Journal of Manufacturing Systems
Vol. 20/No. 4
2001

Experimental Investigation of Iterative Simulation-Based Scheduling in a Dynamic and Stochastic Job Shop

Erhan Kutanoglu, Dept. of Industrial Engineering, University of Arkansas, Fayetteville, Arkansas, USA
Ihsan Sabuncuoglu, Dept. of Industrial Engineering, Bilkent University, Ankara, Turkey

Abstract

A vital component of modern manufacturing systems is the scheduling and control system, which determines companies' overall performance in their respective supply chains. This paper studies iterative simulation-based scheduling mechanisms for manufacturing systems that operate in dynamic and stochastic environments. Also assessed are the issues involved when these mechanisms are used to make higher-level scheduling decisions, such as dispatching rule selection, instead of generation of a full schedule. A typical simulation-based system is outlined and tested under various experimental conditions. Examined are the effects of stochastic events such as machine breakdowns and processing time variations on the system performance, and the effectiveness of the simulation-based approach from the control point of view is evaluated. Finally, different levels of two important factors (look-ahead window and scheduling period) are compared for the iterative approach. Computational results show that, although simulation-based scheduling proves effective when these parameters are properly set, the overall performance diminishes due to the dynamic and stochastic nature of the system, which degrades the multi-pass improvement capability of the simulation runs. Experimental results also support the initial expectation in that frequent updates to the higher-level schedule may not be necessary when these decisions are naturally "adaptive" to the unexpected system changes.

Keywords: *Scheduling and Control, Simulation Methods and Models*

Introduction

Effective production scheduling is becoming an increasingly important component of the supply chain environment that most companies face in today's competitive markets. Discrete-event simulation is a decision support tool that has been proposed to achieve effective scheduling. During the last decade, a significant body of literature has accumulated in this area, mainly either proposing simulation-based scheduling schemes or testing existing schemes in different settings. Most studies in this area share two common characteristics: (1) They use static and deterministic environments where all jobs are available for scheduling and no uncertainty is considered. In these cases, simulation is used as a search heuristic for improved scheduling decisions. (2) Simulation is mostly used to assist with constructing a *complete* schedule of all jobs rather than other types of scheduling decisions; however, a simulation-based scheduling scheme might perform differently in a dynamic and stochastic environment and/or when the main scheduling decision is different from constructing an off-line static and complete schedule. The main goal in this paper is to investigate how simulation-based schemes perform under dynamic and stochastic conditions through a comprehensive experimental study when simulation is used to identify scheduling policies rather than to generate a complete schedule.

Two questions are addressed: (1) Are simulation-based schemes still effective in a *dynamic and stochastic* environment? It is already known that the performance of "optimization-based algorithms" used to generate schedules fine-tuned (or even optimal) with respect to deterministic assumptions deteriorates quickly with the introduction of uncertainty (see Lawrence and Sewell 1997 for processing time uncertainty). This study will show if this is a valid conclusion for simulation-based methods, which are believed to be more flexible, adaptable, and realistic. (2) Is there a difference in the performance when simulation is used to make higher-level decisions? This study evaluates two simulation-based methods: (a) to select the best priority rule among candidates (rule selection), and (b) to fine-tune parameters of a heuristic (parameter tuning). From this perspective, simulation results will not generate a static complete schedule. One can interpret this as a mechanism to separate the higher-level (or more critical) policies

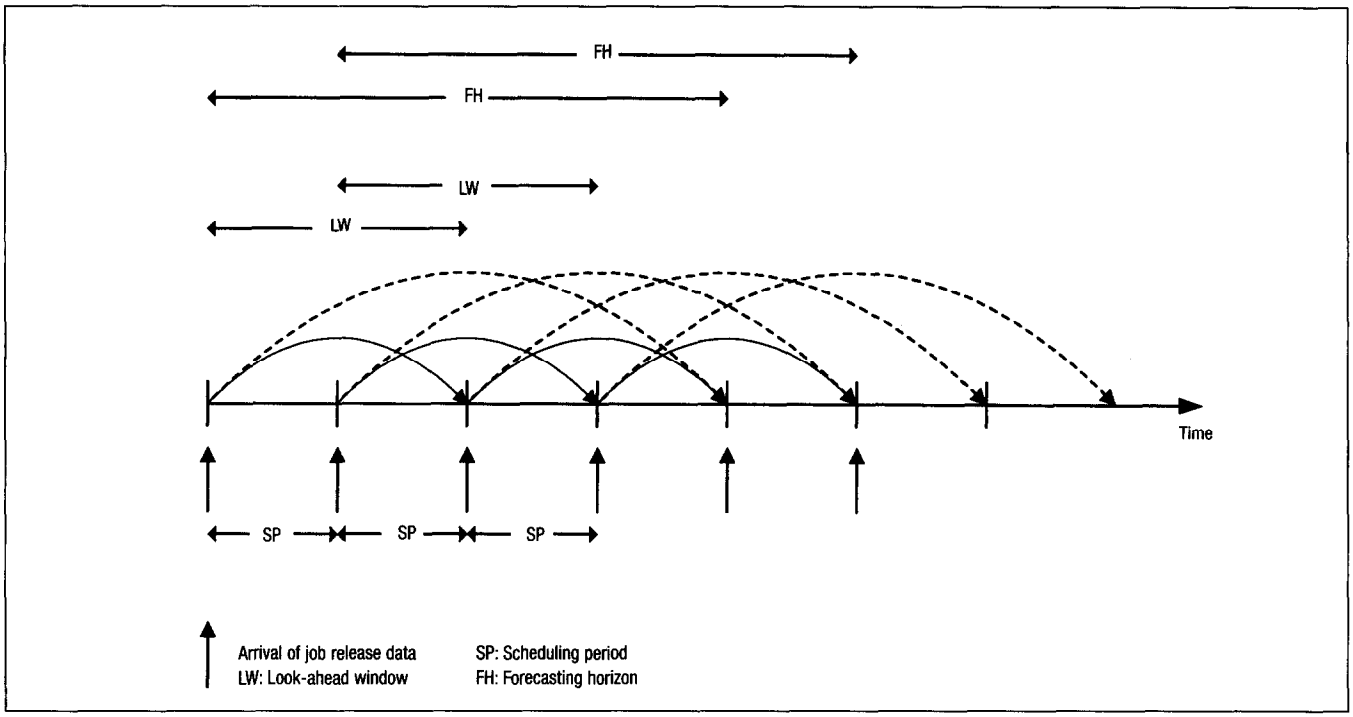

Figure 1
Schematic View of Relations Between Forecasting Horizon, Scheduling Period, and Look-ahead Window

rather than making detailed decisions such as sequencing all jobs and determining start times for operations.

A typical environment is explained where simulation-based schemes can be employed to address the above questions. In this environment, the scheduling and control activity is viewed as an intermediate component of a more global planning system in which decisions regarding production planning and master scheduling are made at a higher level. The scheduling and control level deals with the lower level, short-term decisions using the data provided by the higher levels. Specifically, it is assumed that a *planning module* provides a master schedule of upcoming jobs (called *job release data*). The time span of the job release data is called the *forecast window*. The scheduling module uses simulation runs to make (higher-level) scheduling decisions such as dynamic rule selection and parameter tuning. The length of the simulation runs is usually called the *simulation window* or *look-ahead window*, which may or may not equal the forecast window. The time interval between two successive points in time when the scheduling decisions are made is commonly called the *scheduling period*, which in turn determines the frequency of simulation activation. Simulation-based schemes are usually used in a rolling-horizon basis; that is, if new four-week job release data are available every week, for instance, and decisions are made for 2 weeks using simulation, then only the first-week decisions are implemented, and at the beginning of the next week new decisions are made for the following four weeks using the fresh job release data (see *Figure 1*).

The details of an (iterative) simulation-based mechanism that has been used in past studies are outlined, and the implementation of two specific algorithms for this mechanism is explained. The alternative to simulation-based scheduling will be a well-studied and widely used approach: priority dispatching. Only top-performing priority rules are considered for the problem, job shop scheduling with weighted tardiness objective, which is a surrogate measure of customer service. These priority rules are dynamic and state-dependent and have inherent flexibility to utilize up-to-date information and accommodate changes. We expect to find somewhat different results from previous findings in which fine-tuned static schedules obtained through simulation were compared against these myopic rules under static and deterministic conditions. The contention is that these rules will work well under highly dynamic and stochastic conditions as compared to simulation-based schemes; however, deteri-

oration of simulation-based schemes may not be so severe because they decide higher-level policies and leave the remaining decisions for later while the dynamics of the system take place.

Literature Review

First, the conceptual studies of simulation-based scheduling are summarized. An early study in this area is by Davis and Jones (1988) who decomposed scheduling problems into a hierarchical decision structure (planning, scheduling, and then control). They proposed a mechanism based on the simulation of each scheduling alternative (priority dispatching rules, routing alternatives, etc). Harmonosky (1990) discussed the implementation issues such as modeling, interface to the physical system, saving the system status for evaluating alternatives, and recovery at decision points. Harmonosky and Robohn (1991) addressed issues such as the frequency of invoking the simulation mechanism, dealing with the simulation output, data acquisition, and interface problems. In her later work, Harmonosky (1993) analyzed two key issues: (1) the simulation run length and (2) the type of simulation (deterministic versus stochastic due to machine breakdowns). In another study, Harmonosky and Robohn (1995) investigated the effects of different manufacturing systems on the computational time of simulation runs. The purpose in this paper, though, is not to address these issues; it is assumed that simulation is a viable approach from these perspectives.

Because two decision types (rule selection and parameter tuning) are the focus, the review mainly concentrates on studies that investigated these areas. Most simulation-based methods use a simulation run for each priority rule in a set of candidate rules, and then select the best among them. An early study in this area is by Wu and Wysk (1988, 1989), whose experimental results showed that when simulation run length is accurately determined according to environmental conditions and objectives, then the approach is very effective. After observing the drawbacks of this study where the simulation window coincides with the constant scheduling period, Ishii and Talavage (1991) proposed variable-length, state-dependent scheduling intervals and simulation windows. The simulation experiments showed that using a scheduling interval defined based on the system transient state improves the performance over using a constant scheduling interval, which has a very unstable performance as compared to single-pass rules. The results also showed that the performance becomes poorer than for the single-pass algorithms if the scheduling intervals are not accurately determined. After observing the advantages of switching rules in each time period, Ishii and Talavage (1994) tested the idea of using a different rule on each machine in each period. Cho and Wysk (1993) developed a neural network model to generate alternative dispatching rules based on the current system status, which are then evaluated by multi-pass simulation. Aytug, Koehler, and Snowdon (1994) also considered a rule-selection algorithm strengthened with a genetic algorithm-based machine learning scheme in a parallel machine setting with flow-time objective. Aytug et al. (1994) reviewed the relevant machine learning literature. It should be noted that the current study fills some research needs (testing in *dynamic and stochastic* job shops) identified by Aytug, Koehler, and Snowdon and Aytug et al.

The second area where iterative simulation runs are used is what is called *parameter tuning*, where multiple simulation runs are conducted to tune parameter(s) of an algorithm. An early study by Vepsalainen and Morton (1988) proposed an iterative approach called lead time iteration (LTI) that makes repeated simulation runs to find more consistent queue (waiting) time estimates that are used in priority index calculation. Kiran, Alptekin, and Kaplan (1991) proposed a *feedback heuristic*, where a series of schedules is generated using job-based priorities that are smoothed at every iteration using the respective job's contribution to the performance measure and the job's priority in the previous iteration. Experiments conducted in static and dynamic flexible manufacturing environments showed that the iterative simulation mechanism can be very useful as compared with single-pass rules.

Ovacik and Uzsoy (1994) presented several rolling horizon procedures to minimize maximum lateness on a single machine in the presence of sequence-dependent setup times. Although not simulation-based, the study is reviewed here because it addresses the issues of interactions between *planning horizon* and *forecast window*. The study showed that when forecast window and planning horizon parameters are appropriately selected, the proposed rolling horizon procedures outperform the

earliest due date rule. In a related study, Church and Uzsoy (1992) analyzed several rescheduling policies for dynamic single-machine and parallel-machine problems with the maximum lateness objective. Experimental results showed that periodic scheduling (that is, revising decisions every scheduling period) is very useful when the jobs arrive in batches periodically to the system and the scheduling period coincides with the batch interarrival time. In the case of dynamic and continuous arrivals, the benefit of extra scheduling diminishes rapidly.

All studies reviewed so far assume a deterministic (no stochastic events other than dynamic job arrivals) shop environment. Although there are numerous studies that investigate reactive scheduling, rescheduling, and robustness under uncertainty, there are only a few simulation-based studies that consider stochastic shop conditions. Kim and Kim (1994) proposed a simulation-based, real-time scheduling mechanism that evaluates various dispatching rules for a given job set and selects the best one for a given criterion. The best dispatching rule is used until the difference between the actual performance (under urgent job arrivals and machine breakdowns) and the estimated performance exceeds a given limit (called the *performance limit*); then, a new simulation is performed with the remaining operations, and a new rule is selected. Tayanithi, Manivannan, and Banks (1993a, 1993b) and Manivannan and Banks (1992) proposed an integrated scheduling and control system that combines simulation and knowledge-based concepts to perform an analysis of interruptions in the form of machine breakdowns and rush orders in a flexible manufacturing system. When a control decision cannot be obtained readily from the knowledge base, the alternative actions are evaluated using the simulation mechanism.

From the reviewed literature, it is known that static complete schedules (simulation-based or optimization-based) that do not leave any room for dynamic adaptability perform poorly in the face of disruptions. To the best of our knowledge, there is no single study that (1) uses simulation-based approach to decide high-level policies in a dynamic and stochastic job shop environment, and (2) analyzes them using an extensive experimental study under common conditions that test not only levels of stochastic disruptions (machine availability and processing time variation) but also investigate their interactions with method-specific parameters such as look-ahead and scheduling windows. This study fills this void and even addresses several issues that were raised in previous studies as future research directions: job shops instead of single machines or flow shops, processing time variation, and machine availability (together), and so on. Finally, the study shows whether the believed flexibility of simulation-based systems would hold under dynamic and stochastic conditions or not.

Iterative Simulation-Based Scheduling

A dynamic job shop scheduling problem is considered in which some information about future jobs is available for a certain period of time. The information on these soon-to-arrive jobs becomes available periodically; that is, certain characteristics of jobs that will be released to the shop floor are known, say, every week. The planning module, shown by the "*plant controller*" in *Figure 2* creates the job release data (or master schedule). The job release data consists of

- job arrival times that will occur during the next forecast horizon;
- job characteristics such as due dates, job weights, routing information (that is, number of operations, best estimates of processing times, machines that each job visits).

A typical simulation-based decision support to handle this type of scheduling problem is shown in *Figure 2*. The plant controller lays out the job release data and the managerial objective(s) to the *Parameter Selector (PS)* and the *Iterative Simulation Mechanism (ISM)*. The current shop status is fed back to the ISM and Parameter Selector by the execution and control module, called the "*shop controller.*" The parameter selector determines the look-ahead (or simulation) window, the scheduling period (SP) or planning horizon, and other parameter values that are used in the ISM algorithm. This is done by examining the current shop status, the job release data, and the objectives (the study in a way provides additional insights for the PS to determine what might work in different conditions). The ISM activates the iterative scheduling algorithm using the provided information. The ISM initializes each iteration (or simulation run)

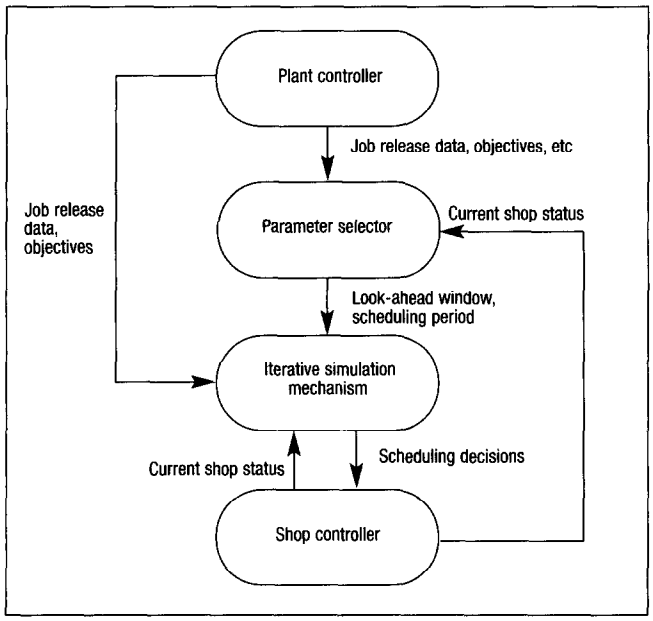

Figure 2
Schematic View of Simulation-based Iterative Scheduling System

with the current system status, and same dynamic events are generated by using the job release data in each iteration. The objective(s) set by management serve as performance measure(s) in the simulation runs. The simulation run length (called the "simulation window") is determined by the look-ahead window parameter provided by the PS. The scheduling period defines the frequency of normal ISM invokes (regular scheduling period). Although the ISM can also be activated in case of unexpected events such as machine breakdowns, this option is left for a future study.

It is implicitly assumed that the ISM consists of a somewhat detailed simulation model of the real system. The ISM uses this model to mimic the real-life system and evaluate the alternative scheduling policies by running the model. From this perspective, the ISM is flexible in terms of the variety of policies that it can test. More specifically, the ISM can be employed for many decision types such as:

- Best-rule selection
- Priority update (or parameter tuning)
- Constructing a complete schedule
- Reactive scheduling and control policy determination

In the current ISM implementation, the first two of these decisions will be considered. A specific algorithm is implemented for each decision:

- Multi-pass Rule Selection Algorithm
- Lead Time Iteration Algorithm

These algorithms are described in detail in the following subsections.

Multi-pass Rule Selection Algorithm

Many researchers have studied priority dispatching rules for more than three decades. The major conclusion that can be drawn from these studies is that there is no single rule that yields the best performance in all conditions and that the performance of a rule is highly affected by the operating conditions and the managerial objective(s). Therefore, changing a dispatching rule over successive time periods based on the current system state, the performance measure, and the current information about the future events can improve overall performance over using a single rule for the entire planning horizon. One issue addressed in this study is to test the effectiveness of this *rule switching approach* as a high-level decision not only in dynamic but also in stochastic environments. Some existing studies that consider similar approaches are reviewed in the previous section (Wu and Wysk 1988, 1989; Ishii and Talavage 1991; Cho and Wysk 1993).

In this specific implementation of this approach, there is a set of candidate rules, each of which can be applied in the shop for the given performance measure (weighted tardiness in this case). The candidate rules can be determined by analyzing their past performances. At each decision point, a new series of simulation runs is performed using each of the candidate priority rules in each run. At the end of each iteration, the performance measure is recorded for that rule, and the rule is selected that produces the best performance for the current look-ahead window. This rule is applied in the system until the next decision point, which is defined by the new information arrival (job release data, see *Figure 3*).

Because the weighted tardiness job shop problem is examined, the following specialized rules in the candidate set are considered: Apparent Tardiness Cost (ATC), Bottleneck Dynamics (BD), Cost OVER Time (COVERT), Modified Operation Due Date (MOD), and Weighted Shortest Processing Time (WSPT). Notation and definitions of the rules are given in *Table 1*. WSPT is selected because it yields good performance for tardiness measures in highly

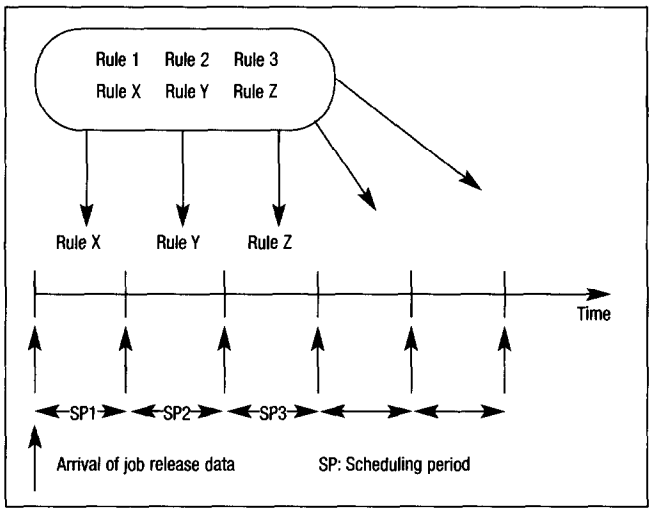

Figure 3
Schematic View of Candidate Rule Set and Implementation of Rule Selection Algorithm

loaded shop conditions. COVERT (Carrol 1965) and MOD (Baker and Kanet 1983) are two rules for minimizing the unweighted tardiness. ATC is similar to COVERT, but it utilizes exponential cost function (urgency factor) instead of linear (Vepsalainen and Morton 1987). BD has been developed as an extension of ATC, with the primary difference in the resource usage computation (Morton and Pentico 1993). The resource usage in the denominator of the formula is calculated by summing the terms (resource price times operation processing times) over the remaining operations for the job under consideration. The resource price of a machine k at time t ($R_k(t)$) is based on both the current jobs in the queue and the overall utilization, as shown below:

$$R_k(t) = \sum_{i=1}^{L_k(t)} w_i U_{ij}(t) + (wU)_{avg} L_k(t) \frac{\rho_k}{1.0 - \rho_k}$$

where $L_k(t)$ is the current queue length, $U_{ij}(t)$ is the urgency factor, $(wU)_{avg}$ is the average delay cost, and ρ_k is the average utilization of the machine. (For a broad discussion of the rules, including BD, refer to Kutanoglu and Sabuncuoglu 1999.)

Lead Time Iteration

Some priority rules such as ATC and BD involve parameters that need to be estimated or tuned. It is well known that the performance of these scheduling rules depends on the right setting of these parameters. Specifically, ATC and BD indices use waiting

Table 1
Priority Dispatching Rules in Candidate Rule Set for Multi-pass Rule Selection Algorithm (priorities are calculated for job *i* waiting for operation *j* at machine *k* at time *t*)

Priority Rule	Description
WSPT (Weighted Shortest Processing Time)	$WSPT_{ij} = \frac{w_i}{p_{ij}}$
MOD (Modified Operation Due Date)	$MOD_{ij}(t) = \max\left\{r_i + \frac{d_i - r_i}{\sum_{q=1}^{m_i} p_{iq}} \times \sum_{q=1}^{j} p_{iq}, t + p_{ij}\right\}$
COVERT (Cost OVER Time)	$COVERT_{ij}(t) = \frac{w_i}{p_{ij}} \times \left(\frac{h\sum_{q=j}^{m_i} W_{iq} - \left(d_i - t - \sum_{q=j}^{m_i} p_{iq}\right)^+}{h\sum_{q=j}^{m_i} W_{iq}}\right)^+$
ATC (Apparent Tardiness Cost)	$ATC_{ij}(t) = \frac{w_i}{p_{ij}} \times \exp\left(-\frac{\left(d_i - \sum_{q=j+1}^{m_i}(W_{iq} + p_{iq}) - p_{ij} - t\right)^+}{Kp_{avg}}\right)$
BD (Bottleneck Dynamics)	$BD_{ij}(t) = \frac{w_i \times \exp\left(-\frac{\left(d_i - \sum_{q=j+1}^{m_i}(W_{iq} + p_{iq}) - p_{ij} - t\right)^+}{Kp_{avg}}\right)}{\sum_{q=j}^{m_i} R_{k(q)}(t) p_{iq}}$

Nomenclature:

w_i	Weight of job i
p_{ij}	Processing time of operation j of job i
r_i	Arrival time of job i to the shop
d_i	Due date of job i
m_i	Number of operations of job i
W_{iq}	Estimated waiting time of job i for operation q
p_{avg}	Average operation processing time
$R_{k(q)}(t)$	Resource price of machine k for operation q
h, K	Constant coefficients

time estimates. One traditional approach is to use a constant multiplier (*lead time constant*) times processing time as a waiting time estimate (Vepsalainen and Morton 1987). In general, single-pass versions of ATC and BD employ this approach. To further improve the performance, there are two alternatives:

- Several alternative values for the lead time constant are evaluated, and the one which produces the best performance measure is selected.
- Lead time iteration (LTI) estimates the waiting time of each individual operation iteratively using realized waiting times in simulation runs. Waiting time estimates that produce the best estimated performance are used in the implementation of the rule.

In this study, the second approach is employed because it revises the individual waiting time estimates independently by using realized waiting times in previous iterations. Therefore, it is expected that it should perform at least as well as the first alternative for the same number of simulation iterations. LTI starts with initial waiting time estimates set as in traditional estimation method, and smoothes the estimated and the actual waiting time estimations at the end of each iteration for the next iteration. The steps of the algorithm are as follows (for the original version of LTI, see Morton and Pentico 1993):

- *Step 1.* Set iteration number n at 1. Initialize estimates (used was three times the processing time, $W_{ij}(1) = 3 \times p_{ij}$) for each job on hand and jobs that will arrive during the look-ahead window.
- *Step 2.* Perform a simulation using ATC (or BD).
- *Step 3.* Record the performance measure obtained and the actual waiting times ($Q_{ij}(n)$) for iteration n.
- *Step 4.* If a termination condition is satisfied, go to Step 7.
- *Step 5.* Compute the new estimates:

$$W_{ij}(n+1) = \alpha W_{ij}(n) + (1-\alpha) Q_{ij}(n)$$

where α is a *smoothing parameter* between 0 and 1 (0.5 is used in the experiments).

- *Step 6.* Increase the iteration number n by 1, and go to Step 2.
- *Step 7.* Report the best performance measure and corresponding waiting time estimates. Use the waiting time estimates in the actual implementation of the rule.

In the implementation of the algorithm, ATC is used because it requires less computation for priority calculation than BD. The smoothing process is used to prevent the estimates from changing drastically and to prevent possible oscillation. A typical termination rule is to set a limit on the number of iterations. An alternative is to terminate when there is no improvement in the performance for the last certain number of iterations. The procedure was set to stop either when the iteration number reaches 30 or when there is no improvement for 10 consecutive iterations.

In the experimental study, when scheduling decisions are made at these decision points either by the rule selection algorithm or LTI, these decisions are implemented in another simulation run representing the actual system operation. If a certain rule is selected during the iterative simulation, then this rule is applied as a priority dispatching rule to select the job next to be processed on an available machine. When the decisions are made by LTI, then ATC is used with the best waiting time estimates. During the iterative simulation process, the mechanism uses the best available information. But during the implementation of the decisions, the actual progress of the operations on the shop floor might be quite different. Two types of events are considered that will affect the actual progress: (1) machine failures and (2) processing time variations. Note that both events are tested simultaneously; that is, machines may unexpectedly fail and processing times may vary in the same experiment (except for the no-breakdown and no-variation cases). Hence in the study, also analyzed are the interactions between the parameters of the scheduling mechanism (that is, the scheduling period and look-ahead window) and these unexpected events that frequently occur in practice.

The extensive experimental study is conducted to test the following conjectures:

1. There is a strong interaction between method-specific factors and system conditions such as utilization and stochastic events.
2. When a simulation-based mechanism is used to determine higher-level policies and these are implemented dynamically over time, there is not a great need to frequently update the policies; that is, short scheduling intervals may not pay off.
3. Single-pass dispatching rules that are inherently dynamic and state-dependent will yield performances comparable (or superior) to simulation-based schemes as more uncertainties are present in the system.

Computational Study

The experiments consider a hypothetical reentrant job shop environment with the following characteristics: Jobs arrive continuously according to a Poisson process. The average utilization of the shop is determined by calibrating the arrival rate of the jobs. The

arrival rate is adjusted to achieve approximately 70% utilization on the average at the low level, and 90% utilization at the high level. The jobs have a fixed number of operations selected from a discrete uniform distribution from 1 to 10. The operations are randomly processed through 10 machines available in the shop. Job weights are drawn from Uniform[1,30]. Due dates are assigned randomly over a full range of flow allowances, with an average of six times the mean job processing time.

Two types of uncertainty are considered in the study: (1) processing time variation and (2) random machine breakdowns. Best estimates of processing times (that are assumed to be released by the plant controller, and that are used in simulation runs) are drawn from the uniform distribution between 1 and 30, U[1,30]. Actual processing times are determined using the best estimates as follows:

$$p'_{ij} = \left(1 + V \times U[-1.0, +1.0]\right) \times p_{ij}$$

where V, as an experimental factor, defines the level of processing time variation and p_{ij} and p_{ij}' are the best estimate and actual processing time for operation j of job i, respectively. In the current experimental study, V is set either at 0.0 (that is, deterministic case, no variation) or at 0.60.

Machine breakdowns are modeled by using the busy time approach proposed in Law and Kelton (1991). With this approach, a random uptime is generated for each machine from a "busy time distribution." The machine is considered "up" until its total accumulated busy time reaches the end of the generated uptime. Then it fails for a random time drawn from a downtime distribution, after which an uptime is generated. Law and Kelton recommended that, in the absence of real data, *busy time distribution* is most likely to be a Gamma distribution with shape parameter (α_b = 0.7) and scale parameter β_b to be specified according to the experimental conditions. They also stated that Gamma distribution with shape parameter (α_d = 1.4) is appropriate for the *distribution of downtimes*. In this framework, the level of machine failure is measured by *efficiency* (or *availability*) level, which gives the long-run ratio of the machine busy time to the total busy and downtime. In fact, this ratio is modified to generate desired levels of the machine failure factor. In this way, the duration of each breakdown comes from

$$Gamma\left(\alpha_d = 1.4, \beta_d = d_{avg} / 1.4\right)$$

and busy time between two successive breaks is drawn from

$$Gamma\left(\alpha_b = 0.7, \beta_b = d_{avg} \times \frac{\varepsilon}{0.7(1-\varepsilon)}\right)$$

Here, d_{avg} represents mean downtime and ε represents the efficiency level. (The former is denoted with D and the latter with E in the statistical analysis, see *Table 2*.) Three levels of efficiency are considered. The first level corresponds to no failure case, in which the efficiency is 100%. In the other two levels, the machines are all fallible with average efficiency of 90% and 80%. Four levels of mean downtimes are defined. The first level represents the no-failure case with 0 mean downtime, which is only possible in the efficiency level of 100%. The other three levels of mean downtime are p_{avg}, $5p_{avg}$, and $10p_{avg}$, where p_{avg} is the overall average processing time (these three cases are only possible in efficiency levels of 80% or 90%).

The system is simulated by using SIMAN simulation language with some additional C subroutines linked in a UNIX environment. Ten independent replications are used for output analysis. Each replication is 1500 job-completions long. The system is initialized with 20 jobs to reach the desired system state faster. Common random numbers (CRN) are used as a variance reduction technique.

By setting the simulation run length, the entire manufacturing horizon is also determined as 1500 job completions. Two different forecast horizons are determined: 250 job arrivals and 100 job arrivals. In this case, the look-ahead window is set equal to the forecast horizon. The scheduling period that defines the decision points has three different levels with respect to the look-ahead window. At the first level, the scheduling period is equal to the forecasting horizon. In the other two cases, the scheduling period is a portion of the forecasting horizon: about a quarter of the forecasting horizon and approximately half of the forecasting horizon. Hence, the scheduling period is equal to 65, 125, or 250 jobs for the case when the forecast horizon is 250, and it is equal

Table 2
Experimental Factors and Their Levels

Factor	Number of Levels	Levels
Rule (R)	5	ATC, BD, COVERT, MOD, WSPT
Utilization (U)	2	70%, 90%
Processing Time Variation (V)	2	0 (low), 0.6 (high)
Efficiency (E)	3	80%, 90%, 100%
Mean Downtime (D)	4	0 (no down), p_{avg}, $5p_{avg}$, $10p_{avg}$
Look-ahead Window (F)	3	100, 250, 1500
Scheduling Period (P)	7	25, 50, 65, 100, 125, 250, 1500

to 25, 50, or 100 jobs for the forecast horizon of 100. The case in which the forecast horizon is equal to the entire manufacturing horizon (1500 jobs) is used as a benchmark. In this case, it is assumed that all the information for a very long time period is available for ISM. (The experimental factors and their levels are summarized in *Table 2*.)

Experimental Results

Results are analyzed in three subsections. First are briefly presented the results of single-pass versions of the rules in the candidate rule set. Then, the results are discussed of the iterative simulation-based scheduling system with the rule-selection algorithm, and with the lead time iteration algorithm.

Before presenting the results, it should be noted that detecting the effects of the downtimes and the length of the scheduling periods requires particular attention. For 100% efficiency level, there is no downtime (it is shown as 0), and for the other levels there are three values for the mean downtime. In addition, the scheduling periods with 25, 50, and 100 jobs are defined according to the 100-job look-ahead window, while 65, 125, and 250-job scheduling periods are defined with respect to the 250-job window. Hence, the scheduling period factor is *nested* in the look-ahead window (shown as P(F)), and the mean downtime factor is nested in efficiency (D(E)). The analysis of the results is performed by using SAS statistics package (SAS 1994).

Single-pass Experiments

Statistical tests (the analysis of variance, ANOVA, and Duncan tests) have been used to identify areas that need further investigation. The reader is cautioned that the ANOVA results are given for expository purposes due to CRN and the resulting lack of sampling independence. The ANOVA test for single-pass experiments is presented in Appendix A. The column "*Pr gt F*" shows the significance level of the source. If the significance level of the analysis is taken as 0.05, the effects of the sources that yield probability smaller than 0.05 are statistically significant. In this table, *B* represents the effect of blocking due to the CRN implementation in simulation replications. According to the *F*-test, the main effects of all the factors are found to be significant. Also, all two-way interactions of rules, utilization, processing time variation, efficiency, and mean downtime are statistically significant. The three-way interaction of rules, utilization, and efficiency is effective on the weighted tardiness criterion.

Duncan's multiple range test is also applied for the main effects of the factors. Appendix B summarizes the Duncan's test results. (In the table, the significance level is 0.05, the levels with the same letter are not statistically different, and *N* is the number of observations for the corresponding level.) Although the rules significantly affect the overall performance, the performances of ATC, BD, and COVERT are not statistically distinguishable. MOD is the worst, producing on average 47% higher weighted tardiness than those of the best rules. An increase in utilization adversely affects the system performance as in the case of processing time variation. Lower efficiency levels significantly increase the weighted tardiness.

Because the two-way interactions of the rules and other factors are significant, these interactions are further depicted in a table and several figures. *Table 3* summarizes the performances of the rules with respect to utilization (U), efficiency (E), and processing time variation (V). Performance measures for different downtime levels are averaged leading to 30 observations ($N = 30$) in 80% and 90% efficiency levels. Note the high variability especially at the low efficiency levels. The performances of the first three rules are very close to each other's in almost every factor combination. One side point is that WSPT gets closer to the best rules with increasing utilization and/or decreasing efficiency. MOD deteriorates sharply as utilization increases as compared with the other rules. The effects of processing time variation are shown in *Figure 4*. In this case, the

Table 3
Summary of Results from Single-pass Experiment. Each pair of values under each rule, respectively, represents average total weighted tardiness and standard deviation using corresponding priority rule under the conditions in each row: U: utilization, E: efficiency, V: processing time variation, and N: number of experiments. When E is less than 100%, there are 30 observations due to three levels of mean downtime.

Rule			ATC	BD		COVERT		MOD		WSPT			
U	*E*	*V*	*N*	Avg	Dev	Avg	Dev	Avg	Dev	Avg	Dev	Avg	Dev
70	100	0	10	320	478	329	491	320	477	335	501	384	574
70	100	0.6	10	356	531	360	538	356	531	377	564	427	638
70	90	0	30	854	1276	859	1282	846	1256	1039	1559	951	1407
70	90	0.6	30	903	1344	919	1362	919	1365	1140	1697	1012	1493
70	80	0	30	1965	2983	1947	2939	1916	2888	2534	3838	2111	3173
70	80	0.6	30	2089	3155	2080	3119	2052	3076	2769	4159	2183	3264
90	100	0	10	470	702	484	723	473	708	605	910	612	915
90	100	0.6	10	538	805	551	825	544	815	706	1064	695	1040
90	90	0	30	1540	2292	1596	2387	1543	2308	2305	3436	1688	2495
90	90	0.6	30	1699	2520	1703	2519	1674	2478	2451	3616	1831	2697
90	80	0	30	4040	6054	3879	5763	3923	5878	6558	9765	4028	6009
90	80	0.6	30	4190	6237	4243	6285	4251	6345	6764	968	4155	6158

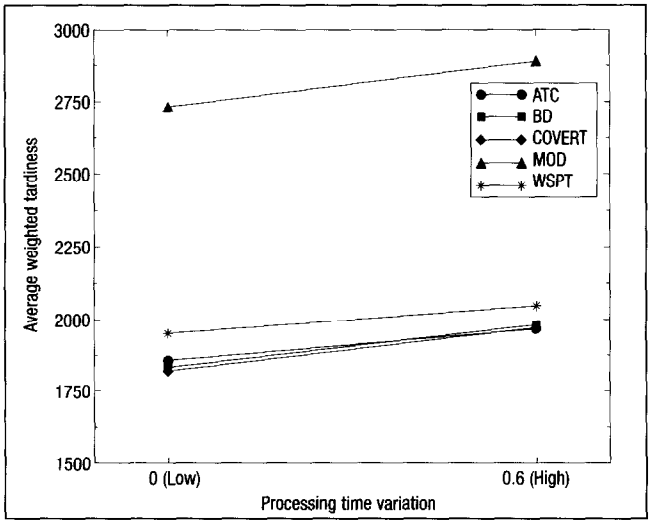

Figure 4
Average Weighted Tardiness vs. Processing Time Variation (Single-pass)

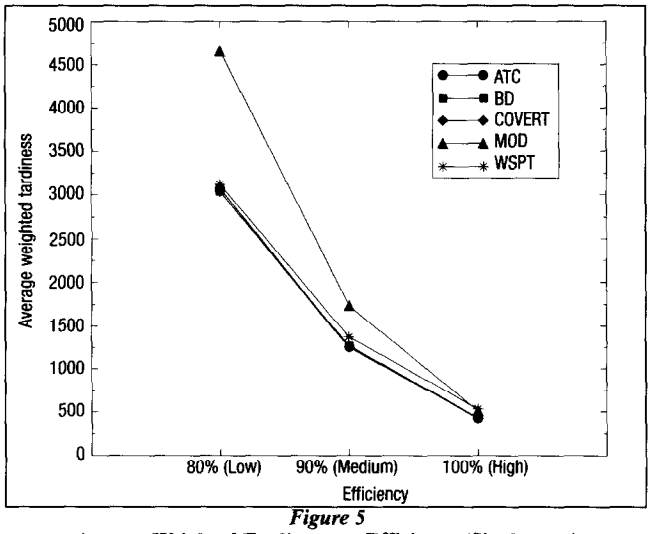

Figure 5
Average Weighted Tardiness vs. Efficiency (Single-pass)

deterioration in the performance of the rules is almost in the same rate as the processing time variation increases. The effects of efficiency and downtimes are depicted in *Figures 5* and *6*. In general, MOD is very sensitive to the efficiency and downtimes. It is also noted that the performance of WSPT is very close to the performances of ATC, BD, and COVERT at 80% efficiency, whereas it is worse than those rules at 90% efficiency.

Results of Multi-pass Rule Selection Algorithm

The ANOVA table for the multi-pass rule selection algorithm is presented in Appendix C. The main effects of all the factors except the scheduling period

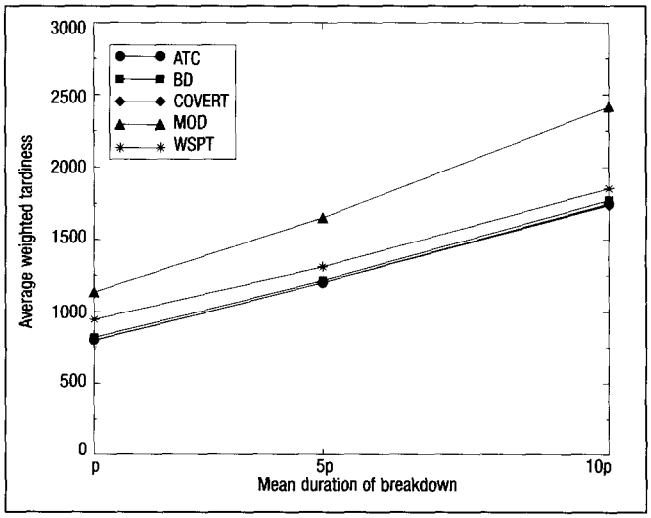

Figure 6
Average Weighted Tardiness vs. Mean Duration of Breakdown (Single-pass, Efficiency = 90%)

are statistically significant. All two-way interactions of utilization, efficiency, downtime nested in efficiency, and look-ahead window are effective on the performance. A significant three-way interaction of utilization, efficiency, and look-ahead window is observed, as well as a significant three-way interaction of utilization, downtime, and look-ahead window. The results also indicate that the scheduling period is effective only with downtime and efficiency.

Appendix D summarizes the Duncan's multiple range test for the multi-pass algorithm. The results support the findings of ANOVA for utilization, processing time variation, and efficiency. The Duncan grouping shows that the look-ahead of 100 jobs is significantly the worst among the tested look-ahead windows. The look-ahead windows with 250 and 1500 jobs produce the lowest measures without any significant statistical difference between them. The effects of the look-ahead window are listed in *Table 4* along with scheduling period levels. It is observed that a look-ahead window set at 250 jobs with scheduling periods of either 65 jobs or 250 jobs produces slightly better performance.

Table 5 shows the selection percentages of each rule along with the average CPU time for one full replication of simulation. The results show that the selection percentages are highly dependent on the lengths of the look-ahead window and scheduling period. Although, ATC and COVERT are more and more selected for longer look-ahead windows, the other rules are alternatively selected in the shorter look-ahead windows. This shows that several rules can be preferred in the short term, although their long-term performances are dominated by others. The table also shows the effects of the look-ahead window and scheduling period on the simulation time. Especially the scheduling period affects the CPU time because it directly determines the frequency of the multi-pass algorithm executions.

A summary of results is provided in *Table 6* for selected combinations of look-ahead window (F) and scheduling period (P). It is observed that F = 100 is inferior mainly due to its performance in low efficiency levels. The effect of utilization on the performance of the 100-job look-ahead window is depicted in *Figure 7*. This figure shows that a scheduling period of 25 jobs is mostly affected by the utilization. The effects of efficiency and mean downtime are displayed in *Figures 8* to *10*. In higher efficiency levels or in short downtimes, the different look-ahead windows seem to produce very similar performances. This effect becomes stronger when the efficiency is high; however, at low efficiency levels or with long downtimes, longer look-ahead windows yield better performances. In summary, the length of the look-ahead window is effective on the performance, but the scheduling period is not very significant. That is, it is important to select a proper time period for the look-ahead window; if an appropriate look-ahead window is selected, then the scheduling period is not so important.

Table 4
Effects of Look-ahead Window and Scheduling Period on Weighted Tardiness (Multi-pass Rule Selection Algorithm)

Scheduling Period	Look-ahead Window	Average
25	100	2012.26
50	100	2051.58
100	100	2028.73
65	250	1894.89
125	250	1922.09
250	250	1890.54
1500	1500	1912.96

Table 5
Selection Percentages of Rules at each Decision Point and the Average CPU Time per Replication

L-ahead Win.	Sched. Per.	ATC	BD	COVERT	MOD	WSPT	CPU
100	25	32.49	16.47	34.59	10.00	5.95	1520.08
100	50	33.73	15.68	34.65	10.51	5.43	798.11
100	100	32.25	16.50	35.86	8.59	6.79	444.32
250	65	37.65	12.76	43.31	5.82	0.45	1256.92
250	125	38.23	12.64	43.52	4.98	0.61	731.55
250	250	38.57	11.57	43.57	5.71	0.57	416.12
1500	1500	54.00	2.00	40.00	4.00	0.00	297.76

Table 6
Summary of Results for Multi-pass Rule Selection Algorithm

(F,P)				(100,100)		(250,250)		(1500,1500)	
U	E	V	N	Avg	Dev	Avg	Dev	Avg	Dev
70	100	0	10	309	461	312	465	317	474
70	100	0.6	10	360	537	356	531	356	532
70	90	0	30	844	1258	862	1282	850	1269
70	90	0.6	30	900	1337	919	1362	914	1356
70	80	0	30	1976	2986	1938	2918	1964	2967
70	80	0.6	30	2129	3207	2040	3067	2147	3238
90	100	0	10	461	690	460	688	468	699
90	100	0.6	10	550	823	544	814	535	800
90	90	0	30	1561	2339	1561	2317	1548	2301
90	90	0.6	30	1668	2482	1649	2439	1684	2501
90	80	0	30	4245	6473	3975	5918	3943	5896
90	80	0.6	30	5051	7606	4144	6156	4245	6333

Results of Lead Time Iteration Algorithm

The ANOVA test for the lead time iteration algorithm is presented in Appendix E. The F-test shows

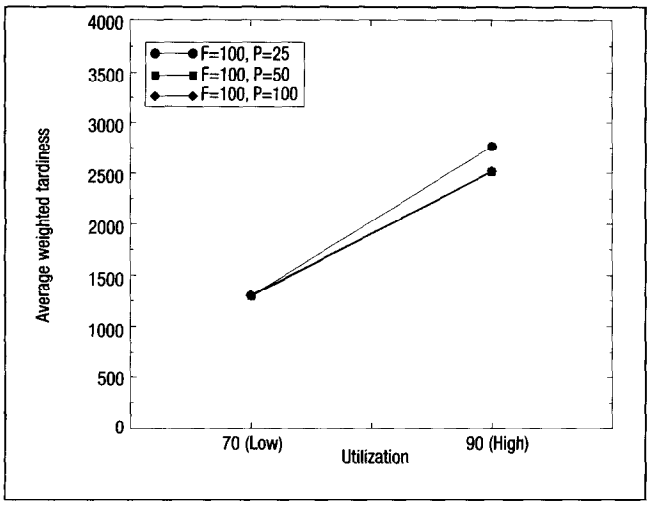

Figure 7
Average Weighted Tardiness vs. Utilization (Multi-pass Rule Selection Algorithm)

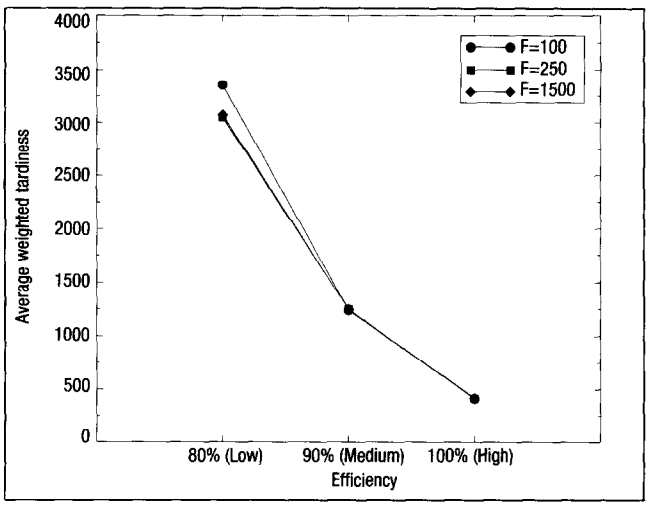

Figure 8
Average Weighted Tardiness vs. Efficiency (Multi-pass Rule Selection Algorithm)

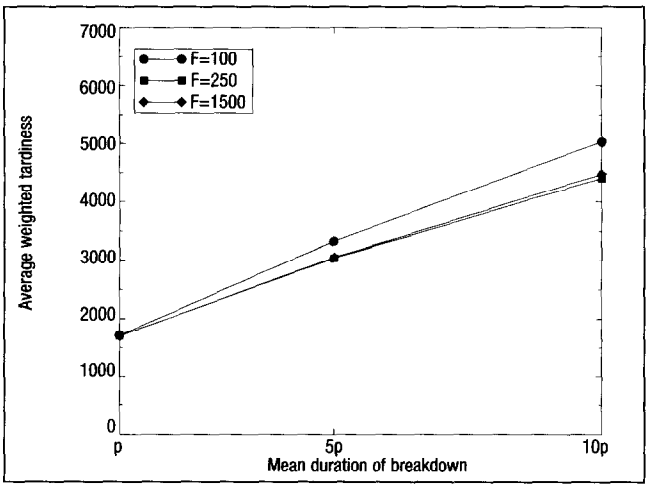

Figure 9
Average Weighted Tardiness vs. Mean Duration of Breakdown (Multi-pass Rule Selection Algorithm, Efficiency=80%)

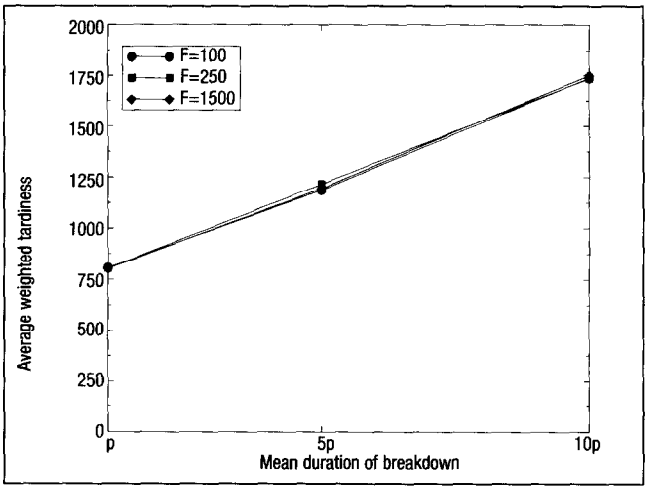

Figure 10
Average Weighted Tardiness vs. Mean Duration of Breakdown (Multi-pass Rule Selection Algorithm, Efficiency=90%)

that only the main effects of utilization, variation, efficiency, and downtime are significant. The lengths of the look-ahead window and scheduling period do not have significant impact on the performance of the lead time iteration algorithm. Among the two and higher-level interactions, the interaction between utilization and efficiency and the interaction between utilization and downtime are found significant. The Duncan's test supports these findings, as shown in Appendix F. For the sake of brevity, a summary table for LTI is provided (see *Table 7*). There is no significant difference among the performances of the selected combinations of F and P. One can choose shorter look-ahead windows with equal scheduling periods to save computational effort. From the same table, the effects of utilization, efficiency, and variation levels are observed.

When compared with the single-pass algorithms, the multi-pass rule selection algorithm provides 7.5% improvement on average over the single-pass rules. When the utilization is high or the processing time variation is low, multi-pass rule selection algorithm is even more advantageous. The lead time iteration algorithm produces lower weighted tardiness with an average improvement of 9.6% over the single-pass rules. When the LTI algorithm is compared with the multi-pass rule selection algorithm, it is seen that, on average, LTI dominates the latter; however, the improvement rate over the rule selection algorithm changes with respect to the efficiency and processing time variation. Even in these cases, however, the improvement does not seem to be signifi-

Table 7
Summary of Results for Lead Time Iteration Algorithm

(F,P)				(100,100)		(250,250)		(1500,1500)	
U	*E*	*V*	*N*	Avg	Dev	Avg	Dev	Avg	Dev
70	100	0	10	300	448	299	447	307	459
70	100	0.6	10	348	520	344	514	345	516
70	9	0	30	839	1250	838	1251	855	1278
70	90	0.6	30	941	1409	910	1351	911	1354
70	80	0	30	1956	2952	2005	3032	2001	3014
70	80	0.6	30	2073	3121	2114	3180	2119	3182
90	100	0	10	443	662	442	662	452	676
90	100	0.6	10	532	797	530	792	537	803
90	90	0	30	1506	2241	1548	2323	1576	2358
90	90	0.6	30	1682	2489	1738	2596	1731	2575
90	80	0	30	4096	6184	4085	6161	4050	6052
90	80	0.6	30	4293	6412	4237	6317	4356	6498

cant especially if the high variability due to processing time variations and machine failures is considered. Hence, either can be implemented in the outlined iterative simulation-based mechanism.

The results of summary tables for alternative schemes are simultaneously analyzed (single-pass, multi-pass rule selection, lead time iteration, see *Tables 3*, *6*, and *7*). The differences observable over all observations may not be consistent with the more direct comparison of the best single-pass rule (say, ATC) and the best setting for multi-pass algorithms ($F = 250$, $P = 250$). Although both multi-pass algorithms improve performance over single-pass rules on average, this is questionable if it is compared with the best among the single-pass rules, especially in low efficiency levels and high processing time variation levels. This implies that multi-pass algorithms may not pay off the computational effort if the best rule to implement in the system is known. These results also show that a simulation-based scheduling mechanism is effective in more deterministic situations ($E = 100$ and $V = 0$). This can be attributed to the nature of the iterative algorithms because they search through the promising improvements over the existing alternatives; however, this positive effect of the search ability reduces in the dynamic and stochastic environment (that is, processing time variations and machine breakdowns seriously diminish the potential improvement expected from the iterative algorithms).

Discussion and Conclusions

This study has defined a scheduling environment where some information about future job arrivals (job release data) is available. A simulation-based scheduling mechanism is outlined. Two types of scheduling decisions were tested for effectiveness: (1) best-rule selection and (2) parameter tuning. Using the multi-pass rule selection algorithm for the first type and the lead time iteration method for the second, an extensive experimental study was conducted by creating both deterministic and stochastic environments (dynamic, however, in all cases). The effects of unexpected events such as machine failures and processing time variation were investigated. The experimental results repeatedly show the significant interactions among the method-specific factors and the system conditions, which supports the first conjecture.

The priority rule is selected to implement during the next scheduling period in the rule selection algorithm, whereas improved waiting time estimates were sought in the lead time iteration method. Namely, higher-level policies are decided at the beginning of a scheduling period, and the selected policies are implemented dynamically over time. The remaining scheduling decisions such as determining the start times of operations (or which job to start next) are determined in a dynamic manner during the implementation of the selected policies (a static *a priori* schedule is not actually generated, as opposed to the traditional approaches). When unexpected events occur in the system, the dynamic nature of these algorithms results in decisions that are state-dependent as the system state changes. Hence, the algorithms that are dynamic and state-dependent in nature inherently develop their reactions to the events until the next decision point. In this sense, the results support our second conjecture that frequent updates of the higher-level policies (short scheduling periods) may not be necessary even in dynamic and stochastic conditions. This observation is especially true when the lead time iteration method is considered. If the algorithm implemented in ISM were not dynamic and robust, then the revision of the policies would be critical.

In this case, the information for the next forecast horizon is utilized explicitly during the decision making process. It is assumed that the arrival times and other characteristics of the jobs are known with certainty at the beginning of scheduling period. The only stochastic events are machine failures and processing time variations; however, the results of the experiments have indicated that these two events do not necessitate frequent revision of the higher-level

policies. It is expected that if there are order cancellations, rush job arrivals, or operational changes in jobs, more frequent scheduling revisions with shorter scheduling periods would potentially improve the performance. This is left as a future study.

The results also show that the multi-pass or iterative algorithm is better than single-pass algorithms (rules) on average, but not better than the best single-pass rule, especially in stochastic cases. This implies that one may just choose the overall best rule and not need to revise this decision for the entire horizon. This can be explained by the dynamic and state-dependent nature of the rules implemented in the study. Even if the rules are single-pass, they use the up-to-date information about shop status because they defer scheduling decisions until when they are needed. Hence, the rules have inherent adaptive/reactive elements such as changing the priorities of jobs, which make them rather robust for uncertain conditions. Additionally, the results show that fine tuning a parameter of a rule in a series of iterative simulations may not be a viable approach in a stochastic environment. The stochastic events may make the fine-tuned rule/parameter an inferior decision for the implementation; that is, the performance of the rule/parameter that performed best under deterministic conditions may turn out to be a poor selection under stochastic events. These observations provide evidence that support the third conjecture listed previously; however, a further investigation is needed for other types of decisions that would be made by means of iterative simulation and for other types of stochastic events.

References

Aytug, H.; Bhattacharyya, S.; Koehler, G.J.; and Snowdon, J.L. (1994). "A review of machine learning in scheduling." *IEEE Trans. on Engg. Mgmt.* (v41, n2), pp165-171.

Aytug, H.; Koehler, G.J.; and Snowdon, J.L. (1994). "Genetic learning of dynamic scheduling within a simulation environment." *Computers and Operations Research* (v21, n8), pp909-925.

Baker, K.R. and Kanet, J.J. (1983). "Job shop scheduling with modified due dates." *Journal of Operations Mgmt.* (v4, n1), pp11-22.

Carrol, D.C. (1965). "Heuristic sequencing of jobs with single and multiple components." Cambridge, MA: Sloan School of Mgmt., Massachusetts Institute of Technology.

Cho, H. and Wysk, R.A. (1993). "A robust adaptive scheduler for an intelligent workstation controller." *Int'l Journal of Production Research* (v31, n4), pp771-789.

Church, L.K. and Uzsoy, R. (1992). "Analysis of periodic and event-driven rescheduling policies in dynamic shops." *Int'l Journal of Computer Integrated Mfg.* (v5, n3), pp153-163.

Davis, W.J. and Jones, A.T. (1988). "A real-time production scheduler for a stochastic manufacturing environment." *Int'l Journal of Computer Integrated Mfg.* (v1, n2), pp101-112.

Harmonosky, C.M. (1990). "Implementation issues using simulation for real-time scheduling, control, and monitoring." *Proc.* of Winter Simulation Conf., 1990.

Harmonosky, C.M. (1993). "Analysis of two key issues for using simulation for real-time production control." *Proc.* of 2nd Industrial Engg. Research Conf., 1993.

Harmonosky, C.M. and Robohn, S.F. (1991). "Real-time scheduling in a computer integrated manufacturing: a review of recent research." *Int'l Journal of Computer Integrated Mfg.* (v4, n6), pp331-340.

Harmonosky, C.M. and Robohn, S.F. (1995). "Investigating the application potential of simulation to real-time control decisions." *Int'l Journal of Computer Integrated Mfg.* (v8, n2), pp126-132.

Ishii, N. and Talavage, J.J. (1991). "A transient-based real-time scheduling algorithm in FMS." *Int'l Journal of Production Research* (v29, n12), pp2501-2520,

Ishii, N. and Talavage, J.J. (1994). "A mixed dispatching rule approach in FMS scheduling." *Int'l Journal of Flexible Mfg. Systems* (v2, n6), pp69-87.

Kim, M.H. and Kim, Y. (1994). "Simulation-based real-time scheduling in a flexible manufacturing system." *Journal of Manufacturing Systems* (v13, n2), pp85-93.

Kiran, A.S.; Alptekin, S.; and Kaplan, A.C. (1991). "Tardiness heuristic for scheduling flexible manufacturing systems." *Production Planning and Control* (v2, n3), pp228-241.

Kutanoglu, E. and Sabuncuoglu, I. (1999). "An analysis of heuristics in a dynamic job shop with weighted tardiness objectives." *Int'l Journal of Production Research* (v37, n1), pp165-187.

Law, A.M. and Kelton, W.D. (1991). *Simulation Modeling and Analysis.* New York: McGraw Hill.

Lawrence, S.R. and Sewell, E.C. (1997). "Heuristic, optimal, static, and dynamic schedules when processing times are uncertain." *Journal of Operations Mgmt.* (v15), pp71-82.

Manivannan, S. and Banks, J. (1992). "Design of a knowledge-based on-line simulation system to control a manufacturing shop floor." *IIE Trans.* (v24, n3), pp72-80.

Morton, T.E. and Pentico, D. (1993). *Heuristic Scheduling Systems with Applications to Production and Project Management.* New York: John Wiley & Sons.

Ovacik, I.M. and Uzsoy, R. (1994). "Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times." *Int'l Journal of Production Research* (v32, n6), pp1243-1263.

SAS Institute. (1994). "SAS/STAT(R) User's Guide." North Carolina: SAS Institute.

Tayanithi, P.; Manivannan, S.; and Banks, J. (1993a). "A knowledge-based simulation architecture to analyze interruptions in a flexible manufacturing system." *Journal of Manufacturing Systems* (v11, n3), pp195-214.

Tayanithi, P.; Manivannan, S.; and Banks, J. (1993b). "Complexity reduction during interruption analysis in a flexible manufacturing system using a knowledge-based on-line simulation." *Journal of Manufacturing Systems* (v12, n2), pp153-169.

Vepsalainen, A.P.J. and Morton, T.E. (1987). "Priority rules for job shops with weighted tardiness costs." *Mgmt. Science* (v33, n8), pp1035-1047.

Vepsalainen, A.P.J. and Morton, T.E. (1988). "Improving local priority rules with global lead-time estimates: a simulation study." *Journal of Mfg. and Operations Mgmt.* (v1, n2), pp102-118.

Wu, S.D. and Wysk, R.A. (1988). "Multi-pass expert control system—a control/scheduling structure for flexible manufacturing cells." *Journal of Manufacturing Systems* (v7, n2), pp107-120.

Wu, S.D. and Wysk, R.A. (1989). "An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing." *Int'l Journal of Production Research* (v27, n9), pp1603-1623.

Authors' Biographies

Erhan Kutanoglu is an assistant professor in the Dept. of Industrial Engineering at the University of Arkansas. Before his current position, he has worked as an operations research analyst and development engineer at

IBM Global Services. He received his PhD in industrial engineering from Lehigh University in 1999 and his MS and BS degrees from Bilkent University, Ankara, Turkey. His current research interests include models and algorithms for distributed decision making, game theoretic approaches to combinatorial problems, and robustness under uncertainty with applications to scheduling, logistics, and supply chain management problems. He has published his work in journals such as *IIE Transactions* and the *International Journal of Production Research*. He is currently involved in funded projects on truck and driver scheduling problems, interactions of e-commerce transactions and transportation flows, and multi-echelon inventory stocking and logistics network design problems. He is a member of IIE and INFORMS.

Ihsan Sabuncuoglu is an associate professor of industrial engineering at Bilkent University, Ankara, Turkey. He received his BS and MS degrees in industrial engineering from Middle East Technical University and his PhD degree in industrial engineering from Wichita State University. Dr. Sabuncuoglu teaches and conducts research in the areas of scheduling, production management, simulation, and manufacturing systems. He has published papers in *Simulation*, *IIE Transactions*, *Decision Sciences*, *International Journal of Production Research*, *International Journal of Flexible Manufacturing Systems*, *International Journal of Computer Integrated Manufacturing*, *European Journal of Operations Research*, *Production Planning and Control*, *Journal of Operational Research Society*, *Computers and Operations Research*, *Computers and Industrial Engineering*, *OMEGA*, and *Journal of Intelligent Manufacturing*. He is on the editorial board of the *International Journal of Operations and Quantitative Management* and the *Journal of Operations Management*. He is an associate member of Institute of Industrial Engineers, Institute of Simulation, and Institute for Operations Research and Management Science.

Appendix A

Analysis of Variance for Weighted Tardiness (WT) (Single-pass Rules)

Source	DF	Sum of Squares	F Value	Pr gt F
Model	148	4305218750.23	141.46	0.0001
B	9	78459689.45	42.40	0.0001
Error	1251	257243013.65		
Source	DF	ANOVA SS	F Value	Pr gt F
R	4	176377062.06	214.44	0.0001
U	1	729224603.36	3546.30	0.0001
R*U	4	66292561.20	80.60	0.0001
V	1	6162597.31	29.97	0.0001
R*V	4	217221.97	0.26	0.9011
U*V	1	500191.15	2.43	0.1191
R*U*V	4	130987.23	0.16	0.9588
E	2	1841908686.72	4478.70	0.0001
R*E	8	88237797.50	53.64	0.0001
U*E	2	275704820.80	670.39	0.0001
R*U*E	8	33849156.53	20.58	0.0001
V*E	2	858029.21	2.09	0.1246
R*V*E	8	322360.01	0.20	0.9915
U*V*E	2	31222.69	0.08	0.9269
R*U*V*E	8	262709.14	0.16	0.9958
D(E)	4	916468601.08	1114.22	0.0001
R*D(E)	16	18004682.52	5.47	0.0001
U*D(E)	4	66818835.40	81.24	0.0001
R*U*D(E)	16	2886616.38	0.88	0.5959
V*D(E)	4	282567.87	0.34	0.8486
R*V*D(E)	16	924265.08	0.28	0.9977
U*V*D(E)	4	229676.86	0.28	0.8915
R*U*V*D(E)	16	1063808.71	0.32	0.9947

Appendix B

Duncan's Multiple Range Test for Weighted Tardiness (Single-pass Rules)

Factor: Rules (R)			
Duncan Grouping	Mean	N	R
A	2810.84	280	MOD
B	1999.84	2809	WSPT
C	1911.66	280	ATC
C	1907.01	280	BD
C	1895.07	280	COVERT
Factor: Utilization (U)			
Duncan Grouping	Mean	N	Utilization
A	2826.60	700	90
B	1383.16	700	70
Factor: Processing Time Variation (V)			
Duncan Grouping	Mean	N	Variation
A	2171.23	700	0.6
B	2038.53	700	0
Factor: Efficiency (E)			
Duncan Grouping	Mean	N	Efficiency
A	3383.75	600	80
B	1373.59	600	90
C	462.12	200	100

Appendix C

Analysis of Variance for Weighted Tardiness (WT) (Multi-pass Rule Selection Algorithm)

Source	DF	Sum of Squares	F Value	Pr gt F
Model	204	5336807534.54	93.04	0.0001
B	9	128727548.77	50.87	0.0001
Error	1755	493474510.01		
Source	DF	ANOVA SS	F Value	Pr gt F
U	1	857480788.75	3049.56	0.0001
V	1	9672458.36	34.40	0.0001
U*V	1	995615.25	3.54	0.0600
E	2	2353820883.82	4185.58	0.0001
U*E	2	370356964.26	658.57	0.0001
V*E	2	1462751.85	2.60	0.0745
U*V*E	2	178918.97	0.32	0.7275
D(E)	4	1362144691.42	1211.09	0.0001
U*D(E)	4	146419209.21	130.18	0.0001
V*D(E)	4	1236815.98	1.10	0.3551
U*V*D(E)	4	1863158.63	1.66	0.1575
F	2	7611557.59	13.53	0.0001
U*F	2	7214977.51	12.83	0.0001
V*F	2	98048.79	0.17	0.8400
U*V*F	2	16597.43	0.03	0.9709
E*F	4	11283954.31	10.03	0.0001
U*E*F	4	9492677.70	8.44	0.0001
V*E*F	4	153526.66	0.14	0.9688
U*V*E*F	4	42890.62	0.04	0.9972
D*F(E)	8	12603550.76	5.60	0.0001
U*D*F(E)	8	10231745.42	4.55	0.0001
V*D*F(E)	8	1458950.07	0.65	0.7371
U*V*D*F(E)	8	1014110.52	0.45	0.8906
P(F)	4	382086.21	0.34	0.8513
U*P(F)	4	363088.62	0.32	0.8628
V*P(F)	4	1105326.33	0.98	0.4157
U*V*P(F)	4	1151821.94	1.02	0.3934
E*P(F)	8	250406.36	0.11	0.9988
U*E*P(F)	8	529866.13	0.24	0.9843
V*E*P(F)	8	1950854.51	0.87	0.5435
U*V*E*P(F)	8	1587157.07	0.71	0.6869
D*P(E*F)	16	9353208.53	2.08	0.0072
U*D*P(E*F)	16	9186562.31	2.04	0.0086
V*D*P(E*F)	16	7835016.21	1.74	0.0338
U*V*D*P(F)	16	7529747.64	1.67	0.0452

Appendix D

Duncan's Multiple Range Test for Weighted Tardiness (Multi-pass Rule Selection Algorithm)

Factor: Utilization (U)			
Duncan Grouping	Mean	N	Utilization
A	2620.44	980	90
B	1297.58	980	70
Factor: Processing Time Variation (V)			
Duncan Grouping	Mean	N	Variation
A	2029.26	980	0.6
B	1888.76	980	0
Factor: Efficiency (E)			
Duncan Grouping	Mean	N	Efficiency
A	3184.70	840	80
B	1247.80	840	90
C	415.60	280	100
Factor: Look-ahead Window (F)			
Duncan Grouping	Mean	N	Look-ahead
A	2030.86	840	100
B	1912.96	280	1500
B	1902.51	840	250

Appendix E

Analysis of Variance for Weighted Tardiness (WT) (Lead Time Iteration Algorithm)

Source	DF	Sum of Squares	F Value	Pr gt F
Model	204	4628127753.47	126.82	0.0001
B	9	115808543.20	71.93	0.0001
Error	1755	313947140.33		
Source	DF	ANOVA SS	F Value	Pr gt F
U	1	750328450.97	4194.42	0.0001
V	1	8583146.32	47.98	0.0001
U*V	1	557577.82	3.12	0.0777
E	2	2168815600.28	6061.96	0.0001
U*E	2	284637329.57	795.58	0.0001
V*E	2	641829.88	1.79	0.1666
U*V*E	2	19358.52	0.05	0.9473
D(E)	4	1201889440.60	1679.67	0.0001
U*D(E)	4	91285489.58	127.57	0.0001
V*D(E)	4	143597.84	0.20	0.9380
U*V*D(E)	4	324093.93	0.45	0.7700
F	2	205876.14	0.58	0.5626
U*F	2	156292.16	0.44	0.6461
V*F	2	79285.56	0.22	0.8013
U*V*F	2	95432.61	0.27	0.7659
E*F	4	108445.48	0.15	0.9623
U*E*F	4	238551.48	0.33	0.8556
V*E*F	4	233195.59	0.33	0.8607
U*V*E*F	4	91062.26	0.13	0.9726
D*F(E)	8	297038.63	0.21	0.9897
U*D*F(E)	8	193969.17	0.14	0.9976
V*D*F(E)	8	359362.95	0.25	0.9807
U*V*D*F(E)	8	152200.06	0.11	0.9990
P(F)	4	152083.15	0.21	0.9316
U*P(F)	4	148489.04	0.21	0.9343
V*P(F)	4	7866.53	0.01	0.9998
U*V*P(F)	4	42043.34	0.06	0.9936
E*P(F)	8	177688.44	0.12	0.9983
U*E*P(F)	8	244903.55	0.17	0.9947
V*E*P(F)	8	159002.95	0.11	0.9989
U*V*E*P(F)	8	11471.68	0.01	1.000
D*P(E*F)	16	458033.14	0.16	0.9999
U*D*P(E*F)	16	830654.05	0.29	0.9972
V*D*P(E*F)	16	391543.55	0.14	1.000
U*V*D*P(F)	16	258803.44	0.09	1.000

Appendix F

Duncan's Multiple Range Test for Weighted Tardiness (Lead Time Iteration Algorithm)

Factor: Utilization (U)			
Duncan Grouping	Mean	N	Utilization
A	2538.99	980	90
B	1301.54	980	70
Factor: Processing Time Variation (V)			
Duncan Grouping	Mean	N	Variation
A	1986.44	980	0.6
B	1854.09	980	0
Factor: Efficiency (E)			
Duncan Grouping	Mean	N	Efficiency
A	3091.71	840	80
B	1254.09	840	90
C	404.50	280	100
Factor: Look-ahead Window (F)			
Duncan Grouping	Mean	N	Look-ahead
A	1944.24	280	1500
A	1919.55	840	100
A	1912.99	840	250