Using Petri Net and Branch and Bound Algorithm for Modeling and Scheduling of a Flexible Manufacturing System

ABOLFAZL JALILVAND
Electrical Engineering Group
Islamic Azad University of Abhar
Abhar
IRAN

SOHRAB KHANMOHAMMADI
Faculty of Electrical Engineering
University of Tabriz
Tabriz
IRAN

Abstract: In this paper a flexible manufacturing system is investigated where the production process is accomplished in two main stages: machining and assembling. For Modeling of the system the timed Petri net is used. The minimum time requirement for completing the machining is discussed so that assembling the parts would be performed at the possible least time. Scheduling the manufacturing system is performed via a Petri net based controller supervised by a branch and bound algorithm. Also for speeding up the running of algorithm a new method is introduced which doesn't need large memory. The proposed approaches are verified through simulation results**.**

Keywords: Manufacturing system, Task scheduling, Assembly, Branch and Bound, Petri net.

1 Introduction

Producing customized products in a short time at low cost is one of the goals of the manufacturing systems. In most of the manufacturing systems the production is performed in two main stages: machining and assembly. At the machining stage, non-standard and raw parts are machined with machines. At the assembly stage the machined parts are assembled with assembly equipments and techniques. The successful implementation of these two stages lies in efficient scheduling of the system [1]. The schematic structure of the manufacturing system that implements these two main stages is shown in Fig. 1 [1].

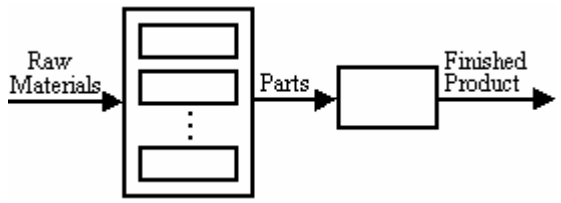

Fig.1: The general structure of a manufacturing system

There are *m* multi-purpose machines at the machining stage and one assembly machine at the assembly stage. The machining Operations include all activities that are required to perform on raw materials. Assembly stage involves in assembling the machined parts to form the final product.

Although in a simple manner each raw part can be machined by a respective machine but for flexibility purpose, it has been assumed that each part can be machined by each of several machines in machining stage. Furthermore we assume that removing different materials and allocating each of them to each machine is performed by a robot manipulator. In deed we consider a system so called flexible manufacturing system which is collection of work-stations that produce a family of related parts that require similar operations. A key feature of a flexible manufacturing system is the way in which raw materials are routed into and finished pieces are routed out of the manufacturing system. A focus in such system is that how the given pieces travel through the system [2-5]. Accordingly in such case we encounter the problem that in which sequence the parts must be removed by robot and in which order they must be allocated to machines so that the total completion time for a product would be minimum.

2 Manufacturing System

In this section we consider a flexible manufacturing system that consists in three multi-purpose machines: M_1, M_2 and M_3 that each of them can process different raw materials: *a*, *b* and *c*. The output parts from these machines will be assembled by M_4 and finally finished part will be produced. The removing and carrying of each raw material and also allocation of it to each of the machines is performed by a robot. In this system, at a production cycle each of three machines receives one of the three different parts and performs its prescribed task on it. When all of the three machines accomplished their tasks and produced their outputs, the assembly machine can

work on them and produce the final product. The production cycle terminates when finished piece is produced. The schematic construction of the manufacturing system is shown in fig. 2.

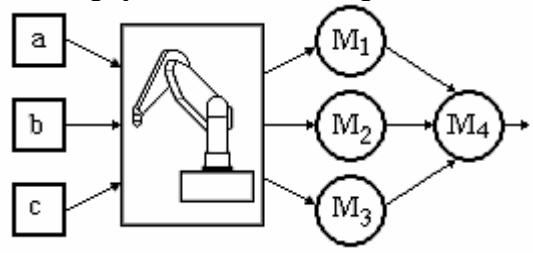

Fig. 2: The flexible robotic manufacturing system

The machines: M_1, M_2 and M_3 must have an input buffer related to each part and one output buffer. Also the machine M_4 has three input buffer and one output buffer (fig. 3).

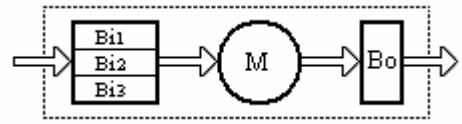

Fig. 3: The complete model of each machine in machining stage and assembly stage.

The Petri net [6] model for each of these machines is depicted in fig. 4. Figure 4(a) shows the Petri net model of each of the machines in machining stage and fig. 4(b) depicts the Petri net model of the assembly machine.

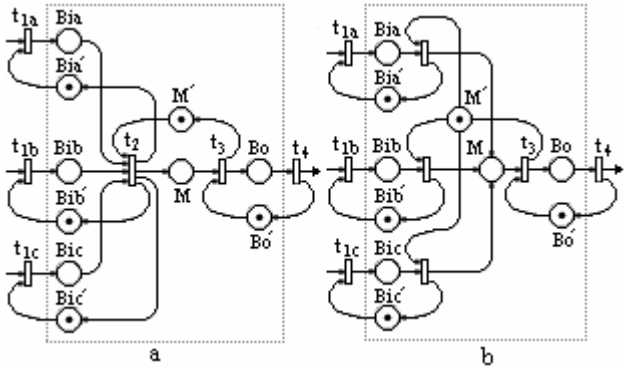

Fig.4: Petri net model of each machine: (a) machining stage (b) assembly stage.

Where each places means as follows:
B_i': The input buffer is empty.
B_i: The input buffer is filled.
B_o': The output buffer is empty.
B_o: The output buffer is filled.
M': The machine is idle.
M: The machine is busy.
Furthermore each of the transitions means as follows:
t_1: The input buffer is loaded.
t_2: The machine starts to work on the part.
t_3: The machine finishes the working on the part.
t_4: The output buffer is unloaded.

Note that in this Petri net models ignoring the times of loading the input buffer and unloading the output buffer, all of time that is consumed for machining the part has been assigned to transition t_3. In other word we use the timed Petri net for modeling.

In this system, based on that the robot's arm has been neighbor which machine after carrying and assigning the previous part and attempts to remove which next part for which machine, it may consume different times. We summarize this times in three matrices: T_{1R}, T_{2R} and T_{3R}. Where t_{ixj} means the time is required to robot moves from neighboring machine M_i to remove part x and carries it to machine M_j.

$$T_{1R} = \begin{bmatrix} t_{1a1} & t_{1a2} & t_{1a3} \\ t_{1b1} & t_{1b2} & t_{1b3} \\ t_{1c1} & t_{1c2} & t_{1c3} \end{bmatrix}$$

$$T_{2R} = \begin{bmatrix} t_{2a1} & t_{2a2} & t_{2a3} \\ t_{2b1} & t_{2b2} & t_{2b3} \\ t_{2c1} & t_{2c2} & t_{2c3} \end{bmatrix} \quad (1)$$

$$T_{3R} = \begin{bmatrix} t_{3a1} & t_{3a2} & t_{3a3} \\ t_{3b1} & t_{3b2} & t_{3b3} \\ t_{3c1} & t_{3c2} & t_{3c3} \end{bmatrix}$$

Furthermore the required processing time for each part relating to each machine is given as follows:

$$T_{M1} = \begin{bmatrix} t_{1aa} & t_{1ab} & t_{1ac} \\ t_{1ba} & t_{1bb} & t_{1bc} \\ t_{1ca} & t_{1cb} & t_{1cc} \end{bmatrix}$$

$$T_{M2} = \begin{bmatrix} t_{2aa} & t_{2ab} & t_{2ac} \\ t_{2ba} & t_{2bb} & t_{2bc} \\ t_{2ca} & t_{2cb} & t_{2cc} \end{bmatrix} \quad (2)$$

$$T_{M3} = \begin{bmatrix} t_{3aa} & t_{3ab} & t_{3ac} \\ t_{3ba} & t_{3bb} & t_{3bc} \\ t_{3ca} & t_{3cb} & t_{3cc} \end{bmatrix}$$

Where t_{ixy} is the time that ith machine consumes in order to process part y whereas its previous processed part was x. for ensuring that assembly machine can produce a finished part at each production cycle, it is required that robot carries one of each parts: a, b and c and machines: M_1, M_2 and M_3, is allocated one of the parts at each cycle. Furthermore, the sequence of robot actions in allocating the parts to machines and also machines actions in machining stage must be considered in such manner that the final product would be produced in minimum time.

To solve the first problem we introduce two Petri net controllers called: sequencer 1 and sequencer 2 that are shown in fig. 5.

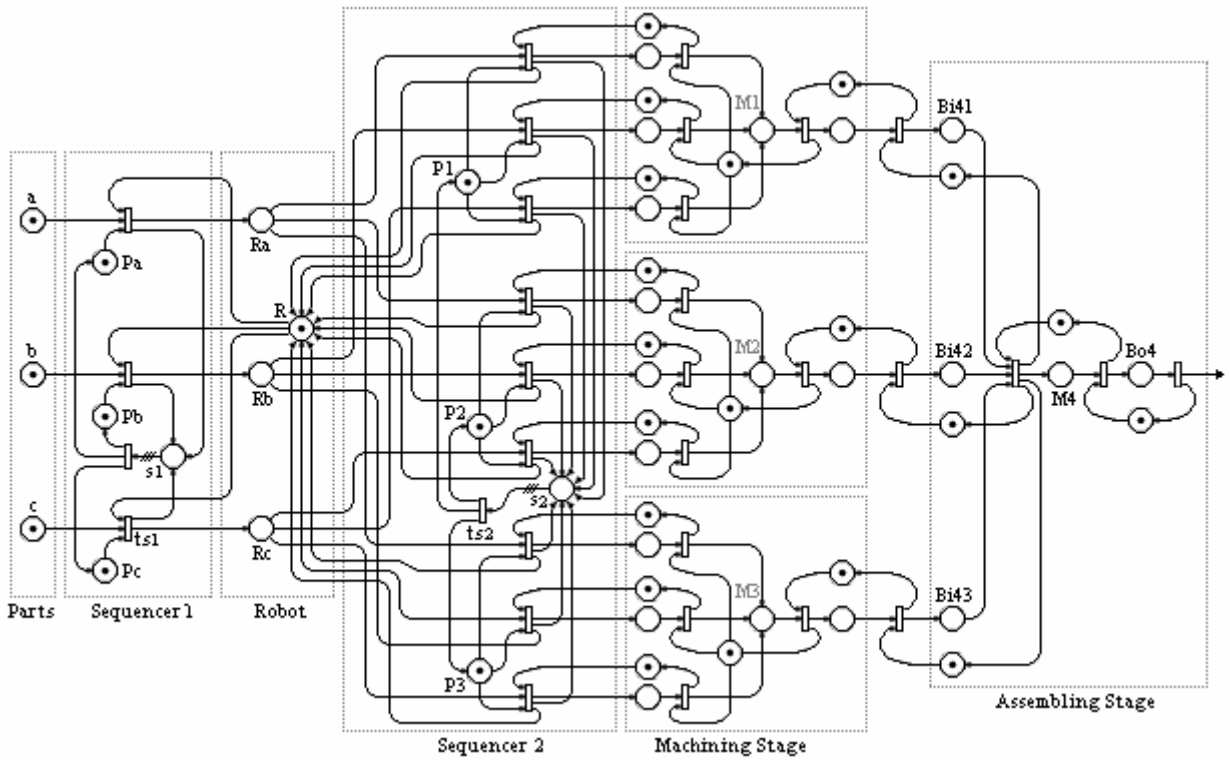

Fig. 5: Petri net model of flexible robotic manufacturing system.

Each of the places P_a, P_b and P_c in sequencer 1 must have one token at first as initial marking. When robot picks up one of the parts: a, b or c the token of its respective place (P_a, P_b or P_c)will be removed. This permits the robot to pick up the other type of parts in next step. When all types of different parts: a, b and c are moved by robot (each of them once), the tokens of places: P_a, P_b and P_c have been removed and the place s has three tokens that are required for firing the transition t_s and consequently deposit one token in each places: P_a, P_b and P_c , for next production cycle. Also places: P_1, P_2 and P_3 in sequencer 2 must have one token initially. When robot delivers the part to one of machines: M_1, M_2 and M_3, the token of its respective place: P_1, P_2 or P_3 will be removed. This permits the robot to delivers the next part to other machines in next step. When all of the machines have their part, the tokens of places: P_1, P_2 and P_3 have been removed and the place s_2 has three tokens that are required for firing the transition t_{s2} and deposit one token in each of the places: P_1, P_2 and P_3, for next production cycle.

For solving the second problem we will use a Branch and Bound based algorithm which enable us to decide which sequence of parts and also which sequence of machines must be chosen by robot in each cycle in order to minimize the total completion time for each finished part.

3 Task scheduling

As it mentioned before, it is required to determine the sequence of picking up each part by robot and allocating it to the machines in such manner that outputs of machining stage would be ready in minimum time for the assembly stage. It must be investigated all of the possible sequences that based on them the robot can pick up a part and allocate it to machines and select the optimum sequence so that the maximum consumed time in order to a part will be ready to assembling will be minimum for each production cycle. Whereas selecting such optimum sequence is dependent on the previous conditions of the system at the end of last cycle, we need an algorithm that can calculate the optimum sequence. The Branch and Bound (BB) is a well known method that can be used for this purpose [7-8]. But its requirement that all of the possible sequences have been produced previously needs a large size memory and causes the running time of algorithm increases considerably [9]. We consider the BB algorithm to determine the optimum sequence for the manufacturing system. But in order to speed up the running of BB algorithm a new method is proposed that doesn't need large size memory.

Regarding the manufacturing system, we must consider that at beginning a new sequence the robot arm may be near each of the three different machines in machining stage and each of the

machines may work on one of the three different parts. The tree of the decision making construction is shown in fig. 6.

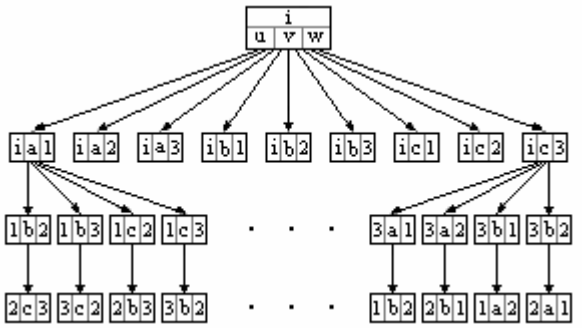

Fig. 6: The decision making tree.

We demonstrate index of the last machine in machining stage by i and the last processed parts by the machines: M_1, M_2 and M_3, by u, v and w respectively. In order to use the BB algorithm, i plus u, v and w is considered as a root node. Content of the other nodes demonstrate the start point, picking up part and the end point of robot in a route. For each of such nodes we can assign a total time, including the robot consuming time and the time that machine consumes in order to process the corresponding part based on its previous processed part. For example for the node 1b2, considering the previous processed part by machine M_2, is v we have the total time $t_{1b2}+ t_{2vb}$ and for the node 2c3, considering the previous processed part by machine M_3, is b we have the total time $t_{2c3}+ t_{3bc}$.

In the tree, each branch from node to one of the leaves demonstrate a possible operation sequence for the manufacturing system which consists in three nodes, each node related to one machine and one part. In order to that the assembly stage would be able to accomplish its task as soon as possible we must consider maximum time of each sequence and finally select the sequence with the minimum of the maximum time. We can describe this by formula as follows:

$$\min(\max\{(t_{ixl}+t_{lpx}),(t_{ixl}+t_{lym}+t_{mqy}),\ldots \\ (t_{ixl}+t_{lym}+t_{mzn}+t_{nrz})\}) \quad (3)$$

$x,y,z\in\{a,b,c\}$, $l,m,n\in\{1,2,3\}$, $p,q,r\in\{u,v,w\}$

Where i is the index of the last machine which robot delivered the part to it at previous sequence. u, v and w are the parts that M_1, M_2 and M_3 processed them at previous sequence respectively. x, y and z are the new order of parts and l, m and n are the indices of the machines which the parts must be delivered them respectively.

Now for applying the BB algorithm we introduce a new method that in each iteration produces and investigates only one branch of tree hence it don't need large memory and speed up the running of the BB algorithm considerably.

4 Branch and Bound Algorithm

For describing the BB algorithm we must determine a route (R) as the sequence of operation order and an array for the time of a route (TR) that includes the total time relating to each part and the machine works on it. For instance for i=2 and [u, v, w]=[c, b, a], for route R=[i, [(b,1) (a,3) (c,2)]] we have the time array TR=[(t_{2b1} + t_{c1b}),(t_{2b1} + t_{1a3} + t_{b3a}),(t_{2b1} + t_{1a3} + t_{3c2}+ t_{a3c})] and the maximum time for this route will be MR= max(TR). Now we describe the BB based algorithm for searching the optimum sequence as follow:

1- Get r as the initial root and [u, v, w].
2- Get the T_R as time matrix of robot, T_M as time matrix of machines
3- Set $S=[u \quad v \quad w]$, $B'=[1 \quad 2 \quad 3 \quad \ldots \quad N']^T$ as the initial value of B' where N' is the length of S, $A'=\left[S^T \quad S^T \quad \ldots \quad S^T\right]^T$ as the initial value of A' and F'=N' as a temporary flag.
4- Select an arbitrary sequence of the N states as the first permutation and locate it in vector P.
5- Set the initial path as R=[r , P]
6- Calculate the maximum time of path R as MR.
7- Set $B=[1 \quad 2 \quad 3 \quad \ldots \quad N]^T$ as the initial value of B, $A=\left[P^T \quad P^T \quad \ldots \quad P^T\right]^T$ as the initial value of A and F=N as a temporary flag.
8- Set R to optimum path (OR) and the MR to minimum value of maximum time (M).
9- Set S'=A'(N',:).
10- *If* MR ≥ M *then* go to step12 *else* go to step 11
11- *If* F = N *then* set OR=R & M=MR and go to 12 *else* set F=F+1 and go to 15.
12- Set B(F)=B(F)+1.
13- *If* B(F)>N and F>1 *then* set B(F)=F, F=F-1 and return to step 12 *else* go to step 14.
14- Substitute the A(F,B(F)) by A(F,F) and vice versa. Set rows F+1 to N of A equal to A(F,:).
15- Set R=[r , A(F,1:F)] and MR=max(TR).
16- *If* B(F)<=N *then* go to step 10 else go to step 17.
17- Set B'(F')=B'(F')+1.
18- *If* B'(F')>N' and F'>1 *then* set B'(F')=F', F'=F'-1 and return to step 17 *else* go to step 19.
19- Substitute the A'(F',B'(F')) by A'(F',F') and vice versa. Set rows F'+1 to N' of A' equal to A'(F',:).

20- *If* B'(F')<=N' *then* set F'=N' and go to step 9 else go to step 21.
21- END.
Figure 10 shows the flow chart of this algorithm.

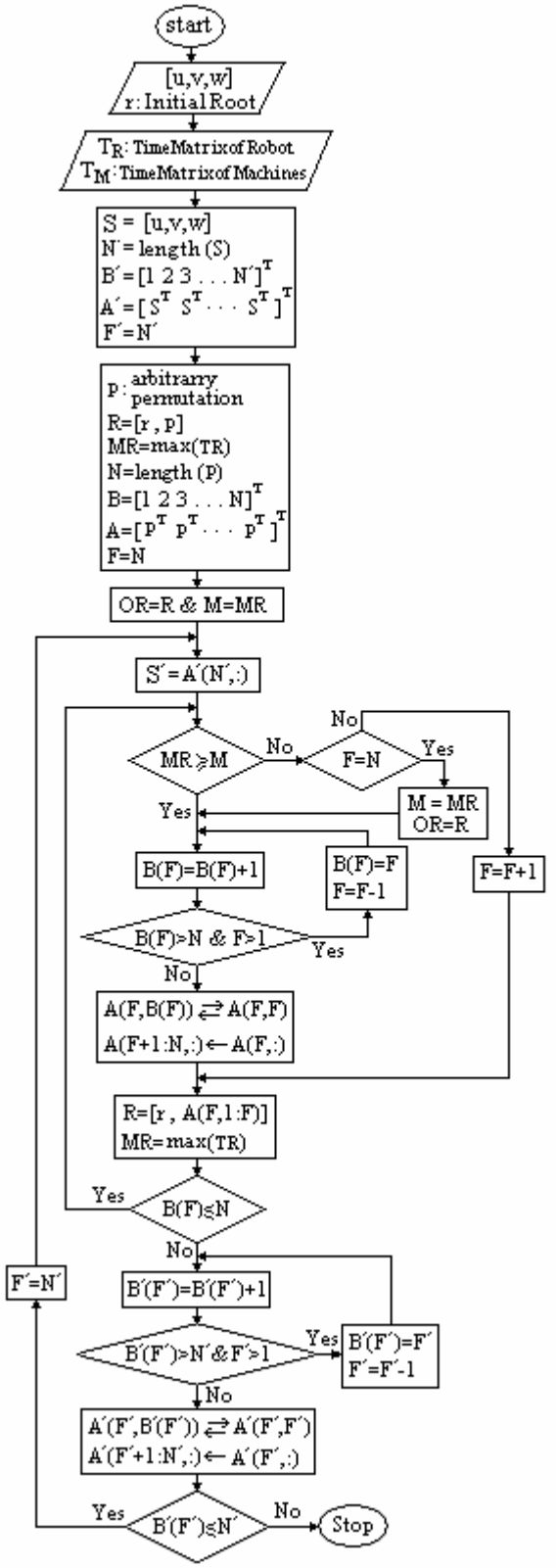

Fig. 7: the flow chart of proposed BB algorithm

This algorithm only needs vectors B and B' and matrices A and A' during its running cycle, hence its memory demanding decreases considerably.

5 Simulation Results

To verify the proposed modeling and scheduling methods we simulated them relating to the described manufacturing system. For this purpose a modified Petri net Tool Box [10] has been used. In order to demonstrate the effect of BB algorithm in operation of system the simulation results are obtained for two different situations:

1) With sequencers and without BB based supervisor.
2) With both sequencer and BB based supervisor.

Figure 8 shows the simulation results of Petri net model controlled by sequencers but without BB algorithm. When the carrying of a part is finished by robot, it is located in input buffer of the specified machine if it would be empty. In this case, selecting the part and allocating it to each of the machines is random. For simulation purpose we considered 20 parts from each type. The results show that the time of machining and assembling all parts is 403 time units.

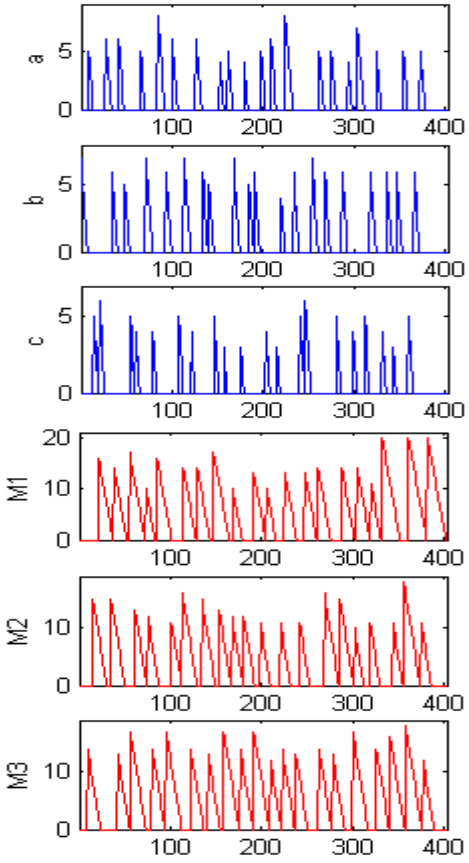

Fig. 8: The simulation result for manufacturing system with sequencer and without BB based supervisor.

The simulation results, when the BB algorithm is used are shown in figure 9. As it is seen from the results the total time of machining and assembling all the parts is reduced to 338 time units. Consequently it can be deduced that the BB algorithm based task scheduling reduces the idle time of assembly machine. In other word applying the proposed method has reduced the total time that is required for finished parts.

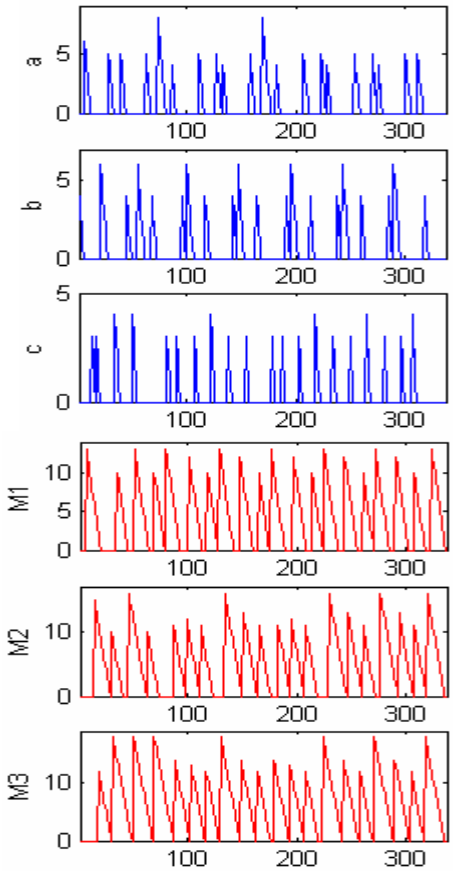

Fig. 9: The simulation result for manufacturing system with sequencer and without BB based supervisor.

6 Conclusion

Modeling and scheduling of a flexible manufacturing system was investigated in this paper. For modeling and control purposes of the system the timed Petri net was used. Task scheduling of the system based on the Branch and Bound algorithm is discussed. In order to apply the Branch and Bound method an efficient algorithm was proposed which doesn't need large memory size and reduces running time of algorithm. The validity of proposed approaches was examined through simulation results.

References

[1] D. He, A. Babayan and A. Kusiak, Scheduling manufacturing systems in an agile environment, *Robotic and Computer Integrated Manufacturing*, Vol. 24, No. 1, 2001, pp. 44-50.

[2] W. S. Newman, A. Podgurski, R. D. Quinn, F. L. Merat and M. S. Branicky Design lessons for building agile manufacturing systems, *IEEE Trans. on Robotic and Automation*, Vol. 16, No. 3, June 2000, pp. 228-238.

[3] D. He and A. Kusiak, Design of assembly systems for modular products, *IEEE Trans. on Robotic and Automation*, Vol. 13, No. 5, Oct. 1997, pp. 646-655.

[4] S. Flake, W. Mueller, U. Pape and J. Ruf, *Analyzing timing constraints in flexible manufacturing systems*, International NAISO Symposium on Information Science Innovations in Intelligent Automated Manufacturing (IAM 2001), Dubai, March 2001.

[5] J. J. westman, F. B. Hanson, and E.K. Boukas, *Optimal production scheduling for manufacturing systems with preventive maintenance in an uncertain environment*, Proc. of 2001 American Control Conference, June 2001.

[6] Tado Murata, Petri nets: properties, analysis and application, *Proc. of IEEE*, Vol. 77, No. 4, April 1989.

[7] P. Y. Gan, K.S. Lee and Y. F. Zhang, *A Branch and Bound algorithm based process planning system for plastic injection mould bases*, the international Journal of Advanced Manufacturing System, Vol. 18, 2001, pp. 624-632.

[8] A. Bemporard, D. Mignone, M. Morari, *An efficient Branch and Bound algorithm for state estimation and control of hybrid systems*, Proc. of the European Control Conference, Karlsruhe, Germany, 1999.

[9] S. Khanmohammadi, Single array Branch and Bound method, *Iranian Journal of Engineering*, Vol. 3, Nos. 1&2, 1990, pp. 71-72.

[10] G. Ribichini, and P. Messina, *Un toolbox per la progettazione la simulazione el' Analisi di Rti di Petri Temporizzate*, Universita Degli Studi Pisa, Facolta di Ingegneria, 7 Marzo 2002.