Expert Systems With Applications, Vol. 6, pp. 219–230, 1993
Printed in the USA.

0957-4174/93 $6.00 + .00

Contingencies for the Design of Scheduling Expert Systems

JAE K. LEE

Korea Advanced Institute of Science and Technology, Seoul, Korea

MIN SOO SUH

Research Institute of Industrial Science and Technology, Pohang 790-600, Korea

MARK S. FOX

University of Toronto, Toronto, Canada

Abstract—*To design a scheduling expert system under the dynamic environment, one needs to consider not only the performances of tardiness and flow time during the generative scheduling phase, but also the robustness and scheduling time requirements during the reactive control phase. To find the best contingent system's architecture (centralized or hierarchical) in combination with the scheduling strategies on activity ordering, allocation time slot selection, and activity partitioning on multiple parallel machines, a series of experiments have been conducted. The findings can be used to guide the design of scheduling expert systems.*

1. INTRODUCTION

TO CONSTRUCT and maintain large-scale job shop scheduling systems effectively, various objectives, requirements, and preferences should be considered. The job shop scheduling problem is intrinsically complex in its computation, and rescheduling becomes further complicated by unexpected events, such as machine breakdowns, raw material stockouts, personnel absences, and so forth. To handle effectively these complex issues of scheduling in manufacturing, various Artificial Intelligence (AI) approaches have been introduced (Fox & Smith, 1984; Keng, Yun, & Rossi, 1988; Lee & Suh, 1988; Ow & Smith, 1987; Sadeh & Fox, 1989). Most of the previous works on AI-based scheduling have adopted *centralized architecture*, where one agent is responsible for the overall scheduling. This architecture seems to produce good performances during the generative scheduling phase; however, it suffers from computational complexity and poor robustness to local changes during the reactive control phase. In this paper, we propose as an alternative architecture a two-layered *hierarchical architecture* for the job shop scheduling (Lee, Suh, & Fox, 1991). The architecture distributes the overall scheduling activities into multiple lower-level dispatchers, each of which schedules and controls a work area. If any uncontrollable conflicts occur within a work area scheduling, the higher level scheduler coordinates to resolve such conflicts. The experiments we performed indicate that no one architecture is superior to the other at all times. This implies that a different scheduling situation requires a different design. To help the design of scheduling expert systems, this paper thus attempts to suggest a proper architecture along with the scheduling strategies on the activity ordering, allocation time slot selection, and activity partitioning on multiple parallel machines.

Remaining sections are organized as follows. Section 2 defines the target job shop scheduling problem. Section 3 suggests a two-layered hierarchical architecture. Section 4 explains the role of a high-level coordinating scheduler in contrast with the lower-level dispatchers for each of the work areas. Section 5 illustrates the experimental result along with the suggestions for design of scheduling expert systems.

Requests for reprints should be sent to Jae K. Lee, Management Information Systems Program, Korea Advanced Institute of Science and Technology, P.O. Box 150, Cheongryang, Seoul, Korea

2. TARGET JOB SHOP SCHEDULING

Let us define the characteristics of the job shop scheduling problem we have explored as follows:

1. An order consists of a set of sequenced activities as shown in Figure 1. An activity is defined as the operation to be accomplished in a work area. Each order may have multiple processing routes that are prescribed by a process plan. An activity may require the production of a batch of items. However, we assume that an activity cannot start without completing the whole batch of preceding activities.
2. A group of machines with similar functions are located in each work area. Thus, it is possible for an activity to be divided among those machines.
3. During the actual manufacturing process, unexpected events, such as machine breakdowns and activity delays, may take place. Thus, reactive controls are required to reconcile the gap between the planned schedule and the actual progress on the shop floor.

For evaluating the performances of the scheduling systems, the following criteria are used in this paper.

Schedule performances during the generative scheduling phase. Important performance evaluation criteria are:

1. Total tardiness
2. Average flow time
3. Total number of setups

Run time necessary to construct a schedule. For an effective operation of large scale job shop scheduling systems, the schedule should be obtained within the acceptable response time.

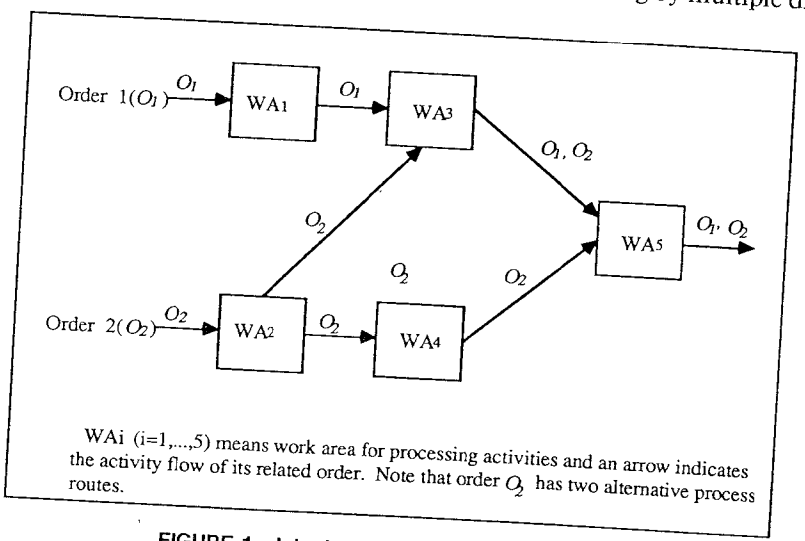
Robustness to changes in the local work areas. This criterion estimates the amount of effort necessary to adjust the global schedule reactively in response to the local changes caused by accidental events. Our goal is to achieve the robust scheduling system that can keep to the original schedule as much as possible (Smith, 1987).

All of the scheduling objectives described above cannot be achieved by a single particular scheduling heuristic. For instance, if the minimization of total tardiness is the scheduling objective of specific order, must use as many machines simultaneously as possibl which may result in an increased number of setup As such, the selection of the heuristic depends upo the objectives and scheduling situations. Therefore, th research seeks to find the appropriate contingencie between "the objectives and scheduling situations" an "the scheduling architectures and strategies" via a serie of experimental simulations.

3. TWO-LAYERED HIERARCHICAL ARCHITECTURE

Let us consider two typical architectures: centralized and hierarchical architectures. Within the centralized architecture as illustrated in Figure 2, the centralized scheduler takes the responsibility of the entire scheduling, such as the selection of the next activity to schedule, assignment of the selected activity to the machines in the work area, and responses to the unanticipated discrepancies from the original schedule.

In contrast to the centralized architecture, we propose a two-layered hierarchical architecture as illustrated in Figure 3 (Lee et al., 1991). In this hierarchical architecture, a large portion of the scheduling problem is delegated to multiple lower-level dispatchers who schedule and control respective work areas. Thus, each dispatcher can handle its own work area independently as far as it satisfies the requirements imposed by the higher-level coordinating scheduler. Decomposition of the scheduling activities becomes possible by appropriately allocating the total slack time of an order to the individual activities in each dispatcher. Thus the higher level *coordinating scheduler* controls the slack time allocation, process routing, and other interactions that may be necessary to resolve the infeasibility in dispatchers. The expected advantages of hierarchical architecture are: (1) reduced complexity by allowing parallel scheduling by multiple dispatchers, and (2) ro-



WAi (i=1,...,5) means work area for processing activities and an arrow indicates the activity flow of its related order. Note that order O_2 has two alternative process routes.

FIGURE 1. Job shop scheduling environment.

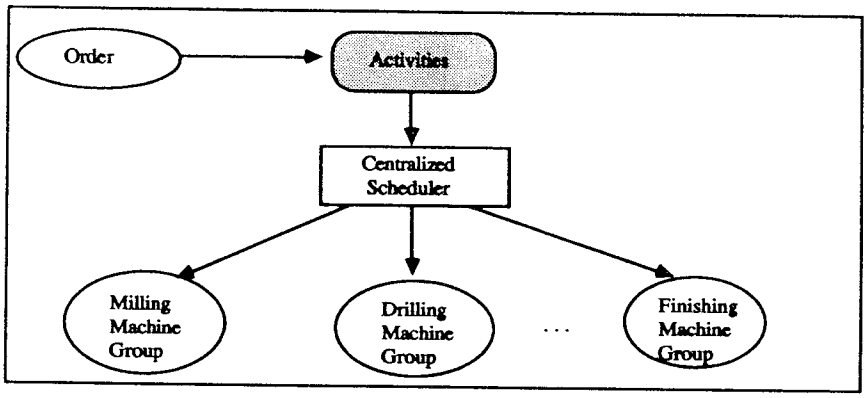


FIGURE 2. Centralized architecture.

bustness of original schedule to local changes. These advantages are confirmed by the experiments shown in Section 5.

4. ROLES OF COORDINATING SCHEDULERS AND DISPATCHERS

Let us describe in detail the roles of coordinating schedulers and dispatchers in the hierarchical architecture. The following notations are used throughout the remainder of this paper.

j: machine
d: dispatcher
$I(j,d)$: idle time intervals in machine j under the dispatcher d
i: order
k: activity
$A(i,k)$: k-th activity of order i, $k = 1, 2, \ldots, K_i$
$DA(i,k)$: demanded amount by the activity $A(i,k)$
$PR(i,k|d,j)$: production rate for the activity $A(i,k)$ by the machine j under the dispatcher d
$PT(i,k|m,d)$: processing time interval to complete the activity $A(i,k)$ using m machines simultaneously under dispatcher d

$$DA(i,k) = PT(i,k|m,d) * \sum_{j=1}^{m} PR(i,k|d,j)$$

$EST(i)$: earliest start time of order i
$LFT(i)$: latest finish time of order i
$EST(i,k)$: earliest start time of the activity $A(i,k)$ $EST(i,1) = EST(i)$
$LFT(i,k)$: latest finish time of the activity $A(i,k)$ $LFT(i,K_i) = LFT(i)$
$SST(i,k)$: scheduled start time of the activity $A(i,k)$
$SFT(i,k)$: scheduled finish time of activity $A(i,k)$
$ST(i)$: total slack time of order i
$ST(i,k)$: slack time of activity $A(i,k)$

4.1. Roles of Coordinating Scheduler

The coordinating scheduler plays the following roles:
1. Selection of a processing route
2. Decomposition of order's slack time to activities
3. Reaction to tardy activities
4. Reaction to machine idleness
5. Trade-off between tardiness and setups

These roles are described in the following subsections.

4.1.1. *Selection of a Processing Route.* When multiple alternative processing routes exist for an order, the scheduler selects one route in consideration of the order's preference and the resource utilization level of machines that are associated with the processing routes.

If an order has a preference on a particular processing route from the technical or economical point of view, the preference can be represented in frames as shown in Figure 4. The symbol p_1 stands for the level of preference which could imply one of the following: (1) utility, (2) priority, or (3) implicit prices. When no particular preference of the order exists, the processing route that utilizes the least busy resources can be applied.

4.1.2. *Decomposition of Order's Slack Time to Activities.* For a selected processing route of an order, usable time intervals for the associated activities can be represented by the earliest start time and the latest finish time. We introduce two types of latest finish time.
Definition: *Generic Latest Finish Time (GLFT).*

$$GLFT(i,k) = LFT(i) - \sum_{r=k+1}^{K_i} PT(i,r|m_r,d_r)$$

$GLFT(i,k)$ denotes that activity $A(i,k)$ should be completed by no later than the time to meet the due date of order i, assuming the succeeding activities can be performed without any slack time.

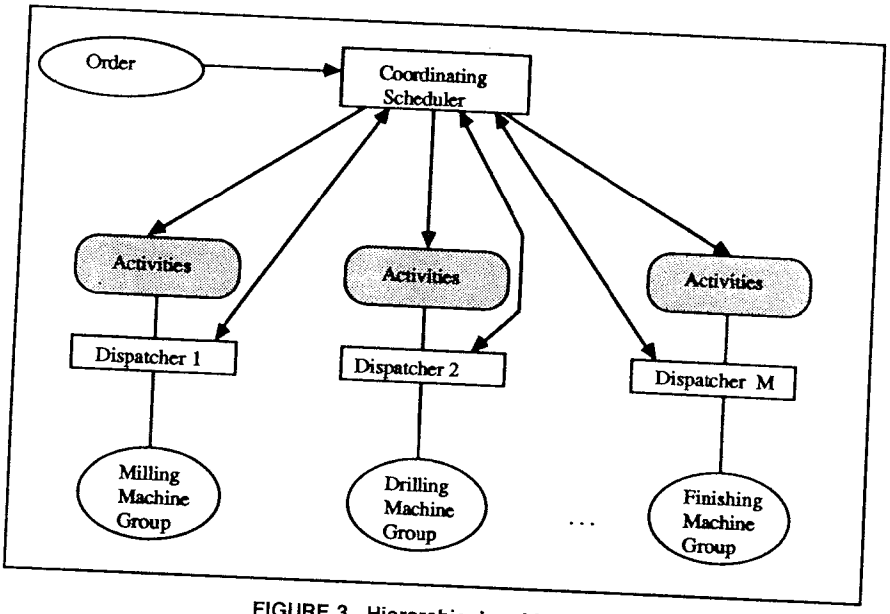


FIGURE 3. Hierarchical architecture.

Definition: *Total Slack Time of Order i, ST(i).*

$$ST(i) = [LFT(i) - EST(i)] - \sum_{r=1}^{K_i} PT(i, r|m_r, d_r)$$

If *GLFT* is used as the latest finish time of an activity, only the first activity in the processing route of the order occupies the total slack time. This might lead to infeasible schedules in the later stages of activities. To distribute the total slack time of an order properly among its activities, we need to consider the load level at each dispatcher. The total slack time may be distributed proportionally to the load level of a work area, or may be distributed equally if the load level information is not available.

$$ST(i, k) = w(i, k)*ST(i),$$

$$\text{where} \quad \sum_{k=1}^{K_i} w(i, k) = 1$$

To establish the notion of *LFT* considering the slack time, *Tighter LFT* (*TLFT*) is defined.

Definition: *Tighter Latest Finish Time* (*TLFT*).

$$TLFT(i, k) = LFT(i) - \sum_{r=k+1}^{K_i} [PT(i, r|m_r, d_r) + ST(i, r)$$

TLFT(*i*, *k*) is tighter than *GLFT*(*i*, *k*) by the sum of the associated slack times of succeeding activities *A*(*i*, *k*). If the user does not have any additional information, we can assume the following defaults:

$$LFT(i, k) = TLFT(i, k),$$

$$EST(i, k) = LFT(i, k - 1).$$

After having determined the time intervals of activities, the coordinating scheduler distributes the activities to the associated dispatchers with other additional information that is necessary for scheduling as illustrated in Figure 5.

In this manner, the coordinating scheduler allocates the slack time to the dispatchers. Then, each dispatcher will start to schedule each work area in parallel. When each dispatcher can handle its own scheduling without any problem, the entire scheduling is completed. How-

```
{{ Order-1
   is-a: Order
   demanded-amount: 300
   EST: 7/1/1992
   LFT: 8/3/1992
   processing-routes: (p1  (Milling Surface-Grinding Inspection ))
                      p2  (Shaping Surface-Grinding Inspection )) }}
```

FIGURE 4. Order's preference on processing routes.

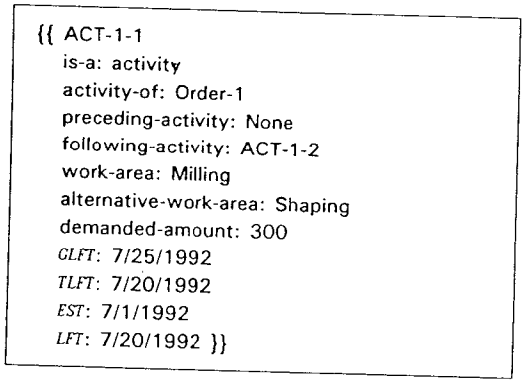


FIGURE 5. An activity frame delivered to its associated dispatchers.

ever, if any problems exist in regard of tardiness and idleness, such problems will be reported to the coordinating scheduler for a reactive scheduling.

4.1.3. *Reaction to Tardy Activities.* If a tardy activity occurs during the dispatcher's scheduling process, the dispatcher reports the tardiness to the coordinating scheduler. The coordinating scheduler then tries to eliminate the tardiness through the following reactions:

1. Check whether the current *SFT* is earlier than *GLFT*. If this is the case, postpone the *LFT* to the current *SFT*. Reset the slack time of the succeeding activities.
2. If the activity is tardy notwithstanding the *LFT* adjustment in step 1, split the tardy activity, assigning a portion to other alternative dispatchers.
3. If none of the above steps can remove the tardiness, the tardiness is unavoidable.

4.1.4. *Reaction to Machine Idleness.* The parallel scheduling of activities through multiple dispatchers may also cause idleness of machines. Removing the idleness may provide an opportunity for reducing the flow time and tardiness of other activities.

4.1.5. *Trade-off Between Tardiness and Setups.* Basically, the two goals, tardiness and the number of setups, are in conflict with each other, and thus, a trade-off between the two is necessary. The activity partitioning strategy on multiple machines affects the goals. By monitoring the status of goal achievements of dispatchers, the coordinating scheduler can initiate the change of such a strategy that is described in Section 4.2.3.

4.2. Roles of Dispatchers

To each dispatcher, a set of activities with their own usable time intervals queue up to be scheduled. The scheduling problem for each dispatcher is a parallel machine scheduling problem (Lee & Suh, 1988), in which the activities are scheduled on the multiple parallel machines with the same or similar functions. The dispatcher's scheduling procedure is composed of the following three steps:

1. Activities ordering
2. Selection of time slot
3. Partitioning of an activity to multiple machines.

4.2.1. *Activities Ordering.* The activities that await scheduling are sequenced by a predetermined ordering strategy. Various ordering strategies can be used in this step, and some of the typical strategies include the *least constraining strategy* (Keng et al., 1988; Sadeh & Fox, 1989), the *earliest due date* (EDD) strategy, and the *shortest processing time first* strategy. Many other dispatching rules in Operations Research (OR) can be found in Blackstone, Phillips, and Hogg (1982) and Panwalker and Iskander (1977). When a scheduling problem is considered as a Constraint Satisfaction Problem (CSP), activity ordering can be regarded as a sort of *variable ordering* (Dechter & Pearl, 1988). Therefore, a proper selection of the activity's ordering strategy contributes to the reduction of search space and good scheduling performance. In our study, two typical ordering strategies are compared with respect to their schedule performances: the *activity's criticality* from the resource-based perspective, and the activity with the *latest finish time.*

4.2.1.1. *Activities' criticality-based strategy.* This strategy selects the next activity to be scheduled based on the activity's criticality. In order to measure the criticality, the ratio of its demanded amount over its total available resource capacity is computed.

Let us assume that there are n machines under the control of dispatcher d. For each machine j, a set of idle time intervals $\{I_{j1}, I_{j2}, \ldots, I_{j,q}\}$ may exist, in which the activities can be assigned. For example, $\{(2, 5), (7, \infty)\}$ means that the machine is available for the interval (2, 5) and all time after time 7. Other time slots are already occupied by scheduled activities, machine breakdowns, etc. Using the information of both the required time intervals of activity $A(i, k)$ and of idleness, it is possible to obtain the total resource capacity (RC) of machines available for activity $A(i, k)$, which is denoted by $RC(i, k|d, j)$.

Definition: *Resource Reservation Ratio* (RR). For each unscheduled activity $A(i, k)$ in dispatcher d with n machines, the resource reservation ratio $RR(i, k|d)$ is defined as follows:

$$RR(i, k|d) = \frac{DA(i, k)}{\sum_{j=1}^{n} RC(i, k|d, j)}.$$

The larger RR means the activity is more critical because it has relatively less available resource capacity

for the demand. Therefore, it is reasonable to select the activity with the maximum RR first. $RR(i, k) > 1$ indicates that activity $A(i, k)$ cannot be scheduled without delay.

4.2.1.2. *Latest finish time-based strategy.* This strategy schedules the activities in an ascending order of the latest finish time. Thus, the order does not change during the scheduling process. This method is similar to the *earliest due date* rule.

4.2.2. *Selection of Time Slot.* The selected activity should be scheduled within the required time interval. Two strategies of selecting the time slot, Least Constraining Time Strategy and Earliest Time Allocation Strategy, are compared.

4.2.2.1. *Least constraining time strategy.* According to this strategy, the activity is allocated to the time slot that is least contended by the activities to be scheduled. As a measure of resource contention for the time slot within the required time interval, the total Sum of **R**esource **R**eservation ratio (SRR) at time t is define the sum of RRs of activities that can be assigne that time.

Let S denote the set of the remaining activitie be scheduled in dispatcher d.

Definition: *Total **S**um of **R**esource **R**eservation R* (SRR).

$$SRR(t) = \sum_{A(i,k)\in S} \delta_t(RR(i, k)),$$

where $\delta_t(x) = \begin{cases} x, & \text{if } t \in [EST(i, k), LFT(i, k \\ 0, & \text{otherwise.} \end{cases}$

A larger SRR value at a certain time point t indicat that this time slot is required by many activities. Thu it is desirable to reserve the time slot for later use lessen the impact on the remaining unscheduled acti ities. Thus the least constraining time strategy selec the time slot with the minimum SRR value.

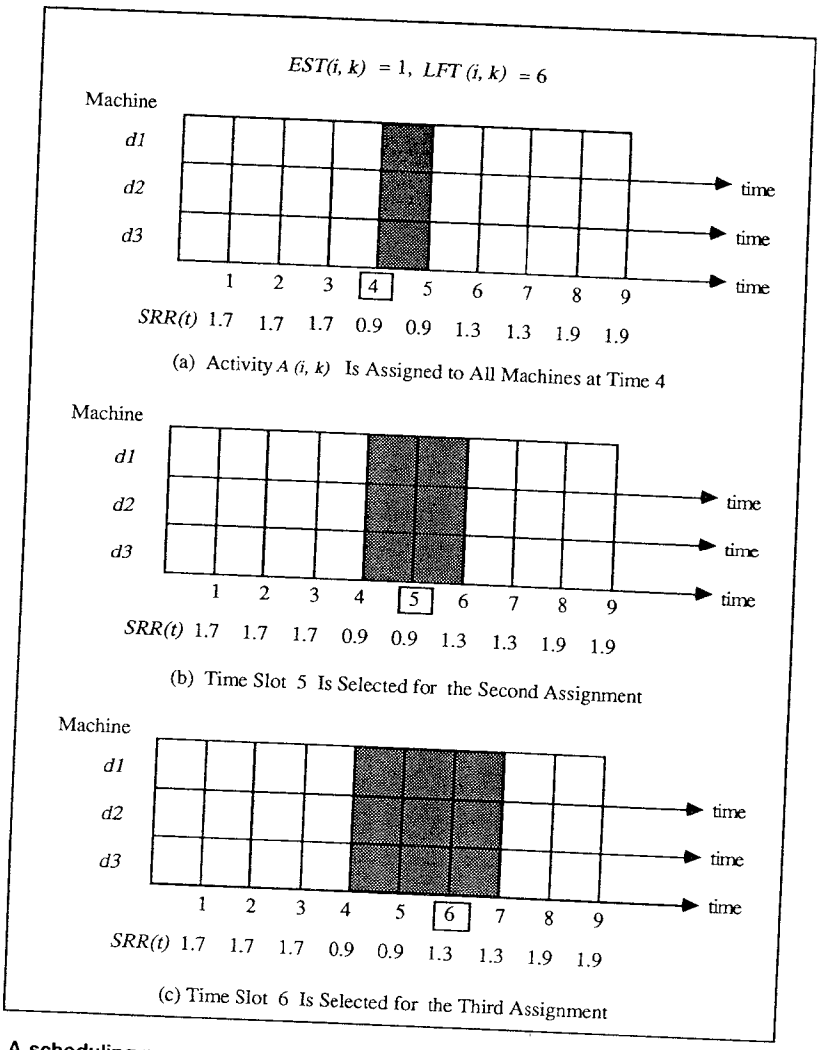


FIGURE 6. A scheduling process by the filling-water strategy with the least constraining time strategy.

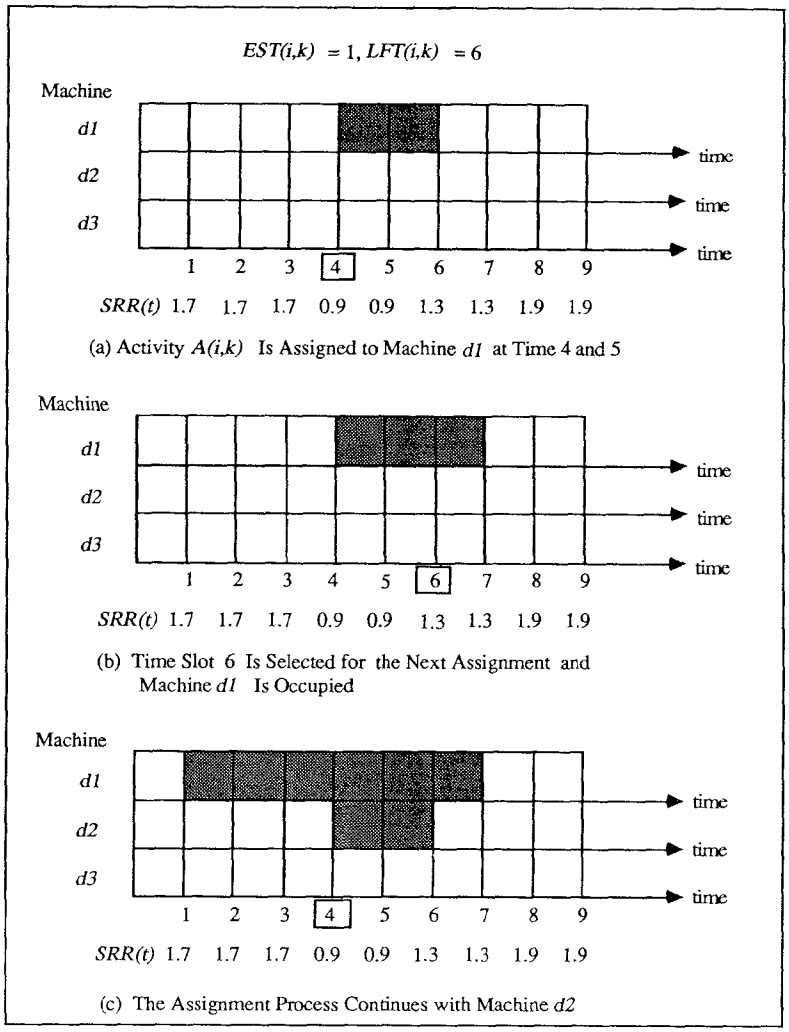


FIGURE 7. A scheduling process by the minimizing-setup strategy with the least constraining time strategy.

4.2.2.2. *Earliest time allocation strategy.* According to this strategy, the current activity is assigned at the earliest possible time of the available resources.

4.2.3. *Partition of an Activity to Multiple Machines.* Another issue is how to partition an activity when multiple parallel machines exist. Two typical strategies that can be considered are the *filling-water strategy* and the *minimizing-setup strategy.*

Definition: *Filling-Water Strategy* (*FWS*). This strategy incrementally assigns the required amount of an activity to multiple machines, starting with the machines with the earliest possible time. This strategy leads to the same finishing times of the partitioned activities at all selected machines.

This strategy resembles filling water of several interconnected buckets that have different depths. The water reaches the same levels after the filling. The schedule generation process employing the *FWS* with the time slot selected by the least constraining time strategy is shown in Figure 6. The *FWS* tends to minimize the total flow time and tardiness of activities, but it also tends to increase the number of setups because all machines that are allocatable to an activity are used simultaneously.

Definition: *Minimizing-Setup Strategy* (*MSS*). This strategy assigns the required amount of an activity to a machine with the earliest possible start time, up to the activity's *LFT*. If all of the amounts of the activity are assigned to the machine, the process then stops. Otherwise, the next earliest possible machine is selected among the remaining machines, and the above procedure is repeated until the entire amount is exhausted.

Figures 7 and 8 illustrate the scheduling processes by the *MSS* combined with the least constraining time strategy and the earliest time allocation strategy, respectively. The *MSS* tries to minimize the number of setups, but it tends to increase the flow time and tardiness.

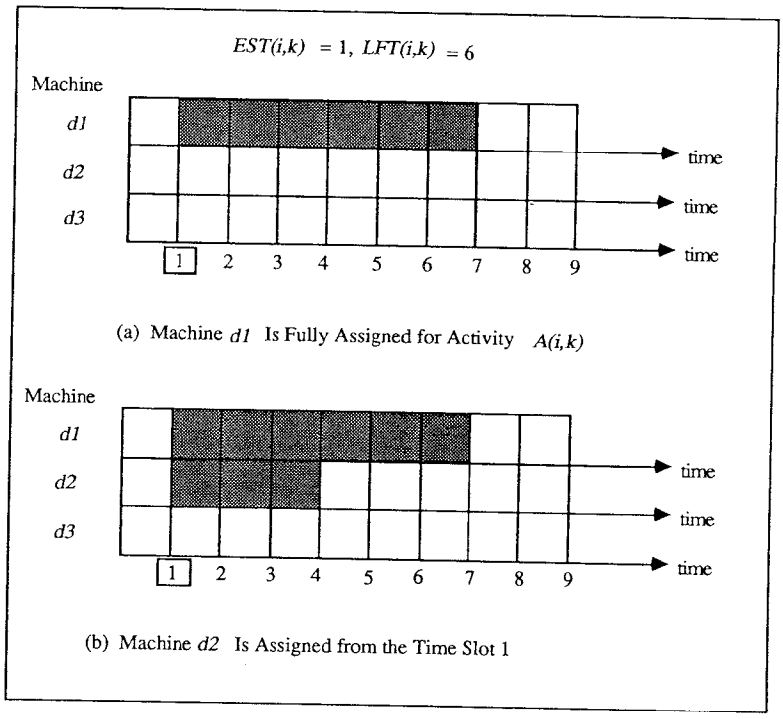


FIGURE 8. A scheduling process by the minimizing-setup strategy with the earliest time allocation strategy.

4.2.4. *Reactive Adjustment of Schedule.* If some changes occur during the manufacturing process, the dispatcher tries to respond to the changes while maintaining the original schedule as much as possible. Such a reactive adjustment is accomplished by using the push-backward and pull-forward actions. (Lee & Suh, 1988)

4.2.5. *Reporting Trouble to Coordinating Scheduler.* When a dispatcher cannot appropriately schedule the activities, it becomes necessary to report its difficulties to the coordinating scheduler during either of the two scheduling phases: *generative scheduling phase* if tardy activities exist; and *reactive control phase* if a dispatcher fails to absorb the impact of the unexpected events.

5. EXPERIMENTAL RESULTS

To evaluate the comparative advantages of architectures with different scheduling strategies, a series of experiments were conducted.

i. To build a test bed scheduling system, we can think of $2^4 = 16$ combinations of strategies by considering the following four factors.

- System's architecture: centralized (CEN) and hierarchical (HIER)
- Activities ordering strategy: resource reservation (RR) and latest finish time (LFT)
- Allocation time slot selection strategy: least constraining (LC) and earliest allocatable (EA)
- Activity partitioning strategy on multiple machines: filling-water (FWS) and minimizing-setup (MSS)

For example, the CEN-RR-LC-FWS strategy inc cates that the system adopts the centralized archite ture; the activity is selected by referring the resour reservation ratio; and the selected activity is assigne to the least constraining time slot by the filling wat strategy.

ii. We simulated the performance of combinatio of strategies under the following manufacturing settin The generated experimental test-bed consists of 5 wo areas, each with 3 or 4 machines and totalling 22 m chines in all. Five different processing routes were ge erated, and one of them was randomly assigned to eac order. For each case, 30 orders were generated, whic can be decomposed into 100 activities. To simulate th performances, 42 cases were generated by manipulatin the following three parameters: mean interarrival tim between the orders that follow exponential distributio required production amount of the order, and due da of the order. These parameters affected the resourc utilization level of the shop varying from 65% to 115% To simulate an actual manufacturing process, realize processing times of activities were also generated wit the deviations from the original time estimates by th error rate ranging from ±10% to ±30%. Under thes test settings, a pilot scheduling system for experimen tation was implemented using the LISP version of th proprietary frame-based tool UNIK-FRAME (Lee e al., 1988), and was tested on a SUN 3/280 workstation The performances were analyzed with respect to tar diness, setup, and flow time during both the generativ scheduling phase and the reactive control phase.

TABLE 1
Total Tardiness and Setup Performances

Architecture	Strategy	Total Tardiness		Total Number of Setups	
		Time (Day)	Percentage	Setups	Percentage
Centralized	CEN-RR-LC-FWS	425.93	100	512.81	100
	CEN-RR-LC-MSS	474.40	111	403.48	78
	CEN-RR-EA-FWS	362.07	85	522.81	101
	CEN-RR-EA-MSS	479.31	112	411.40	80
	CEN-LFT-LC-FWS	196.98	46	470.38	91
	CEN-LFT-LC-MSS	460.19	108	326.17	63
	CEN-LFT-EA-FWS	196.98	46	470.38	91
	CEN-LFT-EA-MSS	460.19	108	326.17	63
Hierarchical	HIER-RR-LC-FWS	258.14	60	501.50	97
	HIER-RR-LC-MSS	273.29	64	494.74	96
	HIER-RR-EA-FWS	260.02	61	501.62	97
	HIER-RR-EA-MSS	273.29	64	494.12	96
	HIER-LFT-LC-FWS	206.12	48	473.95	92
	HIER-LFT-LC-MSS	210.12	49	459.57	89
	HIER-LFT-EA-FWS	206.12	48	463.52	90
	HIER-LFT-EA-MSS	216.90	50	459.57	89

5.1. Tardiness and Setups During the Generative Scheduling Phase

The total tardiness and total number of setups for each strategy are compared as shown in Table 1. Among the 16 strategies, the best strategy in each architecture is:

Tardiness: CEN-LFT-EA (or LC)-FWS and HIER-LFT-EA (or LC)-FWS

Setups: CEN-LFT-EA (or LC)-MSS and HIER-LFT-EA (or LC)-MSS

Based on ANOVA test, we could find the following.

1. For the activity ordering, the latest finish time (LFT)-based strategy was always superior to the resource reservation (RR)-based strategy. The F values in the criteria of tardiness and setups are $F_{1,656} = 0.221$ ($p = 0.639$) and $F_{1,656} = 0.708$ ($p = 0.708$),

TABLE 2
Average Flow Time Performance

Architecture	Strategy	Average Flow Time	
		Time (Day)	Percentage
Centralized	CEN-RR-LC-FWS	59.03	100
	CEN-RR-LC-MSS	63.75	107
	CEN-RR-EA-FWS	53.17	90
	CEN-RR-EA-MSS	63.61	107
	CEN-LFT-LC-FWS	45.96	77
	CEN-LFT-LC-MSS	64.82	109
	CEN-LFT-EA-FWS	45.96	77
	CEN-LFT-EA-MSS	64.82	109
Hierarchical	HIER-RR-LC-FWS	48.24	81
	HIER-RR-LC-MSS	48.81	82
	HIER-RR-EA-FWS	48.31	81
	HIER-RR-EA-MSS	48.81	82
	HIER-LFT-LC-FWS	44.79	75
	HIER-LFT-LC-MSS	45.72	77
	HIER-LFT-EA-FWS	44.79	75
	HIER-LFT-EA-MSS	45.40	76

TABLE 3
CPU Time Performance

Architecture	Strategy	CPU Time	
		Seconds	Percentage
Centralized	CEN-RR-LC-FWS	602.17	100
	CEN-RR-LC-MSS	570.18	94
	CEN-RR-EA-FWS	364.50	60
	CEN-RR-EA-MSS	329.58	54
	CEN-LFT-LC-FWS	675.53	112
	CEN-LFT-LC-MSS	563.50	93
	CEN-LFT-EA-FWS	172.49	28
	CEN-LFT-EA-MSS	163.38	27
Hierarchical	HIER-RR-LC-FWS	218.75	36
	HIER-RR-LC-MSS	208.28	34
	HIER-RR-EA-FWS	192.53	31
	HIER-RR-EA-MSS	186.27	30
	HIER-LFT-LC-FWS	98.56	16
	HIER-LFT-LC-MSS	97.84	16
	HIER-LFT-EA-FWS	87.85	14
	HIER-LFT-EA-MSS	68.26	11

respectively. Table 1 shows that when the *LFT* strategy is used for activity ordering, the effect of the activity start time, whether to choose the least constraining (LC) strategy or the earliest allocating (EA) strategy, was insignificant.

2. The trade-off between the tardiness and the setups could be well accomplished by the *FWS* and *MSS*.
3. When the best combinations of strategies in each architecture are mutually compared, the centralized architecture outperformed the hierarchical architecture by 5% of tardiness and 30% of number of setups, respectively. The inferiority of the hierarchical architecture during the generative scheduling phase is attributable to the inherent coordinating efforts. Therefore, if no revision is necessary during the reactive control phase, the centralized architecture is more recommendable.

5.2. Flow Time During the Generative Scheduling Phase

The average flow time for each strategy is shown in Table 2. The selection of the time slot did not significantly affect the flow time. That is, $F_{1,656} = 1.520$; p 0.218. In each architecture, the CEN-LFT-EA (or LC FWS and the HIER-LFT-EA (or LC)-FWS produc the minimum amounts of average flow time, respe tively. Through a proper allocation of the total sla time to multiple dispatchers, the hierarchical archite ture could achieve less flow time than the centra ized one.

5.3. Scheduling Efficiency During the Generative Scheduling Phase

The average CPU time spent to generate schedules fo each strategy is shown in Table 3. In the hierarchica architecture, the total CPU time was calculated b adding the average CPU time of dispatchers to the CPU time spent by the coordinating scheduler.

1. The effect of the activity's time slot allocation strat egy is significant. ($F_{1,656} = 1977.491$; $p < 0.01$). Th earliest time allocating (EA) strategy recorded les run time than the least constraining (LC) strategy This is due to the computational burden of the leas

TABLE 4
Total Number of Revisional Invocations During the Reactive Control Phase

Strategy	Estimation Error Rate					
	±10%		±20%		±30%	
	Count	Percentage	Count	Percentage	Count	Percentage
CEN-LFT-EA-FWS	426.55	100%	430.71	100%	400.55	100%
HIER-LFT-EA-FWS	408.10	95%	417.79	97%	368.38	91%
t-value	4.35*		3.56*		6.65*	

* Means $p < 0.01$.

TABLE 5
Total Amount of the Changed Schedule Time

	Estimation Error Rate					
	±10%		±20%		±30%	
Strategy	Count	Percentage	Count	Percentage	Count	Percentage
CEN-LFT-EA-FWS	2,612.81	100%	2,976.55	100%	2,491.50	100%
HIER-LFT-EA-FWS	1,993.26	76%	2,287.07	76%	1,871.98	75%
t-value	10.06*		10.93*		7.98*	

* Mean $p < 0.01$.

constraining strategy for calculating the least constrained time slots.

2. As expected, each strategy under the hierarchical architecture took less run time than its corresponding strategy under the centralized architecture because the former distributes the computational burden among the dispatchers. The gap between the two architectures will become wider as the number of work areas increases.
3. Within each architecture, the *MSS* took less run time than the *FWS*. This is because the maintenance of the same finishing time for FWS takes heavy computation. The strategy with the minimum amounts of run time in each architecture was the HIER-LFT-EA-MSS and the CEN-LFT-EA-MSS, respectively.

5.4. Robustness During the Reactive Control Phase

To compare the robustness of scheduling systems, the performances of two strategies, CEN-LFT-EA-FWS and HIER-LFT-EA-FWS, are compared with respect to the tardiness and flow time.

1. The hierarchical architecture exerted less reactive effort than the centralized one. The effort is measured by counting the total number of revisional invocations, such as the pull-forward and the push-backward reactions during the reactive control phase. The hierarchical architecture reduced the total number of revisional invocations by approximately 5% (see Table 4). The total amount of the changed schedule time is reduced by more than 24% (see Table 5).
2. Under the hierarchical architecture, the performance degradation caused by errors in processing time estimation is relatively small. The tardiness and flow time, during the generative scheduling phase and the reactive control phase are shown in Table 6 and Table 7 respectively. Thus, we can see that the hierarchical architecture is more robust to local changes than the centralized one.

6. CONCLUSION

In order to find the best combination of scheduling system's architecture and scheduling strategies, a comprehensive experiment is conducted. We found that during the generative scheduling phase, the centralized architecture outperforms the hierarchical architecture with regard to both the tardiness and the number of setups. The hierarchical architecture, however, requires less computation time; thus, under the real-time scheduling environment, the hierarchical architecture is more recommendable. During the reactive control

TABLE 6
Ratio of Tardiness Between the Generative Scheduling Phase and the Reactive Control Phase

		Estimation Error Rate					
		±10%		±20%		±30%	
Strategy	TGSP[1]	TRCP[2]	Ratio[3]	TRCP	Ratio	TRCP	Ratio
CEN-LFT-EA-FWS	196.98	342.98	1.74	369.69	1.87	359.36	1.82
HIER-LFT-EA-FWS	206.12	312.07	1.51	335.29	1.62	325.02	1.57
t-value	−1.97*	8.04**		8.46**		6.26**	

[1] Tardiness during the Generative Scheduling Phase.
[2] Tardiness during the Reactive Control Phase.
[3] Ratio = TRCP/TGSP.
* $p < 0.05$.
** $p < 0.01$.

TABLE 7
Ratio of Flow Time Between the Generative Scheduling Phase and the Reactive Control Phase

		Estimation Error Rate				
		±10%		±20%		±30
Strategy	FGSP[1]	FRCP[2]	Ratio[3]	FRCP	Ratio	FRCP
CEN-LFT-EA-FWS	45.96	53.61	1.16	55.08	1.19	53.86
HIER-LFT-EA-FWS	44.79	51.29	1.14	52.46	1.17	51.38
t-value	4.11*	7.34*		9.66*		8.00*

[1] Flow time during the Generative Scheduling Phase.
[2] Flow time during the Reactive Control Phase.
[3] Ratio = FRCP/FGSP
* $p < 0.01$.

phase, the hierarchical architecture was more robust in responding to local changes of the original schedule. Therefore, when the robustness is the main objective of scheduling, the hierarchical architecture is more recommendable. Concerning the activity ordering, the latest finish time (LFT)-based strategy outperformed the resource reservation (RR)-based strategy. With regard to the trade-off between the tardiness and the setups, the filling water strategy could outperform with respect to the tardiness and flow time, while the minimizing setup strategy could distinguishably reduce the number of setups.

In conclusion, when we develop scheduling expert systems under the dynamic manufacturing environment, the designer should carefully select the system's architecture together with the scheduling strategies suitable to the nature and objectives of the scheduling situation.

REFERENCES

Blackstone, J.H., Phillips, D.T., & Hogg, G.L. (1982). A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Production Research,* **20**(1), 27–45.

Dechter, R., & Pearl, J. (1988). Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence,* **34**(1), 1–38.

Fox, M.S., & Smith, S.F. (1984). ISIS: A knowledge-based for factory scheduling. *Expert Systems,* **1**(1), 25–49.

Keng, N.P., Yun, D.Y.Y., & Rossi, M. (1988). Interaction-s planning system for job-shop scheduling. In *Proceeding Second Conference on Expert Systems and the Leading Production Planning and Control,* South Carolina (pp. 5

Lee, J.K., Suh, M.S., Oh, S.B., Kim, M.Y., Jung, J.B., Shin, Song, Y.U. (1988). *Development of UNIK for the integra knowledge and optimization models and its application to leum industry,* Technical Report N486-3509-7, KAIST.

Lee, J.K., & Suh, M.S. (1988). PAMS: A domain-specific know based parallel machine scheduling system. *Expert System* 198–213.

Lee, J.K., Suh, M.S., & Fox, M.S. (1991). A hierarchical sche expert system: KAIS-3. In *Proceedings of the World Congr Expert Systems* (pp. 3030–3039). Tarrytown: Pergamon P

Ow, P.S., & Smith, S.F. (1987). Viewing scheduling as an opportu problem-solving process. In R.G. Jeroslow (Ed.), *Annals erations research: Approaches to intelligent decision suppor* 85–108). Baltzer Scientific Publishing Co.

Panwalker, S.S., & Iskander, W. (1977). A survey of scheduling *Operations Research,* **25**(1), 45–61.

Sadeh, N., & Fox, M.S. (1989). Focus of attention in an acti based scheduling. In *Proceedings of the NASA Conference on S Telerobotic,* Pasadena, CA.

Smith, S. F. (1987). A constraint-based framework for reactive m agement of factory schedules. In *Proceedings of the First Co ence on Expert Systems and the Leading Edge in Produc Planning and Control.* (pp. 113–130).