN.
Step 8: Reorder OPEN in the increasing magnitude of f.
Step 9: Go to Step 2.

$L1$ uses a function of the marking m, $f(m) = g(m)+h(m)$. $f(m)$ is an estimate of the cost, i.e., the makespan from the initial marking to the final marking along an optimal path which goes through the marking m. $g(m)$ is the current lowest cost obtained from the initial marking to the current marking m. $h(m)$ is an estimate of the cost from the marking m to the final marking along an optimal path which goes through the marking m.

The scheduling algorithm is admissible [34], i.e., it always finds an optimal path if $h(m)$ satisfies the following condition,

$$0 \leq (m) \leq h^*(m) \text{ for all } m,$$

where $h^*(m)$ is the cost of the optimal path from m to the final marking. The heuristic functions experimented with in this paper do not guarantee this lower bound condition. They are designed to get a reasonably good solution in a reasonable amount of time. Future research includes the development of heuristic functions that guarantee this lower bound condition.

Each arc of the reachability graph corresponds to firing of a transition and has a corresponding cost or time delay. Before a transition can be fired, the tokens of the input places of that transition must be delayed for at least the time delays associated with the input places. Hence, the cost of an arc of the reachability graph, $c(m, m')$ can be considered to be the maximum time delay of the input places, that is,

$$c(m, m') = \max(d_{1j}, d_{2j}, \cdots, d_{kj})$$

where $d_{ij}, i = 1, 2, \cdots, k$ is the time delay of the input place p_{ij} of the transition t_j, and the marking m converts to the marking m' through the transition t_j. But this argument is

not quite correct as explained in the following. Let $d_{\max}$ be $\max(d_{1j}, d_{2j}, \cdots, d_{kj})$. When the marking m converts to m', the tokens of the places that are not input places of t_j, are also concurrently delayed by $d_{\max}$. This is true for every marking update. In fact, therefore, the cost of an arc of the reachability graph is not really the maximum time delay of the input places, but rather the maximum of the remaining time delays of the input places, that is,

$$c(m, m') = \max(r_{1j}, r_{2j}, \cdots, r_{kj})$$

where $r_{ij}, i = 1, 2, \cdots, k$ is the remaining time delay of the input place p_{ij} of t_j. This observation is accounted for in the implementation of $L1$ so that the implementation of $L1$ keeps track of the remaining time delays of the places when it generates the next marking.

The list OPEN maintains the markings generated but not yet explored. These markings form the frontier of the reachability graph. The reachability graph grows pushing this frontier outward from the initial marking until it touches the final marking. Hence, the program does not generate the entire reachability graph, but rather generates the reachability graph layer by layer until the algorithm finds an optimal path to the final marking.

The list CLOSED maintains all the markings generated and explored thus far, i.e., the markings that have been checked against the final marking and found not to be the final marking. In Step 7 of $L1$, if a newly generated marking m' is not on either OPEN or CLOSED, it is put on OPEN to be explored later. If m' is on OPEN, it means a new path to m' from the initial marking has been found. The cost, $g(m')$ of the new path is compared with the cost of the old path. The path, then, is updated to yield the smallest cost. Since m' is still on OPEN, it can be explored in the future.

If m' is on CLOSED, it means m' is already explored and a new path to m' has been found. Since m' is already explored, some descendant markings of m' are already generated. If the paths leading to the descendant markings are followed all the way down, each path ends with a marking which is on OPEN. If the new path to m' has a cost lower than the cost of the old path, there are two options to handle this situation. First, the paths to the descendant markings of m' can be redirected propagating changes all the way down to the markings that are on OPEN. Second, as the algorithm $L1$ does, just the path to m' can be redirected and m' can be put on OPEN for re-exploration.

The first approach has the advantage that the descendant markings of m' and the paths to them that have already been generated will not be discarded. The disadvantage of the first approach is the extra effort to propagate changes to all the descendant markings of m' that may be very large in number. The second approach has the advantage of saving the effort of change propagation. Its disadvantage is that it discards previously generated descendant markings of m' and repeats the effort of generating again some of the descendant markings of m'. But some of the descendant markings of m' may not need to be regenerated at all. The advantages and disadvantages of the above two approaches applied to A^* is discussed in more detail in [34], [38].

TABLE III
JOB REQUIREMENTS OF EXAMPLE 2

	J_1	J_1
1	M_1/M_2	M_1/M_3
2	M_2/M_3	$M_1/M_2/M_3$

If no enabled transition is found in Step 5, the marking m represents a deadlock. Then, no next marking for the marking m would be generated in Step 6. Subsequently, Step 7 and Step 8 would not result in any effects. The algorithm then returns to Step 2 to explore another marking on the list OPEN, i.e., to explore another alternative path that could eventually lead to the final marking. In this way, a path that avoids the potential deadlocks can be found if one exists. If the markings are exhausted in Step 2, there is no path connecting the given initial and final markings.

Since the algorithm $L1$ finds a path between any given pair of the initial and the final markings, a schedule from any given state to any desired state can be generated by appropriately setting the initial and the final markings. Therefore, partial scheduling can be handled without any modification of the model. In this paper, the cost function $g(m)$ is the accumulation of time because the focus of this paper is the makespan. But other functions can be used for $g(m)$. For example, if the operating cost of machines is used for $g(m)$, a schedule with the optimal total operating cost can be obtained. Therefore, other performance criteria or multiple performance criteria can be adapted by using appropriate functions or combinations of functions for $g(m)$ and $h(m)$ in the scheduling algorithm.

IV. THE IMPLEMENTATION OF L1 AND SCHEDULING EXAMPLES

The algorithm $L1$ is implemented in C on a DECstation 5000/200. Reordering OPEN in Step 8 is actually combined with putting a marking on OPEN. Hence, OPEN is always in the increasing order of f. By picking up the first marking from OPEN in Step 3, $L1$ always selects the marking that has the smallest estimate of the cost of an optimal path.

In order to find enabled transitions in Step 5, the implementation of $L1$ identifies marked places. For each marked place, the program then identifies its output transitions. For each output transition, the program checks if all its input places are marked. If they are, the transition is enabled and used in Step 6 to generate the next marking. When generating the next marking, the program updates the marking and calculates the cost functions g and f for the next marking. The program also updates the remaining operation times of the places as discussed in Section III.

Example 2 The problem is to schedule an FMS which has three machines, M_1, M_2 and M_3. The system will handle two jobs, J_1 and J_2. Table III shows the job requirements. The operation times are shown in Table IV. The Petri net model of the system with the initial marking is shown in Fig. 7.

The scheduling problem given in Example 2 is solved by $L1$ using different h functions. $L1$ is executed using the following

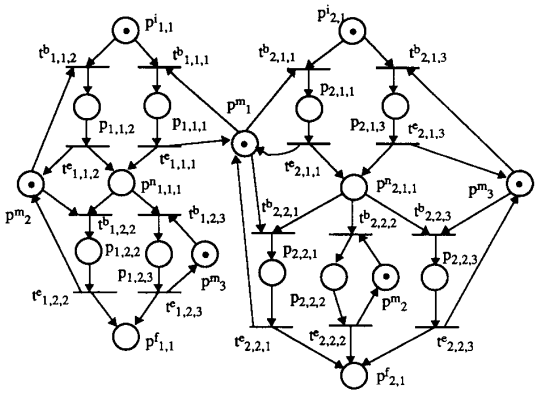


Fig. 7. The Petri net model of the system of Example 2.

TABLE IV
OPERATION TIMES OF EXAMPLE 2

Op.	Time
$O_{1,1,1}$	3
$O_{1,1,2}$	4
$O_{1,2,2}$	3
$O_{1,2,3}$	2
$O_{2,1,1}$	4
$O_{2,1,3}$	2
$O_{2,2,1}$	3
$O_{2,2,2}$	4
$O_{2,2,3}$	4

four h functions.

$$h_1(m) = 0 \text{ for all marking } m. \quad (1)$$

$$h_2(m) = -\ \text{dep}(m), \text{ where dep(m) is the depth of the marking } m \text{ in the reachability graph.} \quad (2)$$

$$h_3(m) = \min(rt_{1m}, rt_{2m}, \cdots, rt_{km}), \text{ where } rt_{im},\ i = 1, 2, \cdots, k \text{ is the remaining operation time of the place } p_{im} \text{ that has a token under the marking } m. \quad (3)$$

$$h_4(m) = h_2(m) + h_3(m) = \min(rt_{1m}, rt_{2m}, \cdots, rt_{km}) - \text{dep}(m). \quad (4)$$

$h_1(m)$ makes $f(m)$ equal to $g(m)$ and the program utilizes no heuristic information. It selects the next marking that has the smallest actual cost to reach the marking from the initial marking. This uninformed best-first search corresponds to the uniform-cost search and is guaranteed to find the optimal path [38].

$h_2(m)$ makes the program prefer markings that are deeper in the reachability graph, that is,

$$f(m) = g(m) + h(m) = g(m) + h_2(m) = g(m) - \text{dep}\ (m). \quad (5)$$

Hence, even if a marking m has a large cost, the cost is compensated if m is deep in the reachability graph. The compensation takes into account that a marking deeper in the reachability graph may be closer to the final marking.

TABLE V
THE EXECUTION RESULTS

	Number of iterations	Depth of final marking	Cost of solution path	Transition firing sequence
$h_1(m)$	50	8	6	$t^b2,1,3t^b1,1,1t^e2,1,3$ $t^e1,1,1t^b2,2,1t^b1,2,3$ $t^e1,2,3t^e2,2,1$
$h_2(m)$	13	8	6	$t^b2,1,3t^b1,1,1t^e2,1,3$ $t^b2,2,2t^e1,1,1t^b1,2,3$ $t^31,2,3t^2,2,2$
$h_3(m)$	54	8	6	$t^b2,1,3t^b1,1,2t^e2,1,3$ $t^b2,2,1t^e1,1,2t^b1,2,3$ $t^e2,2,1t^e1,2,3$
$h_4(m)$	53	8	6	$t^b2,1,3t^b1,1,1t^e2,1,3$ $t^e1,1,1t^b2,2,1t^b1,2,3$ $t^e1,2,3t^e2,2,1$

TABLE VI
THE SCHEDULE OF EXAMPLE 1 GENERATED USING $h_1(m)$

$h_1(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		10
Makespan of schedule		6

TABLE VII
THE SCHEDULE OF EXAMPLE 1 GENERATED USING h_2(m)

$h_1(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,2}$	4
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

$h_3(m)$ favors a marking which has an operation ending soon. Completing an operation makes resources available for other operations. $h_4(m)$ combines $h_2(m)$ and $h_3(m)$. Except for $h_1(m)$, the heuristic functions are intended to find a near optimal schedule in a shorter time. Table V shows the results of the executions employing the above four h functions.

The number of iterations is the number of times the program repeats its main routine. The depth of the final marking indicates the total number of transition firings to reach the final marking from the initial marking. The cost of the path is the makespan of the generated schedule. Since the exploration of a new marking requires an iteration, the number of iterations is always greater than or equal to the depth of the final marking. The transition firing sequence gives the schedule.

The interpretation of the schedule is straightforward from the Petri net model and is given in Table VI–IX. The operation sequence tells the order in which each operation should be initiated at the given machine.

In Table V, all the schedules generated have the same total cost, or makespan, of 6. In Table VI–IX, the sequential sum of operation times of each schedule is 10 or 11. This demonstrates that each schedule fosters concurrency of operations.

TABLE VIII
THE SCHEDULE OF EXAMPLE 1 GENERATED USING h_3(m)

$h_3(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,2}$	4
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

TABLE IX
THE SCHEDULE OF EXAMPLE 1 GENERATED USING h_4(m)

$h_4(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		10
Makespan of schedule		6

TABLE X
THE JOB REQUIREMENTS OF EXAMPLE 3

	J_1	J_2	J_3	J_4	J_5
1	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	M_2/M_3	M_1/M_3
2	M_2	M_3	M_2/M_3	M_1/M_3	M_2/M_3
3	M_1/M_3	M_1/M_2	M_1/M_3	M_2/M_3	M_1/M_2
4	M_1/M_2	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	$M_1/M_2/M_3$

Notice $h_2(m)$ reduced the number of iterations quite significantly, i.e., 74% less than the case of $h_1(m)$, i.e., the uninformed best-first search. This means that the program generated a schedule with the same makespan in a much shorter time using function $h_2(m)$. This suggests the value of heuristic function $h_2(m)$. On the other hand, $h_3(m)$ slightly increased the number of iterations to generate a schedule with the same makespan. This shows that only favoring an operation ending soon, in fact, suggests little heuristic value in a resource-sharing interactive environment. Notice that $h_3(m)$ dominates over $h_2(m)$ as shown in the result of the case of $h_4(m)$.

Example 3: The problem is to schedule an FMS with three multipurpose machines, M_1, M_2, and M_3. There are five jobs, J_1, J_2, J_3, J_4, and J_5 that have four processes each. Each job has a lot size of 10. Table X shows the job requirements. Operation times are given in Table XI.

The Petri net model of the system is decomposed according to each job and the part for J_1 with the initial marking is shown in Fig. 8. The parts for J_2, J_3, J_4, and J_5 can be similarly constructed using the method described in Section II. The places with the same place notations are actually the same places and are drawn separately for clear presentation. The Petri net model of the system has a total of 69 places and 82 transitions. The number inside a place represents the number of tokens.

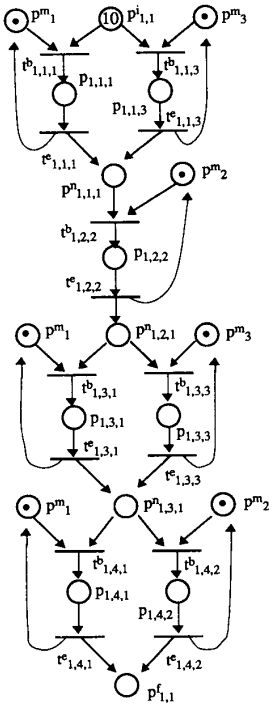


Fig. 8. The model of J_1 of Example 3.

TABLE XI
THE OPERATION TIMES OF EXAMPLE3.

Operation	Time	Operation	Time
$O_{1,1,1}$	7	$O_{3,4,1}$	6
$O_{1,1,3}$	4	$O_{3,4,2}$	3
$O_{1,2,2}$	3	$O_{4,1,2}$	9
$O_{1,3,1}$	3	$O_{4,1,3}$	5
$O_{1,3,3}$	6	$O_{4,2,1}$	6
$O_{1,4,1}$	2	$O_{4,2,3}$	2
$O_{1,4,2}$	4	$O_{4,3,2}$	7
$O_{2,1,1}$	8	$O_{4,3,3}$	12
$O_{2,1,2}$	12	$O_{4,4,1}$	9
$O_{2,2,3}$	4	$O_{4,4,2}$	6
$O_{2,3,1}$	7	$O_{4,4,3}$	3
$O_{2,3,2}$	14	$O_{5,1,1}$	10
$O_{2,4,1}$	8	$O_{5,1,3}$	15
$O_{2,4,3}$	4	$O_{5,2,2}$	7
$O_{3,1,1}$	10	$O_{5,2,3}$	14
$O_{3,1,2}$	15	$O_{5,3,1}$	5
$O_{3,1,3}$	8	$O_{5,3,2}$	8
$O_{3,2,2}$	2	$O_{5,4,1}$	4
$O_{3,2,3}$	6	$O_{5,4,2}$	6
$O_{3,3,1}$	2	$O_{5,4,3}$	8
$O_{3,3,3}$	4		

The problem of Example 3 is solved using the following heuristic function.

$$h(m) = -w \bullet \text{dep}(m), \text{ where w is a weight factor.} \quad (6)$$

This heuristic function compensates the cost of the marking m by the weighted depth of the marking. A marking with a

TABLE XII
THE EXECUTION RESULTS OF EXAMPLE 3

w	Number of iterations	Depth of final marking	Cost of solution path
2	758	400	426
5	412	400	447
10	401	400	439
20	401	400	439
50	401	400	439

TABLE XIII
THE REQUIREMENTS OF J_1 TO J_4 OF EXAMPLE 4

	J_1	J_2	J_3	J_4
1	M_1R_2/M_3	M_2R_2/M_5R_1	$M_2R_1/M_4R_1/M_5$	M_2R_2/M_3
2	M_2R_1/M_5R_1	$M_3R_2/M-4$	$M_2R_3/M_3R_1/M_4R_1$	M_3R_1/M_5R_2
3	M_2R_3/M_4	M_1R_2/M_5R_3	M_1R_3	M_4R_2
4	NA	$M_1/M_3R_1/M_5$	M_2R_1	NA
5	NA	NA	M_4/M_5R_2	NA

bigger depth is hypothesized to be nearer to the final marking. The magnitude of the cost of the marking is related to the magnitude of operation times, the lot sizes, and the number of jobs and operations. Therefore, an adequate value of the weight should reflect the mentioned factors.

Table XII shows the results of executions using different values for w. When the value of w was less than 2, the program did not find a path in a reasonable amount of time. This was because the computer used spent most of the time swapping memory. The computer used, a DECstation 5000/200, has 16 megabytes of RAM. The execution was killed after running in the background a few days. When the program found a solution, it took about 5 to 10 minutes. When the value of w changes from 2 to 5, the number of iterations significantly reduces but the makespan of the solution increases. When the value of w changes from 5 to 10 or greater, the number of iterations reduces a little and the makespan of the solution decreases slightly. The results suggest that the program performs desirably when the value of w is 5 or greater since the number of iterations is close to its lower bound, 400, and the increase of the cost of the path is relatively small.

Example 4 The problem is to schedule an FMS which has multipurpose machines, M_1, M_2, M_3, M_4, M_5, and robots, R_1, R_2, R_3. There are 10 jobs, $J_k, k = 1, \cdots, 10$, each with varied number of processes. Tables XIII, XIV, and XV show the job requirements. The operation times are shown in Table XVI. The Petri net model is constructed using the method discussed in Section II. The Petri net model has a total of 154 places and 182 transitions.

The heuristic function of (6) is used to solve the problem. Table XVII shows the execution results when the lot size is fixed at 5 for each job. Next, an experiment with varied lot sizes is conducted. Table XVIII shows the lot size of each job and the execution results are shown in Table XIX. In Table XVII and Table XIX, the program did not find a path in a reasonable amount of time when w is less than 2 and 1.5, respectively.

TABLE XIV
THE REQUIREMENTS OF J_5 TO J_8 OF EXAMPLE 4

	J_5	J_6	J_7	J_8
1	M_2R_1/M_3R_2	M_1R_3/M_3R_2	M_2R_2/M_4R_1	$M_2R_2/M_3/M_4$
2	M_5	M_4R_1/M_3	M_3/M_5	M_4R_3/M_5R_1
3	M_1/M_3R_1	M_3R_3/M_5R_2	NA	$M_2/M_3/M_4$
4	M_2R_1/M_4R_1	$M_1/M_2/M_3$	NA	M_5
5	M_4R_1	M_2R_1/M_4	NA	M_2R_3/M_4R_1
6	NA	$M_2R_1/M_3/M_5$	NA	$M_1/M_2/M_3R_3$
7	NA	NA	NA	$M_1/M_4/M_5R_1$
8	NA	NA	NA	M_2R_1/M_3R_2

TABLE XV
THE REQUIREMENTS OF J_9 AND J_10 OF EXAMPLE 4

	J_9	J_{10}
1	$M_2R_2/M_3/M_5$	M_5
2	M_2	M_1/M_2
3	M_1R_1/M_4	M_5
4	M_4R_1/M_5	M_2R_1/M_3
5	M_1/M_3	NA

TABLE XVI
THE OPERATION TIMES OF EXAMPLE 4

Op.	Time	Op.	Time	Op.	Time	Op.	Time
$O_{1,1,1}$	3	$O_{3,5,4}$	7	$O_{8,8,3}$	7	$O_{8,6,3}$	4
$O_{1,1,3}$	5	$O_{3,5,5}$	5	$O_{6,5,2}$	4	$O_{8,7,1}$	9
$O_{1,2,2}$	4	$O_{4,1,2}$	4	$O_{6,5,4}$	5	$O_{8,7,4}$	8
$O_{1,2,5}$	5	$O_{4,1,3}$	6	$O_{6,6,2}$	3	$O_{8,7,5}$	7
$O_{1,3,2}$	2	$O_{4,2,3}$	7	$O_{6,6,3}$	7	$O_{8,8,2}$	5
$O_{1,3,4}$	3	$O_{4,2,5}$	8	$O_{6,6,5}$	8	$O_{8,8,3}$	6
$O_{2,1,2}$	4	$O_{4,3,4}$	5	$O_{7,1,2}$	8	$O_{9,1,2}$	3
$O_{2,1,5}$	4	$O_{5,1,2}$	4	$O_{7,1,4}$	8	$O_{9,1,3}$	4
$O_{2,2,3}$	4	$O_{5,1,3}$	3	$O_{7,2,3}$	9	$O_{9,1,5}$	4
$O_{2,2,4}$	5	$O_{5,2,5}$	6	$O_{7,2,5}$	7	$O_{9,2,2}$	6
$O_{2,3,1}$	8	$O_{5,3,1}$	9	$O_{8,1,2}$	5	$O_{9,3,1}$	5
$O_{2,3,5}$	7	$O_{5,3,3}$	7	$O_{8,1,3}$	7	$O_{9,3,4}$	7
$O_{2,4,1}$	6	$O_{5,4,2}$	10	$O_{8,1,4}$	8	$O_{9,4,4}$	2
$O_{2,4,3}$	3	$O_{5,4,4}$	9	$O_{8,2,4}$	10	$O_{9,4,5}$	4
$O_{2,4,5}$	5	$O_{5,5,4}$	4	$O_{8,2,5}$	10	$O_{9,5,1}$	5
$O_{3,1,2}$	3	$O_{6,1,1}$	6	$O_{8,3,2}$	4	$O_{9,5,3}$	7
$O_{3,1,4}$	4	$O_{6,1,3}$	6	$O_{8,3,3}$	5	$O_{10,1,5}$	4
$O_{3,1,5}$	5	$O_{6,2,3}$	7	$O_{8,3,4}$	6	$O_{10,2,1}$	5
$O_{3,2,2}$	5	$O_{6,2,4}$	5	$O_{8,4,5}$	7	$O_{10,2,2}$	6
$O_{3,2,3}$	5	$O_{6,3,3}$	4	$O_{8,5,2}$	4	$O_{10,3,5}$	7
$O_{3,2,4}$	5	$O_{6,3,5}$	4	$O_{8,5,4}$	3	$O_{10,4,2}$	2
$O_{3,3,1}$	6	$O_{6,4,1}$	7	$O_{8,6,1}$	5	$O_{10,4,3}$	3
$O_{3,4,2}$	3	$O_{6,4,2}$	8	$O_{8,6,2}$	5		

From the results shown in Table XVII and Table XIX, the program performs desirably when the weight w is greater than or equal to 5. There is little room for improvement in terms of reducing the number of iterations after w reaches 5. A strong correlation between the number of iterations and the weight w is suggested especially from Table XIX.

V. CONCLUSION

The new scheduling method presented in this paper formulates a scheduling problem with a timed place Petri net

TABLE XVII
THE EXECUTION RESULTS OF EXAMPLE 4 WITH LOT SIZE OF 5

w	Number of iterations	Depth of final marking	Cost of solution path
2	465	450	304
3	465	450	304
4	453	450	304
5	451	450	298
10	451	450	298
100	451	450	298

TABLE XVIII
THE LOT SIZES FOR THE SECOND EXPERIMENT OF EXAMPLE 4

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Lot size	5	6	4	6	4	5	7	3	5	5

TABLE XIX
THE EXECUTION RESULTS OF EXAMPLE 4 WITH VARIED LOT SIZES

w	Number of iterations	Depth of final marking	Cost of solution path
1.5	1121	420	273
1.51	1012	420	273
1.52	1012	420	273
1.55	885	420	273
1.6	778	420	273
2	499	420	281
5	421	420	293
10	421	420	293

model. Once a complete model of the system is constructed, the scheduling algorithm uses this Petri net model to search a partial reachability graph. Depending on the heuristic functions used, the scheduling algorithm finds a globally optimal or near optimal feasible schedule in terms of the firing sequence of the transitions of the Petri net model.

The formulation explicitly and easily handles the important characteristics of flexible manufacturing systems, such as routing flexibility, shared resources, concurrency and lot sizes, and also facilitates maintaining the coherency of problem models. By setting the initial and the final markings appropriately, partial scheduling can be handled without any modification of the model. By using appropriate functions or combinations of functions in the scheduling algorithm, other performance criteria or multiple performance criteria can be adapted.

Since a complete Petri net model of a system is obtained during the problem formulation, the generated schedule in terms of the firing sequence of the transitions can be used directly by employing Petri net controllers in the supervisory control of the modeled system. Once a schedule to reach a desired final state is generated, potential deadlocks can be avoided by following the schedule. Hence the analytical overhead to guarantee the liveness of the model and the system is eliminated.

An experimental study of the efficiency of some heuristic functions is conducted. The function of (6) with properly tuned weight w is a potentially good heuristic function. Further research is under way to explore other heuristic functions which can be used to reduce the search. In this paper, the material transfer among machines is not explicitly discussed but the flow of parts is also important to scheduling. Automated guided vehicle systems (AGVS's) are increasingly used for the material handling. Further research is under way to handle collectively the AGVS's and the part processing facilities such as machines and robots [27], [28].

ACKNOWLEDGMENT

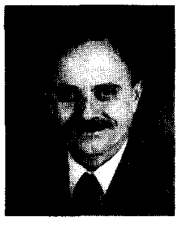
The authors are very grateful to the three anonymous reviewers for their important and constructive comments.

REFERENCES

[1] S. Ahmad and B. Li, "Robot control computation in microprocessor systems with multiple arithmetic processors using a modified DF/IHS scheduling algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1167–1178, 1989.
[2] M. Aicardi, A. D. Febbraro, and R. Minciardi, "Analysis of deterministic discrete event systems via minimax algebra," in *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville, VA, Oct 13–16, 1991, pp. 321–328.
[3] R. Y. Al-Jaar and A. A. Desrochers, "Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 621–639, 1990.
[4] K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, 1974.
[5] E. Bowman, "The schedule-sequencing problem," *Operations Research*, vol. 7, no. pp. 621–624, 1959.
[6] P. J. Brucker, "Scheduling problems in connection with flexible production systems," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 1778–1783.
[7] J. Carlier and P. Chretienne, "Timed Petri net schedules," in *Advances in Petri Nets*, Rozenberg, Ed., Berlin: Springer-Verlag, 1988, pp. 62–84.
[8] C. L. P. Chen, "Time lower bound for manufacturing aggregate scheduling problems," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 830–835.
[9] R. Conway, W. Maxwell, and L. Miller, *Theory of Scheduling*. Reading, MA: Addison-Wesley, 1967.
[10] E. Falkenauer and S. Bouffouix, "A genetic algorithm for job shop," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 824–829.
[11] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood, 1982.
[12] S. B. Gershwin and R. R. Hildebrant, et al., "A control perspective on recent trends in manufacturing systems," *IEEE Control Systems Magazine*, vol. 6, no. 2, pp. 3–15, 1986.
[13] H. P. Hillion and J. Proth, "Performance evaluation of job-shop systems using timed event-graphs," *IEEE Transactions on Automatic Control*, vol. 34, no. 1, pp. 3–9, 1989.
[14] D. J. Hoitomt, J. B. Perkins. and P. B. Luh, "Distributed scheduling of job shops," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 1067–1072.
[15] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A Lagrangian relaxation approach to job shop scheduling problems," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, May 1990, pp. 1944–1949.
[16] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "Job shop scheduling," in *Proceedings of the First International Conference on Automation Technology*, Taipei, Taiwan, July 1990, pp. 565–574.
[17] M. A. Holliday and M. K. Vernon, "A generalized timed Petri net model for performance analysis," *Proceedings of the IEEE International Workshop on Timed Petri Nets*, Torino, Italy, July 1–3, 1985, pp. 181–190.
[18] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Transactions on Automatic Control*, vol. 35, no. 5, pp. 514–523, May 1990.
[19] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. Rockville, MD: Computer Science Press, 1978.
[20] E. S. H. Hou and H. Y. Li, "Task scheduling for flexible manufacturing systems based on genetic algorithms," in *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1991, pp. 397–402.

[21] C. Q. Jiang, M. G. Singh and K. S. Hindi, "Optimized routing in flexible manufacturing systems with blocking," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 589–595, 1991.
[22] P. Kapasouris and D. Serfaty, *et al.*, "Resource allocation and performance evaluation in large human-machine organizations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 521–532, 1991.
[23] R. Larson and A. Odoni, *Urban Operations Research*. Englewood Cliffs, NJ: Prentice-Hall, 1981, pp. 360–361.
[24] A. E. J. Lee, "Integrated tooling and scheduling of flexible machines: Theory and algorithms," Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, May 1989.
[25] D. Y. Lee and F. DiCesare, "FMS scheduling using Petri nets and heuristic search," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 1057–1062.
[26] D. Y. Lee and F. DiCesare, "Experimental study of a heuristic function for FMS scheduling," in *Proceedings of the 1992 Japan-USA Symposium on Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 1171–1177.
[27] D. Y. Lee and F. DiCesare, "Integrated models for scheduling flexible manufacturing systems," in *Proceedings of the 1993 IEEE Intl. Conf. on Robot. & Auto.*, Atlanta, Georgia, May 2–7, 1993, pp. 827–832.
[28] D. Y. Lee and F. DiCesare, "Scheduling of flexible manufacturing systems employing automated guided vehicles," in *Proceedings of the 9th International Conference on CAD/CAM, Robotics, and Factories of the Future*, Newark, New Jersey, USA, August 17–20, 1993, in press.
[29] P. S. Liu and L. C. Fu, "Planning and scheduling in a flexible manufacturing system using a dynamic routing method for automated guided vehicles," in *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 1584–1589.
[30] Z. P. Lo and B. Bavarian, "Job scheduling on parallel machines using simulated annealing," *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1991, pp. 391–396.
[31] P. B. Luh and D. J. Hoitomt, *et al.*, "Schedule generation and reconfiguration for parallel machines," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 687–696, Dec. 1990.
[32] A. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. pp. 219–223, 1960.
[33] P. Mellor, "A review of job shop scheduling," *Operational Research Quarterly*, vol. 17, no. 2, pp. 161–171, June 1966.
[34] N. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980.
[35] J. O'Brien, *Scheduling Handbook*. New York: McGraw-Hill, 1969.
[36] K. Onaga, M. Silva and T. Watanabe, "On periodic schedules for deterministically timed Petri net systems," in *Proceedings of the 4th International Workshop on Petri Nets and Performance Models*, Melbourne, Australia, Dec. 2–5, 1991, pp. 210–215.
[37] T. A. Owens and P. B. Luh, "A job completion time estimation method for work center scheduling," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 110–115.
[38] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley, 1984.
[39] C. Ramamoorthy and G. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Transactions on Software Engineering*, vol. 6, no. 5, pp. 440–449, 1980.
[40] J. Rickel, "Issues in the design of scheduling systems," in *Expert Systems and Intelligent Manufacturing*, Oliff, Ed. Elsevier, 1988, pp. 70–89.
[41] F. Rodammer and J. K. White, "A recent survey of production scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 6, pp. 841–851, 1988.
[42] R. V. Rogers and K. P. White, "Algebraic, mathematical programming, and network models of the deterministic job-shop scheduling problem," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 693–697, 1991.
[43] R. Sengupta and S. Lafortune, "Optimal control of a class of discrete event systems," *Proceedings of the IFAC International Symposium on Distributed Intelligence Systems*, Arlington, VA, Aug 13–15, 1991, pp. 25–30.
[44] L. Shen and Q. Chen, *et al.*, "Truncation of Petri net models of scheduling problems for optimum solutions," *Proceedings of the Japan/USA Symposium on Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 1681–1688.
[45] H. Shih and T. Sekiguchi, "A timed Petri net and beam search based on-line FMS scheduling system with routing flexibility," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 2548–2553.
[46] J. Sifakis, "Performance evaluation of systems using nets," *Net Theory and Applications*. Brauer, Ed. Berlin: Springer-Verlag, 1980, pp. 307–319.
[47] M. Silva and R. Valette, "Petri nets and flexible manufacturing," *Advances in Petri Nets*. Berlin: Springer-Verlag, 1989, pp. 374–417.
[48] A. S. Spachis and J. R. King, "Job-shop scheduling heuristics with local neighbourhood search," *International Journal of Production Research*, vol. 17, no. 6, pp. 507–526, 1979.
[49] A. Vasquez and P. B. Mirchandani, "Concurrent resource allocation for production scheduling," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 1060–1066.
[50] N. Viswanadham and Y. Narahari, "Stochastic Petri net models for performance evaluation of automated manufacturing systems," *Information and Decision Technologies*, vol. 14, no. pp. 125–142, 1988.
[51] D. Zhang, "Planning using timed Pr/T nets," in *Proceedings of the Japan/USA Symposium on Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 1179–1183.
[52] Y. Zhang, "Solution to job-shop scheduling of FMS by neural networks," in *Proceedings of the 1991 IFAC Workshop on Discrete Event System Theory and Applications in Manufacturing and Social Phenomena*, Shenyang, China, June 25–27, 1991, pp. 261–266.

Doo Yong Lee (S'90–M'93) earned the B.S. degree from the Department of Control and Instrumentation Engineering, Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degrees from the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. He is currently a postdoctoral Research Associate at Information Technology Services, Rennselaer Polytechnic Institute. His research interests include the modeling, analysis, and control of discrete event systems; Petri net theory and application; scheduling and supervisory control of flexible manufacturing systems and automated guided vehicle systems; and multimedia theory and application. He received the Charles M. Close Doctoral Prize from Rensselaer Polytechnic Institute in 1993. He is a member of the IEEE Robotics and Automation Society and the IEEE Systems, Man, and Cybernetics Society.



Frank DiCesare (S'60–M'62) received the B.S. and M.S. degrees in Electrical Engineering from Northwestern University, Boston, MA, in 1960 and 1962, respectively.

In 1969 he joined the Rensselaer Polytechnic Institute, Troy, NY, where he is presently a Professor of Electrical, Computer, and Systems Engineering. His current research focus and graduate-level teaching is in the area of modeling, analysis, and control of discrete event systems, with application to manufacturing and intelligent vehicular highway systems. At the undergraduate level he teaches courses in microprocessor systems and embedded control. From 1956 to 1960 he worked as a Cooperative Student Engineer at the M.I.T. Instrumentation Laboratory, Cambridge, MA, and from 1960 to 1962 he was a graduate Research and Teaching Assistant at Northeastern University. As an officer in the U.S. Army Signal Corps, he served from 1962 to 1964 as an Instructor and Course Director at the Ordinance Guided Missile School, Redstone Arsenal, Huntsville, AL. Following this, he spent 1964–1969 at Carnegie-Mellon University as an Instructor in Electrical Engineering and as a Research Engineer in the Transportation Research Institute. He has consulted on information and control systems for numerous companies and agencies at all levels of government and the private sector.

He was a recipient of the 1965 Koppers Fellowship, received the IEEE Centennial Award and Medal in 1984, was cited in the 1987 LEAD Award to Rensselaer Polytechnic Institute by the Society of Manufacturing Engineers, and received the Franklin V. Taylor Award from the IEEE Systems, Man, and Cybernetics Society in 1993. He is very active in the IEEE, both in publication and conferences, and currently serves as Vice President of Long-Range Planning and Finance of the Systems, Man, and Cybernetics Society, and is Co-Chair of the 1994 SMC Conference in San Antonio, TX.