Proceedings of the 2004 IEEE
International Conference on Robotics & Automation
New Orleans, LA • April 2004

Solving the FMS Scheduling Problem by Critical Ratio-Based Heuristics and the Genetic Algorithm

Tsung-Che Chiang and Li-Chen Fu

Dept. of Computer Science and Information Engineering, National Taiwan University,

Taipei, Taiwan, R.O.C.

r88008@csie.ntu.edu.tw, lichen@csie.ntu.edu.tw

Abstract—**This paper addresses the FMS scheduling problem. The objective concerned here is maximizing the meet-due-date rate. The authors propose two rules for job sequencing and job dispatching, two common subtasks in solving this problem. These two rules are designed based on the critical ratio values of jobs. We also propose a mechanism to obtain better performance than the stand-alone scheduling process via genetic algorithms. With the nature of design of the proposed job sequencing rule, the genetic algorithm is designed not only to improve the schedule quality but also to save computation time. All the proposed rules and idea are carefully examined through several different scenarios.**

I. INTRODUCTION

Production scheduling is an interesting research field with long history. The flexible manufacturing system (FMS) scheduling problem is a classical problem among the topics in this field. Theoretically it is a NP-hard problem, which means that there is no polynomial-time solution known to solve this problem. This property attracts many researchers interested in algorithm design from the academia. Meanwhile, engineers in the industry are also concerned about this problem since it can generally model a lot of real-world manufacturing industry.

In the literature, the FMS scheduling problem was usually solved by advanced searching mechanisms. Clyde [2] used a hybrid approach of genetic algorithms (GA) and filtered beam search (FBS) in a two-stage fashion. Yang [3] combined GA and dynamic programming (DP). Chambers [5] and Ponnambalam [6] adopted another well known searching technique, the tabu search (TS) algorithm. Solutions with GA as the core are also very popular [7 – 9]. Conducting searching on the Petri-nets [10, 11] was taken into considerations, too.

Although searching mechanisms like GA, TS and FBS are mature approaches, the searching space will still be too huge if we applied them directly in the granularity of every decision to make a schedule, especially when the scale (like the number of jobs) becomes large. Therefore, we need to seek general guidelines to help building a schedule efficiently. These guidelines, usually known as priority dispatching rules (p.d.r.) or scheduling rules, are pervasively used in the industry. The trend of researches on scheduling rules is toward integration of multiple rules [1, 4, 12, 13]. However, we should notice that the essence of these rules has very significant impact on the performance of their blend. Hence, authors devote to proposing two high-performance rules based on the "critical ratio" idea in this paper. They are shown to have performance much better than the existing rules, even as good as their blend.

In our design of the sequencing rule, it is parameterized to be flexible to control its computation time and solution quality. In this paper, we also propose a GA-based solution that is able to save computation time while keeping schedule quality by taking account of the time factor.

The remainder of this paper is organized as follows. Section II gives the detailed description of our target problem. In Section III, the scheduling mechanism and the proposed job sequencing and dispatching rules will be explained clearly. The design of the time-constrained genetic algorithm is provided in Section IV. Section V contains experiment results and discussions. Finally, the conclusions are given in Section VI.

II. PROBLEM FORMULATION

The FMS problem addressed in this paper can be described as follows:

- There are R routes. For each route r, it consists of s(r) ordered stages s_{r1}, s_{r2}, ... $s_{rs(r)}$. For each stage s_{rk}, it defines a corresponding capability c_{rk}. The capability means the ability to process a certain stage.
- There are M machines. For each machine m, it has at least one capability. In a FMS, a machine usually has several capabilities, and for one capability, there are usually several capable machines.
- There are N jobs. Each job is designated a route. A job is finished if all stages of its route are processed in order (aka precedence constraint) by machines with required capabilities.

- It takes processing time p_{mc} for a station m to process a job at a stage requiring capability c.
- It takes sequence dependent setup (SDS) time T_{kl} for a station to prepare processing a job at a stage requiring capability l right after processing a job at a stage requiring capability k.
- Each machine can process at most one job at a time. There's no preemption issue.
- Each job i has a due date d_i, which indicates the tolerable latest time at which this job should be finished. If a job is finished no later than its due date, we say it meets the due date; otherwise it is a tardy job.
- The primary objective is to maximize the meet-due-date rate of jobs.

III. Scheduling Mechanisms

Solving the FMS problem can be realized as allocations of machines over time for jobs under constraints like capability and precedence. This task is usually divided into two subtasks – job sequencing and job dispatching. The former deals with resource allocation by ordering jobs and let jobs occupy machines in that certain order. The latter is responsible for dispatching the job to an appropriate machine when it has more than one choice.

As we mentioned earlier, there are many methods to accomplish the above two subtasks like tree search, dynamic programming, genetic algorithms, etc. In this paper, we take priority rules as the means. The priority rule is a kind of guideline. The knowledge inside these guidelines is easy to realize, to transfer and to extend. Priority rules, in essence, solves the sequencing and dispatching problem by prioritizing candidates and taking the one with the highest priority. The paradigm of the priority rule-based solution can be generalized:

- When a job tracks into a stage, apply the job dispatching rule to prioritize all capable stations, and dispatch the job to queue at the best one.
- When a machine finishes a job, apply the job sequencing rule to prioritize all jobs queued there, and pick the best one to process next.

In the following two subsections, we will give the details of our proposed sequencing and dispatching rules.

A. Job Sequencing Rule

Before looking at our proposed rule, we list several conventional and famous rules first. These rules will be taken as the benchmarks in our experiments.

- Earliest Due Date (EDD): it gives the highest priority to the job with the earliest due date.
- Shortest Setup Time (SST): it gives the highest priority to the job requiring the shortest setup time if taking it as the next processing target.
- Shortest Processing Time (SPT): it gives the highest priority to the job requiring the shortest processing time for the current stage.
- Shortest Remaining Processing Time (SRPT): it first calculates the sum of processing time of unfinished stages of each job and then gives the highest priority to the job with the smallest sum.
- Critical Ratio (CR): the slack time is the time left from current time to the due date, and the critical ratio is obtained by dividing the remaining processing time by the slack time. This value is usually used to indicate the degree of urgency of a job. CR then gives the highest priority to the job with the largest critical ratio.

Taking a look at these rules, we can find that these rules determine the next job with consideration of only individual information of jobs. It can be expected that they can not provide very good decision with this small amount of information. In our solution, we propose a new sequencing rule that takes account of "group information" based on the concept of the CR rule. The entire procedure is given as follows:

Step0. A machine m now needs to determine which job in queue is the next one. Mark all queued jobs as unevaluated.

Step1. Take an unevaluated job i and assume it is the next job to process. Calculate when its current stage s will be completed by m, say the completion time is t.

Step2. Calculate the critical ratio of all queued jobs at time t.

Step2.1 For the job i, the remaining processing time is estimated by the sum of minimum processing time required to process stages after s. The slack time is $d_i - t$ where d_i is due date of job i.

Step2.2 For other jobs j, the remaining processing time is estimated by the sum of processing time required to process its current stage s' and minimum processing time required to process stages after s'. As for the slack time, it is $d_i - t - T$, where T is the sequence dependent setup time required to change from job i to job j.

Step3. Summing critical ratio values of all queued jobs in Step2, this value V_i is used to estimate the impact caused by taking job i as the next processing target.

Step4. Mark job i evaluated. If there are still unevaluated jobs, go back to Step1.

Step5. Prioritize queued jobs by giving the highest priority to the job with the smallest V_i.

The main idea of this rule is to pick the job such that the critical ratios of all jobs are kept minimal after processing it. It measures the influence upon all related jobs caused by processing the selected job instead of just making decision based on comparatively local information, what the traditional rules do.

To see its time complexity, given n jobs in queue, our rule will take time in proportion to n^2 while the traditional rules only take time in proportion to n. In order to save computation time, we add a filtering preprocess at the head of our rule. We first calculate critical ratios of all queued jobs. Then only the jobs with critical ratios in a predefined interval [L,U] will be considered in our rule. This preprocess can reduce the aforementioned number n and thus reduce the computation time.

Now let us describe the meaning of L and U. The goal of our rule is selecting a job which is itself urgent and processing it first will still keep all jobs slack. Therefore, for jobs with small critical ratios, we think that they are not eager to ask for processing and disregarding it should have little effect on making a good decision.

Reminding that our objective is to maximize the meet-due-date rate, and in some cases we find that pursuing to make very pressing jobs meet their due dates is to burn daylight and even more destroy the chance for other jobs to meet. Hence, we could set an upper bound of the critical ratio to filter out these urgent but hopeless jobs and expect doing that will save a considerable amount of time with the cost of light deterioration of solution quality.

Before ending this subsection, we resolve a detailed issue here – how can we calculate the critical ratio if the job is already overdue (the slack time is negative in this case)? In our rule, we use two predefined values B and D. B stands for the base value of the critical ratio if a job is overdue and D is the increment value added per delay time unit, say a day. Thus the critical ratio of a tardy job is

$$B + D\times(\text{delay\_time})/\text{delay\_time\_unit}$$

B. Job Dispatching Rule

After showing our job sequencing rule, we will describe our dispatching rule in this subsection. Before doing that, similarly, we list several pervasively used rules here. They are also taken as the benchmarks in our experiments.

- Minimum Queue Length (MQL): it gives the highest priority to the machine with the shortest queue.
- Shortest Processing Time (SPT): it gives the highest priority to the machine that can process the job with the shortest processing time.
- Shortest Setup Time (SST): it gives the highest priority to the machine that can process the job with the shortest setup time assuming the job is the next processing target.
- Minimum Workload (MWL): it gives the highest priority to the machine whose sum of processing time of queued jobs is the smallest.

These rules catch the information of either dynamic status of machines (MQL/SST/MWL) or static properties of machines (SPT/SST) and then make decisions accordingly. Since it is known that a decision could be made better with more information, we try to propose a new rule that adopts information combining dynamic status and static properties of both jobs and machines. The answer would be a little surprising – we take the CR rule, a rule that is not thought to be used for dispatching, as the basis of our dispatching rule, Minimum Critical Ratio (MCR). The entire procedure is shown below:

Step0. A job i is waiting to be dispatched to a machine at stage s at time t. Mark all capable machines as unevaluated.

Step1. Take an unevaluated machine m, if it is busy, calculate when it will finish the in-process job, say finish time f. If it is idle, set f equivalent to t.

Step2. Calculate the sequence dependent setup time T for machine m to process job i.

Step3. Estimate the remaining processing time rpt of job i – the sum of processing time of stage s on machine m and minimum processing time required to process remaining stages after s.

Step4. Calculate the slack time slt of job i by $d_i - f - T$, where d_i is the due date of job i.

Step5. Calculate the critical ratio by dividing rpt by slt. This value V_m will be used to decide which machine is the best choice.

Step6. Mark machine m evaluated. If there is still unevaluated machine, go back to Step1.

Step7. Give the highest priority to the machine with the smallest V_m.

From the procedure, we can easily realize the concept of this "minimum critical ratio" dispatching rule. It takes information including remaining processing time of jobs (dynamic status of jobs), due date of jobs (static property of jobs), processing time at current stage (static property of jobs and machines), setup time required (static property and dynamic status of machines) and finish time of machines (dynamic status). With such abounding information, our rule could dispatch the job to a very suitable machine.

IV. TIME-CONSTRAINED GENETIC ALGORITHM

In our testing process of the proposed job sequencing and dispatching rules, the meet-due-date rates of the resulting schedule are usually very satisfying. Only in few cases, the meet-due-date rates are not high enough. We soon think of the parameters in the job sequencing rule, L,U,B and D. After some manual testing, we found that the meet-due-date rate can be improved with appropriate tuning of values of these four parameters. Now the problem is – what are the appropriate values?

Determining the values can be easily formed as a searching problem. We are searching for a "good" quadruple of real numbers. Genetic algorithms have been known to be a simple, elegant and effective searching algorithms for a long time. There are so many applications of GA in the field of production scheduling, and the authors also have experience of applying it to solve semiconductor manufacturing scheduling problem. In our approach here, we again resort to this powerful tool to solve some particularly complicated problem instances.

GA was motivated by simulating the evolution process of the nature. To apply it on problem solving, first we need to encode solutions to be genomes. Then a group of genomes, aka a population, will evolve to be another population based on the fitness values of genomes. In GA, we usually say the population evolves from generation to generation. The fitness value of a genome represents the quality of its corresponding solution, and the evolution process consists of several genetic operators including selection, crossover, mutation and reproduction.

The design of a genetic algorithm usually comprises five parts – genome coding, initial population, fitness function, genetic operators, and genetic parameters. They are given as below:

1. *Genome encoding*: The structure of genome here is very concise. It has four genes representing the parameters L,U,B and D. The range of values of L,U and D is [0,1], and the range of value of B is [1,2]. They are real numbers.
2. *Initial population*: Here we use random creation. Values of four genes are generated randomly according to their ranges.
3. *Fitness function*: In this part, we proposed a new idea. In the traditional use of GA, evaluating genomes usually takes equivalent amount of computation time. However, in our case, different L and B would cause evaluation to take quite different amount of time, as we have mentioned in subsection III.A. In our design here, we take both meet-due-date rate and computation time as indices in the fitness function. To calculate the fitness value of a genome, first we obtain the normalized values of both indices based on the minimum and maximum values of the indices among all genomes in the population. Then the fitness value is the sum of two normalized values.
4. *Genetic operators*:
   - selection: we use the roulette wheel selection operator, which selects a genome with the probability in proportion to its fitness value.
   - crossover: we use the two-point crossover operator. It randomly picks two points and then exchanges the section enclosed by these two points between two selected parents
   - mutation: we use the single-gene mutation operator. It randomly picks one gene and sets its value as a randomly generated value in the range.
   - reproduction: we keep one-fourth best genomes, mate(selection/crossover/mutation) to generate one-half genomes and randomly create the remaining one-fourth genomes.
5. *Genetic parameters*:
   - population size and generation size : 10.
   - mutation rate: 5%.

V. Experiment Results

This section gives the experiment results on comparisons of our proposed approach with representative benchmarks. It will be presented in three subsections, which provide the results about our job sequencing rule (Enhanced Critical Ratio, ECR), job dispatching rule (Minimum Critical Ratio, MCR) and time-constrained GA (TCGA) in order.

A. The Effect of ECR

In testing ECR, we create three data sets "light", "medium" and "tight", each including ten problem instances. The general specification of these three data sets is 10 routes, each with 5 – 10 stages, 20 machines, processing time 0.5 – 4 hours, setup time 0.5 – 2 hours and 100 jobs. Since we focus on testing the effect of sequencing rules, here each capability is owned by only one machine. The ranges of due dates of jobs of three data sets are [20,30], [15,30] and [15,25].

The five benchmarks CR, EDD, SST, SRPT and SPT are already described. For the rule LC5, it means a linear weighted combination of five conventional rules. This is a well-known enhancement. Readers may read the authors' paper [1] to see the details. Here we set the same weights to each rule. As for ECR and ECR(0.2), the former has no filtering preprocess and the later sets the lower bound of the critical ratio of concerned jobs as 0.2. Total tardy hours are the sum of tardy hours of all jobs, and total difference hours are the sum of difference of job finish time and job due date of all jobs. They are given here as a reference to see the effects of our rule. All values in the following tables are average values among ten problem instances in each data set. A symbol * is used to denote the best rule for an index.

Table 1. Effects of ECR in the "light" data set

	meet-due-date rate	computation time (sec)	total tardy hours	total diff. hours
CR	0.77	2.96	1837	13324(*)
EDD	0.82	1.87	1821	20576
SST	0.79	2.22	2034	19409
SRPT	0.78	2.00	3532	26478
SPT	0.79	1.84(*)	2761	21834
LC5	0.91	3.06	628(*)	22374
ECR	0.93(*)	18.35	753	14668
ECR(0.2)	0.90	2.11	1170	17744

Table 2. Effects of ECR in the "medium" data set

	meet-due-date rate	computation time (sec)	total tardy hours	total diff. hours
CR	0.65	2.76	5405	12299(*)
EDD	0.74	1.59(*)	4309	17915
SST	0.67	1.98	4820	17666
SRPT	0.71	1.89	5911	24225
SPT	0.74	1.83	4983	20267
LC5	0.83	3.04	2256	18770
ECR	0.89(*)	18.61	2123(*)	12422
ECR(0.2)	0.83	3.15	3299	13452

Table 3. Effects of ECR in the "tight" data set

	meet-due-date rate	computation time (sec)	total tardy hours	total diff. hours
CR	0.54	2.32	5507	10460
EDD	0.67	1.35(*)	4313	11127
SST	0.57	1.62	5276	10399(*)
SRPT	0.65	1.48	7245	14818
SPT	0.65	1.55	4768	12882
LC5	0.74	2.60	2691(*)	21440
ECR	0.79(*)	14.5	3008	15136
ECR(0.2)	0.75	3.06	3819	14277

From the experiment results, we can see that our ECR rule always behave well on improving the meet-due-date rate. When the due dates of jobs get tighter, the performances of conventional rules appear to be worse. For the other two referenced measurements, total tardy hours and total difference hours, the ECR rule still performs well. It implies that this rule not only raises the meet-due-date rate in the factory but also keeps jobs finished just in time and thus reduces the cost of inventory management.

The only one item over which conventional rules have superiority is computation time. However, in our experience of cooperating with real-world manufacturers, twenty seconds and one second actually do not have huge difference to them. If the computation time is really critical, we are able to reduce computation time by our filtering preprocess while keeping good performance. This is what the last rows, ECR(0.2), in the three tables above tell us.

During cooperation with manufacturers, managers of production lines are always eager to know why the schedule looks like that. They are trying to interpret the schedule and understand the knowledge inside it. Readers may notice that LC5 is somewhat a good rule according to the four measures. It indeed is and the authors have utilized it before. The point is that it is difficult to explain it very well to users of the scheduler. What does it mean to combine five rules? How can we combine rules focusing on difference indices? The combination of rules enhances their ability but meanwhile makes it mysterious and blurs its meaning. The ECR rule does not have this problem. The concept of this rule is well acceptable by users. It follows the same idea from the head to the tail – taking care of the critical ratios.

B. The Effect of MCR

The data sets used to test our dispatching rule, MCR, are similar to those described in the previous subsection. The primary difference between three data sets "few", "some" and "many" is the number of capable machines for one capability. The value is 3, 4 and 5 for them respectively. We use these values to examine the effects of dispatching rules when there are "few", "some" and "many" choices during dispatching process. Besides, in the data set "few", processing time ranges from 0.5 to 3 hours, and in the other two data sets, it ranges from 0.5 to 7 hours. This is used to represent different degrees of the variety of machines.

The benchmarks are all described in subsection III.B. And the concerned performance indices are the same as what we have in testing the job sequencing rules.

Table 4. Effects of MCR in the "few" data set

	meet-due-date rate	computation time (sec)	total tardy hours	total diff. hours
MQL	0.94	3.08 (*)	341	6926
SPT	0.88	12.5	1629	6008
ST	0.78	9.63	2464	5889 (*)
MWL	0.95 (*)	3.42	233 (*)	7029
MCR	0.95 (*)	10.90	726	6436

Table 5. Effects of MCR in the "some" data set

	meet-due-date rate	computation time (sec)	total tardy hours	total diff. hours
MQL	0.33	3.38 (*)	6341	7138
SPT	0.72	13.31	4089	7398
ST	0.32	11.46	22834	24126
MWL	0.34	4.05	6126	6994 (*)
MCR	0.79 (*)	12.91	3537 (*)	7711

Table 6. Effects of MCR in the "many" data set

	meet-due-date rate	computation time (sec)	total tardy hours	total diff. hours
MQL	0.43	3.07 (*)	4357	5365
SPT	0.84	13.36	2009	8564
ST	0.34	14.68	25073	27329
MWL	0.41	3.74	4052	5141 (*)
MCR	0.91 (*)	13.10	1286 (*)	8592

The experiment results show that our MCR rule has the best performance in meet-due-date rate and total tardy hours. With more choices, this superiority is more significant. The rules MQL and MWL perform well in the data set "few". In that case the number of machines to choose is small and the difference between machines is not obvious, and thus distributing jobs evenly will be a good principle. Doing this is also helpful to decrease computation time required when applying the ECR sequencing rule since the number of queued jobs on each machine is approximately equivalent. However, when the number of choices gets large and machines become quite different from each other, these two rules both produce very bad outcomes.

C. The Effect of TCGA

Here we want to examine the consequence resulted after we bring the computation time into the fitness function. We pick ten problem instances from the aforementioned six data sets that the stand-alone ECR rule gives low meet-due-date rates. Now we put focus on meet-due-rates and computation time. Values given are the average among ten problem instances. Three scheduling approaches are used – stand-alone ECR, ordinary GA without considering computation time, and the proposed time-constrained GA.

Table 7. Effects of TCGA

	meet-due-date rate	computation time (sec)
ECR	0.656	18.16
ord-GA	0.720	254.01
TCGA	0.724	221.18

The experiment results show that our new idea can save around 12% computation time on average while the meet-due-date rate is kept as good as what the ordinary GA can do. Actually the time-constrained GA produces the same meet-due-date rates as the ordinary GA does in eight cases, and a higher rate and a lower rate for the other two cases.

VI. CONCLUSIONS

In this paper, we propose two rules for job sequencing and dispatching in solving FMS scheduling problem with maximizing meet-due-date rates as the main objective. These two rules take the critical ratio values of jobs as the basic consideration. Both rules are carefully examined in different scenarios and are shown to have very satisfying performance. In addition to their good performance, these two rules can be understood and realized easily and thus they are more acceptable by real-world users. For the complicated problem that the stand-alone scheduling process can not give good results, we propose a genetic algorithm-based approach to tune the parameters in the sequencing rule and improve the meet-due-date rates. We also propose a new idea to take account of computation time of evaluating genomes in the fitness function. Such idea is verified to be helpful to save substantial computation time through experiments.

References

[1] Jyh-Horng Chen, Li-Chen Fu, Ming-Hung Lin, and An-Chih Huang, "Petri-Net and GA-Based Approach to Modeling, Scheduling, and Performance Evaluation for Wafer Fabrication," IEEE Trans. on Robotics and Automation, vol.17, no.5, pp.619 – 636, October 2001.

[2] Clyde W. Holsapple, Varghese S. Jacob, Ramakrishnan Pakath and Jigish S. Zaveri, "A Genetics-Based Hybrid Scheduler for Generating Static Schedules in Flexible Manufacturing Context," IEEE Trans. on Systems, Man, and Cybernetics, vol. 23, no.4, pp.953 – 972, July/August 1993.

[3] Jian-Bo Yang, "GA-based Discrete Dynamic Programming Approach for Scheduling in FMS Environments," IEEE Trans. on System, Man and Cybernetics, part B., vol.31, no.5, pp.824 – 835, October 2001.

[4] Yung-Feng Chiu and Li-Chen Fu, "A GA Embedded Dynamic Search Algorithm over a Petri Net Model for an FMS Scheduling," Proc. IEEE Int. Conf. Robotics and Automation, pp.513 – 518, April 1997.

[5] John B. Chambers and J. Wesley Barnes, "Flexible Job Shop Scheduling by Tabu Search," Graduate Program in Operations Research and Industrial Engineering, The University of Texas at Austin, Technical Report Series, 1996.

[6] S.G. Ponnambalam, V. Ganapathy, S. Saravana Sankar and R. Karthikeyan, "Scheduling Flexible Manufacturing Systems using Tabu Search Method," Proc. IEEE ICIT, pp. 1043 – 1048, 2002.

[7] In Lee, Riyaz Sikora, and Michael J. Shaw, "A Genetic Algorithm-Based Approach to Flexible Flow-Line Scheduling with Variable Lot Sizes," IEEE Trans. on Systems, Man and Cybernetics, part B., vol.27, no.1, pp.36 – 54, February 1997.

[8] Christian Bierwirth and Dirk C. Mattfeld, "Production Scheduling and Rescheduling with Genetic Algorithms," Evolutionary Compuation, vol.7, no.1, pp.1 – 17, 1999.

[9] Li Wang and Dawei Li, "A Scheduling Algorithm for Flexible Flow Shop Problem," IEEE Proc. Intelligent Control and Automation, pp. 3106 – 3108, June 2002.

[10] Yeong-Dae Kim, Jae-Gon Kim, Bum Choi and Hyung-Un Kim, "Production Scheduling in a Semiconductor Wafer Fabrication Facility Producing Multiple Product Types with Distinct Due Dates," IEEE Trans. on Robotics and Automation, vol17, no.5, pp.589 – 598, October 2001.

[11] Antonio Reyes Moro, Hongnian Yu, and Gerry Kelleher, "Hybrid Heuristic Search for the Scheduling of Flexible Manufacturing Systems Using Petri Nets," IEEE Trans. on Robotics and Automation, vol. 18, no.2, pp. 240 – 245, April 2002.

[12] S.C. Huang and J.T. Lin, "An interactive scheduler for a wafer probe center in semiconductor manufacturing," International Journal of Production Research, vol. 36, no. 7, pp. 1883 – 1900, 1998.

[13] Bo-Wei Hsieh, Shi-Chung Chang, Chun-Hung Chen, "Dynamic Scheduling Rule Selection for Fab Operations," Proceedings of 2000 Semiconductor Manufacturing Technology Workshop, pp. 202 – 210, June 2000.