Petri-Net Based Modeling and Scheduling of a Flexibile Manufacturing System

C. W. Cheng, T. H. Sun and L. C. Fu

Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan, R.O.C.

Abstract

In this paper, a timed place Petri-net (TPPN) model for flexible manufacturing systems with the components of machines, limited buffers, robots and the material handling systems, Automated Guided Vehicles (AGV's) is constructed. Since a firing sequence of the TPPN from the initial marking to the final marking can be seen as a schedule of the modeled FMS, by using an A based search algorithm, namely, Limited-Expansion A algorithm, a near optimal schedule of the part processing can be obtained using reasonable computing time and memory requirement. For large volume of parts, we also propose an adaptive scheduling approach to generate a near-optimal schedule in an economical computing time. In order to show the effectiveness of the proposed method, a prototype FMS in Automation Lab. of Department of Mechanical Engineering, National Taiwan University, is used as a target system for implementation.*

1 Introduction

A flexible manufacturing system (FMS) is an integrated, computer-controlled configuration of automated material handling devices and numerically controlled (NC) machines that can simultaneously process medium-sized volumes of a variety of part types. Such systems combine the advantages of very flexible but inefficient manual job shops with those of highly productive but rigid transfer lines. This new production technology has been designed to attain the efficiency of well-balanced, machine-paced transfer lines, while utilizing the flexibility to simultaneously machine multiple part types.

Despite that the part transportation system provides us flexibility and economy, but it also leaves us numerous ways of routing which unfortunately gives the control problem a combinatorial flavor. Since the environment of an FMS is dynamic and complicated, the most important problems of an FMS is how to assign the given resources to different processes, required in making each product, to achieve the best efficiency.

The production scheduling concerns the efficient allocation of resources over time for manufacturing products. The objective of scheduling is to find a way to assign and sequence the use of these shared resources such that production constraints are satisfied and production costs are minimized. Note that the required ordering of operations within each job (the technological sequence) must be preserved [1, 2, 3, 4]. Production scheduling problems are very complex and it has been proved to be an NP-hard problem [4].

There are various kinds of model of a manufacturing system and various techinques to solve its scheduling problem. The former contains, e.g., network model, mathematical programming model, and finite-state machine model, whereas the latter includes simulation, queuing theory approach, mathematical programming, and heuristic algorithms. But the methods of queueing theory, heuristic algorithm, simulation cannot obtain an exact solution or the solution may far from being optimal. The mathematical programming techniques can offer an exact solution to scheduling problem, but it is really difficult to formulate the optimization problem and to solve it.

Petri-nets are useful tools for the modeling and analysis of a production system. It can provide accurate models of the precedence relations and concurrent, asynchronous events [6]. In this paper, a Petri-net model is built to model the detailed behavior of an FMS and a schedule which is generated based on that Petri-net model is to optimize some a priori assigned performance criterion. Different from the result obtained in [5] , the present work introduces a complete modeling of a general FMS, especially, including AGV transportation system, and suggests a heuristic search algorithm to realistically obtain a near optimal schedule.

Here, we select the timed place Petri-net (TPPN), in which time is associated only with places and all transitions are instantaneous, to model our manufacturing systems. Because the markings of the TPPN are deterministic during the evolution of the firing sequence from initial marking, we can undoubtedly use the markings of the TPPN to describe the states of the system and all the reachable markings can represent the state space of the modeled system. Then, the state-space search method can be applied to obtain an optimal or a near-optimal path of markings and the firing sequences from the initial marking to the final marking can be seen as a schedule of the modeled system.

The organization of this paper is as follows. Section 2 introduces the modeling techinque proposed in this paper. Section 3 presents the limited-expansion A

1050-4729/94 $03.00 © 1994 IEEE

algorithm for searching a near-optimal schedule based on the Petri-net model. Section 4 suggests the implementation on a real prototype FMS in Automatic Lab. of Department of Mechanical Engineering , National Taiwan University. Section 5 is conclusion.

2 Petri-Net Modeling

The problem which we want to solve here is a job shop scheduling problem in flexible manufacturing systems. The job shop consists of a number of processing centers called machines, which are capable of performing multiple types of operations. A job is an ordered set of operations and the ordering is given in precedence relationship. Each job may be routed through alternative machines to perform its required operations.

In this paper, a timed place Petri-net (TPPN) model for flexible manufacturing systems with the components of machines, limited buffers, robots and the material handling systems, Automated Guided Vehicles (AGV's) is constructed. The TPPN model contains two major sub-models. One is called Transportation Model which is stationary , and the other is called Process-Flow Model which may be variable.

The objective of the Transportation Model is to model the behavior of the AGV traveling from the current stop to its destination stop, and that of the Process-Flow Model is to describe the behavior of the part routing and resource assignment. The two sub-models, of course, are interacted with to each other to take up the necessary actions in response to the triggering from another.

In our timed place Petri-net model, the places can be classified into four categories listed below:

Resource Places are used to model the production resources like transportation carrier, machines, loading/unloading station, and the control right of AGV stops in the transportation system. If a resource place is marked, the corresponded resource is free and available.

Operation Places are used to represent the status of usage of resources. Of course, the processing time must be assigned to be associated with the operation places. A token at an operation place represents that a specific operation is being performed.

Intermediate Places are used to model the flow of processing of each job or the movement of AGV's. If an intermediate place is marked, it indicates that the last operation of the part or the last traveling of the AGV is accomplished and is ready for nest operation.

Control Places are used to represent the signals or conditions that are sent to or are received form the other sub-models to indicate some events have occurred. Control places represent the interface between the two basic sub-models: the Transportation Model and the Process-Flow Model.

The Transportation Model and the Process-Flow Model use the places described above to construct its Petri-net model. Generally, the resource places are used to model all shared resources and operation places are associated with time to model the time-related operations. However, the intermediate places are used to model some situations or some place without meaning but to connect two transitions; and control places play a role as an interface between the system sub-models.

2.1 Modeling of a Transportation System

The Transportation Model can be divided into some units by its characteristics. One is Transportation Layout Unit model and the other is Route Control Unit model. However, the Push Control Unit model has to be included if the number of the material handling carriers is greater than one, i.e., multiple AGV system.

Transportation Layout Unit Model

The purpose of the Transportation Layout Unit model is to model the layout of the transportation system. The basic concept of this model is described as follows. When an AGV needs to move from the current stop to the next adjacent stop, it needs to receive a 'ticket' of movement first indicating the destination and then it acquires the control right of the next adjacent stop to make sure that the destination stop is free at the moment. If both of these conditions are satisfied, it can start its traveling to the next adjacent stop. At the same time, the control right of the current stop will be freed to allow another AGV to use it as a destination or pass-through stop.

Route Control Unit Model

The Route Control Unit Model is used to illustrate the decision-making Petri-net model for AGV routing. Each time when an AGV wants to move from the current stop to some other stop, it must determine its route of movement first. The determined route can be sent to the Transportation Layout Unit model through 'ticket' places $TKij$ to entail the AGV to move along that route. Therefore, for each stop, we need to have one corresponding Petri-net model to take care the routing decision, guiding the traveling path from other stops to it.

Push Control Unit Model

In a production system, the material handling system may contain more than one carriers which transport materials between each pair of workstations. However, the collision problems of carriers due to multiple AGV's must be considered and solved. The objective of the Push Control Unit model is to guarantee the collision-free condition between carriers.

The method we provide is a way called *push-AGV* strategy. The basic idea of the push-AGV strategy is described as follows. When one traveling AGV find that a stop which it will pass through is occupied by another freed AGV, it will send a push command to ask that freed AGV to leave that stop. After a 'ticket' is sent, that freed AGV can move to its next adjacent stop and release its current occupied stop. When

the stop is released, the waiting AGV can resume its traveling.

Note that the movement of AGV is stop-to-stop so that the push command will be issued only when the stop which the AGV is heading now is occupied by another freed AGV. Because the push command may be sent to any other AGV, we must construct models for each pair of AGV's to enforce such strategy. However, the effectiveness of this strategy is limited by the layout and the number of AGV. When the number of AGV is growing, the possibility of issuing push commands will be quite high so that the performance of system will decline and the system may incur AGV deadlocks .

2.2 Modeling of a Process Flow

A process flow of manufacturing can be seen as a sequence of *part transportations* by material handling system between two workstations and *part processings* on numerically controlled (NC) machines. The Petri-net model we proposed to illustrate the processing flow of each part type is Process-Flow Model, which models the technological precedence constraints for processing parts and provides normally more than one alternative routes to accomplish the processing.

Therefore, the Process-Flow Model is composed of two major components in a complete Petri-net model. One is the *part transportation unit*, which is used to model the AGV-call request that asks an AGV to move to its current workstation, to load the part onto this AGV, to transport the part by the AGV to its destination stop, and finally to unload the part onto the machine for its next processing. The other one is the *part processing unit*. It is used to model the machine processing. It contains the assignment of local buffers for input and output as well as of operation time for machine processing.

To demonstrate the capability of the above proposed modeling method, we apply it to model the prototype FMS in Automation Lab. of Department of Machanical Engineering, National Taiwan University. Its layout is shown in Fig. 1. Due to shortage of space, here we only persent the final Petri-Net Model shown in Fig. 2, but neglect all the modeling procedure.

3 Near-Optimal Scheduling Method

Once the timed place Petri-net model of an FMS is constructed using the modeling approach described in previous section, the evolution of the system can be described by the changes of marking of the Petri-net. Since the timed place Petri-net model can represent in routing flexibility, shared resources, and various lot sizes, as well as concurrency and precedence constraints , all the possible system behavior described here can be completely tracked down within the reachable markings of the Petri-net.

The general job-shop scheduling problem has been shown to be NP-complete. Therefore, we resort the heuristic search algorithm to solve this problems. The A^* algorithm is known as a heuristic best-first search procedure and guarantees to reach the optimal solution if an approiate cost function is incorporated. However, this would also mean high memory requirement, to store all the nodes generated by the best-first search, and exponential time with respect to the problem size. To avoid these drawbacks, an algorithm, called *Limited-Expansion A algorithm*, modified from A^* algorithm, is used. The relation between this algorithm and $A*$ algorithm is analogous to the one between the beam search and the breadth-first search.

The idea of limited-expansion A algorithm is similar to that of staged search [7]. i.e. , when the number of nodes in $OPEN$ list exceeds a given amount, pruning takes place and only a specified number of the 'best' nodes (of minimum estimated cost)are kept for further processing. Our approach, limited-expansion A algorithm, assumes that the $OPEN$ list only has a given maximum capacity b. At any step, at most b best nodes are kept on $OPEN$ list.

By Modifying the general A^* algorithm, we can construct the following Limited-Expansion A algorithm for non-delay scheduling:

Limited-Expansion A Algorithm for Non-delay Scheduling

Step 1: Place initial marking M_0 on the list $OPEN$.

Step 2: If $OPEN$ is empty, terminate with failure.

Step 3: Choose a marking M from $OPEN$ with minimal cost $f(M)$ and move it from $OPEN$ to $CLOSE$.

Step 4: If M is the final marking, construct the searched path from the initial marking to the final marking and terminate.

Step 5: Generate the successor markings for each enabled transition, and set pointers from the successors to M.

Step 6: For each successor marking M', compute its cost $f(M')$ and do the following:

1. If marking M' is not already on $OPEN$ or $CLOSE$, then Put M' on $OPEN$.
2. Else if marking M' is already on $OPEN$ and a shorter path is found, then direct its pointer along the current path.
3. Else if marking M' is already on $CLOSE$ and a shorter path is found, then direct its pointer along the current path and move M' from $CLOSE$ to $OPEN$.

Step 7: If there are more than b markings on $OPEN$, truncate the marking M_k from $OPEN$ with maximum cost $f(M_k)$. Go to Step 7.

Step 8: Go to Step 2.

Obviously, by using this approach it is not guaranteed that the algorithm will find the optimal schedule, even if an admissible heuristic cost estimate is used. One could see *limited-expansion A* as a relaxed version of A^*. Assuming that the capacity of the $OPEN$ list

in *limited-expansion A* algorithm is b, we can see that *limited-expansion A* becomes A^* when $b \rightarrow \infty$.

For a problem with n operations in total, no more than bn nodes will be expanded in the worst case. The worst-case complexity of this algorithm in terms of the expanded nodes will then be $O(bn)$, if node cost evaluation is considered as the main source of complexity. Therefore, we can say that *limited-expansion A* algorithm trade the schedule optimality for reduced memory requirement and lower algorithm's complexity.

Adaptive Scheduling

In some cases, there may have high-volume shops in production systems. High-volume shops are understood here as models where the total number of parts to be processed is very high, but the total number of different types is low when compared with the total volume of parts (low-variety systems). However, scheduling method proposed above may lead to a situation where we must limit the OPEN list capacity more or find another cost-estimated function, h, that may be non-admissible to guide the search to reach the final goal during the process of Limited-Expansion A search. But this will make the obtained schedule much worse if the computing time.

Because the number of parts to be processed is very high, the use of a search technique in generation of a schedule for the complete set is a heavy computational task due to its NP-complete nature. Moreover, a complete schedule for the jobs is not necessarily flexible, since unexpected events may occur to reduce the effectiveness of the original schedule . This consideration motivates a scheduling strategy which not only provides effective schedules but also is able to deal with the changing environment of production systems, i.e., schedulers needs to be *accurate*, producing optimal or near-optimal schedules, and *responsive*, able to react to changes timely, such as machine failures, variations in demand, etc.

In this paper, we combine this concept to our scheduling method to deal with high-volume-shop manufacturing problems. Not only we obtain a near-optimal schedule due to the adaptive scheduling concept just discussed, but also we retain the rescheduling to react to the changes of the system environment due to the characteristics of Petri-net. For example, rescheduling is straightforward when a new initial marking is given. And we can reduce the number of token in a place to represent that a machine has been breakdown or change the token number of parts to reflect the change of customer demand.

4 Implementation and Scheduling Results

We have shown, in the previous section, that an FMS can be modeled using Petri-net and a near-optimal schedule can be generated using heuristic search method. Now, in order to show how the modeling and scheduling approach can really solve the job shop scheduling problem of a production system, we chose an FMS as a target system for implementation. The target system is a prototype FMS in Automation Lab., Department of Mechanical Engineering, National Taiwan University.

The layout of the target system is shown in Fig. 1. The installed hardware contains (1) Load/Unload Station; (2) Automated Storage/Retrieve System; (3) Automated Guided Vehicles (AGV's); (4) Robot; (5) Lathe; (6) Milling Machine; (7) Machine Center.

In our experiment, there are three part types J_1, J_2, J_3 to be produced and Table 1 shows the operations requirements of these jobs. We use the symbols M_1, M_2, M_3, and R to represent the lathe, milling machine, machine center, and Robot, respectively. The table gives the job requirements of alternative operations and the necessary resources for processing. It also gives the technological precedence constraints among the processes.

Job	O_{i1}	O_{i2}	O_{i3}
J_1	M1/M2	M3	M1/M2
J_2	M3	M1/M2	NA
J_3	M2	M3	M1/M2

Table 1: Job Requirements of Target System

The operation time of each operation of jobs are shown in Table 2. In the table, $OP_{j,i,k}$ means that it is an operation i of job j performed on machine k. Note that operation time includes tool set-up time of machine.

Operation	Operation Time
$OP_{1,1,1}$	82
$OP_{1,1,2}$	102
$OP_{1,2,3}$	27
$OP_{1,3,1}$	45
$OP_{1,3,2}$	32
$OP_{2,1,3}$	50
$OP_{2,2,1}$	53
$OP_{2,2,2}$	25
$OP_{3,1,2}$	95
$OP_{3,2,3}$	88
$OP_{3,3,1}$	112
$OP_{3,3,2}$	94

Table 2: Operation Time of Jobs.

And, Table 3 gives the time of AGV's transportation time between each pair of workstations or stops, e.g., $TP_{i,j}$ means the transportation time from machine M_i to machine M_j. Notice that M_0 represents the load/unload station. It also gives the loading/unloading time of stations, AS/RS, and AGV's. And robot transfer time of the milling machine and lathe is also given. The symbols L/U and ROB are used to represent the loading/unloading operation and robot transportation, respectively.

The initial situation of the system is that all the machines, robot, AGV's are all ready and available to

	Transportation Time
$TP_{0,1}$	5
$TP_{1,2}$	2
$TP_{2,3}$	4
$TP_{3,4}$	1
$TP_{4,0}$	6
$TP_{4,1}$	5
ROB	2
L/U	2

Table 3: Transportation Time of Material Handling Systems

use. All the buffers, common or local, are also free. One AGV stays at Loading/Unloading station, stop 0, and another one stays at output port of AS/RS, stop 4.

In the heuristic search algorithm, the performance criteria we take is the total completion time, i.e. the makespan. This example is solved using the following heuristic function [5] :

$$h(n) = -w * dep(n)$$

where w is a weighting factor and $dep(n)$ is set to be the total number of accomplished operations for all jobs at node n. And the limited-expansion A algorithm is used to solve the job-shop scheduling problem.

Several different job sizes of this example are tested and the results , makespan only, are shown in Table 4. Note that, the capacity of $OPEN$ is limited to 16 and the weighting factor of heuristic function $h(n)$ is set to 3 in our limited-expansion A algorithm.

Lot Sizes			Total Completion Time
Job 1	Job 2	Job 3	
1	1	1	370
2	2	2	536
5	5	5	1403
10	10	10	2435

Table 4: Scheduling Results (cost only).

5 Conclusions

In this paper, we use two basic sub-models to construct the complete Petri-net model of an FMS. One is Transportation Model and the other is Process-Flow Model. The objective of Transportation Model is to model the behavior of AGV traveling from the stop at which it currently stays to its destination stop, which must also ensure that collisions are avoided. And, the Process-Flow Model is used to model the behavior of the part routing and resource assignment.

In order to obtain an optimal schedule and to avoid the NP-complete computing complexity, we use a sub-optimal algorithm, limited-expansion A algorithm, based on the A^* heuristic search algorithm. This method not only can give solutions close to the optimal or a near-optimal one, but also can easily be implemented on computers, since the memory requirement is bounded and adjustable. For large volumes of parts, we propose an adaptive scheduling method to be combined with the limited-expansion A search algorithm to solve the scheduling problem.

References

[1] D. G. Catherine etc., "A Survey of Flexible Manufacturing Systemy," *Journal of Manufacturing System*, Vol. 1, No. 1, pp. 1-16, 1982.

[2] Frederick A. Rodammer, K. Preston White, "A Recent Survey of Production Scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, No. 6, NOV/DEC 1988.

[3] Philippe Solot, "A Concept for Planning and Scheduling in An FMS," *European Journal of Operational Research*, 1990, Vol. 45, pp. 85-95.

[4] Simon France, *Sequencing and Scheduling: An Introduction to the Mathematics of the job-shop*, New York:Wiley, 1982.

[5] D. Y. Lee, F. DiCesare, "FMS Scheduling Using Petri Nets And Heuristic Search," *Proceedings of the 1992 IEEE International Confrerence on Robotics and Automation*, Nice, France, pp. 1057-1062.

[6] Chin-Jung Tai,and Li-Chen Fu, "A Simulation Modeling for a Flexible Manufacturing System with Multiple Task-Flows and Transportation Control Using Modular Petri-Net Approach", Proc. 9th International Conference on CAD/CAM , Robotics & Factories of the Future, August, 1993.

[7] N. J. Nilsson, *Problem-Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

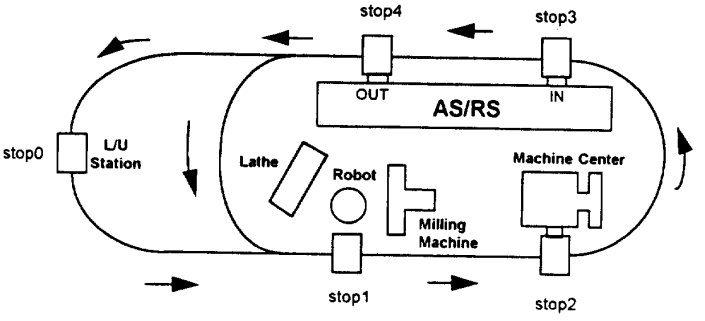

Figure 1: Transportation System Layout of the FMS in Automation Lab. of Department of Mechanical Engineering, National Taiwan University

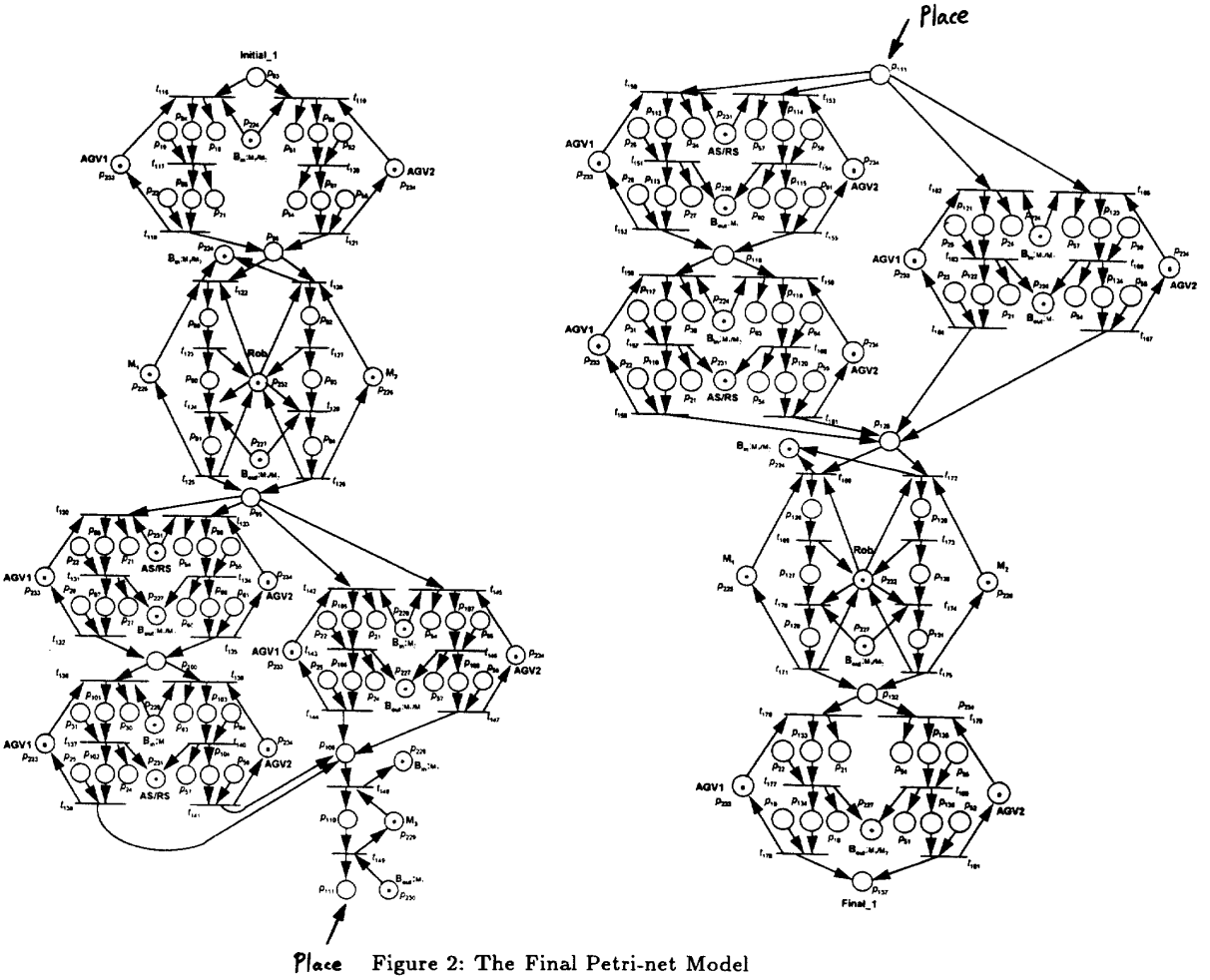

Figure 2: The Final Petri-net Model