Deadlock-Free Scheduling Method Using Genetic Algorithm and Timed S^3PR Nets

Zhonghua Huang* , Zhiming Wu

Abstract—**In this paper, a new kind of deadlock-free scheduling method based on genetic algorithm and reachability analysis of timed S^3PR nets is proposed to solve the scheduling problems of job shop without buffers. Under the framework of timed Petri nets model, the scheduling problem can be described as finding a feasible transition firing sequence in the Petri nets model to avoid deadlock situations and to minimize the makespan. In order to satisfy the deadlock free constraint, a repair procedure is imbedded into the genetic algorithm to improve the quality of infeasible solutions and a penalty item is involved in the fitness computation procedure to prevent the search process from converging to infeasible solutions. The method proposed in this paper can get a feasible scheduling strategy as well as enable the system achieve good performance, and this is empirically shown by simulation results.**

I. Introduction

In the research area of manufacturing systems, much work has been done about the scheduling problems such as job shop and flow shop, however most of the scheduling algorithms ignore both material handling devices and limited buffer space constraints [12]. But for the real-life automated manufacturing systems, auxiliary resources are limited and not always available, so the existing scheduling methods are not readily applied to manage an automated manufacturing system. In automated manufacturing systems, when various parts compete for the limited resources, deadlock may occur if without proper scheduling strategy. Deadlock can cause unnecessary costs (e.g. long down time and low use of some critical and expensive resources), which are particularly important in automated manufacturing systems. Therefore, it is important to develop efficient algorithm to improve and optimize the system performances while preventing deadlock situations. The scheduling problem of job shop without buffers proposed in this paper is a problem that no storage capacity is available between any two machines and the job should wait in its current machine until its following machine is released.

Deadlock problem was firstly addressed by computer scientists developing resource allocation methods in operating systems. In particular, Coffman [1] gave four necessary conditions that must be held for a deadlock to occur: mutual exclusion, no prevention, hold while wait, and circular wait.

The pure logic problem of avoiding deadlock is a well-explored problem, and a lot of deadlock resolution methods, such as deadlock prevention, deadlock detection/recovery, and deadlock avoidance policy have been proposed [2-5]. Although all these methods guarantee the deadlock free operation of a manufacturing system, they don't take operating times into account, so the performance of automated manufacturing systems is not guaranteed, and thus scheduling of systems with potential deadlock is still a rather open field. Deadlock free scheduling methods combine scheduling strategies with deadlock resolution approaches. They can bring us a feasible schedule as well as enable the systems to achieve good performance. However, the joint consideration of scheduling and deadlock avoidance can lead to highly combinatorial scheduling problems. Many classical scheduling techniques don't apply here and their extension to automated manufacturing systems is a challenging work. Until now, not much work about deadlock free scheduling has been done. The main methods are summarized as follows.

Ramaswamy and Joshi [6] provided a mathematical model for deadlock-free scheduling problem of automated manufacturing systems with material handling devices and limited buffers and used a Lagrangian relaxation heuristic to simplify the models to search for the optimized average flow time. Lee and DiCesare [7] proposed a heuristic search algorithm based on A* search technique and Petri nets theory. The algorithm can seek a makespan optimal or near-optimal transitions firing sequence within the reachability graph of Petri nets. Abdallah [8] used timed S^4R nets to model automated manufacturing systems, and proposed a search algorithm, which is based on the branch

Manuscript received September 15, 2003. This work was supported by the National Natural Science Foundation of China. (No. 60074011, 70071017).

Zhonghua, Huang is with the Department of Automation, Shanghai Jiaotong Univerisity, Shanghai, 200030, China (phone: 86-21-62933428-24; e-mail: jacki@sjtu.edu.cn).

Zhiming, Wu is with the Department of Automation, Shanghai Jiaotong University, Shanghai, 200030, China. (e-mail: ziminwu@sjtu.edu.cn).

*To whom correspondence should be addressed

and bound principle, depth-first search strategy and a siphon truncation techniques. A user control factor is added to search algorithm to achieve an acceptable trade-off between the solution quality and search effort. Jeng and Chen [9] developed a heuristic algorithm based on best-first search technique and state equation of timed Petri nets models for minimizing the makespan and their method may need much computation for large systems. They also pointed out that their method couldn't handle larger systems with more than 300 operations. Most of the existing deadlock free scheduling algorithms are based on mathematical programming or heuristic algorithms, and they usually cannot handle larger systems.

In this paper, a new kind of deadlock-free scheduling algorithm based on timed S^3PR nets and genetic algorithm is proposed to solve the scheduling problems of job shop without buffers. Genetic algorithm is an effective global search algorithm. By reachability analysis of Petri nets, the scheduling algorithm can effectively sequence the operations on the resources to minimize the makespan as well as avoid the deadlock. The rest of the paper is organized as follows: section 2 introduces the manufacturing systems under consideration and describes the use of timed S^3PR nets in modeling procedure and defines the scheduling problem under the Petri nets model. Section3 describes the coding and decoding scheme and fitness computation of chromosome. Section 4 proposes design method of genetic operators. Section 5 uses examples to show the effectiveness of the proposed scheduling algorithm. Section 6 is a conclusion.

II. PROBLEM SETTING AND TIMED S^3PR MODEL

In this section, we propose a manufacturing system model called job shop without buffers. In the given system, there is no storage capacity available between any two machines and any job should wait in its current machine until its following machine is released. In the following, we discuss the system in detail. The automated manufacturing system is composed of a set of resources $R = \{R_i \mid i = 1,..., m\}$ where m is the number of the resources. Each resource has a single unit, i.e. all resources are different from each other. There is a set of jobs $J = \{J_i \mid i = 1,..., n\}$ need to be performed, where n is the number of jobs. Each job requires a sequence of operations $O_i = \{O_{i,j} \mid j = 1,...k(i)\}$ in term of the manufacturing process where $k(i)$ is the number of operations of job J_i. Each operation $O_{i,j}$ needs a resource in R to be performed, and requires a given amount of time, called processing time and denoted by x_{ij}. In the manufacturing systems described above, resources cannot be preempted and jobs use resources in an exclusive mode and hold resources while waiting for the next resources that are specified by their operation sequences, so the first three necessary condition of deadlock are always satisfied, and thus deadlock may occur in the system due to improper scheduling strategy. Therefore, the job shop problem without buffers presented here is different from classical job-shop problem [17]. In classical job shop problems, after the completion of one operation, the same resource needed for another operation is released immediately, so the third necessary condition of deadlock "hold while wait" is not satisfied and thus deadlock never occur in classical job shop problems, while job shop problem without buffers is characterized by "hold while wait" as described before, so the possible deadlock should be avoided to enhance the system stability.

Petri nets can be used to describe system dynamic characteristics, such as concurrency, conflict, and deadlock, etc. Many researchers use Petri nets as a formalism to describe automated manufacturing systems and to develop appropriate deadlock resolution methods [14]. In this paper, the Bottom-Up method is used to model the whole system.

The first step is to partition the whole system into several sub-systems in terms of the manufacturing process of each job J_i and construct the Petri nets model for each sub-system. Suppose processing a job J_i need a sequence of resources $\{r_{i1}, r_{i2},..., r_{ik(i)}\}$ ($r_{i1} \in R$), then the places in the Petri nets sub-model of job J_i can be composed of input places O_{i0}, output places $O_{i(k(i)+1)}$, resource places $\{P_{i1}, P_{i2},\ldots, P_{ik(i)}\}$, and operation places $\{O_{i1}, O_{i2},\ldots, O_{ik(i)}\}$. Token in the input place represents that the job waits outside the systems to be operated, and token in the output place represents the completion of the job. Transitions $\{t_{ij} \mid j = 1,...k(i)\}$ correspond to the start of next operation, and $t_{i(k(i)+1)}$ means that the job J_i finishes all of its operations and goes into output place. Fig1 shows a Petri net sub-model of job J_i where $k(i) = 2$.

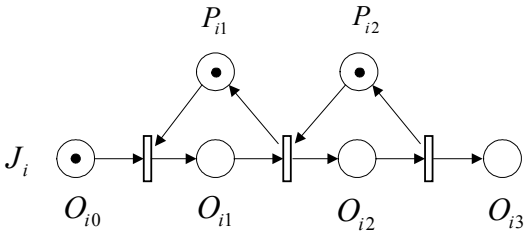

Fig.1 Petri nets sub-model of Ji ($k(i)$=2)

The second step is to merge the sub-models to get the whole Petri nets model by taking into account shared resource places. For example, suppose that there is a set of resources (R_1, R_2, R_3) in the system and two jobs (J_1, J_2) that

need to be processed. The resource sequence that job J_1 needs to require is (R_1, R_2, R_3), and the other one that J_2 needs is (R_2, R_1, R_3). Fig.2 represents the whole Petri nets model after merging the sub-model of J_1 and J_2. The Petri nets model constructed by above Bottom-Up method is called S^{3}PR nets. The detailed definition of the S^{3}PR nets can be found in the references [2][20]. In this paper, the job routing flexibility is not taken into consideration.

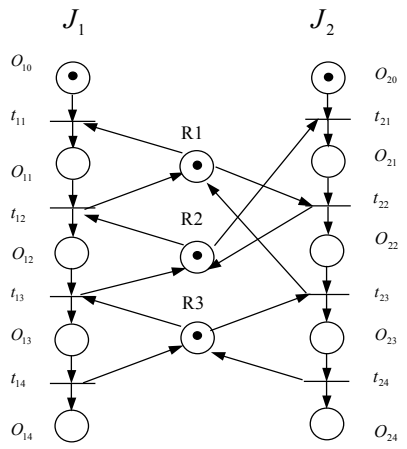

Fig2. Merged Petri nets model

By assigning processing time x_{ij} to the corresponding operation place O_{ij} in S^{3}PR nets, *Timed S3PR nets* can be obtained. In the timed S^{3}PR nets, the tokens in the operation place O_{ij} become available after duration of time x_{ij}. When there are some jobs not finished, all the tokens in the operation place are available, and there are no enabled transitions, we can say that the Timed Petri nets come into a state of deadlock. If the system is in the deadlock situation, all the jobs cannot get progress, so the performance of manufacturing system is dramatically reduced. The goal of deadlock free scheduling algorithm proposed in this paper is to avoid this situation.

Under the framework of timed S^{3}PR nets model, job shop scheduling problem without buffers consists in finding a feasible transition firing sequence in the Petri nets model in order to avoid deadlock and to minimize the makespan. If the system could evolve successfully from the initial state to objective state in term of the feasible transition firing sequence, that is to say, the objective state is reachable under the firing sequence, then the deadlock would not occur.

In the scheduling problem of this paper, the initial state represents the state that all the jobs waits outside the system and all the resources are available, the objective state is the state that all the jobs are completed and all the resources are released and can be reused. In the Petri nets model, we use initial marking M_0 to denote initial state, and objective marking M' to denote objective state. A is the incidence matrix of Petri nets. State equation can be denoted as $M' = M_0 + A \cdot T$ [19]. In the timed S^{3}PR net model, if the system can evolve from initial state to the final state, all the transitions must be fired only once. Then we can get that $T = [1,1,..1]$. If the transition sequence σ composed by all the transitions in Petri nets model is called a *complete sequence*, then a complete sequence is necessary condition for the system to evolve to the objective state.

In timed S^{3}PR net, if all the transitions fire according to the given firing sequence, the system evolves from initial state to target state successfully, then the transition firing sequence is called a *feasible transition sequence*. It is obvious that a feasible transition sequence must be a complete sequence in the timed S3PR net. A complete transition sequence $\sigma(\tau_1, \tau_2, ...\tau_N)$ is feasible, if and only if it satisfies the following reachability constraint: $M_0 + A \cdot \sum_{i=1}^{n} \tau_i \geq 0$ ($n = 1,2,...,N$).

III. Encoding of Transition Sequences and Fitness Computation

In order to get a feasible and optimal transition firing sequence, an effective algorithm based on Petri nets theory and genetic algorithm (GA) is proposed. This algorithm encodes transition firing sequence of Petri nets. With the help of repair procedure and genetic operators, population evolves generation by generation to converge to optimal or near optimal solution.

A. Encoding Scheme

Encoding of transition firing sequence is carried out by using natural numbers. Each chromosome is composed of $N = \sum_{i=1}^{n} (k(i)+1)$ genes. Every gene represents a transition. The number in the gene means the part type corresponding to the transition, and the relative position of gene in the genes with the same number determines the concrete transition. Suppose the number in the gene is i, and the relative position in the genes with the same number i is j, then the gene represents transition t_{ij}. For example, if there is a chromosome 32323211, then the transition sequence corresponding to the chromosome is $t_{31}t_{21}t_{32}t_{22}t_{33}t_{23}t_{11}t_{12}$. If the chromosome could represent a complete transition firing sequence, then the coding scheme of genetic algorithm is reasonable.

B. Check and Repair procedure

Using the method addressed in the above section, we can check up the feasibility of a transition sequence. If the

transition sequence is not feasible, a proper repair procedure is added to the scheduling algorithm to improve the solution.
(1) Assume to be given an encoded chromosome and initial Petri nets marking $M = M_0$. A pointer P is placed initially at the first gene. The number of enabled transitions p is 0, and an iteration parameter q is 0.
(2) If the transition t represented by the gene that is pointed by the pointer P satisfies the firable condition at a marking M, i.e. $M + A \cdot t \geq 0$, then fire the transition and update the marking $M \leftarrow M + A \cdot t$, P $\leftarrow$ P+1, p $\leftarrow$ p+1, and q $\leftarrow$ 0; otherwise, move the pointed gene to the end of the chromosome.
(3) If p+q equals the length of given chromosome, then stop this procedure; else go to (2).

If transition t in the transition sequence pointed by a pointer is not firable successfully from the current state, other transitions behind the pointer would be checked by carrying out the check and repair procedure, so more transitions have the opportunity to be fired under the repair procedure, and the chromosome can be improved to some extent. The parameter p represents the number of transitions, which have been fired successfully from initial marking. The other parameter q shows the number of transitions which have not been judged yet on the right-hand side from the number pointed by the pointer. By calculating a parameter q, we can avoid checking the same transition which has been judged already.

When the system evolves according to the repaired transitions sequence, if the firable transition number p equals N, the system would come to the object state eventually; otherwise if p< N, the system would enter into the deadlock state.

C. Transition controlled timed S^3PR nets

In order to force the timed S^3PR nets model to run automatically according to the friable transition sequence σ ($\tau_1, \tau_2, ...\tau_p$) (p is the number of firable transitions) represented by the repaired chromosome, it is necessary to modify the original Petri nets model. In the original timed S^3PR nets model, place C_i, directed arc (τ_i, C_i) and (C_i, τ_{i+1}) are added between transitions τ_i and τ_{i+1} $(i = 1,...p-1)$. It is noted that if τ_i and τ_{i+1} belong to the transitions of the same job, then there is no need to add place and arcs. The modified Petri nets model is called *Transition controlled timed S^3PR nets*, and it turn out to be a net without conflict. The transitions in transition controlled timed S^3PR nets can be fired according to the first p firable transitions in the given transition sequence. The method can be referenced from [16].

Suppose in the first p firable transitions, the number of transitions related to the same job J_i is s_i, and the firing time of transition t_{ij} is tr_{ij} ($i = 1,...n$, $j = 1,...s_i$). Since conflict phenomenon is avoided in transition controlled timed S^3PR nets, the firing times of the first p firable transitions are easy to get. If $p = N$, all the jobs would be finished, and the makespan is $\max(tr_{i(k(i)+1)})$; if $p < N$, the systems would come into the deadlock situation, and the time when deadlock happens is $t = \max(tr_{is_i} + x_{is_i})$. The whole processing time of unfinished operations $\sum_{i=1}^{n} \sum_{j=s_i+1}^{k(i)} x_{ij}$ can be used to construct the penalty items of infeasible solutions.

D. Computation of fitness:

The procedure of fitness computation can be represented as follows.

1) Given a chromosome, under the action of check and repair procedure, a new chromosome and the number of firable transitions p can be obtained.

2) Construct transition controlled timed S^3PR nets, and compute the firing time of the first p firable transitions.

3) If p equals N, the solution is feasible and fitness= $1/\max(tr_{i(k(i)+1)})$. If p< N, the solution is infeasible and a penalty item should be added to the fitness computation.

fitness= $1/(\max(tr_{is_i} + x_{is_i}) + k \cdot \sum_{i=1}^{n} \sum_{j=s_i+1}^{k(i)} x_{ij})$ where k is an adaptable parameter.

From the computation procedure of fitness, we can see that the smaller the makespan of feasible solutions, the larger the fitness. For the infeasible solutions, they are not discarded by the genetic algorithm directly, and can be included in the populations for a long time. This is because the infeasible solutions may have some fine structures that are important for genetic operations, so the infeasible chromosomes are allowed to exist in the populations. However, their fitness is decreased by adding penalty items.

IV. Design of Genetic Operations

A. Selection operation:

We employ the spinning roulette wheel selection method. The probability of individual being selected is $p_i = \frac{fi}{\sum_{i=1}^{S} fi}$,

where f_i is the fitness of individual i and S is population size.

B. Crossover operation:

Implementation of crossover can introduce new individuals by recombining current population. Based on the coding scheme introduced above, the classic crossover operators for traveling salesman problems cannot be applied here, such as partially mapped crossover and order crossover. So we design a special crossover operator by making full use of the information that the chromosomes have. The detailed algorithm is as follows.

1) Choose two individuals P1 and P2 as parents from the population randomly. Suppose that the number of firable transitions of P1 is $p1$ and the one of P2 is $p2$. Generate a random number $i \in [1, \min(p1, p2)]$ as the crossover point. Record the left part of P1 and P2 from gene 1 to gene i with strings T1 and T2 respectively.

2) For each gene in T1, search for a gene which has the same value with it in parent individual P2, and then replace the value of the gene with 0. After doing that, move all the genes which have the value of 0 in P2 to the left part, and maintain the relative sequence of other genes. P1 is altered in the same way.

3) Replace the left part of P1 with T2, and then we can get a new child individual Child1; by the same way, we can get another child individual Child2.

C. Mutation operation:

Mutation operation serves to maintain diversity in population. In this paper, mutation operation can't be implemented by changing a randomly selected gene, because this operation would bring us a meaningless chromosome. What's more, if the position of changed gene is after the last firable transition, then the chromosome after mutation operation is still an infeasible solution.

The mutation operation presented in this paper can be described as follows.

Given a parent individual P1, suppose the number of the firable transitions related to P1 is p. Select two mutation positions $i \in [1, p]$, $j \in [1, L]$ randomly. If the genes in the two positions are different, then exchange them; otherwise, select exchange positions again.

V. EXPERIMENTS

An commonly used example with 3 machine and 4 jobs which is cited from Ranmaswamy and Joshi [6] is shown in the following table.

Here we chose the following parameters: population 30, crossover probability 0.65, mutation probability 0.2, and parameter k 1. After 15 generations, we can get the optimal solution 512. Comparison of results obtained using the proposed method and others is given in table 2. It is obvious that the algorithm proposed in this paper is very efficient. Furthermore, compared with heuristic methods and mathematical programming methods, genetic algorithm is more fitted to handle large problems.

Here we give another three examples in which more jobs are added into the original system. Processing time and production sequence of jobs are shown in table 3. We select the first 6 jobs as the second example, the first 8 jobs as the third example, and all of the ten jobs as the fourth example. The scheduling results of genetic algorithm are shown in table 4. The Gantt graphs of different examples are shown in figure 3, figure 4 and figure 5. From the above results, we can see that scheduling algorithm proposed in this paper can solve large problems efficiently.

Table 1. Processing time of jobs

J_1	J_2	J_3	J_4
M_1:40	M_2:45	M_1:212	M_1:55
M_1:100	M_1:65	M_1:73	M_1:65
M_1:36	M_3:98	M_1:32	M_1:35

Table.2 comparison of results obtained using the proposed method and others

Method	Optimal makespan	CPU time (s)	Computing machine
Mathematical programming (Ramaswamy and Joshi 1996 [6])	512	0.71	IBM ES/3090-600S
Heuristic search based on Petri nets (Xiong and Zhou 1997 [10])	512	0.13	Sun SPARC 20
Genetic algorithm based on timed S3PR nets (Proposed in this paper)	Best: 512 Mean: 519	0.04	Celeron 800MHz

Table.3 Job information

J1	J2	J3	J4	J5
M_1:40	M_2:45	M_1:212	M_3:55	M_1:50
M_2:100	M_1:65	M_2:73	M_2:65	M_3:120
M_3:36	M_3:98	M_3:32	M_1:35	M_2:30
J6	J7	J8	J9	J10
M_2:95	M_3:155	M_2:15	M_3:55	M_2:20
M_1:50	M_1:55	M_3:45	M_2:85	M_1:45
M_3:40	M_2:75	M_1:50	M_1:25	M_3:95

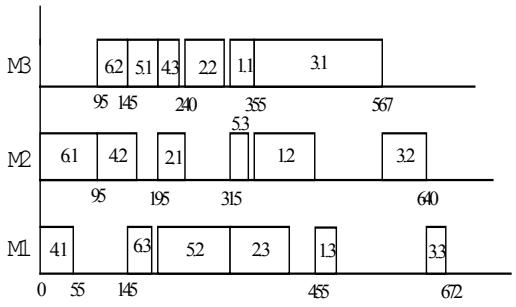

Figure.3 Gantt graph of 6 jobs

VI. CONCLUSION

In this paper, a new scheduling method based on the timed S^3PR nets and genetic algorithm is put forward in order to solve the scheduling problem of job shop without buffers. This algorithm checks up the feasibility of solutions through reachability analysis of Petri nets. The infeasible solutions can be improved in a certain extent through a repair procedure and the fitness of them should be reduced by adding the penalty items. Transition controlled timed S^3PR nets models are constructed according to the firable transitions represented by chromosomes. The fitness of chromosome can be easily acquired from the related transition controlled timed S^3PR nets. By taking full advantage of information in the chromosome, the appropriate genetic operators can be designed to gain fast convergence speed. Examples are offered in this paper to support the validity of the algorithm.

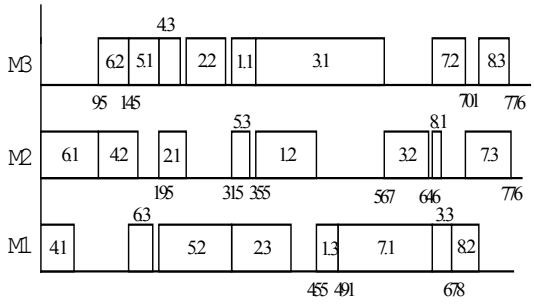

Figure.4 Gantt graph of 8 jobs

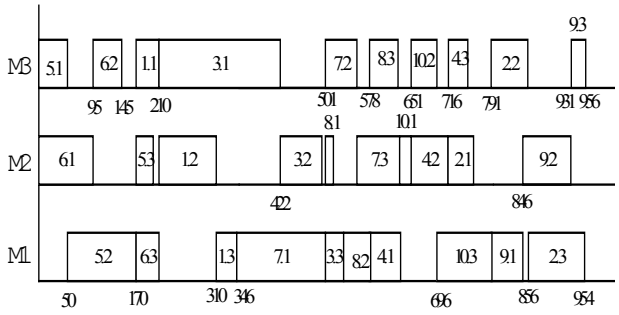

Figure.5 Gantt graph of 10 jobs

REFERENCES

[1] E.G., Coffman, M.J., Elphick, and A., Shoshani, "System deadlocks", ACM Computing surveys, vol.3, 1971, pp. 67-78.

[2] J., Ezpeleta, J. M., Colom, and J., Martinez, "A Petri net based deadlock prevention polity for flexible manufacturing system," IEEE Transactions on Robotics and Automation, vol. 11, no.2, 173-184, 1995.

[3] M. A., Lawley, S. A. Reveliotis, and M., Ferreira. "A correct and scalable deadlock avoidance policy for flexible manufacturing systems," IEEE Transactions on Robotics and Automation, vol. 14, no. 5, 796-809, 1998.

[4] M. P., Fanti, G. Maione, and B. Turchiano, "Digraph-Theoretic Approach for Deadlock Detection and Recovery in Flexible Production Systems," Studies in Informatics and Control, vol. 5, no. 4, pp. 373-383, 1996.

[5] N. Q., Wu and M.C., Zhou, "Avoiding Deadlock and Reducing Starvation and Blocking in Automated Manufacturing Systems", IEEE Transactions on Robotics and Automation, vol. 17, no. 5, pp. 658-669, 2001..

[6] S. E., Ramaswamy, and S. B., Joshi, "Deadlock-free schedules for automated manufacturing workstations", IEEE Trans. on Robotics and Automation, Vol. 12, No. 3, 1996, pp. 391-400

[7] D.Y., Lee, and F., DiCesare, "Scheduling FMS Using Petri Nets and Heuristic Search," IEEE Trans. on Robotics and Automation, vol.10, no.2, pp123-132, 1994.

[8] I. B., Abdallah, and A., Elmaraghy, " Deadlock Prevention and Avoidance in FMS: A Petri Net based Approach", Int. J. Advanced Manufacturing Technology, vol. 14, n. 10, 704-715, 1998.

[9] M. D., Jeng, and S. C., Chen, "Heuristic search approach using approximate solutions of Petri net state equations for scheduling flexible manufacturing systems," International Journal of Flexible Manufacturing Systems, 1998, 10(2), 139-162

[10] H. H., Xiong, and M. C., Zhou, "A Petri net method for deadlock-free scheduling of flexible manufacturing systems", International Journal of Intelligent Control and Systems, 3(3), pp. 277-295, September 1999.

[11] Y., Mati, "Geometric approach and taboo search for scheduling flexible manufacturing systems" IEEE Transactions on Robotics and Automation, v 17, n 6, December, 2001, p 805-818

[12] M., Pinedo, Scheduling: Theory, Algorithm and Systems New Jersey, USA, Prentice Hall 1995.

[13] G., Xu, and Z. M., Wu, "Deadlock-free scheduling method using Petri net model analysis and GA search" Control Applications, 2002. Proceedings of the 2002 International Conference on , Vol 2 , 2002 ,P1153 -1158

[14] T., ElMekkawy, "Deadlock resolution in flexible manufacturing systems: a Petri nets based approach". Ph.D. thesis, Department of Industrial and Manufacturing Systems Engineering, University of Windsor, Windsor, Ontario, Canada

[15] Y. L., Chen, "Petri-net based hierarchical structure for dynamic scheduler of an FMS: rescheduling and deadlock avoidance" Proceedings - IEEE International Conference on Robotics and Automation, n pt 3, 1994, p 1998-2004

[16] B. C., Damasceno, and X. L., Xie, , "Petri nets and deadlock-free scheduling of Multiple-Resource Operations" Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, v 1, 1999, p I-878 - I-883

[17] Y., Mati, "A taboo search approach for deadlock-free scheduling of automated manufacturing systems," Journal of Intelligent Manufacturing, vol. 12, n 5-6, October, 2001, p 535-552.

[18] K., Takahashi, M., Yamamura, and S., Kobayashi, "GA approach to solving reachability problems for Petri nets, " IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, v E79-A, n 11, Nov, 1996, p 1774-1780

[19] W., Resig, "Petri nets: an introduction," Berlin, New York: Springer-Verlag, 1985.

[20] I. B., Abdallah, H. A., ElMaraghy, and T., ElMekkawy, "A logic programming ap proach for finding minimal siphons in S3 PR nets applied to manufacturing systems", Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, vol. 2, 1997, P 1710 -1715