Mullya Satish Anand, Santosh Krishnaji Sindhe / International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 3, Issue 2, March -April 2013, pp.492-496

Modeling and Simulation of Job Shop Scheduling Using Petri-Nets

Mullya Satish Anand*, Santosh Krishnaji Sindhe**
*(Department of Mechanical Engg, IOK COE Pune)
** (Department of Mechanical Engg, IOK COE Pune)

ABSTRACT

In most of the manufacturing units scheduling is a difficult task due to the complexity of the system. Hence powerful tools that can handle both modeling and optimization are required. Most of the research in this area focuses in either developing optimization algorithms, or in modeling complex production systems. However, few tools are aimed to the integration of both of them. In this paper, a Petri Net based integrated approach, for simultaneously modeling and scheduling manufacturing systems, is proposed. The procedure is illustrated with an example problem.

Keywords - **Analysis of Petri Nets, INA, Petri Nets, Scheduling.**

I. INTRODUCTION

The prime subject of research on scheduling problems consists of the optimal allocation of scarce resources to tasks over time. Despite the complexity of many scheduling problems, effective algorithms have been developed. However, most of research focused on the effectiveness of the algorithms, neglecting the issue of flexibility. Research on Petri nets addresses the issue of flexibility. To facilitate the modeling of complex systems many extensions have been proposed. Typical extensions are the addition of 'color', 'time' and 'hierarchy'. These Petri nets have all the advantages of the classical Petri net, such as the graphical nature, mathematical foundation and the various analysis methods. Therefore, it is interesting to investigate the application of Petri nets to scheduling. In this paper we concentrate on timed Petri nets, i.e. Petri nets extended with a timing concept.

II. LITERATURE REVIEW

When Petri Nets were introduced, many papers were published on topics such as resource utilization, bottlenecks, throughput, cycle times and capacity estimations. Petri Nets were evaluated in manufacturing systems using different techniques like simulation, queuing theory, probability and stochastic Petri nets. Most of the results in this area focused on cyclic scheduling problems. Ramamoorthy & Ho (1980) and Hillion & Proth (1989) have used a technique based on a 'marked graphs' (a subclass of Petri nets) to analyze the throughput of cyclic (production) processes. Carlier, Chretienne & Girault (1984, 1988, 1983), Gao, Wong & Ning (1991) and Watanabe & Ya-mauchi (1993) also focused on minimal cycle times for repetitive scheduling problems. In this paper focus is on the traditional non-cyclic scheduling problems such as machine scheduling and job shop scheduling (Pinedo, 1995). In this paper timed Petri net model is used and time is associated with transitions.

III. TIMED PETRI NETS

Petri nets originate from the early work of Carl Adam Petri in 1962. Since then there has been lot of research in study and application of Petri nets. The classical Petri net is a bipartite directed graph with two node types called places and transitions. The nodes are connected via directed arcs. Two nodes of the same type cannot be connected. Places are represented by circles and transitions by rectangles. Places may contain zero or more tokens, which are represented by black dots. The number of tokens may change during the execution of the net. A place 'p' is called an input place of a transition't' if there exists a directed arc from p to t, p is called an output place of t if there exists a directed arc from t to p. The net shown in Fig.I illustrate the classical Petri net model. These Petri net models a machine which processes jobs and has two states free and busy. There are four places *in*, *free*, *busy* and *out* and two transitions *start* and *finish*. In the state shown in Fig. I there are five tokens; four in place *in* and one in place *free*. The tokens in place *in* represent jobs to be processed by the machine. The token in place *free* indicates that the machine is free and ready to process a job. If the machine is processing a job, then there are no tokens in *free* and there is one token in *busy*. The tokens in place *out* represent jobs which have been processed by the machine. Transition start has two input places *in* and *free* and one output place *busy*. Transition *finish* has one input place *busy* and two output places *out* and *free*. A transition is called enabled if each of its input places contains at least one token.

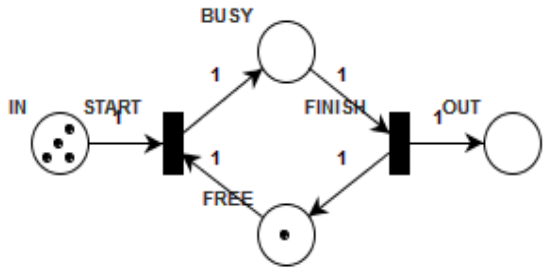

Figure I. A Petri net which represents a machine

An enabled transition can fire. Firing a transition t means consuming tokens from the input places and producing tokens for the output places, i.e. t 'occurs'.

Transition *start* is enabled in the state shown in Fig. I, because each of the input places *in* and *free* contains a token. Transition *finish* is not enabled because there are no tokens in place *busy* . Therefore, transition *start* is the only transition that can fire. Firing transition *start* means consuming two tokens, one from *in* and one from *free*, and producing one token for *busy*. The resulting state is shown in Fig.II. In this state only transition *finish* is enabled. Hence, transition *finish* fires and the token in place *busy* is consumed and two tokens are produced, one for *out* and one for *free*. Now transition *start* is enabled, etc. As long as there are jobs waiting to be processed, the two transitions fire alternately, i.e. the machine modeled by this net can only process one job at a time.

Timing Concept:

To model the real systems it is often important to describe the behaviour of the system with the help of durations and delays. Since the classical Petri net is not easily capable of handling quantitative time, we add a timing concept. In this paper a timing concept is used where time is associated with transitions which determine delays. Firing is instantaneous, i.e. the moment a transition consumes tokens from the input places the produced tokens appear in the output places. However, because of the firing delay it takes some time before the produced tokens become available for consumption. This results in the following definition of a timed Petri net .

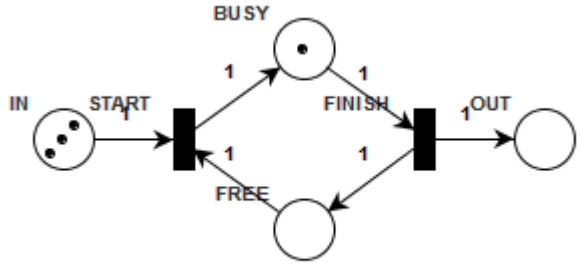

Figure II. Transition start has fired

Definition 1

A timed Petri net is a six tuple TPN = (P, T, I, O, TS, D) satisfying the following requirements:

(i) P is a finite set of places.

(ii) T is a finite set of transitions.

(iii) $I \in T \rightarrow P$ (P) is a function which defines the set of input places of each transition.

(iv) $O \in T \rightarrow P$ (P) is a function which defines the set of output places of each transition.

(v) TS is the time set.

(vi) $D \in T \rightarrow TS$ is a function which defines the firing delay of each transition.

The state of a timed Petri net is given by the distribution of tokens over the places. Firing a transition results in a new state. This way a sequence of states M0, M1, ... Mn is generated such that M0 is the initial state and Mi+1 is the state reachable from Mi by firing a transition. Transitions are eager, i.e. they fire as soon as possible. If several transitions are enabled at the same time, then any of these transitions may be the next to fire. Therefore, in general, many firing sequences are possible. Let M0 be the initial state of a timed Petri net. A state is called a reachable state if and only if there exists a firing sequence M0, M1, ...Mn which 'visits' this state. A terminal state is a state where none of the transitions is enabled, i.e. a state without successors.

IV. THE GENERAL SCHEDULING PROBLEM

Scheduling is concerned with the optimal allocation of scarce resources to tasks over time. Scheduling techniques are used in production planning, project planning, computer control, manpower planning etc.

Resources can be called as 'machines' or 'processors' and tasks as 'operations' or 'steps of a job'. Resources are used to process tasks. However, it is possible that the execution of a task requires more than one resource, i.e. a task is processed by a resource set. There may be multiple resource sets that are capable of processing a specific task. The processing time of a task is the time required to execute the task given a specific resource set. By adding precedence constraints it is possible to formulate requirements about the order in which the tasks have to be processed. It is assumed that resources are always available, but shall not necessarily assume the same for tasks. Each task has a release time, i.e. the time at which the task becomes available for processing. This leads to the following definition.

Definition 2

A scheduling problem is a six-tuple SP = (T, R, PRE, TS, RT, PT) satisfying the following requirements.

(i) T is a finite set of tasks.

(ii) R is a finite set of resources.

(iii) PRE ⊆ T × T is a partial order, the precedence relation.
(iv) TS is the time set.
(a) the resource sets capable of processing task t and
(b) the processing time required to process t by a specific resource set.

This definition specifies the data required to formulate a scheduling problem. The tasks are denoted by T and the resources are denoted by R. The precedence relation PRE is used to specify precedence constraints. If task t has to be processed before task t', then (t, t') ∈ PRE, i.e. the execution of task t has to be completed before the execution of task t' may start. TS is the time set. IN and IR+ ∪ {0} are typical choices for TS. The release time RT (t) of a task t specifies the time at which the task becomes available for processing, i.e. the execution of t may not start before time RT (t).

Scheduling Problem: 3 jobs 4 machines problem

Table I. Path of operations

job	1	2	3	4
1	1	2	3	4
2	1	2	4	3
3	3	1	2	4

Table II. Time of operations

job	1	2	3	4
1	9	8	4	4
2	5	6	3	6
3	10	4	9	2

Above, TABLE I. shows path for operations which is to be followed by 3 jobs and the TABLE II. shows the corresponding processing time of each job on a particular machine.

Assumptions
1. No resource may process more than one task at a time.
2. Each resource is continuously available for processing.
3. Each operation, once started, must be completed without interruptions.
4. The processing times are independent of the schedule. Moreover, the processing times are fixed and known in advance.

Fig. V shows Petri Net model of 3 job 4 machines. The typical characteristics exhibited by the activities in a dynamic event-driven system, such as concurrency, decision making, conflict, sequentiality, mutual exclusion, synchronization and priorities, can be modeled effectively by Petri nets. Here mutual exclusion is used to build the model. This is due to the fact that 3 jobs may be ready to be processed with a single machine, but as per our assumption machine can process 1 job at a time. Mutually exclusive- Two processes are mutually exclusive if they cannot be performed at the same time due to constraints on the usage of shared resources. Fig.III shows this structure. Here either transition t1 or t3 can be fired first. For example, a robot may be shared by two machines for loading and unloading.

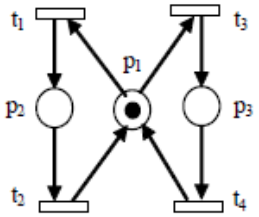

Figure III. Mutual Exclusive

In Fig. V places containing tokens indicate that jobs and machines are available initially. Processing times are given to the transitions. TABLE III shows the transitions and its time. Other transitions given in the model apart from the transitions given in the table has 0 unit time. These transitions indicate start of the process. T25 transition is a switch transition used to complete the cycle; its time is 0 units. Initial state consists of places P0, P1, P2 containing 1 token each indicating start of job 1, 2 & 3. And final state consists of places P38, P39, P40 containing 1 token each indicating completion of job 1, 2, 3. To achieve the final state transitions are fired according to sequence of operations. The sequence of transitions with minimum time is the optimum schedule. Analysis of the Petri Net model is done with the help of INA tool.

Table III. Transitions with time units

T1	9	T6	4	T12	9	T18	2
T3	5	T8	8	T14	3	T20	4
T4	10	T10	6	T16	4	T22	6

V. INTEGRATED NET ANALYZER

With the help of INA, Petri nets of very different kinds can be investigated under different firing rules (in particular timing) with regard to their general properties. Properties which can be verified through the analysis are boundedness of places, liveness of transitions, and reachability of markings or states.
INA combines the following: a textual editor for nets, a by-hand simulation part, a reduction part for Place/Transition-

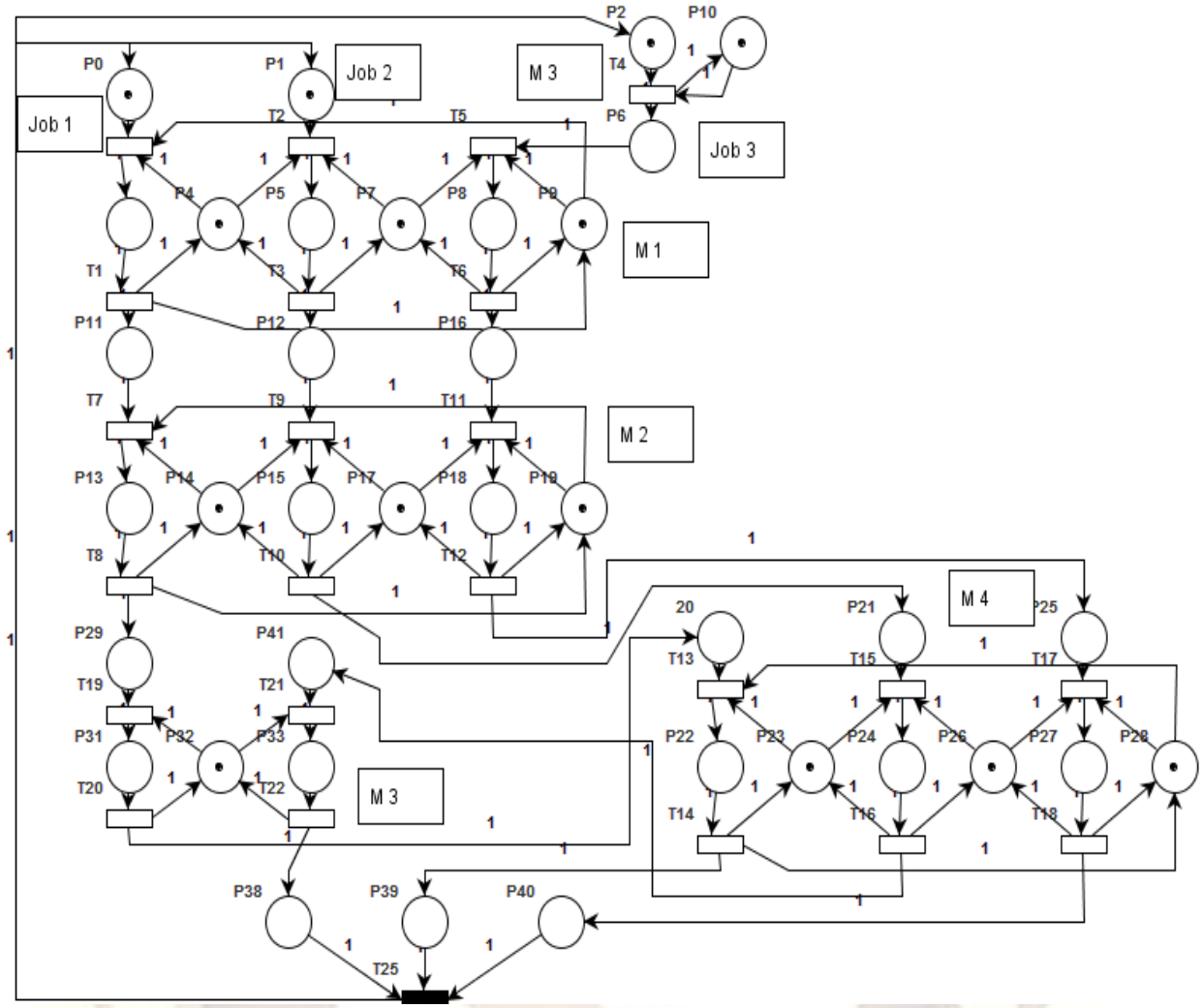

Figure V. Petri Net model of 3 job 4 machines.

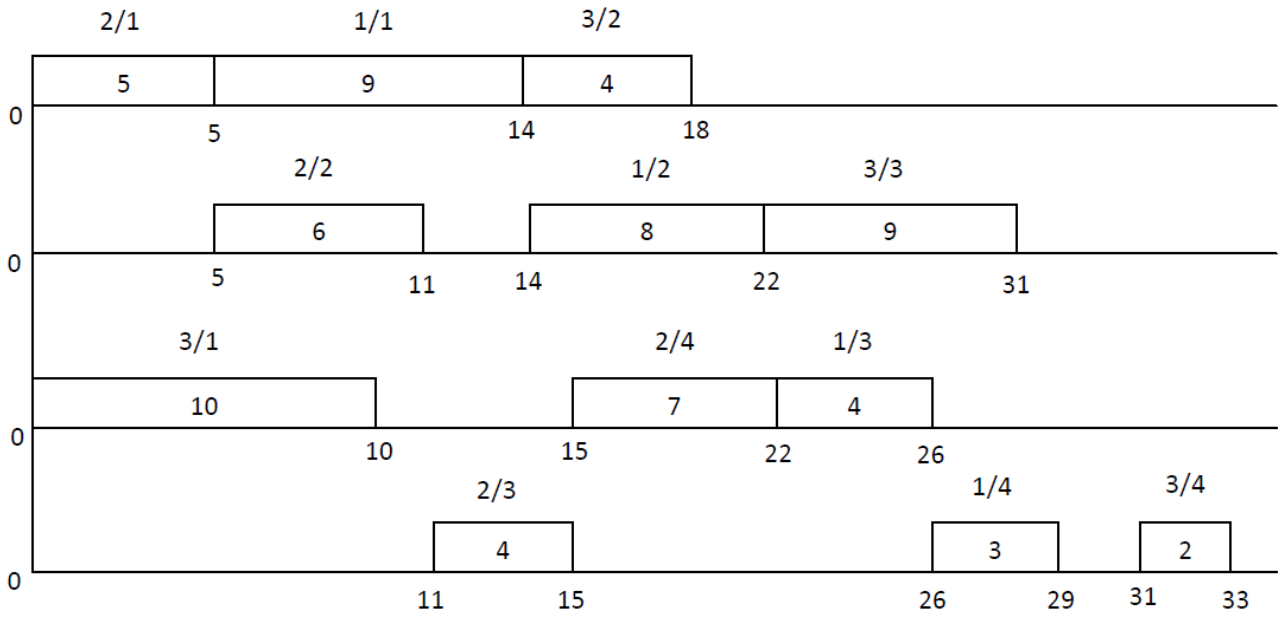

Figure VI. A Schedule for 3 jobs and 4 machines.

nets, an analysis part to compute: structural information, place invariants, transition invariants, reachability graphs.

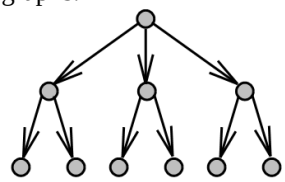

Figure IV. Reachability Graph

Fig. IV shows Reachability Graph. The reachability graph of a timed Petri net is constructed as follows. We start with an initial state M. Then we calculate all states reachable from M by firing a transition. Each node in the reachability graph corresponds to a reachable state and each arc corresponds to the firing of a transition. firing sequences. By computing the reachability graph, it is possible to analyze all possible For a net representing a scheduling problem, each of these firing sequences corresponds to a feasible schedule. Therefore, reachability graph is used to generate many feasible schedules.

VI. RESULTS

INA is a textual tool whereas petri net is a graphical tool so the input to the INA tool is a .PNT file which is in text form. INA gave all the structural properties of the petri net model. INA computed reachability graph which consists of 34 states and the optimum schedule was having a make-span of 33 units. For this schedule the sequence of firing of transitions is T4 T3 T1 T10 T16 T6 T8 T22 T12 T20 T14 T18. The Gantt chart of the schedule is given in Fig. VI.

VII. CONCLUSION

The approach presented in this paper shows that it is possible to model many scheduling problems in terms of a timed Petri net. In fact, a recipe has been formulated for mapping scheduling problems onto timed Petri nets. This recipe shows that the Petri net formalism can be used to model tasks, resources and precedence constraints. By mapping a scheduling problem onto a timed Petri net, we are able to use Petri net theory to analyze the scheduling problem. We can use Petri net based analysis techniques to detect conflicting precedence's, determine lower and upper bounds for the minimal make-span, etc. By inspecting the reachability graph, we can generate many feasible schedules.

REFERENCES

[1] Jiacun Wang. "Petri Nets for Dynamic Event – Driven System modeling.

[2] W.M.P van der Aalst. "Petri Net based Scheduling" Department of Mathematics and Computing Science, Eindhoven University of Technology.

[3] Gonzalo Mejia, Nicholas G. Odrey "Job Shop Scheduling in Manufacturing: An approach using Petri Nets and Heuristic Search".

[4] Tadao Murata "Petri Nets: Properties, Analysis and Applications" *Proceedings of the IEEE, vol 77, no 4, 1989.*

[5] Raida El Mansouri, Elhillali Kerkouche, and Allaoua Chaoui, "A Graphical Environment for Petri Nets INA Tool Based on Meta-Modelling and Graph Grammars". *World Academy of Science, Engineering and Technology 44 2008.*

[6] Ramamoorthy, C.V. And G.S. H O (1980), Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets, *IEEE Transactions on Software engineering 6, 440–449.*

[7] Carlier , J., P. Chretienne , And C. Girault (1984), Modelling scheduling problems with Timed Petri Nets, in: G. Rozenberg (ed.), *Advances in Petri Nets 1984,Lecture Notes in Computer Science 188, Springer-Verlag, Berlin, 62–82.*

[8] Hillion , H.P. And J.P Proth (1989), Performance Evaluation of Job-Shop Systems Using Timed Event Graphs, *IEEE Transactions on Automatic Control 34, 3–9.*

[9] Pinedo,M. (1995), Scheduling: theory, algorithms, and systems, Prentice-Hall, Engle-wood Cliffs.