ELSEVIER

European Journal of Operational Research 126 (2000) 567–586

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Theory and Methodology

# Analysis of reactive scheduling problems in a job shop environment

I. Sabuncuoglu [a,*], M. Bayız [b]

[a] *Department of Industrial Engineering, Faculty of Engineering, Bilkent University, 06533 Ankara, Turkey*
[b] *Kiran Consulting Group, San Diego, CA, USA*

Received 1 August 1998; accepted 1 April 1999

**Abstract**

In this paper, we study the reactive scheduling problems in a stochastic manufacturing environment. Specifically, we test the several scheduling policies under machine breakdowns in a classical job shop system. In addition, we measure the effect of system size and type of work allocation (uniform and bottleneck) on the system performance. The performance of the system is measured for the mean tardiness and makespan criteria. We also investigate a partial scheduling scheme under both deterministic and stochastic environments for several system configurations. 

*Keywords:* Reactive scheduling; Beam search; Job shop scheduling

## 1. Introduction

Scheduling is an important element of production systems because it serves as an overall plan on which many other shop activities are based. By properly planning and timing of shop floor activities, various system performance measures can be optimized. There are two key elements in any scheduling system: schedule generation, and revisions (monitoring and updating the schedule). The first element which acts as a *predictive* mechanism determines planned start and completion times of operations of the jobs. The second element which is viewed as the *reactive* part of the system monitors the execution of the schedule and copes with unexpected events (i.e., machine breakdowns, tool failures, order cancelation, due date changes, etc).

The major criticism brought against the predictive mechanisms in practice is that the actual events on the shop floor can be considerably different compared to the one specified in the schedule due to the random interruptions (i.e., breakdowns, scraps, due date changes, order cancelations, etc.). Thus an appropriate corrective action (or response) should be taken to improve the performance of the degraded schedule. Beside other environmental factors, the nature of a response depends on the way that a schedule is generated (or types of scheduling decisions are made). In the scheduling literature, there are two

* Corresponding author. Tel.: +90-312-266-4477; fax: +90-312-266-4126.
*E-mail address:* sabun@bilkent.edu.tr (I. Sabuncuoglu).

0377-2217/00/$ - see front matter 
PII: S0377-2217(99)00311-2

main scheduling approaches: *off-line scheduling* in which all available jobs are scheduled all at once for the entire planning horizon and *on-line scheduling* in which scheduling decisions are made one at a time when needed. In the on-line approach, the schedule is not determined in advance, but it is constructed over time as the system state changes. Thus, any disturbance can automatically be handled by this event based scheduling system. In the off-line approach, however, there is an a priori generated (predictive) schedule which needs to be revised whenever necessary. These revisions can be performed in several ways, ranging from repair of the existing schedule to generating a new schedule. In this study, we develop and compare several such schedule revision techniques.

Generally speaking, schedules are easily generated by using on-line dispatching rules. But the solution quality is sacrificed due to the myopic nature of these rules. On the other hand, the optimum seeking off-line approaches search in a larger solution space and hence generate high quality schedules at a cost of greater computation times. One of the objectives of this study is to compare these two scheduling methods (i.e., simple heuristic vs more sophisticated optimum seeking method) so that we get some insights into their relative strengths and weaknesses in different environmental conditions.

In general, the majority of the published works in the scheduling area deals with the task of schedule generation. The second part (reactive element) has not been studied well in the literature. In this context, this paper provides an important contribution towards the analysis and better understanding of the reactive scheduling problems.

Another point is that a system with a larger number of machines and jobs is more complex and thus the scheduling problems are more difficult than small systems. However, it is not generally known in the literature how the relative performance of the scheduling methods is affected by this system size factor. In addition, we suspect that the work load distribution in the system (uniform loading vs nonuniform loading or bottleneck system) may also affect the relative performance of scheduling systems. One can intuitively expect better scheduling decisions in the uniform system (long-term utilization of machines are nearly the same) than that of the nonuniform system. However, how these two schedule generation methods (off-line vs on-line) perform in such manufacturing environments is again an open research question. Finally, there is a need to compare the scheduling methods under stochastic disturbances so that we assess their strengths and weaknesses.

The purpose of this paper is to investigate all these issues stated above. Specifically, we analyze the effects of the load allocation (bottleneck vs uniform), system complexity (small vs large), and stochasticity (breakdowns vs not) on the performance of the on-line and off-line scheduling methods. Moreover, we develop and compare several reactive policies. We also investigate the feasibility of using partial scheduling in both deterministic and stochastic environments.

The rest of the paper is organized as follows. We present a literature review in Section 2. This is followed by the discussion on system considerations and experimental conditions in Section 3. In Section 4, the scheduling methods are compared in deterministic and stochastic environments. In Section 5, we investigate a periodic response scheduling policy. We then study partial scheduling in Section 5.2. Finally, we make concluding remarks in Section 6.

## 2. Literature review

As discussed earlier, the majority of the published literature in the scheduling area deals with the task of schedule generation or predictive nature of the scheduling problems. But, reactive scheduling and control is also important for the successful implementation of scheduling systems. In what follows, we review the research papers that are related to reactive scheduling.

In order to provide more organized information about the existing studies, we propose a classification scheme based on seven attributes (see Table 1). We use three main divisions *environment*, *schedule generation* and *implementation* of reactive policies which define the characteristics of the problems. In the environment part, we have shop floor type, job arrival information and source of

Table 1
Classification of the papers in reactive scheduling

| Author | Environment | | | Schedule generation | | Implementation | |
|---|---|---|---|---|---|---|---|
| | Shop floor | Job arrival | Stochasticity | Method | Objective function | When | How |
| Yamamoto and Nof, 1985 | Job shop | Static | Machine breakdown | Branch and Bound | Makespan | Event driven (MB) | Full new schedule |
| Church and Uzsoy, 1992 | Single machine | Dynamic | No | EDD | L max | Periodic and Event driven (urgent jobs) | Full new schedule |
| Holloway and Nelson, 1974 | Job shop | Static | Process time variation | HSP | Tardiness rel. perform. M. | No | Initial Full schedule |
| Nelson et al., 1977 | Job shop | Dynamic | Process time variation | HSP | Tardiness rel. perform. M | Periodic | Full new schedule |
| Ovacik and Uzsoy, 1994 | Single machine | Dynamic | No | Algorithm based on B&B | L max | After scheduling $\lambda$ jobs | Partial |
| Kiran et al., 1991 | FMS | Static–dynamic | No | Multipass heuristic (FH) | Tardiness rel. perform. M. | None periodic | Full new schedule |
| Kim and Kim, 1994 | FMS | Semi-dynamic | Machine break. Urgent job | Dispatch rules | mean FT & T combination | Periodic & Event driven | Full new schedule |
| Matsuura et al., 1993 | Job shop | Semi-dynamic | Mach. break., Specif. change, Rush jobs | B&B, FCFS, SPT | Makespan | After first disruption | Full & Job selection |
| Muhleman et al., 1982 | Job shop | Dynamic | Machine break., Process time variation | Dispatch rules | FT, MT, PL, CMT | Periodic | Full new schedule |
| Farn and Muhleman, 1979 | Single machine | Dynamic | No | DR & Heuristics based on TSP | Changeover time | Periodic | Full new schedule |
| Bean et al., 1991 | Multiple resource | Static | Mach. break., Unavail. tool | MUSA | Weighted total tard. | Event driven | Repair |
| Akturk and Gorgulu (1998) | Modified flow line | Static | Machine break. | RHSA | Earliness and tardiness | Event driven | Repair |
| Nof and Grant, 1991 | Small cell | Static | Machine break. & Unexpected order arrival | | Several | Performance based, Periodic | Full new sch., right shift, rerouting to alter. mac |

Table 1 (Continued)

| Author | Environment | | | Schedule generation | | Implementation | |
|---|---|---|---|---|---|---|---|
| | Shop floor | Job arrival | Stochasticity | Method | Objective function | When | How |
| Sabuncuoglu and Karabuk, 1997 | FMS | Static | Machine break. & Process time variation | Beam search and dispatch rule | Mean tardiness and makespan | Periodic | Full new schedule |
| Kutanoglu and Sabuncuoglu, 1994 | Job shop | Dynamic | Machine breakdown | All reroute, arrival reroute, queue reroute, no reroute | Mean weighted tardiness | Event driven | Dispatching rule |
| Kutanoglu and Sabuncuoglu, 1998a,b | Job shop | Dynamic | Machine breakdown, Process time var. | Iterative simulation | Mean weighted tardiness | Periodic | Dispatch rule selection |
| Lawrence and Sewell, 1997 | Job shop | Static | Processing times | Optimum and heuristic methods | Makespan | Operation completion | Full new schedule |
| Wu and Wysk, 1988 | FMS | Dynamic | No | Dispatching rules | Mean tardiness, Mean flow time | Periodic | Partial simulation window |
| Wu and Wysk, 1989 | FMS | Dynamic | No | Dispatching rules | Mean tardiness, Mean flow time | Periodic | Partial of simulation window |
| Jain and Foley, 1987 | FMS | Static | Machine breakdown | | Mean tardiness | Event driven | Rerouting |
| Bengu, 1994 | Flowline | Dynamic | Machine breakdown | ATC | Mean weighted tardiness | No | Job selection |
| Dutta, 1990 | FMS | Static | Breakdown, New jobs, Change in job priority | KB | Mean completion time, Mean machine utilization | Event driven | Rerouting, Preemption, etc. |

stochasticity attributes. In the job arrival attribute, semi-dynamic refers to the dynamic scheduling problem with a priori known ready times. Under schedule generation division, we specify the method to generate schedules and the objective function of the problem. In Table 1, there are abbreviations in the method attributes. These are the names of the scheduling methods given by authors in their papers. Finally, in the implementation section, we define when and how the reactive scheduling policies are employed. In *when* attribute, we specify the times at which system revision decisions are made. Under this heading, *event driven* refers to the rescheduling which is triggered in response to an unexpected event that alters the current system status. In the *periodic* policy, rescheduling is invoked at the beginning of each period. According to the *performance based* policy, rescheduling is triggered when the performance of the system considerably deviates from the planned performance. In the *how* attribute, the type of corrective action is given. Here, *full* new schedule means that all the available operations are rescheduled according to the current system status. *Partial* means that only a part of the current schedule is updated or a subset of all schedulable operations are scheduled. Job selection refers to the local scheduling decisions using dispatching rules.

The first study in this area is due to Holloway and Nelson (1974) who implement a multi-pass procedure (as described later in Nelson et al., 1977) in a job shop by generating schedules periodically. They concluded that a periodic policy (scheduling/rescheduling periodically) is very effective in the dynamic job shop environments. Later, Farn and Muhleman (1979) compared dispatching rules and optimum seeking algorithms for the static and dynamic single machine scheduling problems. Again, new schedules are generated periodically in a dynamic environment. Their results indicate that the best heuristic for a static problem is not necessarily the best for the corresponding dynamic problem. Muhleman et al. (1982) also analyze the periodic scheduling policy in a dynamic and stochastic job shop system. Their experiments indicate that more frequent revision is needed to obtain better scheduling performance.

Church and Uzsoy (1992) consider periodic and event driven (periodic revision with additional considerations on tight due date jobs) rescheduling approaches in a single machine production system with dynamic job arrivals. The results indicate that the performance of periodic scheduling deteriorates as the length of rescheduling period increases and event driven method achieve a reasonably good performance. Later, Ovacik and Uzsoy (1994) propose several rolling horizon procedures in a single machine environment with sequence dependent set-up. Kiran et al. (1991) propose another rolling horizon type heuristic for manufacturing systems. The experiments with their model in a dynamic environment indicate that the proposed heuristic performs well for several tardiness related criteria.

Yamamoto and Nof (1985) study a rescheduling policy in a static scheduling environment with random machine breakdowns. Rescheduling is triggered whenever a machine breakdown occurs. The results indicate that the proposed approach outperforms the fixed sequencing policy and dispatching rules. Similarly, Nof and Grant (1991) develop a scheduling/rescheduling system and analyze the effects of process time variation, machine breakdown and unexpected new job arrival in a manufacturing cell. In their scheduling system, monitoring is performed periodically and either rerouting to alternative machines or order splitting policies are activated in response to unexpected disruptions.

Bean et al. (1991) consider the rescheduling of the shop with multiple resources when unexpected events prevent the use of a preplanned schedule. The authors reschedule to *match-up* with the preschedule at some point in the future whenever a machine breakdown occurs. The match-up approach is compared with the no response policy and several dispatching rules. The results of the test problems indicate that the proposed system is more advantageous. Later, Akturk and Gorgulu (1998) apply this approach to the modified flow shop. The results indicate that the match-up approach is very effective in terms of schedule quality, computation times, and schedule stability.

Simulation based approaches are also widely reported in the scheduling literature. In this

studies, various control policies are tested by using simulation. For example, Wu and Wysk (1988, 1989) propose a multi-pass scheduling algorithm that utilize simulation to make scheduling decisions in an FMS. Specifically, the multi-pass scheduling system simulates the system for each alternative rule by using the current shop status information and selects the best one rule to implement. The results show that the multi-pass approach is considerably better than using a single rule for the entire horizon. Jain and Foley (1987) use the simulation methodology to investigate the effects of the machine breakdowns in an FMS. Their experiments indicate that rerouting is always a better policy.

Matsuura et al. (1993) study the problem of selection between sequencing and dispatching as a rescheduling approach in a job shop environment involving machine breakdowns, specification changes, and rush jobs. The authors propose a method that switches from sequencing to dispatching when an unexpected event occurs. Their results show that this combined approach performs very well. In another study, Kim and Kim (1994) develop a simulation based real time scheduling methodology for an FMS. In this system, there are two major components: a simulation module and a real time control system. The simulation module evaluates various dispatching rules and select the best one for a specified criterion. The real time control module monitors the shop floor and a new schedule is generated at the beginning of each period when there is a major disturbances in the system. In another study, Bengu (1994) develops a simulation based scheduler that uses the up-to-date information about the status of the system and improve the performance of a scheduling rule Apparent Tardiness Cost (ATC) under dynamic and stochastic production environments.

Kutanoglu and Sabuncuoglu (1994) compare four reactive scheduling policies under machine breakdowns. The policies (all rerouting, arrival rerouting, queue rerouting and no rerouting) are tested using a job shop simulation model. A material handling system (MHS) is also considered in the model. The results show that the all rerouting is preferred reactive policy when the MHS is ignored. In the later study, Kutanoglu and Sabuncuoglu (1998a,b) propose an iterative simulation based scheduling mechanism for dynamic manufacturing environments. The authors test the proposed method by using multi-pass rule selection algorithm and lead time iteration algorithm in both deterministic and stochastic environments. The results indicate that the iterative improvement procedure improves the performance of the dispatching rules significantly. Later, Sabuncuoglu and Karabuk (1997) study the scheduling rescheduling problem in an FMS environment. The authors propose several reactive scheduling policies to cope with machine breakdowns and processing time variations. Their results indicate that it is not always beneficial to reschedule the operations in response to every unexpected event and the periodic response with an appropriate period length can be quite effective in dealing with the interruptions.

The reactive scheduling problems are also studied by using knowledge based and artificial intelligence (AI) techniques. For example, Dutta (1990) develops a knowledge based (KB) methodology to perform real time production control in FMS environments. The proposed mechanism monitors the system and takes a corrective action whenever a disruption event occurs. The author considers machine failures, dynamic introduction of new jobs and dynamic increases in job priority as shop floor disruptions. The results show that the KB mechanism with such corrective actions renders effective and robust production control. There are also other AI based studies in the literature. Among them, ISIS developed by Fox and Smith (1984) and OPIS proposed by Smith et al. (1990) are the most well-known systems. Other AI or KB systems are can be found in Szelke and Kerr (1994).

There are other studies that investigate scheduling problem under certain stochastic events and variations. He et al. (1994) examine the effect of processing time variation (PV) on the dispatching rules and find that the relative performances of the rules remain the same under PV. In a recent study, Lawrence and Sewell (1997) compare optimum and heuristic methods in a job shop environment when processing times are uncertain. The results indicate that dynamic scheduling heuristics (i.e.,

dynamically updated heuristic sequences) perform better than the static optimum schedules for even moderate amount of processing time uncertainty. The authors also report that the performance of simple heuristics converges to that of optimum seeking methods (even sometimes better) as uncertainty increases. This may be the reason why the practitioners often use simple scheduling methods.

## 3. The proposed study

In this section, we first describe the on-line and off-line scheduling approaches. Then we explain the job shop system and experimental conditions.

### 3.1. Scheduling methods

The off-line scheduling method used in this study is a heuristic algorithm which is based on the filtered beam search. This search method is an approximate Branch and Bound method (i.e., breadth-first-search without backtracking). Unlike breadth-first-search, it moves downward from a certain number of best promising nodes. The solution space is explored by heuristics that estimate a certain number of best paths by permanently pruning the rest. The degree of pruning is controlled by the beamwidth parameter that indicates the number of solutions saved at each level of the search tree. The number of nodes generated is controlled by the filterwidth parameter. By that way, only a subset of nodes are generated and the rest are filtered out by the local evaluations function. The remaining nodes are subject to global evaluation.

The structure and thorough analysis of this search technique are given in Sabuncuoglu and Bayiz (1999). Based on the results of this study, for the mean tardiness criterion, we use the active schedule generation scheme with the local evaluation function of Modified Operation Due Date (MODD) priority rule and the global evaluation function of the Shortest Processing Time (SPT) dispatching heuristic. For the makespan criterion, however, we use the nondelay schedule generation method with the local evaluation function of Most Work Remaining (MWR) priority rule and the global evaluation function of the MWR dispatching heuristic. As the on-line scheduling method, SPT and MWR dispatching rules are used for the mean tardiness and makespan criteria, respectively.

The off-line scheduling method described above is a constructive algorithm. Hence, operations are scheduled sequentially in the forward direction similar to the Branch and Bound method. This feature of the beam search algorithm allows us to generate partial schedules. We define *partial scheduling* as the one that does not schedule all the operations of the jobs but rather a subset of schedulable operations in the system. In general, the length of a partial schedule is important since it affects both the schedule quality and CPU time requirements. This length can be measured by either in terms of clock time or number of operations. In this study, we prefer the latter approach in order to be consistent with the definition of the period length defined in Section 3.4. According to this approach, for example, a partial schedule of half length corresponds to the case in which half of the operations are scheduled at a time.

### 3.2. Job shop environment

A classical job shop is used in this paper. We assume that the jobs are available for processing at time zero. Number of operations of a job are drawn from a discrete uniform distribution between 5 and 15. Processing times are generated from a discrete uniform distribution between 20 and 80. We assume that preemption is not allowed and set-up times are included in the processing times. Number of machines and jobs in the system determines the size of the problem. Since one of the purposes of this study is to examine the effect of problem complexity on the performance of the algorithms, we use four different sizes of the problem (Table 2).

Note that the large system configurations (Cases 2 and 4) are obtained from Cases 1 and 3 by increasing the number of machines with a factor of two. In order to make both the small and large

Table 2
Sizes of the shop analyzed

| Case | Number of jobs | Number of machines |
|---|---|---|
| 1 | 9 | 6 |
| 2 | 18 | 12 |
| 3 | 12 | 6 |
| 4 | 24 | 12 |

systems comparable, we try to achieve the same work load per machine. Hence, we increase the number of jobs in the large systems. The following expression is used to compute the average load per machine:

$$E[\text{Work load/machines}] = \frac{E[\text{Process time}] \times E[\text{Number of operations}] \times \text{Number of jobs}}{\text{Number of machines}}.$$

Two types of problem instances are generated: uniform and nonuniform. In the uniform case, machine loads are nearly the same. In the nonuniform case, processing times are multiplied by constants. For example, in the system with 6 machines processing times are multiplied with the coefficients of 0.7, 0.8, 0.9, 1.1, 1.2, 1.3, respectively. In the shops with 12 machines, processing times on two machines are multiplied with the coefficients of 0.7, 0.8, etc. By this perturbation, the speed of some machines are decreased to form bottleneck stations whereas others are accelerated. But, the total work load in the uniform and nonuniform systems are kept the same.

### 3.3. Machine breakdowns

In this study, we use the busy times approach to model machine breakdowns. This method allows the machine to break down when it is busy. A random up time is generated from a busy time distribution and the machine operates until its total accumulated busy time reaches the end of that time. When a failure occurs, a repair time is generated and the machine is kept down during this time period. After that, another up time is generated from the busy time distribution.

In the absence of real data, Low and Kelton (1991) recommends the Gamma distribution as a busy time distribution with a shape parameter of 0.7 and a scale parameter to be specified. The authors also state that Gamma distribution with the shape parameter of 1.4 is appropriate for the down time distribution. In this framework, the level of machine breakdowns is measured by efficiency level which gives the long run ratio of busy time to sum of busy and down time. In our experiments, we use the 90% efficiency level with 360 minutes of mean busy time and 40 minutes of mean down time.

### 3.4. Frequency of scheduling

In response to machine breakdowns or other unexpected interruptions in the system, we can either take no action (i.e., use the fixed sequence previously established by hoping that the system recovers from the undesirable effects of interruptions by itself) or reschedule the system from scratch at every machine breakdown (i.e., continuous rescheduling). The former approach has the disadvantage that alternative schedules might improve the system performance. The disadvantage of the latter approach is that too frequent schedule revision can increase the system nervousness and computational times.

Between these two extremes policies we can use so-called periodic rescheduling as an alternative approach. In this approach, system is continuously monitored but the necessary actions are taken periodically by considering the unscheduled operations and the current system status. First issue in periodic scheduling is to determine an appropriate period length. *Fixed time* or *variable time interval* approaches can be used for this purpose. In this study, we use variable time method. According to this approach (Sabuncuoglu and Karabuk, 1997), the system is monitored at each time increment and if the cumulative processing time realized on all machines in the system reaches a multiple of the specified length of the period, then rescheduling is triggered at this point. In the fixed time interval method, however, the period length is solely determined by the absolute clock time. The variable

time interval approach has some advantages over the fixed interval approach. First, since we use busy time method to model machine breakdowns, the probability of breakdowns is the same in each scheduling period in this approach. Also, this method divides the entire scheduling horizon into equal intervals in terms of processing times so that amount of schedule executed is the same in each time interval. By this way, we can measure the level of responsiveness of the rescheduling without being affected by the system load or any other scheduling factor.

We use 10 levels of frequency of scheduling in our experiments. These are 0, 2, 4, 6, 8, 10, 12, 14, 16, 1000. Here, 0 corresponds to no rescheduling (fixed sequencing) case in which a schedule is generated at the beginning of the horizon and the sequence determined in this preschedule is used through the scheduling horizon regardless of any future event. If a machine breakdown event occurs, then the unexecuted operations on this machine are simply right shifted for the duration of down time. Another extreme level is 1000 that represents the continuous rescheduling. In this case, rescheduling frequency is so high that reschedule is triggered at any event that alters the system status (i.e., machine breakdown, job completion, etc.). Between these two extreme cases, eight levels of periodic scheduling are analyzed. For instance, level 4 results in the schedule to be revised approximately four times during the makespan of the schedule. However, if there is no machine breakdown during the period, the schedule is not revised.

The scheduling algorithms and simulation model are coded in the C language. Experiments are conducted in the Unix environment with Sun Spark 2 stations.

## 4. Computational results

In this section, the two algorithms (i.e., on-line and off-line methods) are applied to the scheduling systems of different sizes. For each system size, the uniform and nonuniform loading schemes are also considered. In the experiments, 10 randomly generated problems are used at each experimental condition. The averages of 10 replications are presented in the tables. Both the makespan and mean tardiness criteria are used to evaluate the performance of the scheduling methods.

In the first phase, the analysis is performed in the deterministic environment. In the second phase, a stochastic environment is created by including machine breakdowns and the analysis is repeated under the new conditions.

The results are presented in Tables 3 and 4 for the mean tardiness and makespan criteria, respectively. In these tables, % Diff. 1 stands for the absolute difference between the solutions of the off-line and on-line algorithms. Diff. 2 represents the difference between solution quality in deterministic and stochastic environments. Diff. 3 is the difference between solution quality of algorithms for uniform and nonuniform systems in the deterministic environment. Finally, Diff. 4 represents the differences for the stochastic environment. Paired $t$-test is used to determine if the differences are significant. In the tables, '*' indicates that respective term is statistically significant at $\alpha = 0.05$.

### 4.1. Deterministic environment

The analysis is first performed for the mean tardiness criterion. As seen in Table 3, the differences (Diff. 1) are statistically significant. The off-line method (based on beam search algorithm) performs better than on-line method (the SPT rule) under all the experimental conditions. This is due to the fact that the optimum seeking methods should yield better results than simple myopic rules under the static and deterministic conditions where the most of the assumptions of optimization algorithms hold. We also note that the mean tardiness values are higher in the larger systems than small ones (e.g., the off-line algorithm produces the mean tardiness of 76.32 and 90.76 for the 9 jobs 6 machines and 18 jobs 12 machines, respectively). This may be due to slightly lower machine utilization achieved in the large systems.

The results also indicate that the difference between the scheduling methods (off-line vs on-line) are not much effected by the number of machines. In contrast, as the number of jobs in the system

Table 3
Mean tardiness results

| | | | Deterministic | Stochastic | Diff. 2 | Diff. 3 | Diff. 4 |
|---|---|---|---|---|---|---|---|
| 9 Jobs 6 Machines | Uniform | Algorithm | 76.32 | 118.9 | 42.58* | | |
| | | Dispatch | 123.7 | 157.22 | 33.52* | | |
| | | Diff. 1 | 47.38* | 38.32* | | | |
| | Nonuniform | Algorithm | 127.55 | 169.97 | 42.52* | 51.23* | 51.07* |
| | | Dispatch | 187.68 | 221.29 | 33.61* | 63.98* | 64.07* |
| | | Diff. 1 | 60.13* | 51.32* | | | |
| 18 Jobs 12 Machines | Uniform | Algorithm | 90.76 | 138.94 | 48.18* | | |
| | | Dispatch | 132.58 | 168.12 | 35.54* | | |
| | | Diff. 1 | 41.82* | 29.18* | | | |
| | Nonuniform | Algorithm | 119.38 | 172.08 | 52.70* | 28.62* | 33.14* |
| | | Dispatch | 166.93 | 204.2 | 37.27* | 34.35* | 36.08* |
| | | Diff. 1 | 47.55* | 32.12* | | | |
| 12 Jobs 6 Machines | Uniform | Algorithm | 171.32 | 242.52 | 71.20* | | |
| | | Dispatch | 231.37 | 280.14 | 48.77* | | |
| | | Diff. 1 | 60.05* | 37.62* | | | |
| | Nonuniform | Algorithm | 220.38 | 288.26 | 67.88* | 49.06* | 45.74* |
| | | Dispatch | 291.68 | 339.30 | 47.62* | 60.31* | 59.16* |
| | | Diff. 1 | 71.30* | 51.04* | | | |
| 24 Jobs 12 Machines | Uniform | Algorithm | 212.61 | 286.78 | 74.17* | | |
| | | Dispatch | 279.29 | 335.35 | 56.06* | | |
| | | Diff. 1 | 66.68* | 48.57* | | | |
| | Nonuniform | Algorithm | 245.35 | 312.98 | 67.63* | 32.74* | 26.20* |
| | | Dispatch | 317.78 | 370.53 | 52.75* | 38.49* | 35.18* |
| | | Diff. 1 | 72.43* | 57.55* | | | |

increases, (9 jobs 6 machines vs 12 jobs 6 machines), we observe an increasing trend in the absolute differences between the scheduling methods. Specifically, the off-line method performs better than the on-line approach in the more crowded systems. This is due to the fact that there are a large number of alternative schedules to search through by the off-line algorithm when there are more jobs to schedule.

Another observation is that in the nonuniform system the mean tardiness performance is worse than that of the uniform system. This is due to the bottleneck machines which delay job completion times in the nonuniform system. We also note that the performance of dispatching rules are affected more severely than the off-line algorithm. However, this degradation is less significant in the large systems (51.23 for 9 jobs 6 machines, 28.62 12 jobs 12 machines).

We also note that the difference between the performance of the off-line algorithm and the dispatching rules gets larger for all the shop size combinations in the nonuniform shops. In general, we also notice that the off-line algorithm performs better than the dispatching rules when the variability in processing times is high (i.e., variations between processing times of different jobs are relatively high). This makes sense because schedules developed by different algorithms will be similar if the processing times of jobs are similar. In terms of the CPU times, the on-line method (i.e., dispatching rule) is very fast since it generates schedules by only evaluating a priority function. It is on the average $10^{-2}$ seconds for 9 jobs 6 machines system and $6 \times 10^{-2}$ seconds for 24 jobs 12 machines system. As these numbers show from the smallest to the largest system, the CPU times only change with a factor of 6. On the other hand, the CPU times of the off-line algorithm vary significantly according to the system size. Average CPU times are 12.2, 31.5, 98.7, 260.6 seconds for 9 jobs 6 machines, 12 jobs 6 machines, 18 jobs 12

Table 4
Makespan results

| | | | Deterministic | Stochastic | Diff. 2 | Diff. 3 | Diff. 4 |
|---|---|---|---|---|---|---|---|
| 9 Jobs 6 Machines | Uniform | Algorithm | 1048.2 | 1119.2 | 71* | | |
| | | Dispatch | 1109.2 | 1180.7 | 71.5* | | |
| | | Diff. 1 | 61* | 61.5* | | | |
| | Nonuniform | Algorithm | 1203.5 | 1292.4 | 88.9* | 155.3* | 173.2* |
| | | Dispatch | 1284.6 | 1350.4 | 65.8 | 175.4* | 169.7* |
| | | Diff. 1 | 81.1* | 58* | | | |
| 18 Jobs 12 Machines | Uniform | Algorithm | 1114.3 | 1218.8 | 104.5* | | |
| | | Dispatch | 1171.8 | 1253.9 | 82.1* | | |
| | | Diff. 1 | 57.5* | 35.1* | | | |
| | Nonuniform | Algorithm | 1225.7 | 1317.3 | 91.6* | 111.4* | 98.5* |
| | | Dispatch | 1310.7 | 1374.3 | 63.6* | 138.9* | 120.4* |
| | | Diff. 1 | 85* | 57* | | | |
| 12 Jobs 6 Machines | Uniform | Algorithm | 1237.9 | 1372.3 | 134.4* | | |
| | | Dispatch | 1315.4 | 1400.5 | 85.1* | | |
| | | Diff. 1 | 77.5* | 28.2 | | | |
| | Nonuniform | Algorithm | 1361.8 | 1472.1 | 110.3* | 123.9* | 99.8* |
| | | Dispatch | 1438.7 | 1521.7 | 83* | 123.3* | 121.2* |
| | | Diff. 1 | 76.9* | 49.6* | | | |
| 24 Jobs 12 Machines | Uniform | Algorithm | 1437.4 | 1562.8 | 125.4* | | |
| | | Dispatch | 1514.2 | 1611.1 | 96.9* | | |
| | | Diff. 1 | 76.8* | 48.3 | | | |
| | Nonuniform | Algorithm | 1590.2 | 1714.2 | 124* | 152.8* | 151.4* |
| | | Dispatch | 1679.3 | 1740.4 | 61.1* | 165.1* | 129.3* |
| | | Diff. 1 | 89.1* | 26.2 | | | |

machines, 24 jobs 12 machines, respectively. This means that that as the number of jobs increases with a factor of 4/3, the CPU times increases about 2.5 times. This is consistent with the $O(n^3)$ complexity of the beam search algorithm. Note that the CPU times of the scheduling methods are not comparable ($6 \times 10^{-2}$ seconds for the dispatching rule and 260.6 seconds for the beam search method). But the absolute difference between the scheduling methods is only 66.68 (which corresponds to 31.36%) in the larger system.

When the simulation experiments are repeated for the makespan criterion, we observe that the results of the makespan case are very similar to those of the mean tardiness case. One exception is that degradation in the schedule in the nonuniform load allocation is almost equal for both the off-line algorithm and the dispatching rule (see Diff. 1 values in Table 4). Moreover, the difference between the scheduling methods gets larger in the nonuniform shops for all the system sizes.

### 4.2. Stochastic environment

In this section, we consider the stochastic case with random machines breakdowns. The breakdowns occur according to the busy time approach as discussed earlier. We first study the effects of machine breakdowns on the relative performance of the scheduling methods (dispatching rules representing the on-line scheduling approach and the beam search algorithm as the off-line scheduling method) at various levels of the system complexity and work load allocation types.

As summarized in Tables 3 and 4, the effects of system size and work allocation on the relative performance of the scheduling methods in the stochastic environment are very similar to the deterministic case. Specifically, the off-line algorithm performs better than the on-line method. Again, the performance measures are worse in the large systems than the small systems due to the increasing system complexity. Similar to the

deterministic environment, the performance of the uniform case is better than the nonuniform case. We also note that the difference between the scheduling methods increases for all the system sizes in the nonuniform systems.

An interesting observation is that the performance of the on-line scheduling rules degrades less than the off-line scheduling algorithm in the stochastic environment (refer to Diff. 2 values which are smaller for dispatching rules in all the problem sets). This means that the dispatching rules are quite robust to variability and uncertainty in the system. This result is consistent with that of Lawrence and Sewell (1997) who also observe that there is not much difference between the optimum methods and heuristics when uncertainty in processing times is high. Similar observations are also made by the studies of Sabuncuoglu and Karabuk (1997) and Yamamoto and Nof (1985) in that the potential benefits of using optimum seeking methods diminish as the system experiences random interruptions (i.e., machine breakdowns). From these results, one can infer that the effort to reduce the variability and uncertainty in the systems might worth more than the difficulties in using more sophisticated algorithms.

In our experiments, the number of breakdowns in the shop with 12 machines is about two times larger than the system with 6 machines (there are on the average 7.2, 16, 10 and 18.9 machine breakdowns in the shops with 9 jobs 6 machines, 18 jobs 12 machines, 12 jobs 6 machines and 24 jobs and 12 machines, respectively). Hence, one can intuitively expect that schedules of the large systems are affected more than the small systems. However, we did not observe this phenomenon in our experiments. This is probably due to the additional slack in the schedule. The system with 12 machines has twice more slack than the system with 6 machines. Therefore, the amount of deterioration in the schedule remains approximately the same in the both systems (42.58 for 9 jobs 6 machines, 48.19 for 18 jobs 12 machines).

We also note that if there are more jobs in the system, the difference between the scheduling methods both in the deterministic and stochastic systems become more significant (42.58 for 9 jobs 6 machines, 71.20 for 12 jobs 6 machines). Thus as the number of jobs increase, the system gets more congested and the average utilization increases (0.677 for 9 jobs, 0.728 for 12 jobs). Thus, machine breakdowns considerably affect the quality of schedules in the system with more number of jobs.

In the nonuniform system, the same pattern of changes are observed (Table 3). Specifically, the performances of the deterministic and stochastic cases get closer to each other in both small and large systems. As the number of jobs increases, the system performance is affected more severely from the machine breakdowns in the uniform case.

The results for the makespan criterion are given in Table 4. In general, the differences between the off-line and on-line algorithms in the stochastic environment for the 12 jobs 6 machines and 24 jobs and 12 machines systems are not statistically significant. In a way, this result verifies our previous observation that the performance of the off-line algorithm is affected more than the on-line method in a stochastic environment. The makespan results are very similar to the mean tardiness case (i.e., all the previous results hold for the makespan case.)

## 5. Periodic response and partial scheduling

In general, the on-line scheduling approach which employs simulation concept with the application dispatching rules takes into account machine breakdowns as they occur since the decisions are made one at a time as these stochastic events occur. In the off-line case, however, a predictive schedule needs to be revised to recover from the negative effects of the interruptions. In this section, we first study a periodic revision (or response) policy. Then, we present the results of partial scheduling as a part of the rolling horizon scheme.

### *5.1. Analysis of the periodic response policy*

The analysis is first conducted for the mean tardiness criterion and the uniform shop conditions. As seen in Fig. 1a and b, the performance of the off-line algorithm improves as the scheduling frequency increases. This is consistent with those

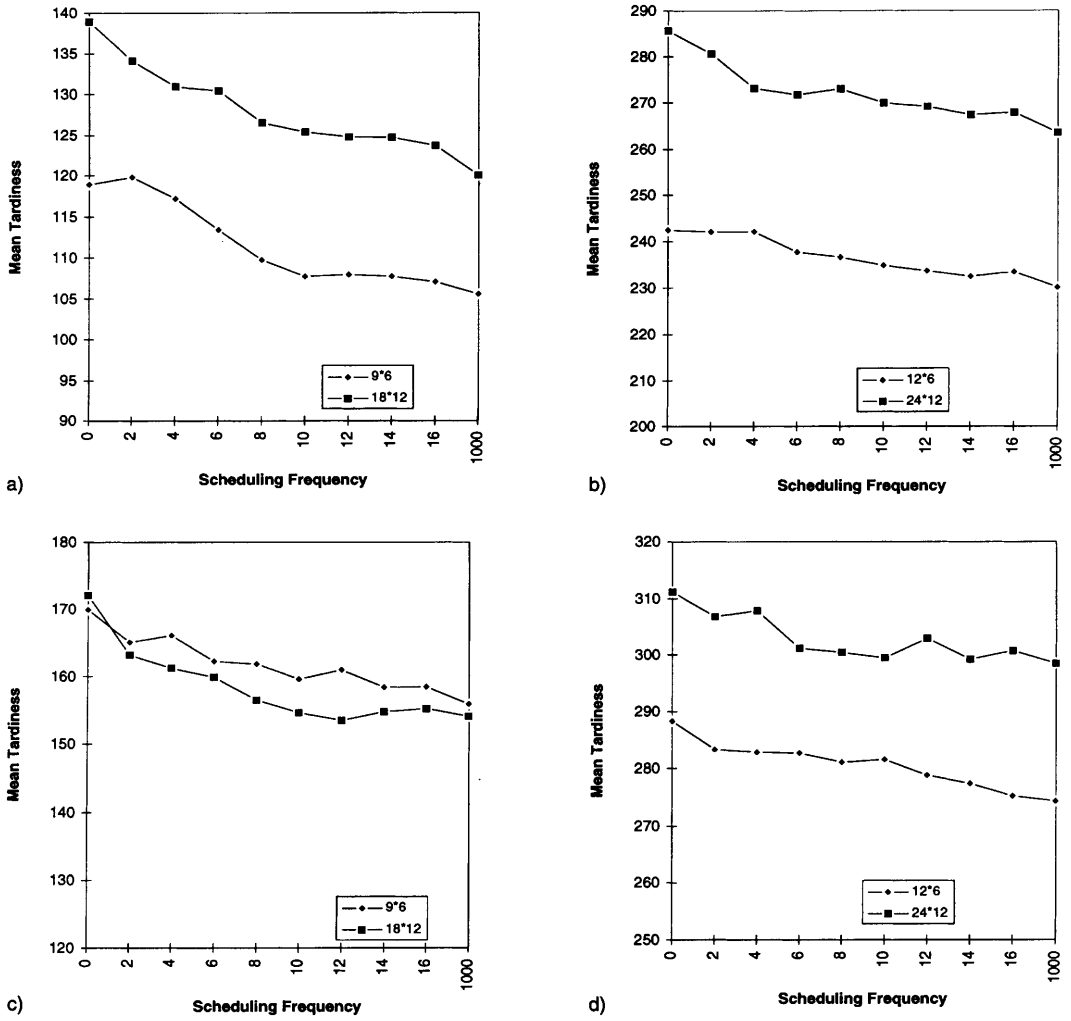


Fig. 1. Interactions between scheduling frequency and mean tardiness. (a) Scheduling frequency vs mean tardiness, uniform case. (b) Scheduling frequency vs mean tardiness, uniform case. (c) Scheduling frequency vs mean tardiness, nonuniform case. (d) Scheduling frequency vs mean tardiness, uniform case.

of Sabuncuoglu and Karabuk (1997) who analyses the scheduling/rescheduling problem in an FMS environment and those of Church and Uzsoy (1992) who study the problem for single machine scheduling. Note that the segmented line (representing the results of the small system) is flatter than the larger system (Fig. 1b). This is due to the fact that there are more number of machine breakdowns in the large systems and rescheduling the operations helps more to recover from the effects of these interruptions. This observation is less clear in Fig. 1a, but still the level of improvement is slightly better for the large system (see for example, 15.8% for 18 jobs 12 machines case vs 12.7% for 9 jobs 6 machines case).

Another observation is that the positive effect of rescheduling is more erratic in the nonuniform systems (Fig. 1c and d). The ups and downs of the mean tardiness in these systems may be due to the high variability in processing times. From the graphs in Fig. 1, one can see that the effect of rescheduling is less significant for the nonuniform

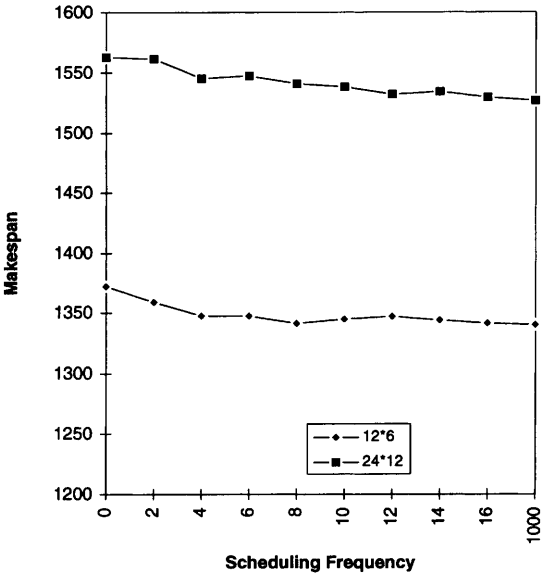


Fig. 2. Interactions between scheduling frequency and makespan values.

systems. Because, the excessive amount of slack in the nonuniform environment absorbs negative effects of breakdowns. In other words, the effect of machine breakdowns is less disruptive in the nonuniform environment unless the bottleneck machine experiences frequent failures.

Since the above results mostly hold for the makespan criterion, we present the results for only one experiment condition. As seen in Fig. 2, the makespan is not significantly improved as the scheduling frequency increases since the number of alternatives generated by the nondelay schedule generation scheme of the beam search algorithm are not too large for the problem under consideration. We also observe that the changes in the makespan values for the small systems are relatively less than those of the large systems and the effects of rescheduling draws more erratic behavior in the nonuniform environment as in the tardiness case.

### 5.2. Analysis of partial scheduling

As stated before, the partial scheduling method is implemented as a part of the periodic response policy. In the experiments, we set the partial schedule length by taking into account the scheduling frequency because it should at least be enough to cover the period length (i.e., total processing time of the scheduled operations should at least be equal to the period length). The resulting partial scheduling system with a periodic response is implemented in a rolling horizon scheme i.e. at each scheduling point a partial schedule with certain length is generated and used until the next decision point. During any period, if all the scheduled operations are executed and the next scheduling point has not been reached (i.e., the length of the schedule is not enough to cover the period), the scheduling scheme is triggered to generate a new partial schedule at this point in time.

To conduct simulation experiments, we created the same experimental environment (system size and uniform vs nonuniform) used in the previous section. We analyze the effects of partial scheduling for two levels of scheduling frequency. We choose one level for low frequency (4) and one level for the high frequency (14). Recall that for a high frequency level, the period lengths should be shorter than those of a low frequency. Therefore, we use more number of partial schedule alternatives for the scheduling frequency level of 14 than the level 4. Specifically, we use 1/10, 1/8, 1/6, 1/4, 1/2, 1 as the partial schedule lengths for the scheduling frequency level of 14. Since the period lengths are long at the scheduling frequency of level 4, we only use 1/3, 1/2, 1 partial schedule lengths for that level. Here, 1/3 means that at each scheduling point, 1/3 of the total number of operations are scheduled and needless to say, 1 refers to generating complete schedules.

The effect of partial scheduling is first measured for the mean tardiness criterion. The results of the analysis for the uniform case is shown in Fig. 3. These graphs show the changes in the mean tardiness and CPU times as a function of partial schedule length for both the low and high frequency levels. The first observation is that as the length of partial schedule increases the quality of the scheduling decisions improves regardless of the level of scheduling frequency. Because in the short lengths, myopic decisions are made by the partial

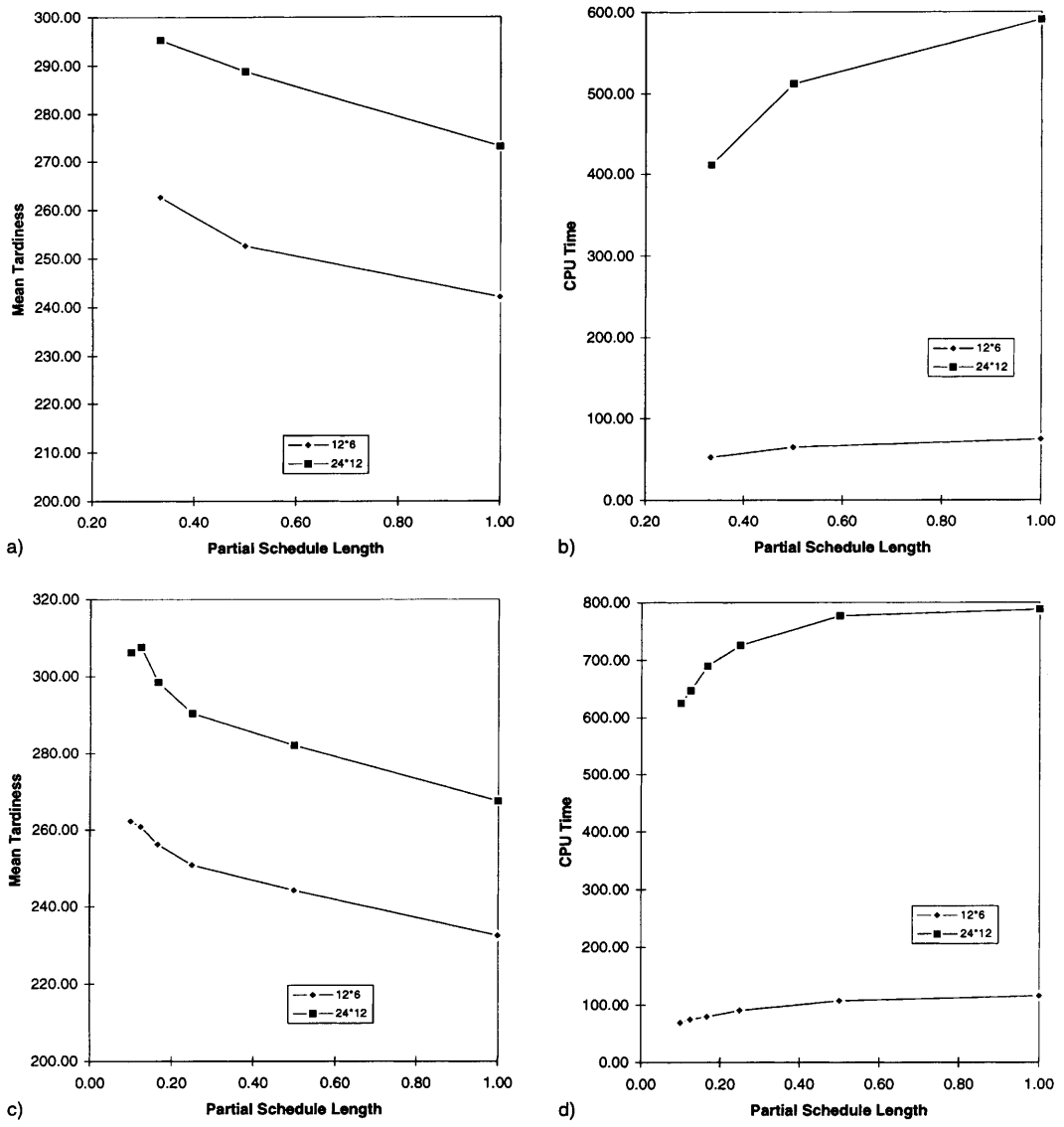


Fig. 3. Mean tardiness and CPU time as a function of partial schedule lengths (uniform case). (a) Frequency level of 4. Mean tardiness vs partial schedule length. (b) Frequency level of 4. CPU time vs partial schedule length. (c) Frequency level of 14. Mean tardiness vs partial schedule length. (d) Frequency level of 14. CPU time vs partial schedule length.

scheduling and this situation negatively effects the quality of schedules. Besides, additional idle times are inserted in the schedules since only a subset of all operations are scheduled and some idle time remains between the consecutive partial schedules.

We also observe that the mean tardiness does not linearly change with the partial schedule length. We have analyzed the effects of partial scheduling at two scheduling levels. The behavior of the mean tardiness is more informative at the scheduling frequency level of 14 (see Fig. 3c) since more alternatives are evaluated at this level. From Fig. 3c, it can be noted that for the short partial lengths (i.e., the lengths of 1/10 and 1/8), the amount of deterioration in the schedule quality is negligible. Because the number of operations in the

partial schedules are close to each other for the lengths of 1/10 and 1/8. Thus, the proposed partial scheduling system produce similar schedules with comparable performances. However, as the partial schedule length increases (i.e., the lengths of 1/6 and 1/4), we observe a significant improvement in the mean tardiness. The reason for that is the reduction of inserted idle times due to the increase in the partial schedule length. We also note that the marginal improvement in the mean tardiness gets smaller for the long partial schedule lengths (i.e., the lengths of 1/2 and 1). Because there is a small amount of inserted idle time that slightly worsens the schedule quality.

In summary, we conclude that the mean tardiness performance of the job shop system is significantly affected by the moderate level of partial schedule length rather than the short and very large partial schedule lengths.

Our experiments also indicate that the system complexity does not effect the performance of the partial scheduling system. We arrive at this conclusion because the mean tardiness lines are almost parallel for small and large systems as seen in Fig. 3a and c. In terms of CPU times, it increases as the partial schedule lengths increases (see Fig. 3b and d). This is an expected behavior because we somehow eliminate extra work to generate complete schedules at every scheduling period.

The pattern of segmented lines (Fig. 3b and d) in CPU times is consistent with the pattern of the mean tardiness case. The changes in the CPU times are also negligible for the small partial schedule lengths. Similar to the mean tardiness case, we note significant increases in the CPU times for the moderate lengths of the partial schedules. Marginal increases in the CPU times are also negligible for the long partial schedules. Because in the beam search algorithm, generating the first half of the search tree requires more computation time than the generating the other half of the tree since global evaluations takes less time at the higher levels of the search.

We also note that, by using the partial scheduling concept, the solution quality is sacrificed on the average 10.6% and 11.2% for 12 jobs 6 machines system and 24 jobs 12 machines system, respectively. However, we gain from the CPU times on the average 35.2% and 31.3% for small and large systems, respectively. In terms of percent differences, the gain of CPU times seems to be more significant than the degradation in the schedule quality. For that reason, Here, we advice to the practitioners to consider both the pros and cons of alternatives before determining a suitable policy.

The same analysis is performed in the nonuniform system. In order to compare the changes in the uniform and nonuniform cases, we plot all the results in the same graph. As seen in Fig. 4, the pattern of the segmented lines are the same in the both cases except that the mean tardiness values are slightly higher in the nonuniformly loaded systems. But the effect of partial scheduling on the system performance is the same under both the uniform and nonuniform environments.

Similar observations are made (Fig. 5). One difference is that deterioration in makespan is not significant for the small partial schedule lengths. This is probably due to the fact that the makespan performance measure is not considerably affected by the length of the partial schedule. In this case, an additional inserted idle time due to partial scheduling has less probability to delay the completion time of the job that determines the makespan of the schedule. Another reason is that a small amount of inserted idle time is resulted from the nondelay schedule generation scheme. Because this scheme tries to first assign the operation with the minimum starting time. Therefore, the distribution of operations among the machines are homogeneous in the partial schedules, which reduces the idle times. In addition, the scale of the makespan axis is too large that the changes in these values doe not look significant. Other observations are similar to the mean tardiness case.

In the proposed partial scheduling system, there are two key parameters (partial schedule length and scheduling frequency) that affect the solution quality. Since partial scheduling lengths are determined considering the scheduling frequency, these parameters are not independent. In the simulation experiments, the effects of partial schedule lengths on the solution quality are analyzed for the fixed scheduling frequency. To investigate the ef-

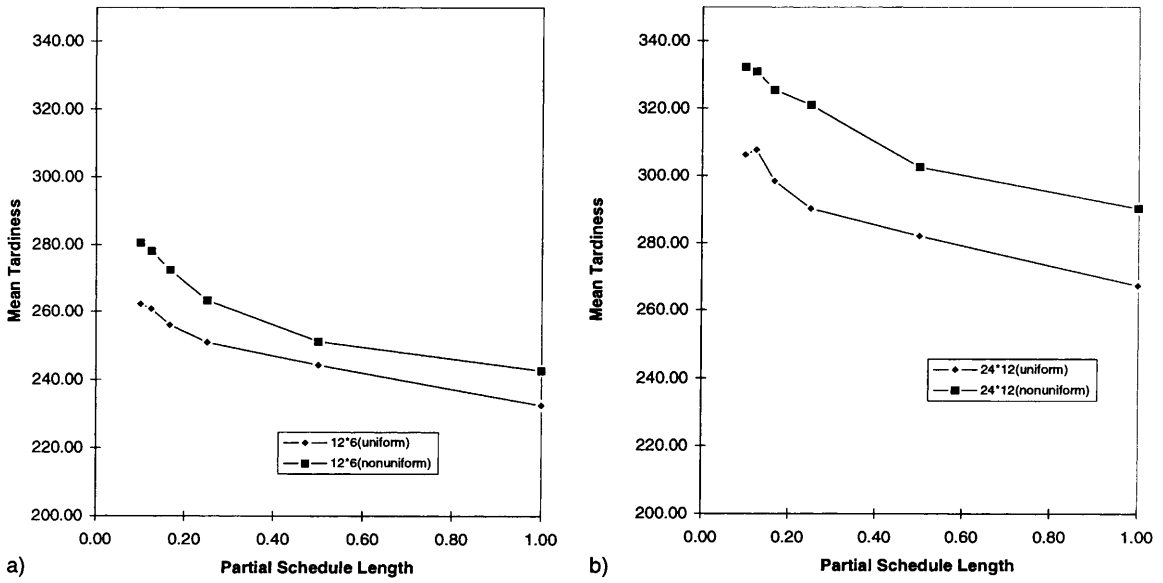


Fig. 4. Changes in mean tardiness as a function partial schedule length. (a) Mean tardiness vs partial schedule length. System size of 12 jobs and 6 machines. (b) Mean tardiness vs partial schedule length. System size of 24 jobs and 12 machines.

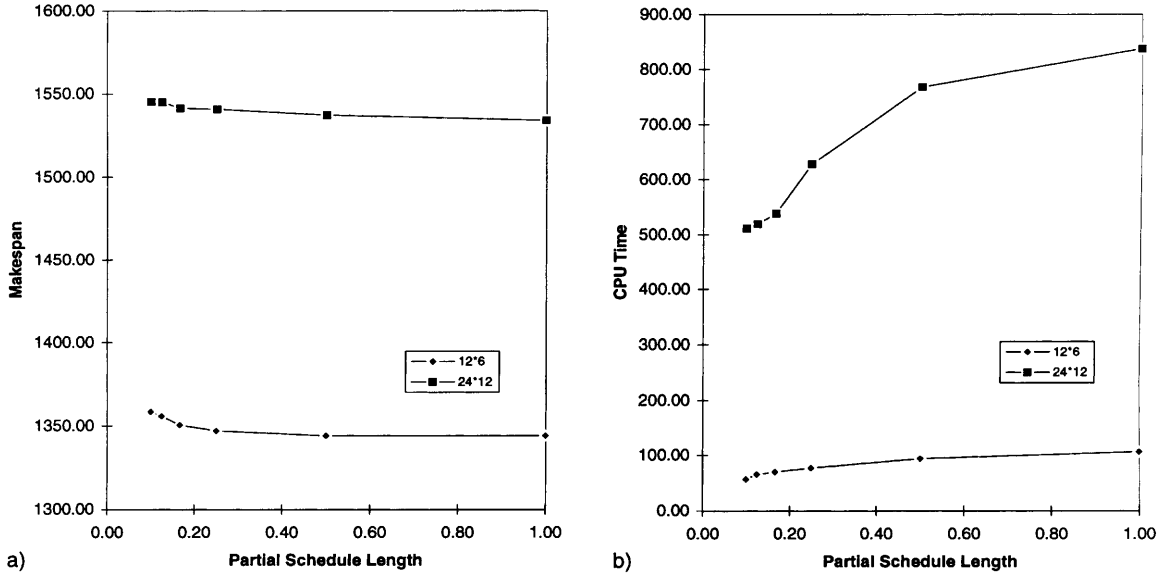


Fig. 5. Interactions between makespan and CPU time and partial schedule length (uniform case). (a) Frequency level of 14. Makespan vs partial schedule length. (b) Frequency level of 14. CPU time vs partial schedule length.

fects of scheduling frequency for the same partial schedule length, we compare the solutions of the partial schedule length of 1/2 at the scheduling frequency levels of 4 and 14. The results indicate that the solution quality at the level 14 is always better that the that of at the level 4 for each

problem. This observation confirms our previous finding that the solution quality improves as the scheduling frequency increases.

We also examine the effect of machine breakdowns on the solution quality of partial scheduling. For this purpose, additional simulation experiments are conducted for the 80% efficiency level (with 320 time units mean uptime and 40 time units mean down time). The results of both 90% and 80% efficiency levels are displayed in Fig. 6 for the mean tardiness measure. As expected the solution quality is worse in the 80% efficiency level than the 90% level since at this low level the system experiences longer down periods.

In summary, the quality of the schedule deteriorates as the length of the partial schedule decreases. This effect is more significant in the tardiness case than the makespan case. We also observe that the required CPU time decreases as the length of partial schedule decreases since the extra work to generate a complete schedule is eliminated. In general we observe that the percent gain in the CPU times is more than the percent loss in the objective function for each criterion.

## 6. Conclusion

In this paper, we studied the reactive scheduling problems and measure the effect of shop floor configurations (system size and load allocation) on the performance of scheduling methods (off-line and on-line scheduling methods). We also examine the effectiveness of partial scheduling in stochastic manufacturing environment. The following conclusions are drawn from this study:

First, we observe that the relative performances of the scheduling methods are not seriously affected by the systems size (i.e., system being big or small), but rather they are more affected by the system load (i.e., system being congested or not).

Second, we note that distribution of the load in the system have significant impact on the performance of the scheduling methods. Specifically, the optimization based off-line scheduling method performs better than on-line dispatching rules when the load across the machines are not uniform (i.e, there are bottleneck and/or underutilized machines in the system). These two observations also hold in stochastic environment.

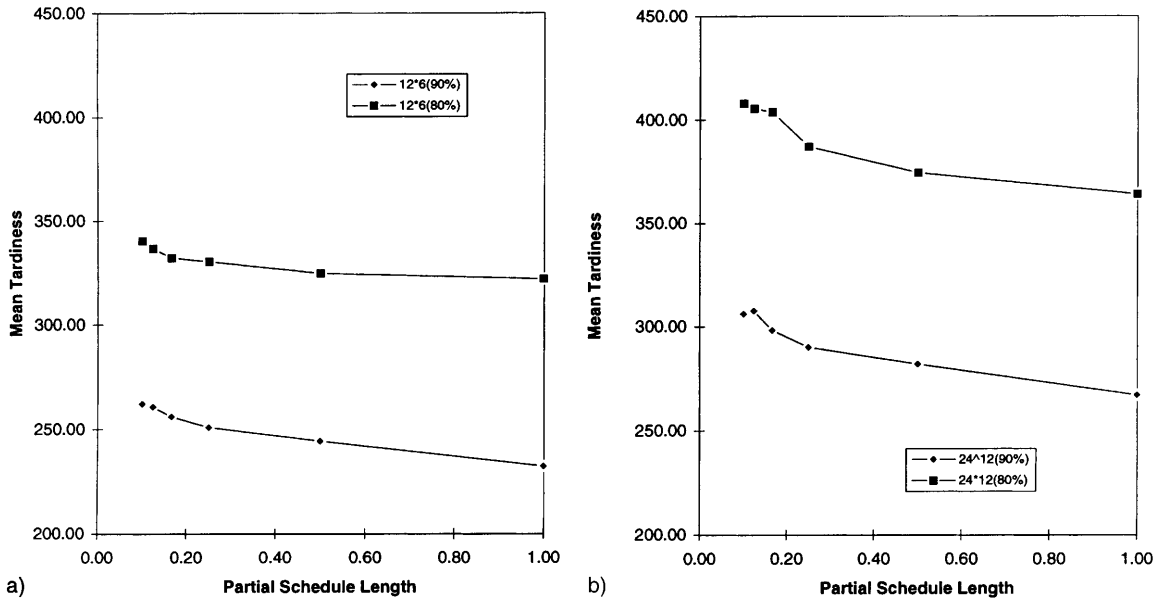


Fig. 6. Mean tardiness vs partial schedule length in 90% and 80% efficiency levels, uniform case. (a) Frequency level of 14, 12 jobs and 6 machines. Mean tardiness vs partial schedule length. (b) Frequency level of 14, 24 jobs and 12 machines. Mean tardiness vs partial schedule length.

Third, the off-line scheduling method is affected more than the on-line dispatching mechanisms when there is stochastic disturbances such as machine breakdowns. This is consistent with the previous studies (Lawrence and Sewell, 1997; Sabuncuoglu and Karabuk, 1997) that the performance of optimization methods and simple heuristics get close to each other when there is considerable uncertainty and variability in the system.

Fourth, under the stochastic disturbances the performance of the off-line scheduling method improves as the level scheduling frequency increases. But the marginal improvement is not significant for the high levels of scheduling frequency. The results also indicate that frequent scheduling (i.e., high level scheduling frequency) is more effective in the large systems with uniform load across the machines.

Fifth, partial scheduling with optimization based scheduling algorithms can be a very practical scheduling tool in a highly dynamic and stochastic environment. Even though the performance of the schedule can be inversely affected by the extra amount of inserted idle times and myopic characteristics of the partial scheduling decisions, the amount of deterioration in the schedule is not always very significant. Besides, the potential saving in CPU times is great when the partial scheduling can be employed. These observations are for each scheduling criteria under both uniform and nonuniform (i.e., bottleneck) system regardless of the system size.

We finally note that more research is needed in this area. An immediate extension would be testing the results of this paper under the dynamic job shop environment with various random disturbances. Another further research direction would be to investigate new schedule revision techniques together with considerations of the partial scheduling.

## References


Akturk, S., Gorgulu, E., 1998. Match-up scheduling under a machine breakdown. European Journal of Operational Research 112 (1), 80–96.

Bean, J., Birge, J.R., Mittenthal, J., Noon, C.E., 1991. Match-up scheduling with multiple resources, release dates and disruptions. Operations Research 39 (3), 470–483.

Bengu, G., 1994. A simulation-based scheduler for flexible flowlines. International Journal of Production Research 32 (2), 321–344.

Church, L.K., Uzsoy, R., 1992. Analysis of periodic and event driven rescheduling policies in dynamic shops. International Journal of Computer Integrated Manufacturing 5 (3), 153–163.

Dutta, A., 1990. Reacting to scheduling exceptions in FMS environments. IIE Transactions 22 (4), 300–314.

Farn, C.K., Muhleman, A.P., 1979. The dynamic aspects of a production scheduling. International Journal of Production Research 17 (15).

Fox, M.S., Smith, S.F., 1984. ISIS – A knowledge based system for factory scheduling. Expert Systems 1, 25–49.

He, Y., Smith, M.L., Dudek, R.A., 1994. Effect of inaccuracy processing time estimation on effectiveness of dispatching rule. In: Third Industrial Engineering Research Conference, pp. 308–313.

Holloway, C.A., Nelson, R.T., 1974. Job shop scheduling with due dates and variable processing times. Management Science 20 (9).

Jain, S., Foley W.J., 1987. Real time control of manufacturing systems with redundancy. Computer and Engineering (2).

Kim, M.H., Kim, Y., 1994. Simulation based real time scheduling in a flexible manufacturing systems. Journal of Manufacturing Systems 13 (2), 85–93.

Kiran, A.S., Alptekin, S., Kaplan A.C, 1991. Tardiness heuristic for scheduling flexible manufacturing systems. Production Planning and Control 2 (3), 228–241.

Kutanoglu, E., Sabuncuoglu, I., 1994. Experimental investigation of scheduling rules in a dynamic job shop with weighted tardiness costs. In: Third Industrial Engineering Research Conference, pp. 308–312.

Kutanoglu, E., Sabuncuoglu, I., 1998a. Simulation-based scheduling: Part 1: Background and literature review. Technical Report IEOR-9820. Department of Industrial Engineering, Bilkent University, Ankara.

Kutanoglu, E., Sabuncuoglu, I., 1998b. Simulation-based scheduling: Part 2: Experimental study. Technical Report IEOR-9821. Department of Industrial Engineering, Bilkent University, Ankara.

Lawrence, S.R., Sewell, E.C., 1997. Heuristic, optimal, static, and dynamic schedules when processing times are uncertain. Journal of Operations Management 15, 71–82.

Low, A.M., Kelton, W.D., 1991. Simulation Modeling and Analysis. Mcgraw-Hill, New York.

Matsuura, H., Tsubone, H., Kanezashi, M., 1993. Sequencing, dispatching, and switching in a dynamic manufacturing environment. International Journal of Production Research 31 (7), 1671–1688.

Muhleman, A.P., Lockett, A.G., Farn, C.K., 1982. Job shop scheduling heuristics and frequency of scheduling. International Journal of Production Research 20 (2), 227–241.

Nelson, R.T., Holloway, C.A., Wong, R.M., 1977. Centralized scheduling and priority implementation heuristics for a dynamic job shop model. AIIE Transactions 9 (1).

Nof, S.Y., Grant, F.H., 1991. Adaptive/predictive scheduling: review and a general framework. Production Planning and Control 2 (4), 298–312.

Ovacik, I.M., Uzsoy, R., 1994. Rolling horizon algorithms for a single machine dynamic scheduling problem with sequence dependent setup times. International Journal of Production Research 32 (6), 1243–1263.

Sabuncuoglu, I., Bayiz, M., 1999. Job shop scheduling with beam search. European Journal of Operational Research 118 (2), 390–412.

Sabuncuoglu, I., Karabuk, S., 1997. Analysis of scheduling–rescheduling problems in a stochastic manufacturing environment. Technical Report IEOR-9704. Department of Industrial Engineering, Bilkent University, Ankara.

Smith, S.F., Ow, P.S., Potvin, J.Y., Muscettola, N., Matthy, D., 1990. An integral framework for generating and revising factory schedules. Journal of the Operational Research Society 41 (6), 539–552.

Szelke, E., Kerr, R.M., 1994. Knowledge-based reactive scheduling. Production Planning and Control 5 (2), 124–145.

Wu, S.D., Wysk, R.A., 1988. Multipass expert control system – A control/scheduling structure for flexible manufacturing cells. Journal of Manufacturing Systems 7 (2), 107–120.

Wu, S.D., Wysk, R.A., 1989. An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. International Journal of Production Research 27 (9), 1603–1623.

Yamamoto, M., Nof, S.Y., 1985. Scheduling in the manufacturing operating system environment. International Journal of Production Research 23 (4), 705–722.