

論文 / 著書情報
Article / Book Information

題目(和文)	
Title(English)	An intelligent scheduling system for management of flexible production
著者(和文)	石井信明
Author(English)	Ishii Nobuaki
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第3081号, 授与年月日:1995年9月30日, 学位の種別:課程博士, 審査員:
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第3081号, Conferred date:1995/9/30, Degree Type:Course doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

**AN INTELLIGENT SCHEDULING SYSTEM
FOR
MANAGEMENT OF FLEXIBLE PRODUCTION**

Nobuaki Ishii

May, 1995

**Submitted in Partial Fulfillment
of the Requirements for the Degree of
Doctor of Engineering**

Thesis Advisor: Associate Professor Masaaki Muraki

**Department of Industrial Engineering and Management
Tokyo Institute of Technology**

ABSTRACT

Today's manufacturers, especially in industrialized nations, are facing a dynamic business environment where business conditions including production information, such as demand, product variety, production technologies, change frequently and unforeseeably. In the circumstances, the intelligent scheduling system, which can effectively manage flexible production systems subject to frequent changes of production information within this dynamic business environment, has been a big issue for manufacturers. In this research, a generic framework for the intelligent scheduling system is created. The detailed structures and mechanisms for the intelligent scheduling system are investigated for FMS (Flexible Manufacturing System) and batch processes which are currently employed in practice as typical flexible production systems for, respectively, discrete part production and process type production in the chemical industries.

As an application in FMS, a transient based real-time scheduling system, which selects a dispatching rule dynamically for the next short period, to respond to changes of system status, is proposed as one of the structures of the intelligent scheduling system. The proposed system defines the next short period for a schedule revision based on the system transient state, and selects a dispatching rule for the period by discrete event simulation in consideration of the problem of censored data. In addition, a mixed dispatching rule approach which can assign different dispatching rules for each machine to improve the performance of real-time scheduling systems is proposed as a mechanism within the structure.

As an application in batch processes, an on-line scheduling system, which modifies a schedule in a certain short period according to the changes of production information, is proposed as one of the structures of the intelligent scheduling system. The proposed system uses different schedule modification strategies to correct a schedule within the modification space specified according to the disruptions or performance degradations of a schedule. In this research, key schedule modification strategies used within the structure are proposed. Namely, an extended dispatching rule approach which rapidly generates a complete schedule under various scheduling constraints, and a process-variability-based on-line scheduling system which modifies a schedule in consideration of waiting time constraints between tasks under processing time variations, are proposed.

The performance of the proposed systems is evaluated and discussed based on the simulation experiments with production models.

TABLE OF CONTENTS

ABSTRACT	ii
LIST OF TABLES	vii
LIST OF FIGURES	ix
ACKNOWLEDGEMENTS	xi

PART 1:

Chapter 1

INTRODUCTION	2
1.1 Background	2
1.2 Thesis Objectives	5
1.3 Overview of the Research	6
1.4 Flexible Production System	9
1.4.1 FMS (Flexible Manufacturing System)	9
1.4.2 Batch Process	11
1.5 Overview of Intelligent Scheduling System	14
1.5.1 Fundamental Scope of Intelligent Scheduling System	14
1.5.2 A Framework for Intelligent Scheduling System	17
1.5.3 Structures of Intelligent Scheduling System	21
1.6 Literature Review	23
1.7 Chapter Summary	28

PART 2:

Chapter 2

A TRANSIENT BASED

REAL-TIME SCHEDULING SYSTEM IN FMS 31

2.1	Chapter Overview	31
2.2	Problem Description and Background	32
2.3	Transient Based Real-Time Scheduling System	38
2.3.1	Transient Detection and Scheduling Interval Setting Module	40
2.3.2	Dispatching Rule Selection Module	44
2.4	Experiments on Real-Time Scheduling System	49
2.4.1	An FMS Model	50
2.4.2	Design of Experiments	53
2.4.3	Results of Experiments	57
2.5	Chapter Summary	65

Chapter 3

A MIXED DISPATCHING RULE APPROACH

IN FMS SCHEDULING 66

3.1	Chapter Overview	66
3.2	Background	67
3.3	Search Algorithm for MDR Approach	70
3.3.1	Basic Dispatching Rules	70
3.3.2	Search Strategy	71
3.4	Example and Evaluation of MDR Approach	74
3.4.1	An FMS Model	74
3.4.2	Detailed Explanation of MDR Search Algorithm	77
3.4.3	Effectiveness and Efficiency of MDR Approach	81
3.5	Chapter Summary	87

PART 3:

Chapter 4

A FUNDAMENTAL STRUCTURE OF ON-LINE SCHEDULING SYSTEM IN BATCH PROCESS MANAGEMENT

4.1	Chapter Overview	89
4.2	Previous works and Research Needs	90
4.3	Fundamental Structure for On-line Scheduling System (FOSS)	94
4.3.1	Scope of the System	94
4.3.2	Concept of FOSS	96
4.3.3	Overview of the Structure	100
4.3.4	Modules and Functions for FOSS	102
4.4	Experiment for Evaluation	106
4.4.1	Example Problem for Experiment	106
4.4.2	Results of Experiment	122
4.5	Chapter Summary	132

Chapter 5

AN EXTENDED DISPATCHING RULE APPROACH WITHIN ON-LINE SCHEDULING STRUCTURE IN BATCH PROCESS

5.1	Chapter Overview	133
5.2	Background	134
5.3	Extended Dispatching Rule Approach	137
5.3.1	Dispatching-Simulation Mechanism	137
5.3.2	Concept of EDR Approach	140
5.3.3	EDR Search Algorithm	140
5.4	Evaluation of EDR Approach	145
5.4.1	Design of Experiments	145
5.4.2	Results of Experiments	151
5.5	Chapter Summary	161

Chapter 6	
A PROCESS-VARIABILITY-BASED SCHEDULING SYSTEM WITHIN ON-LINE SCHEDULING STRUCTURE IN BATCH PROCESS	162
6.1 Chapter Overview	162
6.2 Background and Problem Description	163
6.2.1 Background and Previous Works	163
6.2.2 Problem Description	167
6.3 Process-Variability-Based On-line Scheduling System	168
6.3.1 Overview of PVSS	168
6.3.2 Details of PVSS	169
6.4 Evaluation by Simulation Experiments	180
6.4.1 Design of Experiments	180
6.4.2 Results of Experiments	186
6.5 Chapter Summary	199
PART 4:	
Chapter 7:	
CONCLUSIONS	202
7.1 Summary and Conclusions of the Research	202
7.2 Promising Areas for Further Study	205
REFERENCES	207

LIST OF TABLES

Table 1.1	Summary of promising structures and research needs for the intelligent scheduling system	29
Table 2.1	Process sequence	52
Table 2.2	Processing time (in minutes) and due date	52
Table 2.3	Part release schedule for EX-A (Part arrival rate)	54
Table 2.4	Part release schedule for EX-B (Part arrival rate)	54
Table 2.5	Part release schedule for EX-C (Part arrival rate)	55
Table 2.6	Comparison of real-time scheduling system (Part release schedule: EX-A)	61
Table 2.7	Comparison of real-time scheduling system (Part release schedule: EX-B)	62
Table 2.8	Comparison of real-time scheduling system (Part release schedule: EX-C)	63
Table 2.9	Summary of the percentage improvement against single-pass scheduling algorithm	64
Table 3.1	An example of Effective Rule Table (ERT)	73
Table 3.2	An example of Bottleneck Machine Table (BMT)	73
Table 3.3	Results of experiments (Part release schedule: EX-A)	84
Table 3.4	Results of experiments (Part release schedule: EX-B)	85
Table 3.5	Results of experiments (Part release schedule: EX-C)	86
Table 4.1	Changeover times between products	114
Table 4.2	Initial inventory level	114
Table 4.3	Production plan received on September 1st	115
Table 4.4 (a)	A space decision table	121
Table 4.4 (b)	A strategy decision table	121
Table 5.1	Production order and inventory level	148
Table 5.2	Summary of experiments (Single dispatching rules)	152

Table 5.3	Summary of experiments (EDR, BDR, RS, EDR_R)		153
Table 5.4	Paired comparison between RS, RS_AVE, and BDR		155
Table 5.5	Paired comparison between BDR, RS, EDR_R, EDR_R_AVE, and EDR		156
Table 5.6	ANOVA tables (Effects of waiting time constraints)		158
Table 5.7	ANOVA & mean tables (Effects of performance criteria)		159
Table 6.1	Processing times		182
Table 6.2	Results of experiments (Product campaign scenario: SPC1)		192
Table 6.3	Results of experiments (Product campaign scenario: SPC2)	...	193
Table 6.4	Results of experiments (Product campaign scenario: SPC3)		194
Table 6.5	Results of experiments (Product campaign scenario: SPC4)		195
Table 6.6	Results of experiments (Product campaign scenario: MPC1)	...	196
Table 6.7	Percentage increase on mean makespan by Strategies 1 and 2		197
Table 6.8	Percentage increase on mean makespan by Strategy 2 under WTC1		197
Table 6.9	Percentage increase on mean makespan by slack time insertion		198

LIST OF FIGURES

Figure 1.1	Overview of the research	7
Figure 1.2	An example of FMS	12
Figure 1.3	An example of batch process in chemical industries	12
Figure 1.4	Boundaries of intelligent scheduling system within production management	15
Figure 1.5	A classification of production information associated with the intelligent scheduling system	16
Figure 1.6	A framework for an intelligent scheduling system	19
Figure 2.1	Boundaries of real-time scheduling system	33
Figure 2.2	Next scheduling interval by transient based approach and constant interval approach	37
Figure 2.3	An overview of the transient based real-time scheduling system	39
Figure 2.4	Transient detection algorithm	43
Figure 2.5	Minimum Scheduling Interval (MSI)	45
Figure 2.6	Simulation window in Strategy 2	47
Figure 2.7	Physical layout of an FMS model	51
Figure 3.1	An example of simple FMS	69
Figure 3.2	A search algorithm for MDR approach	78
Figure 3.3	Example iterations of MDR search algorithm	79
Figure 4.1	Boundaries of on-line scheduling system	95
Figure 4.2	A schedule modification model for on-line scheduling system	98
Figure 4.3	Overview of FOSS	101
Figure 4.4	Overview of variability check module (VCM)	103
Figure 4.5	Material flow in example plant	107

Figure 4.6	Equipment configuration	108
Figure 4.7	Product interdependency and mass balance per batch	110
Figure 4.8 (a)	Process time chart of IP1 and IP2	112
Figure 4.8 (b)	Process time chart of FP1, FP2, and FP3	113
Figure 4.9	Decision table approach: mechanism to select corrective action strategy	120
Figure 4.10 (SC0)	Original schedule in SC0	127
Figure 4.10 (SC1)	Modified schedule in SC1	128
Figure 4.10 (SC2)	Modified schedule in SC2	129
Figure 4.10 (SC3)	Modified schedule in SC3	130
Figure 4.10 (SC4)	Modified schedule in SC4	131
Figure 5.1	A conceptual search process	136
Figure 5.2	A schema of dispatching and simulation mechanism	139
Figure 5.3	Overview of two-stage EDR search algorithm	142
Figure 5.4	Example of search process by EDR search algorithm	144
Figure 5.5	A multiproduct process model	147
Figure 5.6	Product interdependency and materials balance	147
Figure 6.1	Effects of processing time variations	165
Figure 6.2 (a)	An example of schedule modification by Strategy 1 (S1)	172
Figure 6.2 (b)	An example of schedule modification by Strategy 2 (S2)	172
Figure 6.3	Schedule modification algorithm	173
Figure 6.4	Shift time (S_{time}) and schedule modification with fixed processing times	176
Figure 6.5	Shift time (S_{time}) and schedule modification based on processing time variations by PVSS	177
Figure 6.6	Task network for experiments	181
Figure 6.7	Maximum deviation from waiting time constraint under PV2 in case of no slack time insertion	187

ACKNOWLEDGEMENTS

I wish to express my deepest gratitude to Associate Professor Masaaki Muraki, my thesis advisor, for his insightful guidance throughout my studies at Tokyo Institute of Technology (T.I.T.). His enlightening advice and useful direction were indispensable in planning and carrying out the research herein.

I am very grateful to Professor Joseph J. Talavage, School of Industrial Engineering, Purdue University, West Lafayette, Indiana, for his beneficial advice and support while I was at Purdue. My research at Purdue with his support formed the basis of this thesis.

I would also like to thank Research Associate Tetsuo Fuchino and all colleagues in Muraki Lab. who have given me warm encouragement and so much help.

In addition, I would like to acknowledge with thanks the financial support of JGC Corporation (Nikki K.K.) which made my research possible.

Last of all, I am thankful to all my family for their patience and continuing encouragement during my studies at T.I.T.

PART 1:

INTRODUCTION

TO

INTELLIGENT SCHEDULING SYSTEM

Chapter 1

INTRODUCTION

1.1 Background

Today's manufacturers, especially in industrialized nations, are facing a dynamic business environment where business conditions including production information, such as demand, product variety, production technologies, change frequently and unforeseeably. In the circumstances, manufacturers must quickly grasp changes in business conditions, and then take strategic actions immediately and flexibly on a worldwide basis to survive and succeed. The dynamic business environment requires totally different guiding principles for manufacturers who have conducted high volume production of a limited number of standardized products under virtually fixed business conditions since the industrial revolution.

A large number of new concepts and theories for manufacturing industries have been proposed from various points of view against the changes in the business environment. Typical examples of the emerging concepts and theories on organization, engineering, and corporate management are the dynamic human networking organization by Savage (1990), the flotilla structure by Drucker (1990), agile manufacturing enterprise (Hormozi, 1994) to realize quick response to market conditions, and concurrent engineering (Parsaei and Sullivan, 1993) to shorten the period of new product development.

Today's dynamic business environment also necessitates new approaches to production systems and their operations including production scheduling. In the past, production systems which provided high volume production of a limited number of

standardized products at low cost brought increased profit for manufacturers. In today's dynamic business environment, however, establishment of flexible production systems which can produce a wide variety of products efficiently is a vital issue for manufacturers. (In this research, a production system is defined as flexible if it is capable of processing a number of different products in any sequence.)

For example, manufacturers have employed FMS (Flexible Manufacturing System) (Talavage and Hannam, 1988) for discrete part production, and batch processes for process type production to realize flexible production in practice. In addition, extensive researches to increase the flexibility of production systems, such as bionical manufacturing systems (Okino, 1992), have been stimulated in the last decade. (FMS will be used for both singular (system) and plural (systems) in this thesis.)

However, no flexible production systems can be well utilized in a dynamic business environment with supervision by conventional production management systems which have been established based on a static business environment, and they cannot rapidly adapt to the frequent changes of production information due to said dynamic business environment. Therefore excess capacity, such as excess inventory, excess buffer usage, and low facility utilization, to absorb changes of production information, is often associated with today's flexible production systems. Over the last decade, however, many manufacturers have been faced with severe worldwide competition. To survive and prosper in this environment, manufacturers must have the ability to produce with low price, high quality, and short turn around. For this reason, reducing inventory, buffer size, and production capacity, and improving the utilization of production facilities are vital issues in flexible production systems. Therefore new concept production management to bring out the potential ability of flexible production systems in the dynamic business environment is strongly required.

There are at least two issues in realizing a new concept production management system for flexible production systems, as follows;

- Establishment of production information infrastructure to acquire and control a wide variety of information associated with production activities in a distributed environment,

- Establishment of mechanisms, such as production planning, scheduling, and control, to manage flexible production systems according to the dynamic changes of the production information acquired.

The production information infrastructure can be established by using today's advanced information technologies, such as high performance computers, data bases, and networks. However, the mechanisms to manage flexible production systems according to the dynamic changes of production information have not been well established so far because of the complexity and difficulty of the problem.

In the static business environment, production planning (Vollmann *et al.*, 1988) which has usually been established annually or quarterly has played a major role in production management. For example, purchasing of materials and production scheduling have been done based on the results of the planning. In the dynamic business environment, however, virtually no long-term production plan makes sense in production management because of frequent changes of the business conditions which require frequent revisions of the plan. Therefore scheduling systems which coordinate equipment, materials, and resources within production systems according to the latest production information become much more important than ever before in production management to improve the performance of flexible production systems in the dynamic business environment.

However, conventional scheduling systems assume static or off-line scheduling. For example, production demands and processing times are assumed to be unchanged over a scheduling period. Furthermore, production systems are assumed to be operated exactly as scheduled. In practice, hardly any production system can be operated on a schedule generated in advance because it is quite usual to face, for example, frequent demand changes and variations of processing times during a schedule execution in the dynamic business environment. These dynamic changes cause increased waiting times, increased work-in-process, low equipment utilization, and low product quality, and therefore reduced performance of flexible production systems. In addition, it is impossible to make complete schedules for flexible production systems in advance, because of their complex and dynamic

behavior due to multi-product production in various sequences and frequent stop-and-go operations. Therefore the schedules in practice tend to include much slack to absorb uncertainties. In the circumstances, production staff in factories have to carry out the hard task of schedule revision. However, increasing product variety and the frequent changes of production information make manual schedule revisions virtually impossible. For this reason, an intelligent scheduling system which can effectively manage flexible production systems under the frequent changes of production information due to the dynamic business environment is strongly required in modern production management.

1.2 Thesis Objectives

The objectives of this research are to create a framework, structures, and mechanisms within each structure, for an intelligent scheduling system which manages flexible production systems effectively, according to the dynamic changes of production information in today's dynamic business environment. More specifically, this research investigates and proposes the following.

- A generic framework for the intelligent scheduling system
- Detailed structures for the intelligent scheduling system depending on the characteristics of the following flexible production systems, and mechanisms within each structure
 - FMS in discrete part production
 - Batch processes in chemical industries

In this research, a framework is defined as scope and fundamental functions of the intelligent scheduling system. A structure, on the other hand, is defined as a fundamental approach and a scheme to realize the intelligent scheduling system within the framework in each target flexible production system. In addition, mechanisms are defined as methodologies or algorithms used within a structure. Several structures within a framework could be available depending on the characteristics of each flexible production system.

Therefore this research focuses on FMS and batch processes to study structures and mechanisms in detail.

1.3 Overview of the Research

This research consists of four parts, i.e., introduction, application of intelligent scheduling system in FMS, application of intelligent scheduling system in batch process, and conclusions as shown in Figure 1.1. The following presents an overview of this research in each part.

Part 1: Introduction

The following section in this chapter identifies FMS and batch processes in chemical industries as typical flexible production systems in practice for, respectively, discrete part production and process type production, and investigates their characteristics from the intelligent scheduling perspective.

In addition, the scope and a generic framework for the intelligent scheduling system are presented, together with the possible structures within the framework to realize the intelligent scheduling system for each flexible production system selected.

Followed by general descriptions on the intelligent scheduling system, previous works in production scheduling systems are briefly surveyed, and then the research needs to realize the intelligent scheduling system are clarified.

Part 2: Application of intelligent scheduling system in FMS

Part 2, consisting of Chapters 2 and 3, provides a real-time task based scheduling system as one of the structures of the intelligent scheduling system in FMS, together with mechanisms within the structure.

Namely, Chapter 2 proposes a transient based real-time scheduling system as a structure of the intelligent scheduling system in FMS, as well as algorithms within the structure. The transient based real-time scheduling system selects a dispatching rule

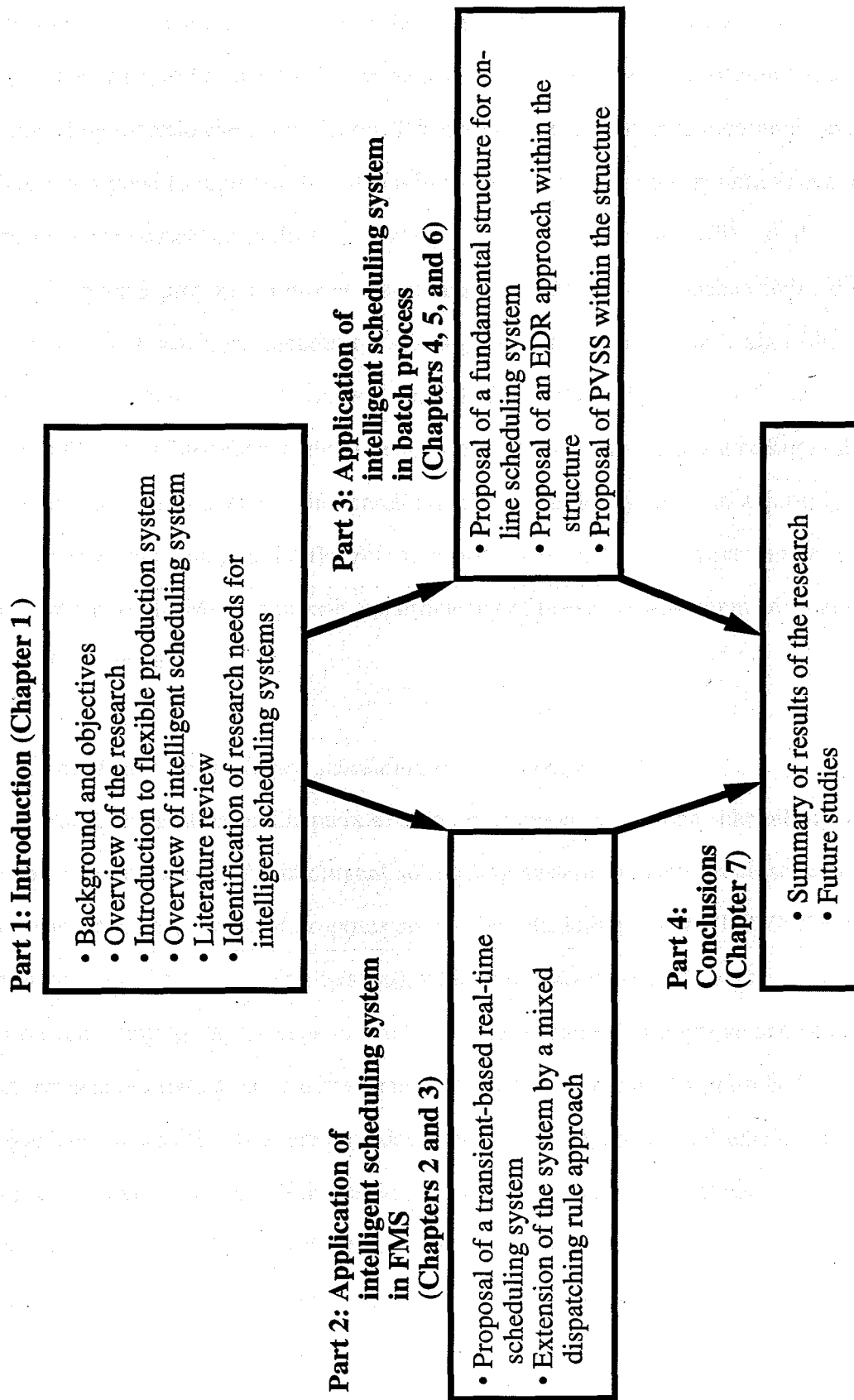


Figure 1.1 Overview of the research

dynamically for the next short period to respond to the changes of system status. The system consists of two modules. The first module defines the scheduling interval associated with the dispatching rule, to be selected in the second module, based on the system transient state predicted by discrete event simulation. The system transient state is measured based on an INDEX designed to represent the load on the FMS. The proposed system is evaluated by simulation experiments, and the statistical significance of the results is discussed.

Chapter 3 proposes a mixed dispatching rule (MDR) approach which extends the transient based real-time scheduling system, together with a search algorithm for the approach. The MDR approach can assign a different dispatching rule for each machine in an FMS, although ordinal dispatching rule approaches assign a single dispatching rule for all machines in a system. The MDR search algorithm efficiently finds an appropriate mixed dispatching rule using a bottleneck approach with discrete event simulation. The effectiveness of the MDR approach and efficiency of the search algorithm are demonstrated on an FMS model.

Part 3: Application of intelligent scheduling system in batch process

Part 3, consisting of Chapters 4, 5 and 6, provides an on-line scheduling system as one of the structures of the intelligent scheduling system in batch processes in chemical industries. Namely, chapter 4 proposes an on-line scheduling system (FOSS: Fundamental structure of On-line Scheduling System), which modifies a schedule within a certain short period according to the changes of production information, to improve and maintain the performance of production, as a fundamental structure for the intelligent scheduling system in batch processes. The structure provides a fundamental approach and scheme to realize an on-line scheduling system within the framework for the intelligent scheduling system. The structure defines schedule modification timing and a modification space within each timing according to the changes of production information, and dynamically modifies a schedule to minimize the effect of the changes. For each schedule modification, the structure selects an appropriate modification strategy, such as complete re-scheduling, resource re-assignment, or time shift of each task, depending on the disruptions or the performance degradations on a

schedule predicted by a simulation mechanism. Therefore the structure requires several schedule modification mechanisms to conduct schedule modifications for different modification spaces and strategies.

Chapters 5 and 6 propose the EDR (Extended Dispatching Rule) approach and PVSS (Process-Variability-based on-line Scheduling System) as schedule modification strategies within FOSS. The EDR approach executes a complete re-scheduling efficiently by using a dispatching-simulation mechanism. PVSS, on the other hand, modifies task start and completion times to improve the performance of production under the waiting time constraints between tasks specified in a recipe under processing time variations.

The effectiveness of the proposed structure (FOSS) for the intelligent scheduling system in batch processes as well as the EDR approach and PVSS within the structure are demonstrated in each chapter.

Part 4: Conclusions

Chapter 7 summarizes the results of this research, as well as promising areas for further studies.

1.4 Flexible Production System

In research and practice, a large number of flexible production systems have been proposed in many ways. Among them, typical flexible production systems widely implemented in practice so far can be summarized into FMS and batch processes in chemical industries.

1.4.1 FMS (*Flexible Manufacturing System*)

FMS is a highly integrated and computer controlled production system which provides mid-to-low volume discrete part production, such as machine parts production. The system, as shown in Figure 1.2, usually consists of load/unload stations, numerically controlled machining centers, parts storage facilities used for temporary storage for WIP (Work In

Process), and automated transportation systems for parts and tools, together with the control system to integrate the facilities. FMS has been of major interest to manufacturers, because of its advantages, as follows:

- Shortened lead time, reduced WIP, and improved market response,
- High product quality,
- High equipment utilization.

From the scheduling point of view, FMS has the following characteristics.

Characteristics of FMS:

(1) Requirements of real-time decision

FMS is a highly integrated production system involving systems which require real-time decisions such as tool management, automated transportation control like AGV (Automated Guided Vehicle), and dispatching control of parts movement, because of its dynamic and stochastic nature.

(2) Flexible routing

FMS usually has alternative processors to conduct a task. Therefore the routing to produce a product can be defined flexibly according to the system states. (In this research, a set of serial operations within a certain machine or process unit is called a task.)

(3) Unlimited waiting times between tasks

In general, products produced in FMS such as machine parts can virtually have unlimited waiting times between tasks. In addition, FMS usually have buffer spaces between tasks for WIP to increase utilization of production facilities.

(4) Task based scheduling

In a discrete part production system like an FMS, a task which indicates a set of serial operations within a certain machine can be seen as an unit for scheduling because the

constraints between tasks are usually precedence relations only and unlimited waiting times between tasks are possible. Therefore scheduling systems for FMS can specify task schedules dynamically on a real-time basis during a schedule execution.

1.4.2 Batch Process

Batch process is widely used in chemical industries for producing mid-to-low volume chemical related products, such as pharmaceutical bulk actives, paints, lube oil, and so on. Batch processes usually consist of reactors, dryer, filters, and vessels. For example, Figure 1.3 shows an equipment configuration of a batch process which produces approximately 3 different kinds of pharmaceutical bulk actives simultaneously. In addition, new concept batch processes which realize flexible connection and pipeless materials transfer, and therefore increased flexibility, are emerging in the last decade (Ishii, 1994).

From the scheduling point of view, batch processes have the following characteristics.

Characteristics of batch process:

(1) Waiting time constraints between tasks

Waiting time constraints to proceed to the next task constitute a critical factor in batch process scheduling especially from the quality control point of view because of the instability of intermediate products. Violation of the constraints will cause increased by-products, low product quality, and low production volume, therefore an undesirable schedule. Three types of waiting time constraints can be seen in an ordinary batch process, as follows (Ku *et al.* 1987);

- ZW (Zero Wait) where an intermediate product must be immediately processed in the succeeding task,
- FW (Finite Wait) where an intermediate product is allowed to be held for a specified period,
- UW (Unlimited Wait) where an intermediate product is allowed to be held unconditionally.

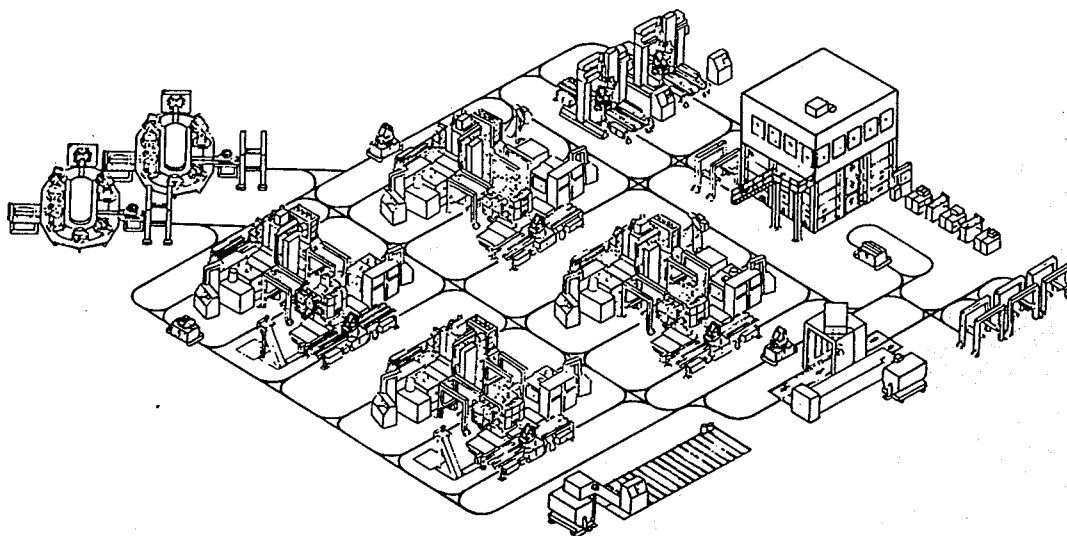


Figure 1.2 An example of FMS (Talavage and Hannam, 1988)

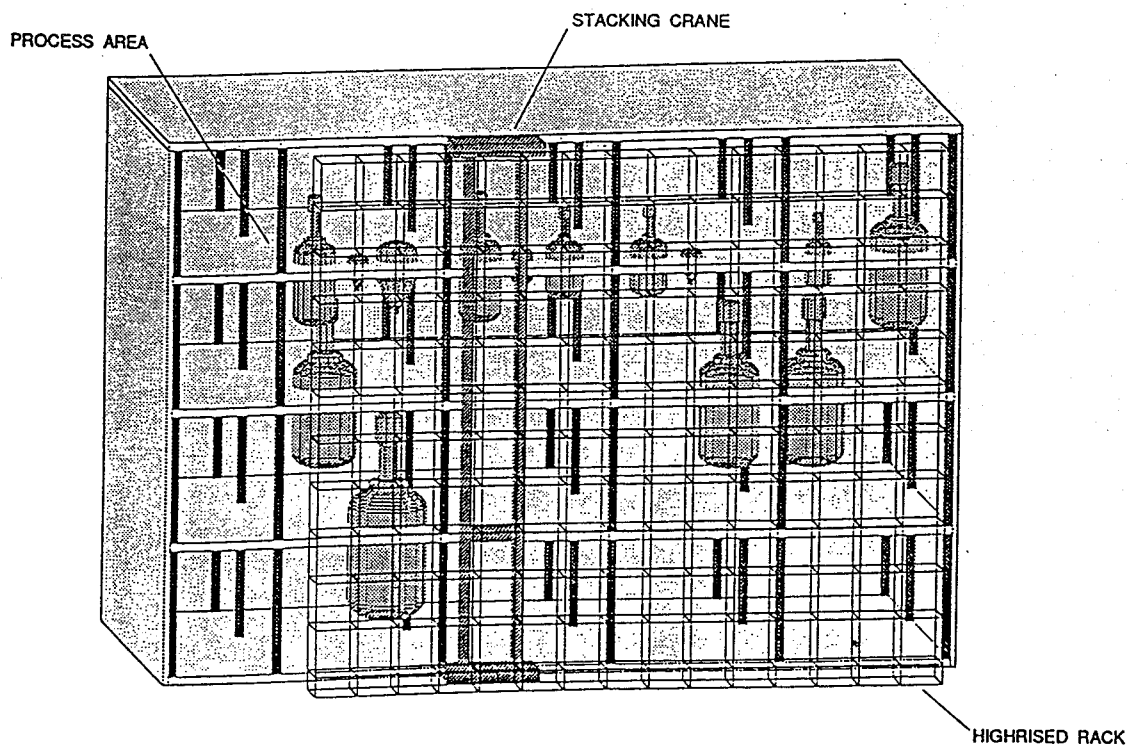


Figure 1.3 An example of batch process in chemical industry
(Overview of SHUTTLENET by JGC Corporation (Ishii, 1994))

(2) Processing time variability

Processing times in batch processes usually vary even in normal operation due to fluctuations of quality of raw materials or catalyst, fouling of equipment during production, and fluctuation of cleanness of equipment. (Cott and Macchietto (1989b) report that variations of 5% of the processing time and upwards are typical.) Processing time variations will cause not only low product quality level because of the violation of waiting time constraints between tasks but also low productivity because of increased idle time.

(3) Product interdependency

In batch processes, producing several intermediate products and final products using the same facility is a normal practice. In this case, the product interdependency among intermediate products and final products arises as a constraint in generating a feasible schedule.

(4) Limited process configuration and buffer capacity

Batch processes usually employ fixed piping networks to transfer materials between process equipment. In addition, only limited intermediate storage tanks are usually available because of the high investment cost for the facilities and the instability of intermediate products. Therefore the degree of schedule modification, such as changes of the process equipment usage, and the production sequence, is limited after starting production.

(5) Product based scheduling

In batch processes, the schedules of each task are strongly connected with each other. For example, prediction of waiting times of all tasks for a product is necessary at the beginning of the production to generate a schedule in consideration of the waiting time constraint between tasks under processing time variations. Therefore task based scheduling as in FMS is not applicable in batch processes because of its nature of real-

time decisions on a task basis. In batch process scheduling, product based scheduling which generates schedules on a product unit basis is required

1.5 Overview of Intelligent Scheduling System

1.5.1 Fundamental Scope of Intelligent Scheduling System

Figure 1.4 shows the basic system boundaries between the intelligent scheduling system and the systems relating to production management. The corporate system provides production planning information and performance criteria for scheduling to the intelligent scheduling system at certain time intervals, for example, every week, except for urgent situations. The process control system provides on-line process data regarding the current situation of production systems. Based on the production planning information, master data for scheduling, and the on-line process data, the intelligent scheduling system generates an instruction which best suits performance criteria over the scheduling period, and then informs the process control system of the instruction. The process control system executes production according to the instruction in a scheduling period. The corporate system also refers to the instruction as feedback information from the intelligent scheduling system, and eventually updates production planning information or performance criteria for scheduling, if the instruction provided is unacceptable according to the corporate policy.

Figure 1.5 describes the classification of production information associated with the intelligent scheduling system, in detail. The information in Figure 1.5 has quite different degrees of variability. Master process data, for example, are usually unchanged at least during a scheduling period. On the other hand, on-line process data are dynamically updated during production. The intelligent scheduling system takes corrective actions toward the current instruction to the process control system during production to adapt to changes of the widely variant information shown in Figure 1.5, and then improves or maintains the performance of production systems according to the criteria given by the corporate system.

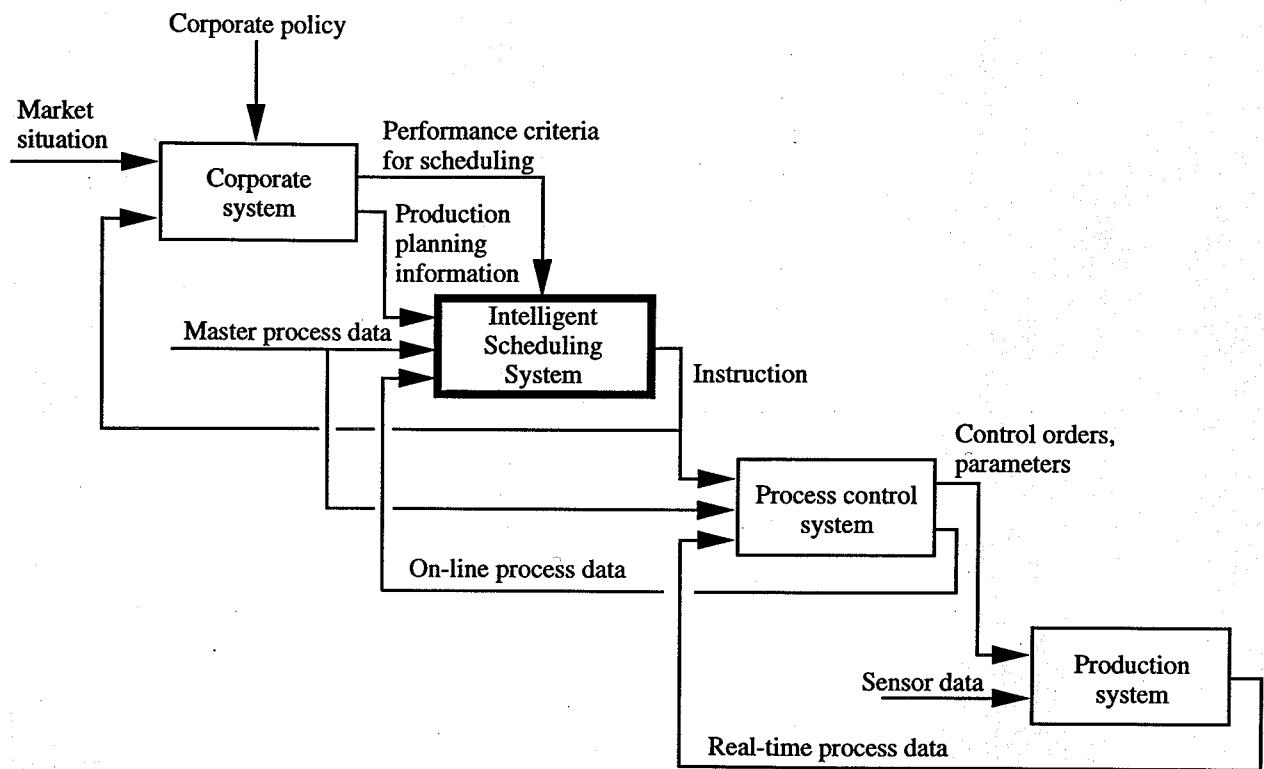


Figure 1.4 Boundaries of intelligent scheduling system within production management

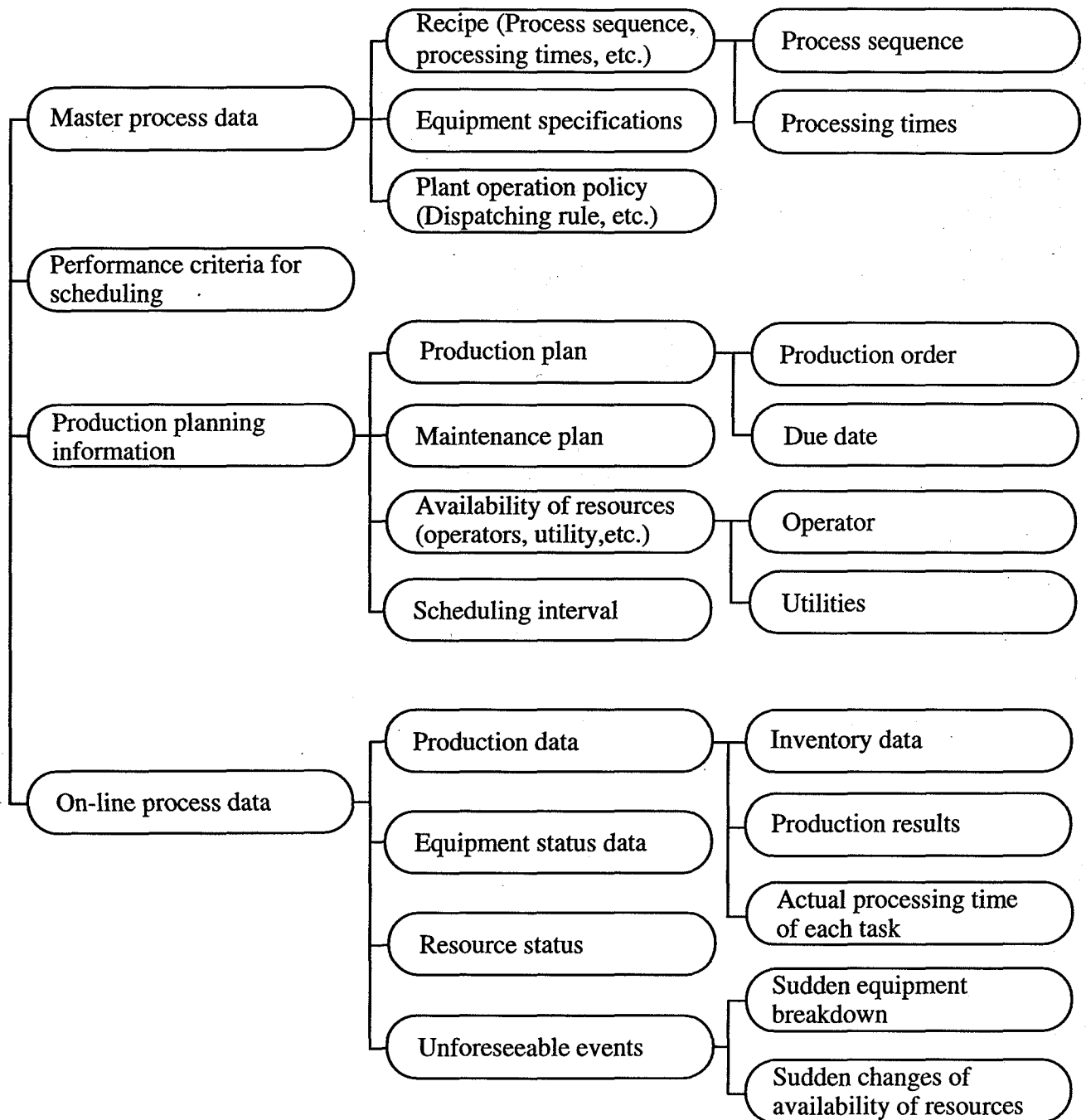


Figure 1.5 A classification of production information associated with the intelligent scheduling system

1.5.2 A Framework for Intelligent Scheduling System

The intelligent scheduling system reacts to changes of production information, and revises the instructions to the process control system to improve or maintain the performance of production systems according to the performance criteria provided by the corporate system. The detailed features of the intelligent scheduling system could be different in each application area. However, any intelligent scheduling system is established based on the framework shown in Figure 1.6 which consists of the following core functions to realize the system.

Core Functions:

- Monitoring of production information
- Defining the timing for instruction revision
- Instruction revision

(1) Monitoring of production information

The intelligent scheduling system requires the functions for data acquisition and monitoring of the changes of production information, such as production planning information, on-line process data. These are essential functions in the intelligent scheduling system to react to dynamic changes of the production information.

These functions can be realized using today's information technologies. For example, DCS (Distributed Control System) which can acquire and monitor on-line process data via information networks is provided by a large number of control system vendors (Procyk, 1991).

(2) Defining the timing for instruction revision

The intelligent scheduling system revises the instruction to the process control system during production according to the changes of production information. Therefore it requires the function to specify suitable timing for instruction revision. There are several ways to define the timing, as follows;

- By constant interval,

- By events, such as change of system status, change of production demand,
- By combination of constant interval and events,
- By some index showing the degree of the change of production information.

In any case, defining suitable timing for the instruction revision is a critical issue in the intelligent scheduling system. Too short an interval for the revision leads to unnecessary schedule revisions, and therefore causes instability of production systems. It also requires huge computational resources to generate the instructions rapidly. Too long an interval for the revision, on the other hand, could fail to identify the most suitable timing of instruction revisions which are necessary to improve or maintain the performance of production systems, and therefore reduce the performance of production systems.

The way to define the most suitable timing highly depends on the characteristics of each production system, the type of instruction to the process control system, and the dynamic behavior of production information. Therefore each intelligent scheduling system requires its own way to define the timing for the instruction revisions according to the characteristics of each application area.

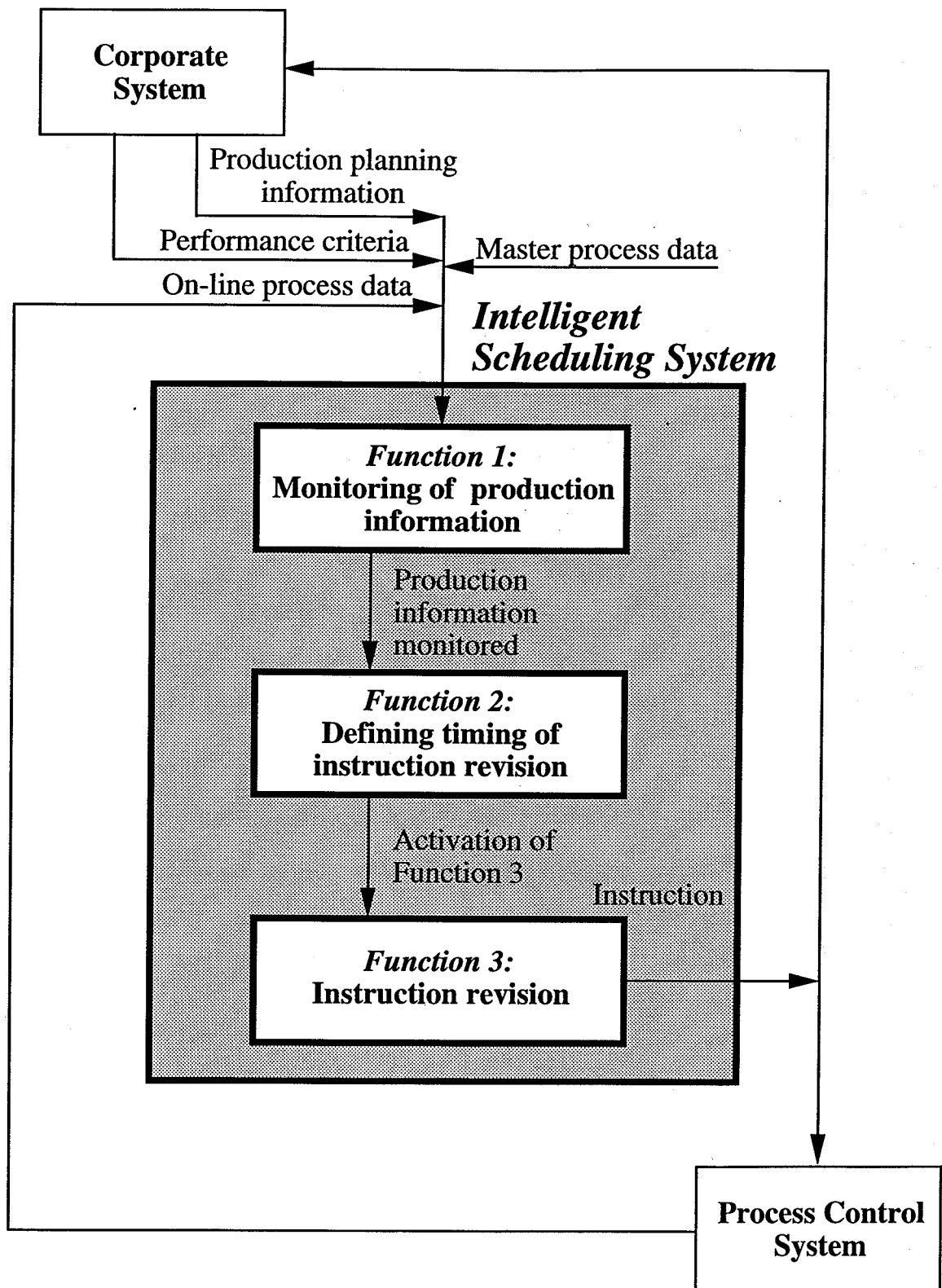


Figure 1.6 A framework for an intelligent scheduling system

(3) Instruction revision

The intelligent scheduling system generates instructions for the process control system according to the changes of production information, to improve or maintain the performance of production systems. The instruction to the process control system could be at one of several levels depending on various factors, such as the characteristics of each production system, and the degree of changes of production information associated with the production system considered. There are, however, at least two levels of instruction as follows;

Instruction level:

- Level 1: Instruction of operating rules, such as dispatching rules,
- Level 2: Instruction of a complete schedule.

In level 1, the intelligent scheduling system specifies operating rules as instructions to the process control system at every timing interval for instruction revision. The process control system dynamically decides process operations, such as production sequence of each product, start time of production at each equipment unit, and so on, on a real-time basis according to the operation rules specified by the intelligent scheduling system. In this instruction level, a scheduling problem can be decomposed into several small problems, and resolved in distributed systems. For example, the part dispatching, which selects the next part to be loaded, is decided at each machine based on the rules. In this case, no global optimization according to the performance criteria for a production system can be guaranteed because the performance is determined as a result of a series of local decisions. However, the intelligent scheduling system in this form requires less computational resources than centralized scheduling which solves complete schedules with a centralized system at every short period. In addition, the level 1 instruction can be adaptable to a wide variety of production systems under practical constraints.

In level 2, on the other hand, the intelligent scheduling system specifies a complete schedule which defines the production sequence, resource usage, and start and completion times of each task as an instruction to the process control system at every timing interval of

instruction revision. The process control system controls the production system so that the production system executes the production based on the schedule specified by the intelligent scheduling system as the instructions. The instructions in level 2 attempt to coordinate the operations in a production system in advance. Therefore this is useful for production systems having constraints that real-time decisions cannot satisfy, such as waiting time constraints between tasks in batch processes. To implement level 2 instructions in the intelligent scheduling system, however, the system requires high performance mechanisms which can modify the current working schedule on a real-time basis according to the latest production information.

There could be several intermediate levels between the level 1 instruction and the level 2 instruction. For example, an instruction which specifies a production sequence together with operation rules to define equipment usage on a real-time basis would be effective in some cases. Therefore the instruction level in the intelligent scheduling system would be defined based on the characteristics of each application.

1.5.3 Structures of Intelligent Scheduling System

The framework described in the above is common in any intelligent scheduling system. The detailed structures within the framework, however, could take several forms depending on the characteristics of each production system, such as production process, equipment used, physical properties of materials, and the dynamic nature of production information such as frequency of demand changes, equipment breakdown, and so on. (In this thesis, a structure is defined as a fundamental approach and a scheme to realize the intelligent scheduling system.) For example, the flexible production systems introduced in the previous section could form different structures within the framework of the intelligent scheduling system as follows.

(a) Structure for FMS (Flexible Manufacturing System)

FMS usually produces a large number of products simultaneously. In addition, it consists of machines, such as AGVs and cranes, where detailed scheduling is virtually

impossible in advance. Processing time variations, especially manual operations like parts set-up, are also associated with FMS. Therefore a real-time task based scheduling system which generates the instruction of suitable operation rules for the process control system at short intervals based on the latest production information is promising as a structure of the intelligent scheduling system in FMS. In this case, the instructions would be operation rules and the timing of instruction revision would not be event-driven because of the large number of events occurring in the system. As stated in the above, too short an interval of instruction revision causes unnecessary schedule revisions, and therefore revisions at every event would fail.

(b) Structure for batch processes in chemical industries

Batch processes usually require the product based scheduling which generates schedules on a product unit basis because of the characteristics of batch processes, such as waiting time constraints between tasks, associated with physical properties handled in batch processes. Therefore the on-line scheduling system which generates the instruction of a complete schedule at short intervals according to the latest production information is promising as a structure for the intelligent scheduling system in batch processes. In this case, the instruction would be a complete schedule and the timing of the instruction revision would be specified based on events in production information, because the detailed schedule, such as the task start times, could be changed for every event in batch processes.

As described in the above, the detailed structure within the framework for the intelligent scheduling system could be different in each application area. Therefore this research identifies FMS and batch processes as typical flexible production systems currently employed in practice, and proposes structures and mechanisms for the structures to establish intelligent scheduling systems for management of the systems. The following chapters present intelligent scheduling systems in FMS and batch processes, respectively, based on the characteristics associated with each system.

1.6 Literature Review

Conventional scheduling systems, which assume static or off-line scheduling, have been an extensive and intensive research area over the past several decades. Comprehensive reviews in this area are provided by Blackstone *et al.* (1982), Ku *et al.* (1987), Musier and Evans (1990), Reklaitis (1991), Rippin (1993), Blazewicz *et al.* (1994), Basnet and Mize (1994), and so on.

Scheduling approaches proposed so far in conventional systems can broadly be categorized as follows;

- Analytical approach,
- Exact mathematical approach,
- Heuristic mathematical approach,
- Simulation approach,
- Artificial Intelligence (AI) approach.

The analytical approach, for example Johnson's rule (Johnson, 1954), provides an exact algorithm to obtain optimum solution for a specified scheduling problem. The exact mathematical approach, such as Kondili *et al.* (1993), Fuchino *et al.* (1992), and so on, formulates a scheduling problem and solves the problem by mathematical programming, such as the integer programming technique, to obtain an optimum solution. The heuristic mathematical approach tries to solve the mathematical problem only approximately. For example, the local search approach (Kuriyan and Reklaitis, 1989) is widely employed in literature. In addition, modern heuristic search approaches which employ search techniques such as simulated annealing, generic algorithm, tabu search, and beam search, have been studied over the last decade (Morton and Pentico, 1993). The simulation approach (Kiran and Smith, 1984) usually generates a schedule with a process model and a simple dispatching rule. A dispatching rule is a rule to select the next part to be served from a set of parts awaiting processing or transportation. This approach is widely employed in research

and practice because of its flexibility. The AI approach in scheduling has a wide variety of forms. (Prosser and Buchanan (1994) provide a comprehensive review paper on AI based scheduling.) The typical AI approach using an expert system (Pedersen, 1989) attempts to enable complex dispatching scheduling with a realistic process model, which is impossible to realize through simple dispatching scheduling by simulation technique only. The AI technique is also used in combination with mathematical programming, simulation, and so on.

Research in the intelligent scheduling area, which assumes dynamic changes of production information, has been stimulated in the last decade. Comprehensive reviews in this area are provided by Reklaitis (1991), Harmonosky and Robohn (1991), and so on.

The intelligent scheduling system is more complex and difficult than the conventional scheduling system because it must satisfy the following requirements.

- On-line schedule modifications during production according to the changes of production information.
- High flexibility able to consider practical constraints, such as processing time variations, changes of demand and resource availability.
- Scheduling on a real-time basis.

Scheduling problems, with a few exceptions, are classified in the NP-class problem from the computational complexity point of view (King and Spachis, 1980). It is, therefore, virtually impossible to solve a practical size problem by either the analytical approach, the exact mathematical approach, or the heuristic approach in conventional scheduling systems. In addition, conventional scheduling approaches ignore mechanisms for schedule modifications during production according to the dynamic changes of production information. For example, the simulation approach and the AI approach in conventional scheduling systems usually employ fixed rules to generate a schedule over a scheduling period, although the fixed rules cannot always perform effectively under the dynamic changes of production information. Therefore the fruits of research on static or off-line scheduling in conventional scheduling systems cannot apply directly to the intelligent

scheduling system.

Approaches in the intelligent scheduling system for the discrete part production system including FMS usually apply a simulation based approach or AI approach because of their flexibility. For example, Thesen and Lei (1986) propose an expert system approach for a real-time robot scheduling problem. Wu and Wysk (1988) provide a structure and mechanisms of a real-time scheduling system for FMS based on the expert system and discrete event simulation. Cho and Wysk (1993) apply neural network technology (Rumelhart *et al.*, 1986) in real-time FMS scheduling. In addition, Hatono, *et al.* (1995) propose a distributed real-time scheduling system in FMS, which manages a production system through a series of local decisions by each machine and AGV in an FMS. The local decisions are made on a real-time basis applying several heuristic rules.

In any case, previous research shows that the real-time task based scheduling system which defines the next short period and dispatching rule for the period as an instruction to the process control system is promising as a structure for the intelligent scheduling system in FMS because of its capability for real-time based decisions and high suitability for the characteristics of FMS. Within the structure, a real-time scheduling system defines a certain length scheduling period as the timing for an instruction revision, and specifies a dispatching rule as an instruction to the process control system for the period. As a result, a real-time scheduling system continually alters different dispatching rules, and thus yields a schedule which performs well under the dynamic changes of production information. Within the structure, however, at least the following issues must be studied extensively to establish a sophisticated intelligent scheduling system in FMS.

Research needs for an intelligent scheduling system in FMS:

- Mechanism to define the most suitable short period to provide a dispatching rule.
- Mechanism to identify the best dispatching rule for the next short period according to the latest production information.

The first issue, corresponding to the Function 2 within the framework for the

intelligent scheduling system shown in Figure 1.6, strongly affects the performance of the system. For example, a constant length interval would fail to grasp big system changes which could affect the system performance significantly. However, no mechanism to define the most suitable scheduling period has been proposed so far. The second issue, corresponding to the Function 3 within the framework, arises because of the short time window to evaluate dispatching rules. This could cause selection of an inadequate dispatching rule through the problem of censored data (Blackstone *et al.*, 1982). (A number of parts remaining in a system within a short scheduling period affect different dispatching rules in an unequal way, and therefore an inadequate dispatching rule could be selected.) Therefore mechanisms to reduce the problem of censored data are very necessary for the Function 3 within the framework for the intelligent scheduling system.

In batch processes, for example, Egli and Rippin (1986) state the importance of flexible and dynamic scheduling procedure in the face of changing conditions, and then develop SRSBP (Short Range Scheduling for Batch Plants) which can restart scheduling from any intermediate state to handle changes of production information. Cott and Macchietto (1989a) propose a computer-based system which integrates planning and plant control of batch processes. In the system, they propose an on-line scheduling and control mechanism consisting of POMA (Projected Operation Modification Algorithm) (Cott and Macchietto, 1989b) and a scheduling algorithm. POMA is a simple time shift algorithm to modify a schedule in response to the processing time variations. Terpstra *et al.* (1994) propose a prototype reactive scheduler, intended to handle a wide variety of uncertainties in production information, for mixed-batch/continuous plants with a real-time expert system. In addition, scheduling systems which assume the processing time variations have been proposed, such as, Onogi *et al.* (1986), and Cott and Macchietto (1989b), because the processing time variations are the most critical in batch processes among the variations of production information.

In any case, previous works show that on-line scheduling which instructs a complete schedule to the process control system at certain short intervals is effective as a structure for the intelligent scheduling system in batch processes. The on-line scheduling system defines

the timing of an instruction revision by the events in production information, and modifies the current working schedule. The modified schedule is provided to the process control system as new instruction. However, the detailed structure of the on-line scheduling system for batch processes, mainly associated with the Function 3 in the framework shown in Figure 1.6, have not been well investigated yet. Therefore, at least the following issues must be studied extensively to establish an intelligent scheduling system for batch processes.

Research needs for an intelligent scheduling system in batch processes:

- Structure for on-line scheduling system in batch processes in the framework for the intelligent scheduling system.
- Schedule modification mechanisms used within the structure.

One of the most primitive structures for the on-line scheduling system is to adapt complete re-scheduling based on the latest production information at regular short term intervals. It is obvious, however, this is infeasible in practice because of its cost and the computational time required. A structure, which provides an appropriate scheduling modification space and modification strategy at appropriate timing according to the disruptions and performance degradations of the current schedule due to the changes of production information, should be established in the on-line scheduling system to minimize the computational effort for schedule modifications. Such a structure requires different schedule modification mechanisms for different modification spaces and modification strategies. In addition, the mechanisms must execute a schedule modification on a real-time basis. Therefore schedule modification mechanisms within the structure should also be developed to establish the on-line scheduling system.

1.7 Chapter Summary

This chapter presents the background and objectives of this research by clarifying the importance of the intelligent scheduling system for management of flexible production systems under the frequent changes of production information inherent in today's dynamic business environment.

Because there are a large number of flexible production systems in research and practice, this chapter identifies FMS and batch processes in chemical industries as typical flexible production systems currently employed in practice for, respectively, discrete part production and process type production, and then investigates the characteristics of the systems from the scheduling point of view.

Followed by the investigation, this chapter presents the fundamental scope and a generic framework for the intelligent scheduling system. In addition, promising structures and research needs for the intelligent scheduling system in FMS and batch processes, respectively, are presented as summarized in Table 1.1, based on the characteristics of each production system and the results of literature review on scheduling systems proposed over the past several decades.

Table 1.1 Summary of promising structures and research needs for the intelligent scheduling system

	FMS	Batch process
<i>Promising structure</i>		
Approach	Real-time scheduling	On-line scheduling
Instruction to process control system	Dispatching rules	Complete schedule
Timing of instruction revision	By short intervals	By event in production information
<i>Research needs</i>	Mechanism to define the most suitable short period to provide dispatching rules. Mechanism to search the best dispatching rule for the next short period according to the latest production information.	Structure for on-line scheduling system in batch processes. Schedule modification mechanisms to be used within the structure.

PART 2:

APPLICATION OF INTELLIGENT SCHEDULING SYSTEM IN FMS

Chapter 2

A TRANSIENT BASED REAL-TIME SCHEDULING SYSTEM IN FMS

2.1 Chapter Overview

FMS is a highly integrated and computer controlled flexible manufacturing system which provides mid-to-low volume discrete part production, such as machine parts production. Because of its dynamic behavior and the frequent changes of production information due to today's business environment, establishment of an intelligent scheduling system, which effectively manages flexible production systems under the frequent changes of production information, is essential in FMS.

As mentioned in the previous chapter, a real-time scheduling system, which generates a suitable dispatching rule for the process control system at short intervals, is promising as a structure of the intelligent scheduling system in FMS. However, there are several considerations in improving the performance of the system, such as following;

- Mechanism to define the most suitable short period to provide a dispatching rule,
- Mechanism to search the best dispatching rule for the next short period according to the latest production information.

In this chapter, a transient based real-time scheduling system is proposed as one of the structures of an intelligent scheduling system in FMS, together with the above mechanisms within the structure. The transient based real-time scheduling system defines a short term

scheduling interval based on a system transient state. The change in system state which denotes the beginning of a transient state is predicted by discrete event simulation. After that, the system selects a dispatching rule as an operating rule for that scheduling interval, among candidate dispatching rules, again using discrete event simulation of the FMS.

This chapter organized as follows: Section 2.2 provides an overall introduction to the real-time scheduling system, together with literature review on this subject. Section 2.3 presents the structure of the transient based real-time scheduling system and algorithms used in the structure in detail. Section 2.4 evaluates the performance of the transient based real-time scheduling system with an FMS model.

2.2 Problem Description and Background

The relations between a real-time scheduling system and other management systems in FMS are shown in Figure 2.1. The major information given from outside systems is as follows :

- Production planning information (part release schedule, due dates, etc.).
- Master data (process sequence, processing time, etc.).
- Performance criteria for scheduling.
- Candidate dispatching rules.
- On-line process data.

Based on the above data, the real-time scheduling system selects a dispatching rule which makes selected performance criteria best in the next short period. (This short period is termed "a scheduling interval".) A dispatching rule is a rule used to select the next part to be serviced from a set of parts awaiting processing or transportation.

Many researchers have been studying dispatching rules since the 1960s. As a result, more than one hundred dispatching rules, such as shortest processing time (SPT), minimum slack time (SLACK), first in first out (FIFO), have been developed.

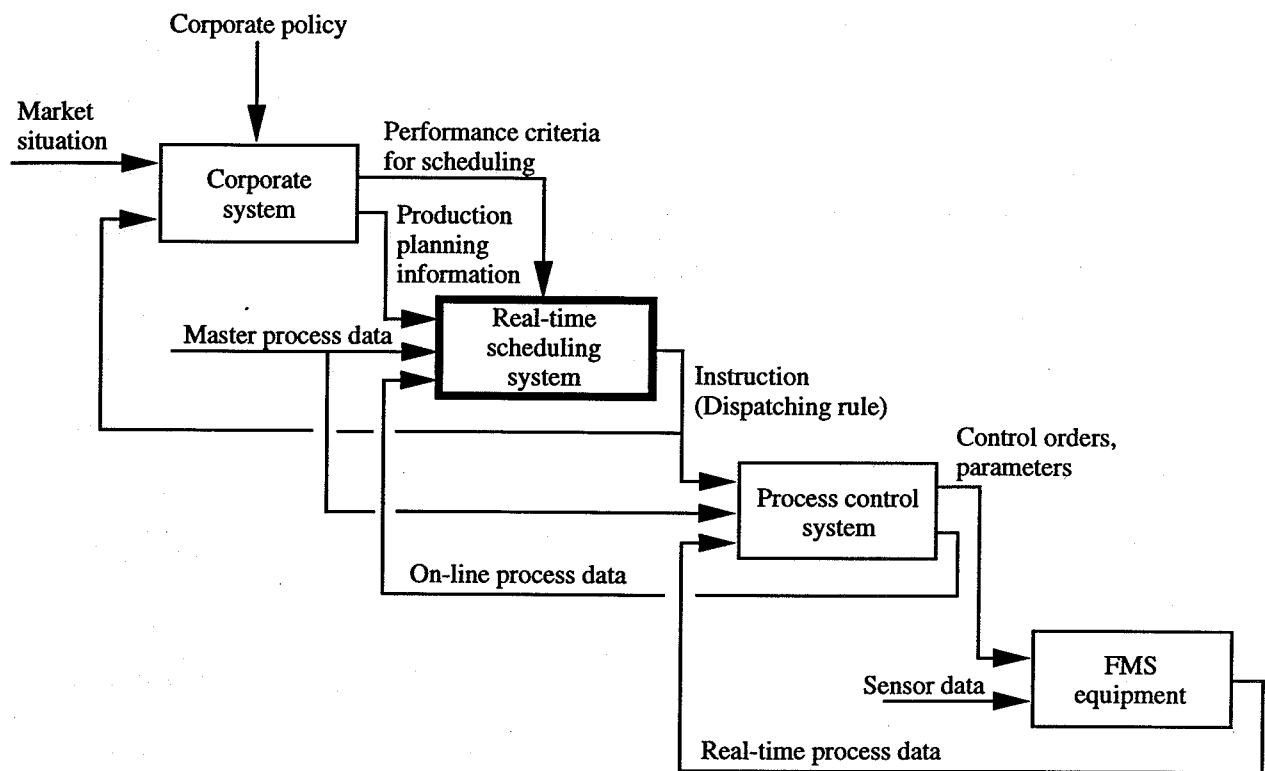


Figure 2.1 Boundaries of real-time scheduling system

However, Blackstone *et al.* (1982) have stated that "No dispatching rule has been shown to consistently produce better results than all other rules under a variety of shop configuration and operating conditions. It is impossible to identify any single rule as the best in all circumstances."

Although a single dispatching rule will not perform the best for a long period, some rules perform better than others under certain conditions. Therefore changing a dispatching rule over successive short periods based on current system state could make performance better than using a single dispatching rule for a long time.

To evaluate the performance of dispatching rules under a specified system state, almost all studies employ discrete event simulation due to the difficulty and inflexibility of analytical methods.

Kiran and Smith (1984) describe the appropriate conditions for the study of dispatching rules by simulation. For example, they found that machine utilizations in many studies range from 0.85 to 0.95 in order to avoid transient conditions so as to get a suitable queue length for the study of dispatching rules. However, results obtained from such studies would not be applicable for a highly dynamic manufacturing system like FMS, which would not obtain steady state with multiple order launching during each shift.

Thesen and Lei (1986) propose an expert system approach for a real-time robot scheduling problem. In their research, they point out that a scheduling system where an expert changes dispatching rules at appropriate points in time performs better than a scheduling system where a single rule is used at all times. In their system, several heuristic rules are evaluated in advance under different system states by simulation. Based on the simulation results, an expert scheduling system is developed. It performs significantly better than the traditional scheduling system which uses a single dispatching rule for the entire manufacturing period. However, the performance of their scheduling system is limited to their selection of initial system states.

In FMS, different part types enter the system so often and at varied times that the system state considered by most studies would not be obtained. To evaluate the performance of dispatching rules for all possible system states is almost impossible. For this reason, a

system which selects a dispatching rule at the beginning of each short period based on the current system state is proposed in this chapter.

Although the concept of real-time scheduling system has the potential to improve the performance of FMS beyond that of using a single dispatching rule all the time, there are two difficulties with such a system:

- The definition of the next scheduling interval length.
- Evaluation of the performance of candidate dispatching rules for the next scheduling interval.

For this problem, Wu and Wysk (1989) propose a multi-pass scheduling system. In their system, a dispatching rule is selected in each constant short period by discrete event simulation. In the long run, this process results in a combination of different dispatching rules. They conclude that the multi-pass scheduling system performs better than the single-pass scheduling system which uses a single dispatching rule for the entire manufacturing period. In their paper, they indicate that the length of scheduling interval is a significant factor for the performance of multi-pass scheduling system. Although they discuss this problem from several aspects, no general procedure to define the scheduling interval has been proposed. They use the multiple (i.e., one, two, three times) of average total processing time and find that triple the average total processing time is the best scheduling interval length in their experiments with an FMS model.

A constant length scheduling interval can not follow the changes of production information, hence it cannot always make performance better particularly in a dynamic system like FMS. Therefore this research proposes the interval length which is defined dynamically based on system state, as shown later.

In addition, Wu and Wysk define the simulation window, i.e., the time window to simulate a model and evaluate the performance of candidate dispatching rules, as equal to the scheduling interval in their system.

The simulation window defined as such, however, will result in the problem of censored data. (Blackstone *et al.* (1982) discuss the problem of censored data in detail.) A

number of parts remaining in a system at the end of each simulation run affect different dispatching rules in an unequal way. The evaluation of dispatching rule based on censored data could cause a selection of an inadequate dispatching rule. This problem is more important when the simulation window is a short period as in the real-time scheduling problem. However, no methodology to eliminate the problem of censored data is shown in the multi-pass scheduling system.

In this chapter, a transient based approach to define the next scheduling interval is proposed. This approach adapts the length of the next scheduling interval automatically based on system transient state. For illustration and comparison, Figure 2.2 shows an example of a scheduling interval defined by a transient based approach and by a constant interval approach. As shown in the figure, the transient based approach defines scheduling intervals from valley to valley of the system state. On the other hand, the constant interval approach defines an interval independent from the system state.

The transient based approach assumes that changing the dispatching rule for each transient interval would improve the system performance. In addition, defining a transient interval from valley to valley for a component of the system state such as number of parts in the system could reduce the problem of censored data to some extent, because the parts remaining at the end of scheduling interval is smaller than some peak value earlier in the interval. To precisely identify the transient interval, a function termed INDEX is proposed in Section 2.3.

Three algorithms designed to reduce the problem of censored data on dispatching rule selection are also proposed in this chapter. By reducing censored data, more accurate evaluation of the candidate dispatching rules is expected.

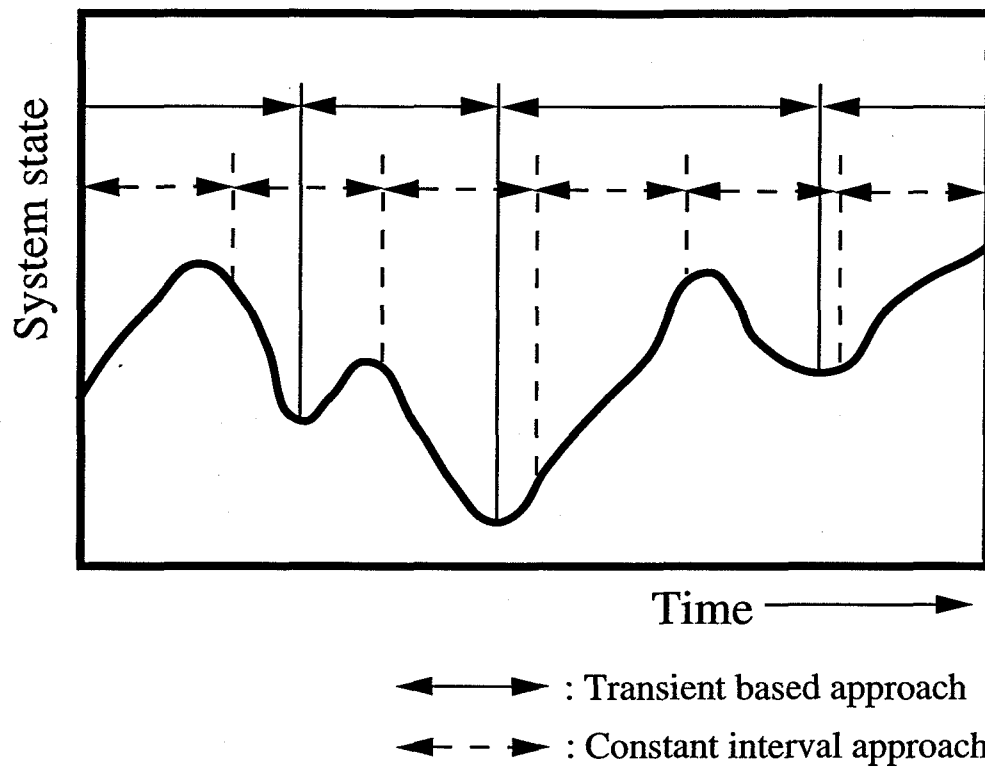


Figure 2.2 Next scheduling interval by transient based approach and constant interval approach

2.3 Transient Based Real-Time Scheduling System

There are three modules in the transient based real-time scheduling system, i.e., transient detection and scheduling interval setting, dispatching rule selection, and simulator, as shown in Figure 2.3.

(a) *The transient detection and scheduling interval setting module*

In this module, the next scheduling interval is defined by analyzing the time series data of a performance measure called INDEX calculated based on data from the simulator. This module corresponds to the Functions 1 and 2 within the framework for the intelligent scheduling system shown in Figure 1.6.

(b) *The dispatching rule selection module*

In this module, each candidate dispatching rule is evaluated from the results of simulation provided by simulator. The selection is based on the dispatching rule which performs best to the performance criteria over the next scheduling interval. This module corresponds to the Function 3 within the framework for the intelligent scheduling system shown in Figure 1.6.

(c) *The simulator module*

The simulator models the FMS and is initialized with:

- On-line process data (machine status, inventory, etc.).
- Part release schedule.
- Part data (due date, sequence route, part ID, etc.).
- Dispatching rule.
- Terminating conditions of the simulation.

On-line process data are assumed to be updated automatically by the process control system. The part release scheduling system would update the part data. The dispatching rule

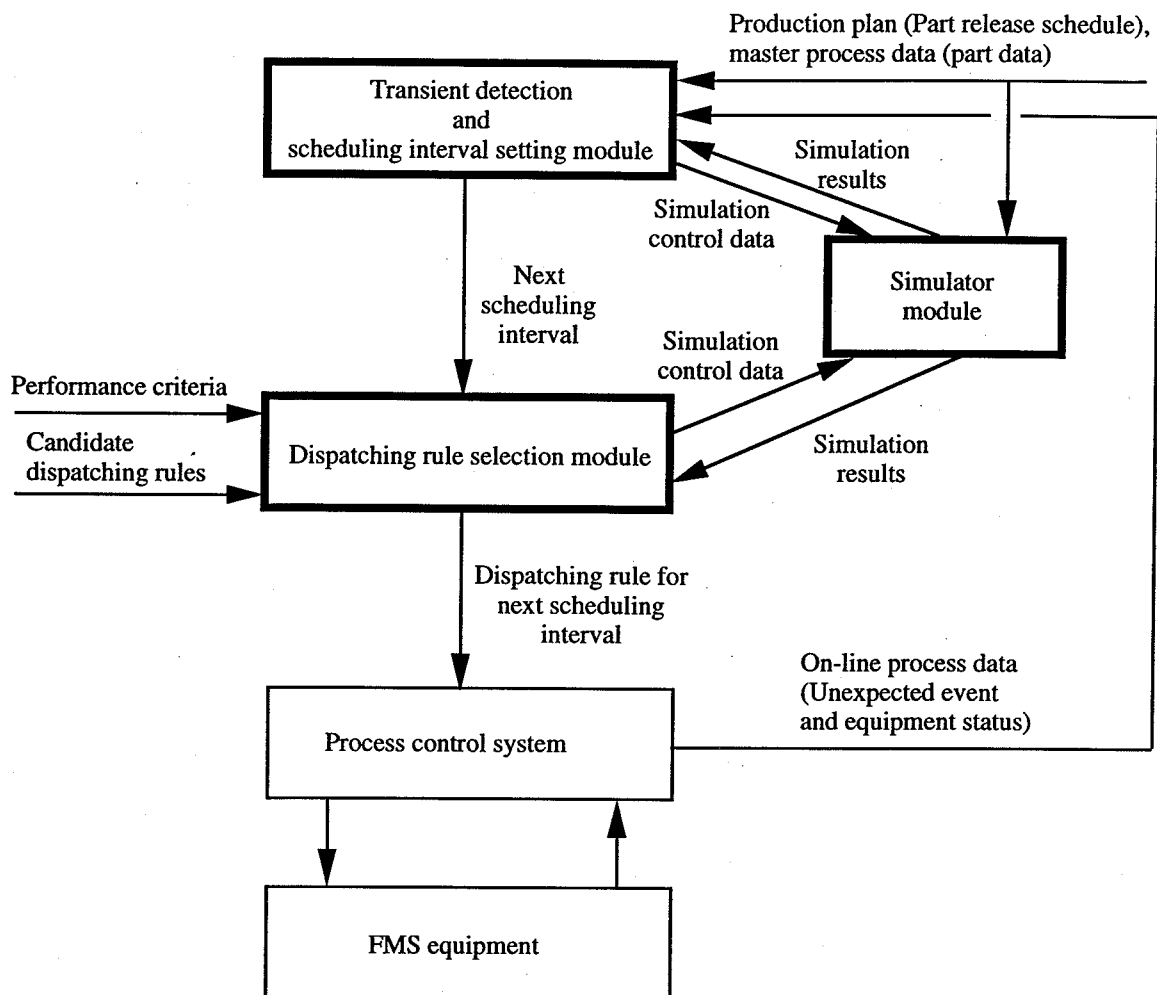


Figure 2.3 An overview of the transient-based real-time scheduling system

and terminating conditions are provided by either the transient detection and scheduling interval setting module or the dispatching rule selection module depending on which module activated the simulator. The terminating conditions of simulation are described in Sections 2.3.1(2) and 2.3.2.

The transient based real-time scheduling system will be activated by several events:

- *Normal activation cycle*
 - At the end of each real-world scheduling interval.
- *Unexpected events* (The events which are not considered in a simulation model.)
 - Machine breakdown.
 - Changes of part release schedule, candidate dispatching rules, or performance criteria.

2.3.1 Transient Detection and Scheduling Interval Setting Module

A next scheduling interval is defined from current time to the beginning of the next "upward" transient state as shown in Figure 2.2. This type of definition may reduce the problem of censored data to some extent because the number of parts remaining in a system at the end of the interval is smaller than the peak value during the interval. The transient state is detected by analyzing the time series data of INDEX which shows system congestion, lateness, flow time, and so on. The following describes in detail the INDEX and the transient detection algorithm, respectively.

(1) *Measuring system state*

A transient system state is determined by means of performance criteria. There are many conventional performance criteria, such as flow time, lateness, and so on. In this research, a criterion called INDEX is employed for measuring system state. The INDEX is defined as follows :

$$\text{INDEX}(t) = (1/n) * \sum_{j=1}^n (PT_j + WT_j + TR_j) / DT_j$$

n : number of parts in system at simulation time t , PT_j : part j 's total processing time, WT_j : part j 's waiting time (predicted by simulator), TR_j : part j 's transportation time (predicted by simulator), DT_j : part j 's due date, t : simulation time at which INDEX is calculated.

The INDEX has several characteristics which are worthy of note :

- Part entries in excess of the system capacity increase the INDEX because of increasing waiting time.
- Urgent parts increase the INDEX because of decreasing due date.
- The INDEX shows the ratio of work load per system capacity at current conditions. An INDEX greater than 1.0 indicates an over capacity condition.

For the INDEX calculation, the simulator provides the predicted waiting time and transportation time of each part that will exist in the system during the scheduling interval.

To predict the end of the transient interval, the simulator always uses the same type of dispatching rule. In this research, First In First Out (FIFO) rule is used for this purpose. FIFO performs substantially the same as random selection in regard to many measurement criteria (Blackstone *et al.* 1982). Since the main purpose of the INDEX is to predict the fluctuations of the system state, FIFO seems to be a relatively unbiased dispatching rule for this purpose.

The time series data of INDEX is calculated in each period defined by the following:

- The time interval to calculate the INDEX
= minimum scheduling interval / sampling number
- Sampling number = 10 to 15

The minimum scheduling interval (MSI) is a time window to detect system changes and is equal to the shortest total processing time of all part types. The following section describes this point in detail.

(2) *The transient detection algorithm*

Transient state is predicted by analyzing the time series data of INDEX. Figure 2.4 shows the transient detection algorithm employed in this research.

In this algorithm, the time series data of INDEX are divided into short time intervals called Minimum Scheduling Interval (MSI) as shown in Figure 2.5. MSI is a time window to calculate the average and mean square of INDEX. (MSI is defined equal to the shortest total processing time of all part types.) System state changes are detected by comparing those data for consecutive MSI.

In Figure 2.4, MSI_j , m_j , and V_j indicate j -th Minimum Scheduling Interval, the average of INDEX in MSI_j , and the mean square of INDEX in MSI_j , respectively.

In this algorithm, system state at MSI_j is defined based on the decision rules described as follows. In these rules, ST_i indicates the system state at i -th MSI interval.

• *The decision rules to define system state*

- (a) If ST_i is a steady state and $m_i = m_j$ then ST_j is a steady state.
- (b) If ST_i is a steady state and $m_i \neq m_j$ and $m_i > m_j$ then ST_j is a downward transient state.
- (c) If ST_i is a steady state and $m_i \neq m_j$ and $m_i < m_j$ then ST_j is an upward transient state.
- (d) If ST_i is an upward transient state and $m_i < m_j$ then ST_j is an upward transient state.
- (e) If ST_i is an upward transient state and $m_i > m_j$ and $m_i = m_j$ then ST_j is a steady state.
- (f) If ST_i is an upward transient state and $m_i > m_j$ and $m_i \neq m_j$ then ST_j is a downward transient state.
- (g) If ST_i is a downward transient state and $m_i > m_j$ then ST_j is a downward transient state.
- (h) If ST_i is a downward transient state and $m_i < m_j$ and $m_i = m_j$ then ST_j is a steady state.
- (i) If ST_i is a downward transient state and $m_i < m_j$ and $m_i \neq m_j$ then ST_j is an upward transient state.

To compare whether m_i is equal to m_j , Welch's method (Snedecor and Cochran 1989), one of the methods to test the average of sampling data, is employed in this algorithm. The mean square of INDEX is used in Welch's method.

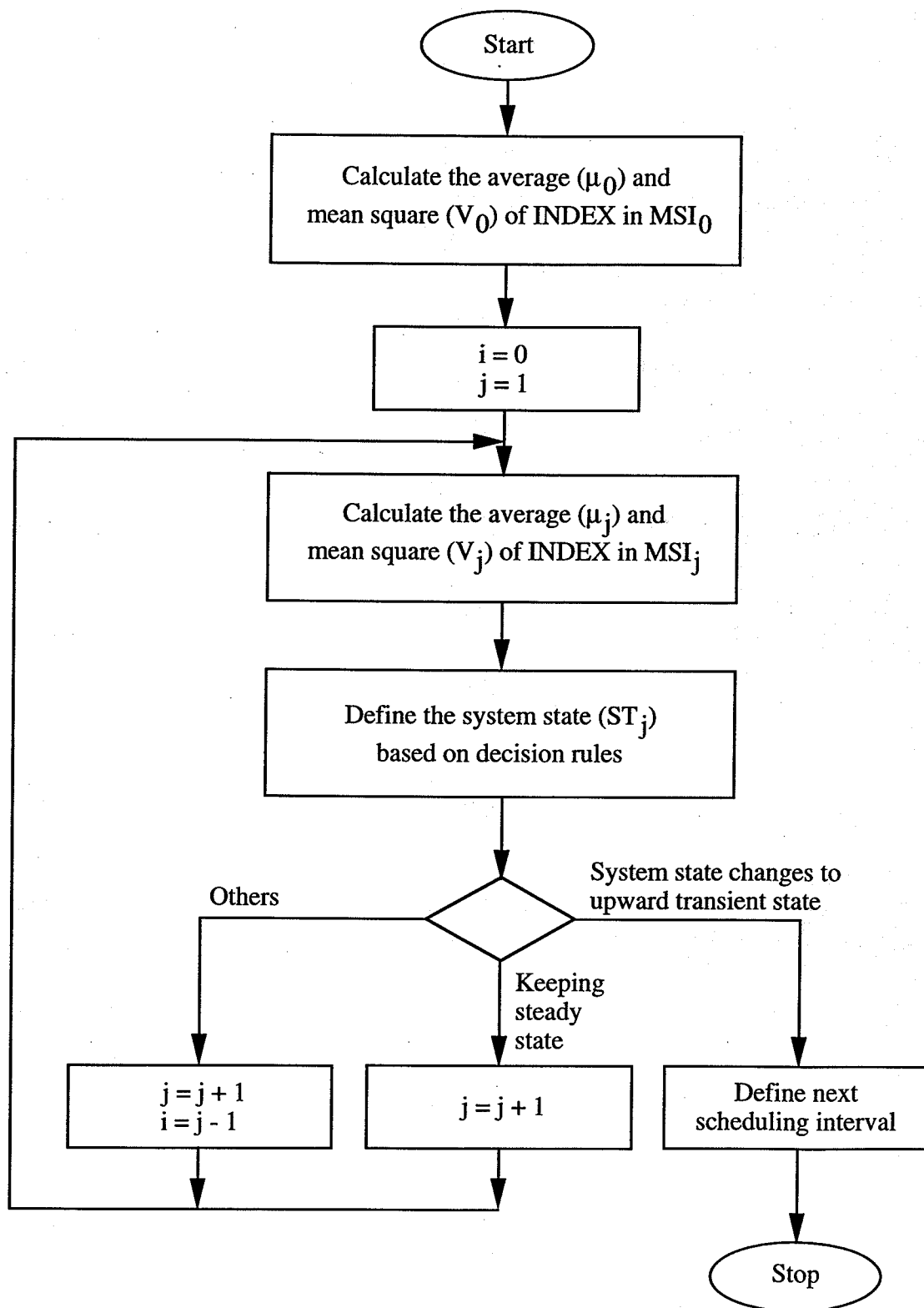


Figure 2.4 Transient detection algorithm

If a predicted system state (ST_j) changes to upward transient state from steady state or downward transient state, a next scheduling interval is defined from current time (real time) to the simulation time at which the upward transient state is detected. For example in Figure 2.5., since an upward transient state begins at MSI_j , the scheduling interval is defined from current time (MSI_0) to the beginning of MSI_j .

If the system state at MSI_j does not change to upward transient state, the system state at the next MSI is tested. In addition, the base data, m_i , V_i , and ST_i , to compare the system state with the next MSI is changed if system state at MSI_j does not remain in steady state.

This algorithm terminates the procedure when the next scheduling interval is defined. The Simulator and the INDEX calculation are terminated at the same time.

2.3.2 Dispatching Rule Selection Module

This module selects a dispatching rule to use in the next scheduling interval defined by the transient detection and scheduling interval setting module.

At first stage in this module, candidate dispatching rules are defined for evaluation. (Wu and Wysk (1988) developed an expert system which generates promising single dispatching rules for current system state.)

Each candidate dispatching rule is evaluated with the simulation results provided by the simulator, and the rule which performs the best to performance criteria is selected as a dispatching rule for the next scheduling interval.

To evaluate the performance of candidate dispatching rules is a difficult problem because of the short time window for evaluation. As described in Section 2.2, the problem of censored data strongly affects the evaluation. In this research, four methodologies to evaluate the performance of candidate dispatching rules are proposed. Those methodologies are called strategy 1, 2, 3, and 4, in this research. The strategy 1 has no mechanism to eliminate the problem of censored data. The others, i.e., strategies 2, 3, and 4, have different mechanisms for this problem. The following sections describe the strategies in detail.

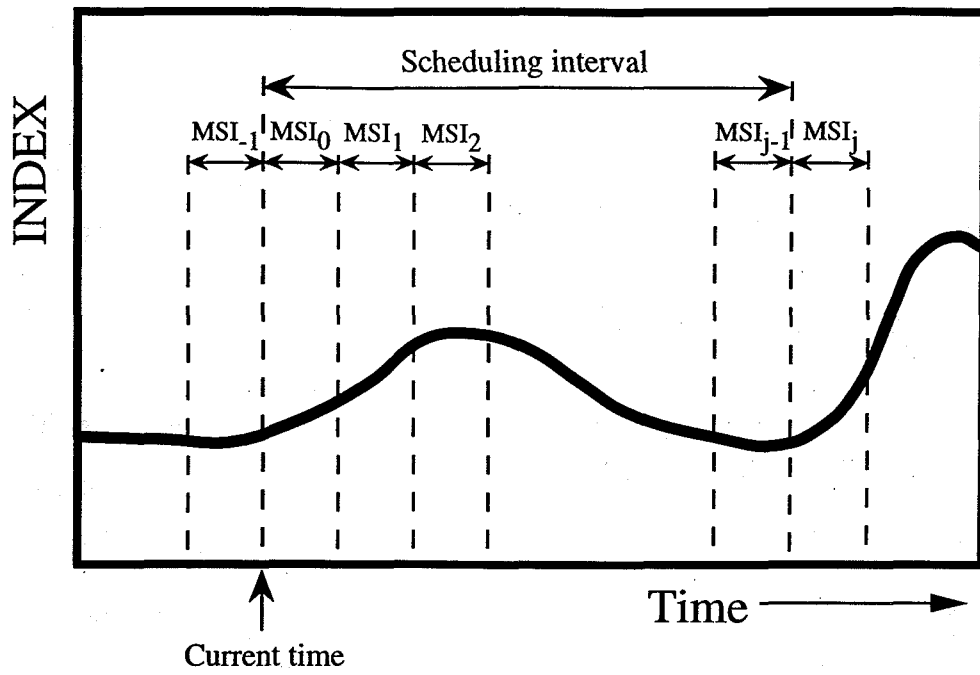


Figure 2.5 Minimum Scheduling Interval (MSI)

(1) Dispatching rule selection algorithm: Strategy 1

In this algorithm, a simulation window is defined of equal length to the next scheduling interval. In other words, this algorithm evaluates the performance of each candidate dispatching rule with the simulation terminated at the end of the next scheduling interval.

This is a same concept employed in the multi-pass scheduling algorithm described in Section 2.2 and does not consider the censored data problem at all.

(2) Dispatching rule selection algorithm: Strategy 2

In this algorithm, the best dispatching rule for the next scheduling interval is selected based on the results of simulation under the following conditions.

- Simulation window is defined from the current time to the time until all parts which will exist in the system during the next scheduling interval depart from the system.
- Dispatching rule after the end of the next scheduling interval is changed from the one under investigation during the interval to FIFO (First In First Out) rule. (FIFO rule is employed to allow unbiased part movement after the next scheduling interval.)

Figure 2.6 explains this more clearly. In Figure 2.6, T₀ is the next scheduling interval defined based on system state. Since parts A, B, and D will exist in a system during T₀, simulation will be terminated and the performance of a candidate dispatching rule will be evaluated after the completion of part B in this case. Part E is not included in evaluation because it will enter the system after T₀ interval. In this case, a candidate dispatching rule is used in T₀ and FIFO rule is used in T₂ as dispatching rules for simulation.

In this algorithm, the length of simulation window is slightly different depending on the candidate dispatching rule because different candidate dispatching rules produce different part movement in a system.

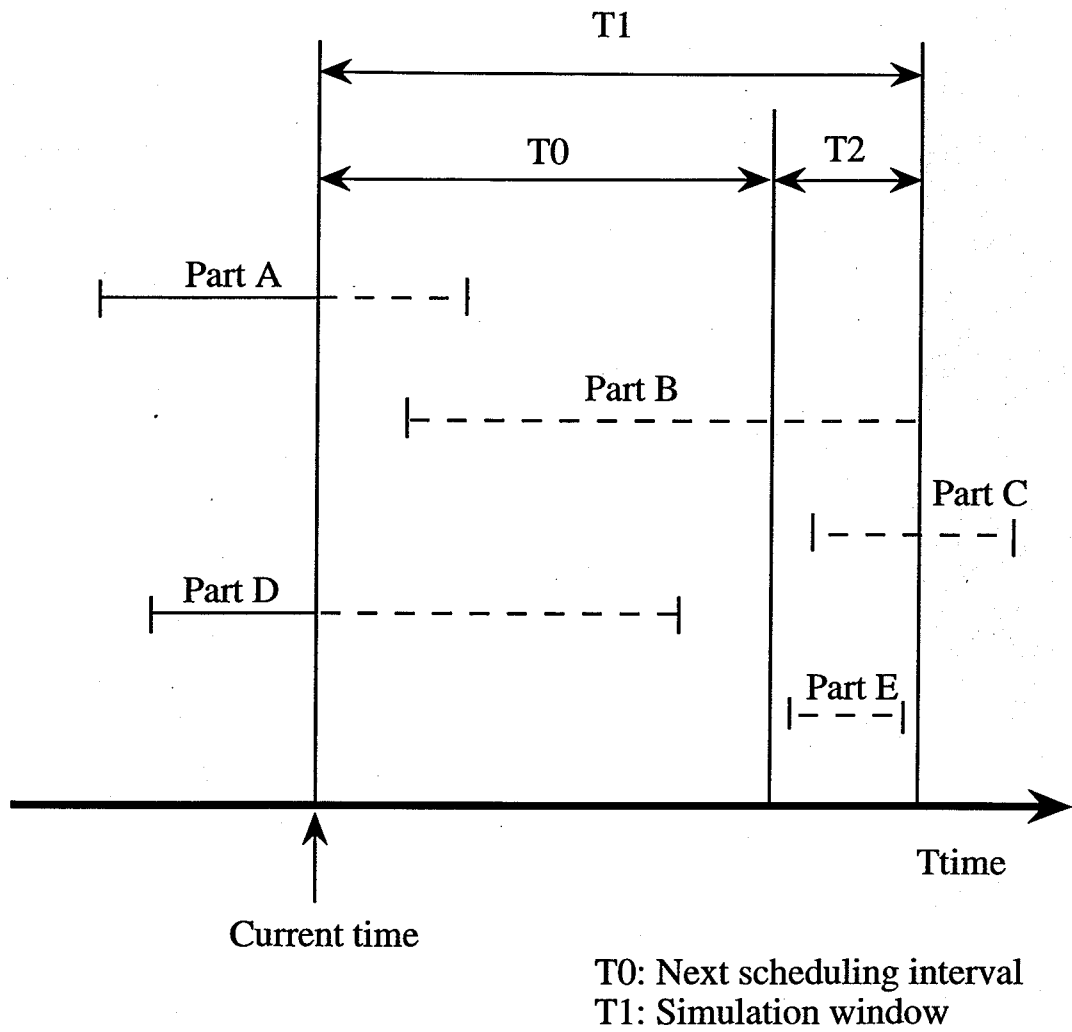


Figure 2.6 Simulation window in Strategy 2

(3) *Dispatching rule selection algorithm: Strategy 3*

In this algorithm, a simulation window is defined as from current time to the end of the entire manufacturing period. (The entire manufacturing period should be specified by a corporate system in advance. For example, the time bucket of the medium term scheduling system could be the entire manufacturing period.)

The following steps describes how this algorithm selects the best dispatching rule for the next scheduling interval.

Step 1: Select the best single dispatching rule for the entire manufacturing period with simulation. (This is termed the single-pass scheduling algorithm.) Step 1 is activated at the beginning of the entire manufacturing period only. The simulation is terminated at the end of the entire manufacturing period, and the performance of each candidate dispatching rule is evaluated.

Step 2: Select the best dispatching rule for the next scheduling interval from candidate dispatching rules based on the results of simulation under the following conditions.

- Simulation window is defined from the current time to the end of the entire manufacturing period.
- Dispatching rule after the end of the next scheduling interval is changed from the one under investigation during the interval to the dispatching rule selected in step 1.

This algorithm evaluates the performance of each candidate dispatching rule with the simulation terminated at the end of the entire manufacturing period. For the simulation, it uses the best single dispatching rule selected in step 1 after the end of the next scheduling interval. For this reason, the combination of dispatching rules by this algorithm guarantees equal or better performance than the best single dispatching rule by the single-pass scheduling algorithm which uses a single dispatching rule for the entire manufacturing period.

(4) Dispatching rule selection algorithm: Strategy 4

This algorithm selects a dispatching rule for the next scheduling interval (dt_i) based on the results of simulation terminated at the end of the following scheduling interval (dt_{i+1}) which immediately follows the next scheduling interval.

The following steps describes how this algorithm selects the best dispatching rule for the next scheduling interval (dt_i).

Step 1: Define dt_i and dt_{i+1} intervals in advance by the transient detection algorithm described in Section 2.3.1.

Step 2: Evaluate all combinations of candidate dispatching rules for dt_i and dt_{i+1} intervals with simulation. Simulation window is defined from the current time to the end of the dt_{i+1} interval.

Step 3: Select the best dispatching rule for the dt_i interval. The dispatching rule for the dt_i interval in the best combination of candidate dispatching rules for two intervals is selected as a dispatching rule for the dt_i interval.

Since this algorithm changes the candidate dispatching rules for evaluation at dt_i and dt_{i+1} intervals in a complete enumeration manner, it needs n^2 simulation runs if there are n candidate dispatching rules.

2.4 Experiments on Real-Time Scheduling System

In this section, the performance of real-time scheduling system, i.e., the transient based scheduling system, the single-pass scheduling system, and the multi-pass scheduling system, is tested on an FMS model.

The multi-pass scheduling system selects a dispatching rule in each short scheduling interval like the transient based scheduling system. However, the multi-pass scheduling system employs a constant length scheduling interval. In addition, it has no mechanism to reduce the problem of censored data on dispatching rule selection.

2.4.1 An FMS Model

The following sections describe the FMS model in detail. (The FMS model was built using SLAM II (Simulation Language for Alternative Modeling) (Pritsker, 1986).)

(1) *Physical layout*

Figure 2.7 shows the physical layout of an FMS model. The FMS model has two load/unload stations (L/U), four machines (M), and three Automated Guided Vehicles (AGV). Each L/U and M has a pallet storage carousel for pre/post process inventory area. Parts enter the FMS at L/U1 and leave at L/U2. Parts travel between machines by the AGV system.

(2) *Material handling*

Three AGVs are used to transport parts between stations. The detailed specification of AGV is as follows.

- AGV speed : 15.0 m/min. (when loaded)
: 16.5 m/min. (when unloaded)
- Length of AGV : 1.8 m
- Maximum capacity : One part / AVG
- AGV can move counterclockwise direction only.

(3) *Pre/Post process inventory*

The pallet storage carousel on each M and L/U has the following maximum capacity.

- Pre-process inventory : 6 parts
- Post-process inventory : 4 parts

(4) *Part Data*

There are six part types in this model. Each part type has a different part sequence in the FMS and has different processing times for each machine. In addition, a different fixed due date is defined for each part type. Tables 2.1 and 2.2 show these data in detail.

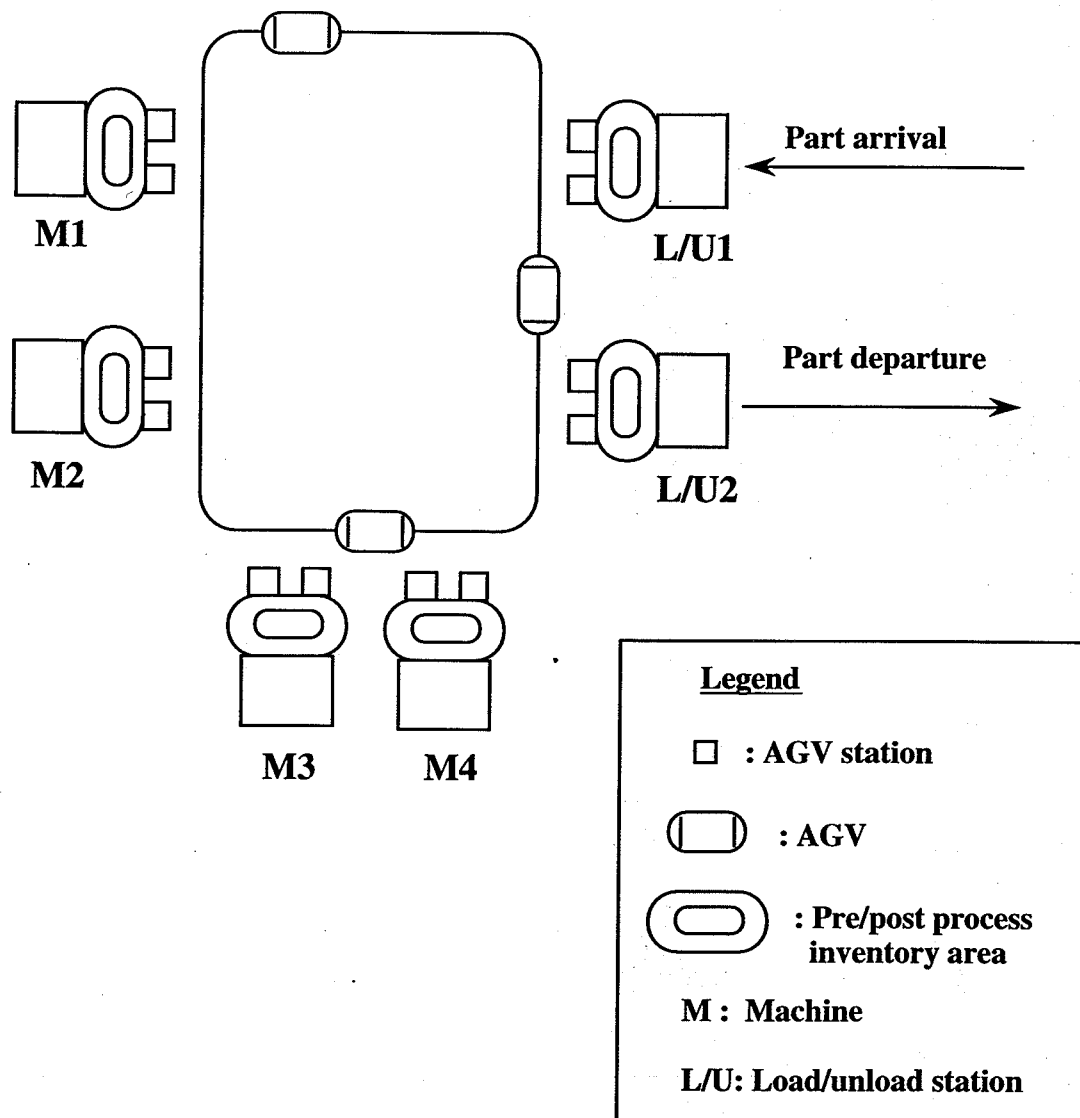


Figure 2.7 Physical layout of an FMS model

Table 2.1 Process sequence

Part Type	Sequence No.					
	1st	2nd	3rd	4th	5th	6th
A	L/U1	M1	M4	M3	L/U2	***
B	L/U1	M2	M1	M3	M4	L/U2
C	L/U1	M2	M1	M3	M4	L/U2
D	L/U1	M3	M2	M1	L/U2	***
E	L/U1	M4	M2	M1	L/U2	***
F	L/U1	M4	M3	M2	M1	L/U2

M: Machine

L/U: Load/Unload station

Table 2.2 Processing time (in minutes) and due date

Part Type	Process						Due Date in min. After Entry
	M1	M2	M3	M4	L/U1	L/U2	
A	10.0	***	18.0	20.0	9.0	5.0	180.0
B	23.0	15.0	15.0	21.0	8.0	5.0	220.0
C	23.0	15.0	15.0	21.0	8.0	5.0	180.0
D	10.0	20.0	22.0	***	7.0	5.0	150.0
E	15.0	16.0	***	18.0	12.0	5.0	200.0
F	28.0	21.0	17.0	19.0	15.0	5.0	310.0

2.4.2 Design of Experiments

(1) Part release schedule

In a real-time scheduling system, the part release schedule which defines a part arrival pattern is provided by an external source as shown in Figure 2.3. In the experiments, three different part release schedules are employed. One of these is essentially random, one is small transient, and one is highly transient. These experiments are called EX-A (small transient), EX-B (highly transient), and EX-C (random) in the following sections. (Tables 2.3, 2.4, and 2.5 shows part release schedules for, respectively, EX-A, EX-B, and EX-C.)

In EX-A and EX-B, the part release schedules are designed so as to make transient states intentionally in the FMS. However, the magnitude of transient is designed so that EX-A has less transient than EX-B. To generate such part release schedules, three part types of all enter the FMS regularly and other part types intended to make transient states enter the FMS occasionally. In EX-C, on the other hand, the part release schedule is designed so that all part types enter the FMS almost random.

(2) Time intervals

Time intervals related to real-time scheduling system are defined as follows.

- The entire manufacturing period (Total scheduling time): 2880 min. (2 days)
- The scheduling interval of the multi-pass scheduling system is tested as either 240 or 360 (min.). This should be reasonable judging from the computational results of Wu and Wysk (1989).
- The minimum scheduling interval (MSI) of the transient based scheduling system is defined as 60.0 min. from the shortest total processing time of all part types.

(3) Candidate dispatching rules

In the real-time scheduling system, candidate dispatching rules are assumed to be provided as shown in Figure 2.3. In the experiments, the following rules are selected as candidate dispatching rules.

Table 2.3 Part release schedule for EX-A
(Part arrival rate) [part/minute]

Time [minute]	Part Type					
	A	B	C	D	E	F
0 - 80	0.025	0.02	0.0	0.0	0.0	0.0
80 - 230	0.025	0.02	0.033	0.0	0.0	0.0
230 - 650	0.025	0.02	0.0	0.0	0.0	0.0
650 - 770	0.025	0.02	0.0	0.033	0.0	0.0
770 - 1100	0.025	0.02	0.0	0.0	0.0	0.0
1100 - 1220	0.025	0.02	0.0	0.0	0.033	0.0
1220 - 1600	0.025	0.02	0.0	0.0	0.0	0.0
1600 - 1800	0.025	0.02	0.0	0.02	0.0	0.0
1800 - 1900	0.0	0.02	0.0	0.02	0.0	0.0
1900 - 2020	0.0	0.02	0.0	0.02	0.0	0.033
2020 - 2200	0.0	0.02	0.0	0.02	0.0	0.0
2200 - 2320	0.0	0.02	0.0	0.02	0.033	0.0
2320 - 2880	0.0	0.02	0.0	0.02	0.0	0.0

Table 2.4 Part release schedule for EX-B
(Part arrival rate) [part/minute]

Time [minute]	Part Type					
	A	B	C	D	E	F
0 - 100	0.025	0.02	0.0	0.0	0.0	0.0
100 - 340	0.025	0.02	0.0	0.033	0.0	0.0
340 - 800	0.025	0.02	0.0	0.0	0.0	0.0
800 - 1120	0.025	0.02	0.025	0.0	0.0	0.0
1120 - 1200	0.0	0.02	0.025	0.0	0.0	0.0
1200 - 1400	0.0	0.02	0.0	0.0	0.0	0.0
1400 - 1500	0.0	0.0	0.0	0.0	0.0	0.0
1500 - 1800	0.0	0.0	0.0	0.029	0.04	0.0
1800 - 2100	0.0	0.0	0.0	0.029	0.0	0.0
2100 - 2460	0.0	0.0	0.0	0.029	0.0	0.033
2460 - 2880	0.0	0.0	0.0	0.029	0.0	0.0

Table 2.5 Part release schedule for EX-C
(Part arrival rate) [part/minute]

Time [minute]	Part Type					
	A	B	C	D	E	F
0 - 250	0.02	0.02	0.017	0.0	0.0	0.0
250 - 400	0.02	0.02	0.017	0.0	0.017	0.0
400 - 480	0.0	0.02	0.017	0.0	0.017	0.0
480 - 500	0.0	0.02	0.0	0.0	0.017	0.0
500 - 730	0.0	0.02	0.0	0.017	0.017	0.0
730 - 750	0.0	0.02	0.0	0.017	0.0	0.0
750 - 800	0.0	0.02	0.0	0.017	0.0	0.02
800 - 980	0.0	0.0	0.0	0.017	0.0	0.02
980 - 1000	0.0	0.0	0.0	0.0	0.0	0.02
1000 - 1150	0.02	0.0	0.02	0.0	0.0	0.02
1150 - 1250	0.02	0.0	0.02	0.0	0.0	0.0
1250 - 1400	0.02	0.0	0.02	0.0	0.015	0.0
1400 - 1500	0.0	0.0	0.02	0.0	0.015	0.0
1500 - 1750	0.0	0.0	0.02	0.017	0.015	0.0
1750 - 1900	0.0	0.0	0.02	0.017	0.015	0.017
1900 - 2000	0.0	0.0	0.0	0.017	0.015	0.017
2000 - 2230	0.0	0.015	0.0	0.017	0.015	0.017
2230 - 2880	0.0	0.015	0.0	0.017	0.015	0.0

NINQ: Select the part from the current queue whose next process machine has the shortest queue.

SPT: Select the part which has the shortest processing time at the machine.

SLACK: Select the part which has the minimum slack time.

(Slack time = due date - present time - remaining process time.)

FIFO: Select the part which enters the queue at the machine first.

(4) Performance criteria

In experiments, the performance of real-time scheduling system is compared using the following six criteria.

(a) Mean flow time

$$\sum_i (C_i - R_i) / n$$

(b) Mean tardiness

$$\sum_i \text{Max}(0, L_i) / NT$$

(c) Mean flow time + Mean tardiness

$$\sum_i (C_i - R_i) / n + \sum_i \text{Max}(0, L_i) / NT$$

(d) Weighted mean flow time

$$(\sum_j \sum_i W_j * (C_i - R_i)) / n$$

(e) Weighted mean tardiness

$$(\sum_j \sum_i W_j * \text{Max}(0, L_i)) / NT$$

(f) Weighted mean flow time + Weighted mean tardiness

$$(\sum_j \sum_i W_j * (C_i - R_i)) / n + (\sum_j \sum_i W_j * \text{Max}(0, L_i)) / NT$$

where C_i : Completion time of a part of type i , R_i : Time of entry of a part of type i , D_i : due date of a part of type i , L_i : Lateness of a part of type i ($= C_i - R_i - D_i$), W_j : Performance criterion weight of part type j . (The weight is specified for each part type and is defined randomly between 1 and 5.), NT : Number of tardy parts of all types, n : Number of completed parts of all types.

2.4.3 Results of Experiments

The results of experiments are summarized in Tables 2.6, 2.7, and 2.8. In these tables, the numbers inside parentheses show the percentage improvement by each algorithm against the best schedule by the single-pass scheduling algorithm. In addition, Table 2.9 shows the average, standard deviation, minimum, and maximum of those data. The following sections describe the results in detail. (Other experiments were done under several weight's (W_j) values for performance criteria. However, no significant differences which change the following conclusions were observed.)

(1) Performance of the multi-pass scheduling system

The results of experiments show that the performance of the multi-pass scheduling system is highly variable. The performance improvements by this system versus the single-pass scheduling system range from -54.5% to 17.4%.

The reasons for instability of the multi-pass scheduling system can be speculated as follows. In the real-time scheduling system, the selected dispatching rule controls part movement in FMS during the next scheduling interval. It also controls part movement in the following scheduling interval indirectly if those intervals are not independent. (If no part is in the system at the end of scheduling interval A, it is defined that the scheduling interval A and the following scheduling interval B are independent.)

The dispatching rule selected by evaluating its performance only for the next scheduling interval is a local optimum solution. It could make performance criteria of the succeeding scheduling intervals worse. This seems to happen particularly when scheduling

intervals are begun in an upward transient state or in a very congested system state, because more parts controlled by the current dispatching rule enter the next scheduling interval.

The multi-pass scheduling system uses a constant scheduling interval which is insensitive to transient state. In addition, it has no mechanism to reduce the problem of censored data. It selects a dispatching rule based on local evaluation. For this reason, the results for the multi-pass scheduling system are not unexpected.

(2) Transient based scheduling system vs the multi-pass scheduling system

The transient based scheduling system (strategy 1) generally performs better than the multi-pass scheduling system. A single factor analysis of variance (Mason *et al.*, 1989) is employed to test the significant difference between strategy 1 and the multi-pass scheduling system by using the data which show the percentage improvement against the best schedule by the single-pass scheduling system. The result of the analysis of variance shows a significant difference between these systems at 95% confidence level.

Both strategy 1 and the multi-pass scheduling system have no mechanism to avoid the problem of censored data. The only difference is the definition of scheduling intervals. The multi-pass scheduling system uses a constant scheduling interval. On the other hand, the transient based scheduling system (strategy 1) uses scheduling intervals defined based on system transient state. For this reason, the results of experiments support the following conjecture.

- Using a scheduling interval defined based on system transient state makes the performance of real-time scheduling system better than using a constant scheduling interval.

(3) Performance of the transient based scheduling system

(a) Strategy 1

Strategy 1 does not always performs better than the best schedule by the single-pass scheduling system. In the single-pass scheduling system, a single dispatching rule is used throughout the entire manufacturing period.

In this algorithm, the simulation window where candidate dispatching rules are evaluated is of equal length to a scheduling interval. For this reason, the problem of censored data could seriously affect the selection of dispatching rules.

(b) Strategy 2

Strategy 2 makes results more stable than strategy 1. It improves the performance between -1.01% to 14.3% compared with the best schedule by single-pass scheduling system.

Although it is difficult to recognize a significant difference between strategy 1 and strategy 2 from the results of experiments, strategy 2 seems to reduce the problem of censored data to some extent compared with strategy 1.

(c) Strategy 3

Strategy 3 always performs better than the single-pass scheduling system. In addition, it provides nearly the best schedules over all the experiments. It performs up to 16.5% better than the best schedule by the single-pass scheduling system.

Strategy 3 evaluates the performance of each candidate dispatching rule with simulation terminated at the end of the entire manufacturing period, hence it guarantees equal or better performance than the best schedule by single-pass scheduling system. In addition, strategy 3 successfully reduces the problem of censored data on dispatching rule selection because of the longer simulation window than other strategies. By reducing censored data, strategy 3 seems to improve the ability to select the adequate dispatching rule for the next scheduling interval compared with other strategies.

(d) *Strategy 4*

Most results by strategy 4 are better schedules than for the other strategies and better than the results by the single-pass scheduling system. However, EX-A (performance criterion = weighted mean flow time) shows that strategy 4 does not always work well.

In strategy 4, a dispatching rule X is selected not because it is the best for the next scheduling interval, but because the combination of rule X for the next scheduling interval and some rule Y for the succeeding scheduling interval is the best. In other words, when it selects rule X for the next scheduling interval, it expects rule Y for the succeeding scheduling interval. However, rule Y may not be selected at the next stage. For this reason, strategy 4 sometimes performs worse than expected.

Table 2.6 Comparison of real-time scheduling system (Part release schedule: EX-A)

Real-time scheduling system	Criteria					
	(1) Mean flow time	(2) Mean tardiness	(3) Mean flow time plus mean tardiness	(4) Weighted mean flow time	(5) Weighted mean tardiness	(6) Weighted mean flow time plus weighted mean tardiness
<i>Single-pass scheduling</i>						
NINQ	203.90	48.80	252.70	309.20	69.10	378.20
SPT	179.10	55.10	234.30	276.50	87.60	364.10
SLACK	191.30	18.90	210.20	298.20	33.90	332.10
FIFO	190.50	23.20	213.70	294.90	40.30	335.20
<i>Transient based scheduling</i>						
Strategy 1	185.90	16.20	204.10	277.00	30.30	331.60
	(-3.80%)	(+14.30%)	(+2.90%)	(-0.18%)	(+10.60%)	(+0.15%)
Strategy 2	175.20	16.20	204.10	272.90	33.40	321.90
	(+2.18%)	(+14.3%)	(+2.90%)	(+1.30%)	(+1.47%)	(+3.07%)
Strategy 3	174.90	16.60	203.00	267.10	28.30	302.70
	(+2.35%)	(+12.20%)	(+3.43%)	(+3.40%)	(+16.50%)	(+8.85%)
Strategy 4	174.90	16.20	204.10	300.50	28.30	323.50
	(+2.35%)	(+14.30%)	(+2.90%)	(-8.68%)	(+16.50%)	(+2.59%)
<i>Multi-pass scheduling</i>						
Scheduling interval =240.0 min.	192.00	16.50	209.80	320.30	28.00	422.60
	(-7.20%)	(+12.70%)	(+0.19%)	(-15.80%)	(+17.40%)	(-27.30%)
Scheduling interval =360.0 min.	185.40	16.40	210.70	300.50	30.30	323.80
	(-3.52%)	(+13.20%)	(-0.24%)	(-8.64%)	(+10.60%)	(+2.50%)

(): The percentage improvement by each algorithm against the best schedule by the single-pass scheduling algorithm

Table 2.7 Comparison of real-time scheduling system (Part release schedule: EX-B)

Real-time scheduling system	Criteria					
	(1) Mean flow time	(2) Mean tardiness	(3) Mean flow time plus mean tardiness	(4) Weighted mean flow time	(5) Weighted mean tardiness	(6) Weighted mean flow time plus weighted mean tardiness
<i>Single-pass scheduling</i>						
NINQ	181.00	43.50	224.50	345.50	81.10	426.60
SPT	175.10	71.30	246.40	326.70	121.00	447.70
SLACK	182.50	38.20	220.70	357.50	76.20	433.70
FIFO	179.10	38.30	217.40	343.40	76.60	420.00
<i>Transient based scheduling</i>						
Strategy 1	174.50 (-0.34%)	37.00 (+3.14%)	219.60 (-1.01%)	323.10 (+1.10%)	76.20 (+0.00%)	411.10 (+2.12%)
Strategy 2	174.00 (+0.63%)	37.00 (+3.14%)	219.60 (-1.01%)	323.10 (+1.10%)	76.20 (+0.00%)	411.10 (+2.12%)
Strategy 3	174.00 (+0.65%)	35.30 (+7.59%)	214.30 (+1.43%)	323.10 (+1.10%)	65.70 (+13.80%)	394.50 (+6.07%)
Strategy 4	174.00 (+0.63%)	35.30 (+7.59%)	214.30 (+1.43%)	323.10 (+1.10%)	65.70 (+13.80%)	394.50 (+6.07%)
<i>Multi-pass scheduling</i>						
Scheduling interval =240.0 min.	174.30 (+0.46%)	38.90 (-1.83%)	233.60 (-2.85%)	331.00 (-1.32%)	97.70 (-28.20%)	447.80 (-6.62%)
Scheduling interval =360.0 min.	173.80 (-0.74%)	44.80 (-17.30%)	233.30 (-7.31%)	337.20 (-3.21%)	80.20 (-5.25%)	423.80 (-0.90%)

(): The percentage improvement by each algorithm against the best schedule by the single-pass scheduling algorithm

Table 2.8 Comparison of real-time scheduling system (Part release schedule: EX-C)

Real-time scheduling system	Criteria					
	(1) Mean flow time	(2) Mean tardiness	(3) Mean flow time plus mean tardiness	(4) Weighted mean flow time	(5) Weighted mean tardiness	(6) Weighted mean flow time plus weighted mean tardiness
<i>Single-pass scheduling</i>						
NINQ	187.80	58.90	246.60	245.50	68.60	314.10
SPT	170.50	65.90	236.40	230.40	104.60	335.00
SLACK	187.10	33.90	221.00	244.30	38.60	282.80
FIFO	180.00	33.60	213.60	239.00	46.80	285.80
<i>Transient based scheduling</i>						
Strategy 1	170.50 (+0.00%)	31.00 (+7.74%)	211.20 (+1.12%)	226.80 (+1.56%)	37.10 (+3.89%)	282.40 (+0.14%)
Strategy 2	170.50 (+0.00%)	31.60 (+5.95%)	211.90 (+0.80%)	225.30 (+2.21%)	38.60 (+0.00%)	276.00 (+2.19%)
Strategy 3	169.60 (+0.53%)	31.00 (+7.74%)	211.20 (+1.12%)	225.30 (+2.21%)	37.90 (+1.81%)	276.10 (+2.37%)
Strategy 4	169.60 (+0.53%)	31.00 (+7.74%)	211.20 (+1.12%)	225.30 (+2.21%)	37.90 (+1.81%)	276.10 (+2.37%)
<i>Multi-pass scheduling</i>						
Scheduling interval =240.0 min.	172.50 (-1.17%)	33.00 (+1.79%)	212.80 (+0.37%)	234.20 (-1.66%)	39.20 (-1.66%)	315.00 (-11.40%)
Scheduling interval =360.0 min.	172.60 (-1.23%)	51.90 (-54.50%)	218.80 (-2.43%)	235.80 (-2.36%)	42.10 (-9.18%)	297.30 (-5.12%)

(): The percentage improvement by each algorithm against the best schedule by the single-pass scheduling algorithm

Table 2.9 Summary of the percentage improvement against single-pass scheduling algorithm

Real-time scheduling system	Average	Standard deviation	Minimum	Maximum
<i>Transient based scheduling</i>				
Strategy 1	2.45	4.39	-3.80	14.30
Strategy 2	2.35	3.38	-1.01	14.30
Strategy 3	5.17	4.91	0.53	16.50
Strategy 4	4.24	6.00	-8.68	16.50
<i>Multi-pass scheduling</i>				
Scheduling interval = 240.0 min.	-4.12	11.40	-28.20	17.40
Scheduling interval = 360.0 min.	-5.23	14.10	-54.50	13.20

2.5 Chapter Summary

In this chapter, a transient based real-time scheduling system designed as one of the structures of the intelligent scheduling system for management of FMS in a dynamic business environment is described.

The system selects a dispatching rule dynamically for the next short period, in order to respond to the changes of system state due to the changes of production information. In this chapter, an algorithm which defines the next short period based on the system's transient state is proposed. To select the best dispatching rule for the next short period, three methodologies designed to eliminate the problem of censored data are proposed.

The performance of the transient based real-time scheduling system was tested with an FMS model. It improves the performance up to 16.5% against the conventional scheduling system which uses a single dispatching rule for the entire manufacturing period. The following observations on the real-time scheduling system in FMS are gained as a result of experiments:

- As one of the structures of the intelligent scheduling system in FMS, a real-time scheduling system which replaces dispatching rules as revisions of the instruction to a process control system at various short periods could perform better than a conventional single-pass scheduling system where a single dispatching rule is used throughout the entire production period.
- Using a short period defined based on system transient state for the next scheduling interval makes the performance of the system much more stable than using a constant short period.
- The problem of censored data strongly affects the selection of dispatching rules because of the short simulation window for evaluation. Methodologies designed to eliminate the problem, such as strategy 2, 3, or 4 proposed in this chapter, should be employed in a real-time scheduling system.

Chapter 3

A MIXED DISPATCHING RULE APPROACH IN FMS SCHEDULING

3.1 Chapter Overview

In the previous chapter, a transient based real-time scheduling system is proposed as one of the structures of the intelligent scheduling system in FMS. The proposed system, however, provides a single dispatching rule as an instruction to a process control system, and the single rule is assigned for all machines in a system for a given scheduling interval. (In this research, this approach is called a *single dispatching rule (SDR) approach*, whereas the approach which can assign different dispatching rules for each machine is called a *mixed dispatching rule (MDR) approach*.) In FMS, it is possible for each part type to have different sequencing routes and processing times. This can make the situation at one machine's queue completely different from the situation at other machines' queues. For this reason, the use of different dispatching rules for each machine could extend the capability of the intelligent scheduling system and improve FMS performance by adapting dispatching to the unique situation at each machine.

In this chapter, the MDR approach is adapted to a real-time scheduling system in FMS. In Section 3.2, a real-time scheduling system in FMS is introduced and surveyed briefly. In addition, the MDR approach to a real-time scheduling system is discussed. Section 3.3 describes a search algorithm which selects an appropriate MDR by focusing on bottleneck machines in an FMS. The effectiveness (with respect to performance criteria) of the MDR

approach and the efficiency (relative to exhaustive search) of the search algorithm are displayed in Section 3.4 for a particular FMS model.

3.2 Background

In short term FMS scheduling, dispatching rules are usually used to dynamically control part movement. However, Blackstone *et al.* (1982) have stated that no single dispatching rule has been shown to consistently produce better results than all other rules under a variety of shop configurations and operating conditions. For this reason, a real-time scheduling system which changes dispatching rules dynamically in successive short time intervals is appropriate as a structure for the intelligent scheduling system in FMS.

Figure 2.1 shows the relationships between a real-time scheduling system and other management systems in FMS. The corporate system provides the following information to the real-time scheduling system.

- Production planning information (part release schedule, due dates etc.).
- Master data (process sequence, processing time, etc.).
- Performance criteria.
- Candidate dispatching rules.
- On-line process data.

Based on the above data, the real-time scheduling system selects a dispatching rule which best suits the selected performance criteria in the next short period. The information on the on-line process data of the FMS is provided by the process control system.

Wu and Wysk (1988, 1989) propose a multi-pass scheduling algorithm for a real-time scheduling system. In this algorithm, a dispatching rule which performs best according to selected performance criteria is employed in successive short-term scheduling intervals. This algorithm uses a constant short time window for each scheduling interval. The evaluation of candidate dispatching rules is made based on predictions resulting from discrete event

simulation of the FMS. This algorithm can change dispatching rules dynamically for each short-term interval and can adapt to the current situation of an FMS.

Ishii and Talavage (1991) propose a transient based scheduling algorithm for a real-time scheduling system. This algorithm defines a short-term scheduling interval based on the system's transient state. The system state which denotes the beginning of a transient state is predicted by discrete event simulation of the FMS. After defining a short-term scheduling interval, the algorithm selects the best dispatching rule for that interval, again using discrete event simulation of the FMS. Whereas the Wu and Wysk approach uses a scheduling interval of constant size, the size of the scheduling interval is variable in the Ishii and Talavage approach.

Since both the multi-pass scheduling algorithm and the transient based scheduling algorithm change the dispatching rule for short-term intervals to adapt to the current system state, they can improve the performance of an FMS compared to that when the same dispatching rules are used for the entire scheduling period. However, neither of these algorithms have a mechanism to implement mixed dispatching. In other words, both algorithms employ the SDR approach for the problem of dispatching rule selection.

The simple example of an FMS with three machines shown in Figure 3.1 indicates the potential of the MDR approach. Three part types which have different process sequences and different processing times for each machine enter the system randomly. Parts type A enters at machine 1 and 2. Part type B enters at machine 1, 3, and 2. Part type C enters at machine 2 and 3. As a result, the dynamic behavior is different at the three machine queues. In addition, the dispatching rule used at machine 1 affects the part arrival pattern at machine 2 and 3, and so the dispatching rule for machine 1 should be selected with this in mind. The use of possibly different dispatching rules for machine 1, 2, and 3 then has the potential of improving the performance of the FMS.

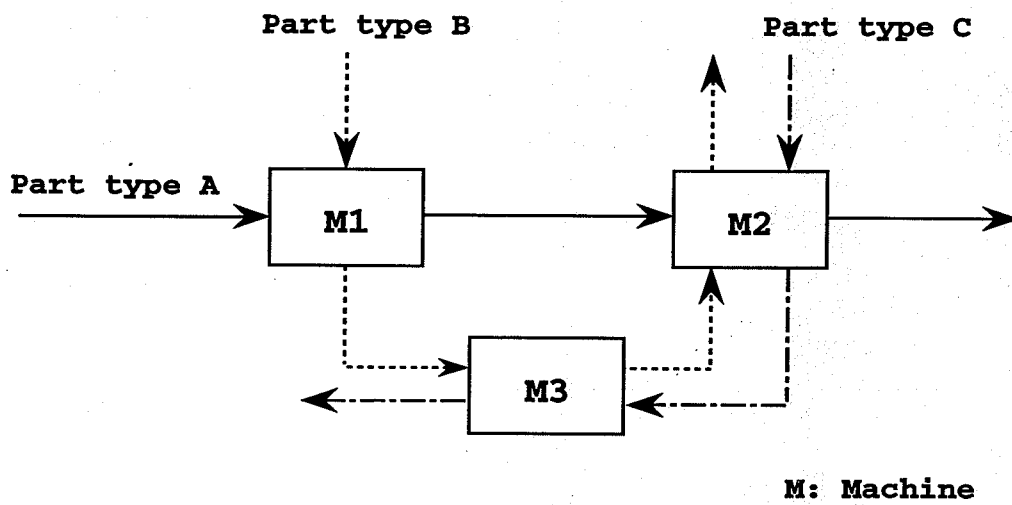


Figure 3.1 An example of simple FMS

In the SDR approach, the number of required simulation runs to determine the best rule is the same as the number of basic dispatching rules in a deterministic case because a single dispatching rule is used for all machines. In the MDR approach, however, it is necessary to specify a combination of the basic dispatching rules because a different dispatching rule may apply at each machine. If one searches for the best combination using complete enumeration for the n -machine m -dispatching rule problem, there are m^n simulation runs in a deterministic case. It is generally not feasible to simulate for all possible combinations in practical cases.

Since the MDR approach is intended to be implemented in a real-time scheduling system, it is necessary that the algorithm find a satisfactory combination in a short time. In this chapter, a heuristic search algorithm which focuses on bottleneck machines in an FMS is proposed in order to reduce the search space of the candidate combinations and, thus, reduce the search time.

3.3 Search Algorithm for MDR Approach

The objective of the search algorithm for the MDR approach is to find the MDR combination which provides the best values of selected performance criteria during a scheduling interval. The scheduling interval is specified by a real-time scheduling system.

3.3.1 Basic Dispatching Rules

In the MDR approach, basic dispatching rules are assumed to be provided. In experiments, the following rules are used as basic dispatching rules. The best MDR here is found as a combination of these.

NINQ: Select the part from the current queue whose next process machine has the shortest queue.

SPT: Select the part which has the shortest processing time at the machine.

SLACK: Select the part which has the minimum slack time.

(Slack time = due date - present time - remaining process time.)

FIFO: Select the part which enters the queue at the machine first.

3.3.2 Search Strategy

The search algorithm employs the following search strategy which focuses on the bottleneck machines to guide the search process.

Stage 1 : Identify the best SDR for the FMS according to the performance criteria and use it as the basis for the search process.

Stage 2 : Select a part type (P_i) which performs worst according to the performance criteria.

Stage 3 : Select a bottleneck machine on a route of part type P_i .

Stage 4 : Test for better dispatching rules at the bottleneck machine. (A better dispatching rule is one which improves the the performance of part type P_i .) The dispatching rule selected as this possibly cause simultaneous degradation in performance of other part types, with an inferior overall effect. Therefore several alternative dispatching rules are tested in this stage.

Stage 1 identifies the starting point of the search process. The SDRs, for example all-SLACK rule for all machines, are evaluated by utilizing discrete event simulation. As a result, the best SDR according to the performance criteria is selected as the starting combination of dispatching rules for the search process. The selected SDR is called the current Best Dispatching Rule (BDR). The Effective Rule Table (ERT) and the Bottleneck Machine Table (BMT), which are used in the search algorithm, are also generated based on the simulation results in this stage.

The ERT shows the ranking of effectiveness (according to selected performance criteria) of dispatching rules for each part type. Table 3.1 shows an example of ERT. Each row in ERT shows the rank of effectiveness of dispatching rules for each part type. In Table 3.1, for example, the most effective dispatching rule for part type A is SPT, the second most effective is FIFO.

The BMT shows the ranking of the bottleneck machines for each part type. The machine with the highest utilization on the part sequence route is defined as the highest rank bottleneck for each part type. Table 3.2 shows an example of BMT. Each row in BMT shows the rank of bottleneck machines for each part type. So, for part type A, the machine presenting the largest bottleneck is machine 3 (which had a utilization of 80%). A detailed example corresponding to Tables 3.1 and 3.2 is given in Section 3.4.

Stage 2 selects the part type for which the performance needs to be improved. It seems reasonable to select the part type that has the worst initial performance because it has more potential for improvement.

Stage 3 selects a bottleneck machine of the part type selected in stage 2. The bottleneck machine is found based on the BMT prepared in stage 1. (In this regard, Lozinski and Glassey (1988) show that focusing on a bottleneck machine is effective for scheduling problems.)

Stage 4 selects dispatching rules which are to be tested for the bottleneck machine identified in stage 3. The most promising dispatching rule for the part type is tested first, where such dispatching rules are ranked based on the ERT prepared in stage 1.

Table 3.1 An example of Effective Rule Table (ERT)

Part type	Rank			
	1st	2nd	3rd	4th
A	SPT	FIFO	SLACK	NINQ
B	NINQ	FIFO	SPT	SLACK
C	NINQ	SLACK	FIFO	SPT
D	SLACK	FIFO	NINQ	SPT
E	NINQ	SPT	FIFO	SLACK
F	NINQ	FIFO	SLACK	SPT

Table 3.2 An example of Bottleneck Machine Table (BMT)

Part type	Rank		
	1st	2nd	3rd
A	M3 (80)	M1 (74)	M4 (63)
B	M3 (80)	M1 (74)	M2 (67)
C	M3 (80)	M1 (74)	M2 (67)
D	M3 (80)	M1 (74)	M2 (67)
E	M1 (74)	M2 (67)	M4 (63)
F	M3 (80)	M1 (74)	M2 (67)

M: Machine

(.): Average Utilization

3.4 Example and Evaluation of MDR Approach

In this section, the MDR approach is explained using an example, and the effectiveness and efficiency of the search algorithm for the MDR approach is tested on an example FMS model.

3.4.1 An FMS Model

Figure 2.7 shows the physical layout of an FMS model which has two load/unload stations (L/U), four machines (M), and three Automated Guided Vehicles (AGV) for part transport. Each L/U and M has a pallet storage carousel for pre/post process inventory. Parts enter the FMS at L/U1 and leave at L/U2.

The detailed specification for each AGV is as follows:

- AGV speed : 15.0 meters per minute (when loaded)
: 16.5 meters per minute (when unloaded)
- Length of AGV : 1.8 meters
- Maximum capacity : One part/AGV
- An AGV can move counterclockwise direction only.

The pallet storage carousel on each M and L/U has the following maximum capacities:

- Pre-process inventory : 6 parts
- Post-process inventory : 4 parts

There are six part types in this model. Each part type has a different process sequence in the FMS and a different deterministic processing time at each machine. A different due date is defined for each part type so that the ratio of the due date to the total processing time is between 2.0 to 3.0. Table 2.1 shows the data for the process sequence of each part type. For example, the process sequence of part type A is L/U1 -> M1 -> M4 -> M3 -> L/U2. Table 2.2 shows the data for the processing time and the due date of each part type. For

example, the processing time of a part of type A at M1 is 10.0 minutes and the due date is 180.0 minutes.

The part release schedule defines a part arrival pattern and is assumed to be provided by the corporate system shown in Figure 2.1. In the experiments, three alternative deterministic part release schedules called EX-A (a small transient part release scenario), EX-B (a highly transient part release scenario), and EX-C (a random part release scenario) are employed. In these schedules the deterministic part arrival rates (parts per minute) are changed periodically to evaluate the effectiveness of the MDR approach in dynamic situations. Tables 2.3, 2.4, and 2.5 show the part release schedules in detail. These tables describe the part arrival rates for each part type within certain time windows. In the schedule of Table 2.3, for example, the mean part arrival rates of the part type A, B, and C between time 80 and time 230 of the simulation are 0.025, 0.02, and 0.033 parts per minute, respectively. In other words, the production requirements of the part type A, B, and C during the period are 3, 3, and 4, respectively.

The search algorithm for the MDR approach has three parameters:

- The number of columns of the BMT corresponding to the ranking of bottleneck machines to be selected,
- The number of columns of the ERT corresponding to the ranking of dispatching rules to be selected,
- The number of ranking of part types to be searched.

These parameters define the size of the search space of the algorithm. In the experiments, the number of columns of BMT is 3, the number of columns of the ERT is 4, and the number of part types to be searched is 6.

The system performance is based on one or more of the following six criteria. In each case, the summation over i is the sum over completed parts, while the summation over j is the sum over part types.

(a) Mean flow time

$$\sum_i (C_i - R_i)/n$$

(b) Mean tardiness

$$\sum_i \text{Max}(0, L_i)/NT$$

(c) Mean flow time + Mean tardiness

$$\sum_i (C_i - R_i)/n + \sum_i \text{Max}(0, L_i)/NT$$

(d) Weighted mean flow time

$$(\sum_j \sum_i W_j * (C_i - R_i))/n$$

(e) Weighted mean tardiness

$$(\sum_j \sum_i W_j * \text{Max}(0, L_i))/NT$$

(f) Weighted mean flow time + Weighted mean tardiness

$$(\sum_j \sum_i W_j * (C_i - R_i))/n + (\sum_j \sum_i W_j * \text{Max}(0, L_i))/NT$$

where C_i : Completion time of a part of type i , R_i : Time of entry of a part of type i , D_i : due date of a part of type i , L_i : Lateness of a part of type i ($= C_i - R_i - D_i$), W_j : Performance criterion weight of part type j . (The weight is specified for each part type and is defined randomly between 1 and 5.), NT : Number of tardy parts of all types, n : Number of completed parts of all types.

In the above criteria, mean flow time and mean tardiness are added to test the effectiveness of the MDR approach and the performance of the search algorithm for a multi-objective problem. (In this research, a problem for which the performance criterion is either mean flow time, or mean tardiness, or weighted mean flow time, or weighted mean tardiness is defined as a single-objective problem. On the other hand, a problems for which the performance criterion is the *sum* of mean flow time plus mean tardiness, or the *sum* of

weighted mean flow time plus weighted mean tardiness is defined as a multi-objective problem.)

3.4.2 Detailed Explanation of MDR Search Algorithm

The example FMS will now be used to explain the MDR search algorithm shown in detail in Figure 3.2. Figure 3.3 shows a portion of the search space of the example, and displays how the algorithm works to find the best solution. For illustration, mean tardiness and the EX-B schedule are used as the performance criterion and the part release schedule, respectively.

The details of the search algorithm are described as a series of steps which correspond to stages 1 through 4 in Section 3.3.

In step 1, SDRs generated from basic dispatching rules are evaluated by discrete event simulation of the FMS. The example FMS was modeled using SLAM II (Pritsker, 1986). As shown in Figure 3.3, four basic dispatching rules are used; i.e., NINQ, SPT, SLACK, and FIFO.

In step 2, the SDR which performs best for the selected performance criteria is set to the current BDR. In Figure 3.3, the SLACK rule is shown as being selected as the current BDR: (SLACK,SLACK,SLACK,SLACK,SLACK,SLACK) because the SLACK rule provides the minimum mean tardiness of all the SDRs. (The vector (R1,R2,R3,R4,R5,R6) shows the combination of dispatching rules employed for M1 to M4 and L/U1, L/U2, respectively.)

In step 3, the ERT is generated based on the simulation results in step 1. Table 3.1 shows the ERT generated for the example of Figure 3.3.

In step 4, the BMT is generated based on the part type's sequence of operations and average utilization of machines for the current BDR. Table 3.2 shows the BMT generated for the example of Figure 3.3. (The average utilization of machines for this case is 74%, 67%, 80%, 63%, 23%, and 42% for M1 to M4 and L/U1, L/U2, respectively.)

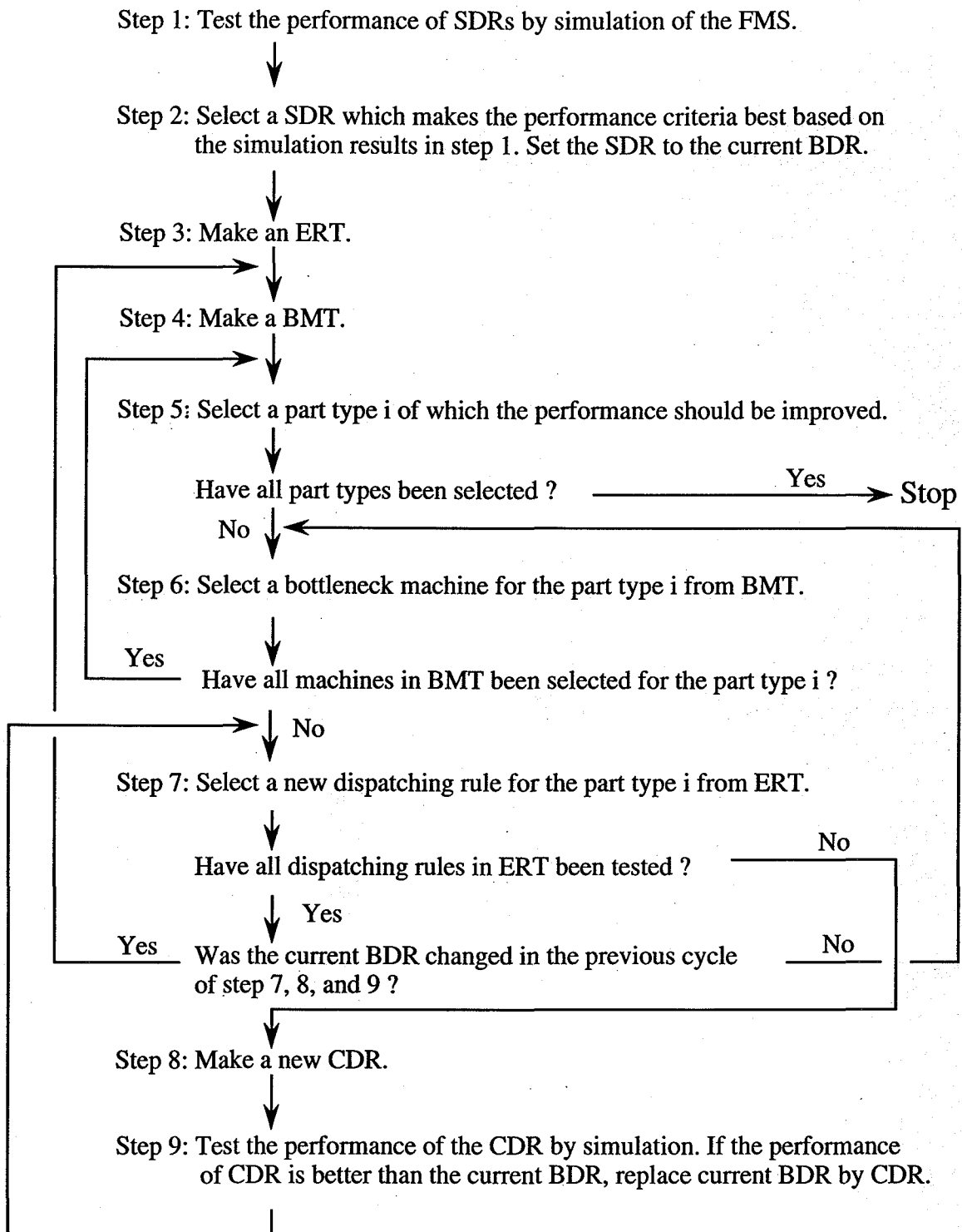


Figure 3.2 A search algorithm for MDR approach

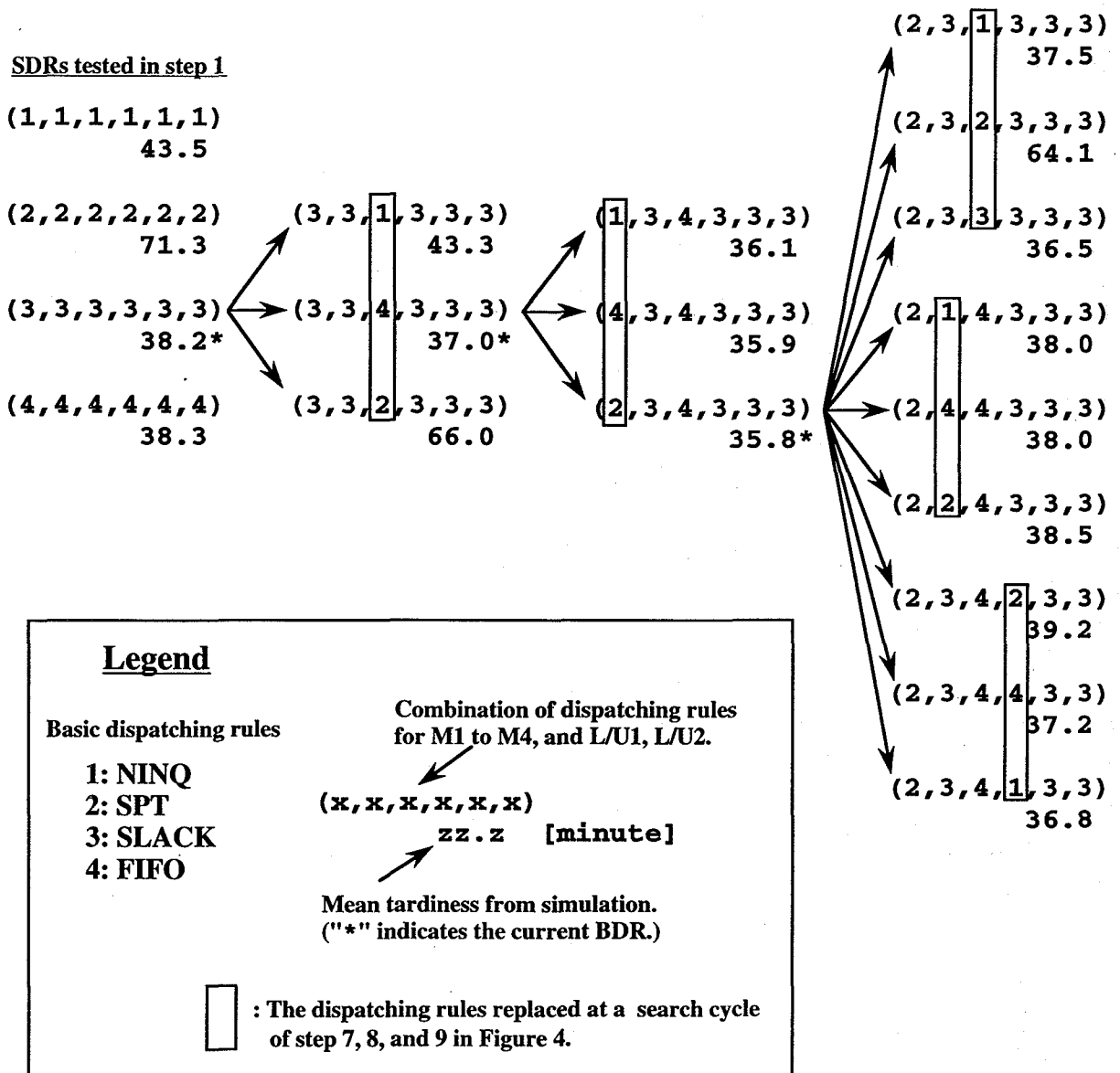


Figure 3.3 Example iterations of MDR search algorithm

In step 5, the algorithm selects a part type for which the performance should be improved, starting with the type that has the worst performance for the current BDR. After all part types have been selected in step 5, the algorithm terminates the search and the current BDR is defined as the final BDR. In this example, part type B was selected because it is the most tardy of all part types in the FMS. (Though not shown in a table, the simulation revealed that, starting with the worst, the rank of the tardiness of each part type is B, A, C, D, F, and E.)

In step 6, the bottleneck machine for the part type selected in step 5 is defined based on the BMT. If all bottleneck machines in the BMT have been selected for the part type in previous runs of step 6, then the part type which has the next worst performance for the current BDR is selected by returning to step 5.

In step 7, a promising dispatching rule is selected based on the ERT for the part type selected in step 5.

In the example of Figure 3.3, part type B was selected in step 5 as the one for which the performance should be improved. Tables 3.1 and 3.2 show the ERT and BMT for this example indicating that the algorithm selects, respectively, M3 as a bottleneck machine and the NINQ rule as a promising dispatching rule in step 6 and 7.

In step 8, a new candidate dispatching rule (CDR) results by replacing the dispatching rule in the current BDR with the new one selected in step 7. Only the dispatching rule for the bottleneck machine is replaced in this step. In the example of Figure 3.3, the new CDR consisting of NINQ for M3 and SLACK for others is generated. The new CDR is tested by simulation and the mean tardiness for the CDR is evaluated.

The algorithm repeats steps 7, 8, and 9 until all dispatching rules in a row of ERT are tested by simulation. If the current BDR is changed in step 9 during the cycle of step 7, 8, and 9, the algorithm goes to step 4 to construct a new BMT because the bottleneck machine for the new BDR could be different. If the current BDR is not changed in this cycle, the algorithm goes to step 6 and the next bottleneck machine is selected. In the case of Figure 3.3, since SLACK has already been tested, the three dispatching rules, i.e., NINQ, FIFO, and SPT in the row of part type B of the ERT, are tested for M3 in the cycle of steps 7, 8,

and 9. Because the performance of CDR: (SLACK, SLACK, FIFO, SLACK, SLACK, SLACK) was better than the current BDR: (SLACK, SLACK, SLACK, SLACK, SLACK, SLACK), the current BDR is replaced and the algorithm goes to step 4 to construct the new BMT based on the new current BDR: (SLACK, SLACK, FIFO, SLACK, SLACK, SLACK).

The search algorithm repeats the above steps until it tests all possible CDRs. In the case of Figure 3.3, the search is continued until the algorithm finds the BDR: (SPT, SLACK, FIFO, SLACK, SLACK, SLACK) as the best MDR. In this case, a total of 19 CDRs are tested by the algorithm and the solution is 6.3% better than the best solution by the SDR approach.

3.4.3 *Effectiveness and Efficiency of MDR Approach*

The total scheduling time for each experiment on the example FMS to test the MDR approach was set at 2880 minutes (two days). Tables 3.3, 3.4, and 3.5 summarize the results of experiments for the part release schedule EX-A, EX-B, and EX-C, respectively. Each table shows the results for the SDR approach and the MDR approach for each performance criterion. In the example of Table 3.3, since SPT was the best SDR for mean flow time (179.10 minutes), it is defined as the solution for the SDR approach. The rows termed "Search algorithm" in these tables show the solutions for the MDR approach. In the rows, the numbers inside the parentheses show the percentage improvement of the performance provided by the MDR approach over the (best) solution from the SDR approach.

The row titled "the number searched" shows the number of search iterations or the number of tested combinations of dispatching rules by the search algorithm. In the example of Table 3.3, 25 search iterations were done by the search algorithm to find the MDR which obtained the best mean flow time.

The row titled "rank of solution" shows the rank of the solutions obtained by the search algorithm from a ranking of all possible combinations of basic dispatching rules. (The ranking was done by testing all possible combinations of the basic dispatching rules on M1,

M2, M3, and M4 ($4^4=256$ combinations)). In the example of table 3.3, the solution obtained by the search algorithm with respect to mean flow time was ranked 1st or the best MDR among all possible combinations. (Other experiments were done using different part type weights (W_j) for performance criteria. However, no significant differences were observed which would change the conclusions described later.)

The MDR approach improves the performance of the example FMS up to 15.9% compared with the best result using the SDR approach. The average improvement is 4%.

Since there are six performance criteria and three part release schedules, 18 experiments were done in the study. If there were no significant difference between the MDR approach and the SDR approach, the number of experiments (NE) in which the MDR approach performs better than the SDR approach would distribute binomially with p (probability) equal to 0.5. Therefore the estimate of NE in 99% confidence interval is calculated by the following equation (Bethea *et al.*, 1975).

$$n \cdot p - 2.58 \cdot [n \cdot p(1-p)]^{1/2} < NE < n \cdot p + 2.58 \cdot [n \cdot p(1-p)]^{1/2}$$

Since n (the number of experiments) is 18 and p is 0.5, the NE is estimated as from 4 to 14. In the experiments, the NE was 18 because the MDR approach always performed better than the SDR approach. For this reason, it is possible to say that there is a significant difference between the MDR approach and the SDR approach.

The results of the experiments support the following statement:

- A real-time scheduling system in FMS by the MDR approach in which different dispatching rules can be used for each machine improves the performance of FMS compared with a real-time scheduling system by the SDR approach for the specific scenarios and performance criteria selected. It can be speculated that the results are applicable in general.

The number of search iterations required for the various experiments ranges from 16 to 33. If the search space which contains all possible combinations ($4^6 = 4096$) is considered, the algorithm searched about 0.4% to 0.8% of the entire space. Most MDR solutions were the best of all combinations. In the worst case, the MDR solution ranked 9th.

The computation time for the search algorithm depends on the computer used and the scheduling conditions. In the experiments which employed a DEC VAX Station 3200, the CPU times were as follows;

- SDR approach: 35 to 41 seconds,
- MDR approach: 150 to 350 seconds.

Although the CPU time for the the MDR approach is longer than that for the SDR approach, it is short enough to utilize the algorithm in real-time scheduling systems. On the other hand, if all possible combinations must be tested to find the best MDR, it would take more than 10 hours.

The MDR approach seems to work efficiently for single-objective problems. The rank of the solutions for single-objective problems ranges from 1st to 3rd among all 4096 combinations. However, the ranking was not as high for multi-objective problems, ranging from 1st to 9th. (Multi-objective problems may have more local optimum solutions than single-objective problems because of the trade-off between different objectives.) To improve the performance of the search algorithm for multi-objective problems, some backtracking mechanism could be employed. The backtracking mechanism could backtrack to the previous CDR and test the alternative CDR for better solutions. In the example of Figure 3.3, the search started from an all SLACK combination. The backtracking mechanism, for example, could re-start the search from all-SPT combination and could reach another solution. As one of the backtracking mechanisms, the concept of simulated annealing algorithm (Manz *et al.*, 1989) could improve the performance of the algorithm for the multi-objective problems. However, any backtracking mechanism would likely increase the number of iterations of the search. Ultimately, one must consider the trade-off between rank of solution and the CPU-time requirement since the MDR approach is assumed to be used in a real-time environment within the intelligent scheduling system.

Table 3.3 Results of experiments (Part release schedule: EX-A)

FMS scheduling approach	Criteria					
	(a) Mean flow time	(b) Mean tardiness	(c) Mean flow time plus mean tardiness	(d) Weighted mean flow time	(e) Weighted mean tardiness	(f) Weighted mean flow time plus weighted mean tardiness
<i>SDR approach</i>						
NINQ	203.90	48.80	252.70	309.20	69.10	378.20
SPT	179.10	55.10	234.30	276.50	87.60	364.10
SLACK	191.30	18.90	210.20	298.20	33.90	332.10
FIFO	190.50	23.20	213.70	294.90	40.30	335.20
<i>MDR approach</i>						
Search algorithm	174.80	16.40	206.30	266.90	31.80	321.70
	(+2.40%)	(+13.20%)	(+1.86%)	(+3.47%)	(+6.19%)	(+3.13%)
The number searched	25	25	22	33	16	16
Rank of solution	1st	1st	2nd	1st	3rd	1st

Table 3.4 Results of experiments (Part release schedule: EX-B)

FMS scheduling approach	Criteria					
	(a) Mean flow time	(b) Mean tardiness	(c) Mean flow time plus mean tardiness	(d) Weighted mean flow time	(e) Weighted mean tardiness	(f) Weighted mean flow time plus weighted mean tardiness
<i>SDR approach</i>						
NINQ	181.00	43.50	224.50	345.50	81.10	426.60
SPT175.10	71.30	246.40	326.70	121.00	447.70	
SLACK	182.50	38.20	220.70	357.50	76.20	433.70
FIFO179.10	38.30	217.40	343.40	76.60	420.00	
<i>MDR approach</i>						
Search algorithm	175.00	35.80	215.50	320.50	64.10	409.80
	(+0.00%)	(+6.28%)	(+0.874%)	(+1.90%)	(+15.9%)	(+2.43%)
The number searched	25	19	22	28	19	25
Rank of solution	1st	2nd	2nd	1st	1st	5th

Table 3.5 Results of experiments (Part release schedule: EX-C)

FMS scheduling approach	Criteria					
	(a) Mean flow time	(b) Mean tardiness	(c) Mean flow time plus mean tardiness	(d) Weighted mean flow time	(e) Weighted mean tardiness	(f) Weighted mean flow time plus weighted mean tardiness
<i>SDR approach</i>						
NINQ	187.80	58.90	246.60	245.50	68.60	314.10
SPT	170.50	65.90	236.40	230.40	104.60	335.00
SLACK	187.10	33.90	221.00	244.30	38.60	282.80
FIFO180.00	33.60	213.60	239.00	46.80	285.80	
<i>MDR approach</i>						
Search algorithm	169.40 (+0.645%)	31.10 (+7.44%)	212.10 (+0.702%)	225.00 (+2.34%)	37.90 (+1.81%)	277.90 (+1.74%)
The number searched	25	31	19	31	22	28
Rank of solution	1st	1st	9th	1st	1st	2nd

3.5 Chapter Summary

In this chapter, the development of a mixed dispatching rule (MDR) algorithm for assigning different dispatching rules to FMS machines is described to improve the performance of real-time scheduling systems in FMS. The performance of the search algorithm to select an appropriate MDR was tested on an example FMS model.

The following conclusions can be drawn from the experiments:

- The mixed dispatching rule approach in FMS scheduling improves FMS for the selected performance criteria compared with the conventional approach which uses a single dispatching rule for all machines.
- The proposed search algorithm for the mixed dispatching rule is efficient enough to be applied in real-time scheduling systems.
- The effectiveness of the search algorithm seems to be somewhat limited for multi-objective problems. Some backtracking mechanism might improve the performance of the algorithm in that case.

PART 3:

APPLICATION OF INTELLIGENT SCHEDULING SYSTEM IN BATCH PROCESS

Chapter 4

A FUNDAMENTAL STRUCTURE OF ON-LINE SCHEDULING SYSTEM IN BATCH PROCESS MANAGEMENT

4.1 Chapter Overview

The intelligent scheduling system, which effectively manages flexible production systems under the dynamic changes of production information, is a vital issue in modern batch processes because of their dynamic behavior and the frequent changes of production information due to the unpredictable business environment. In batch processes, as mentioned in Chapter 1, the on-line scheduling system, which dynamically adjusts production schedules according to the changes of production information, is promising as a structure of the intelligent scheduling system.

However, the details of on-line scheduling systems for batch processes, mainly associated with the Function 3 in the framework for the intelligent scheduling system shown in Figure 1.6, have not been well investigated yet. In addition, no single system can work effectively for all batch processes because of the diversity of batch process configuration and the operation associated with batch process scheduling. This obliges the development of dedicated systems according to the characteristics of each batch process, and therefore causes a long system development time and also inconsistency of the system's quality.

In this chapter, a fundamental structure of an on-line scheduling system within the framework of the intelligent scheduling system is proposed. The structure, which

corresponds to Functions 2 and 3 of the framework shown in Figure 1.6, provides a fundamental approach and scheme to establish on-line scheduling systems for batch process management. Because the structure is flexibly adaptable to the characteristics of each batch process, it realizes on-line scheduling systems for a wide variety of batch processes with less system development time and a certain quality level. In addition, evolution of the system is possible by updating the configuration of the mechanisms in the structure.

This chapter is organized as follows: Section 4.2 presents a brief review of previous works and research needs on an on-line scheduling system as one of the structures of the intelligent scheduling system in batch process management. Section 4.3 describes a fundamental structure of on-line scheduling system, and functions required within the structure. Section 4.4 demonstrates the effectiveness of the proposed structure under several scenarios frequently observed in ordinary batch process operations.

4.2 Previous Works and Research Needs

Many researchers have extensively studied batch process scheduling problems over the last decade. Comprehensive reviews in this field are provided by Ku *et al.* (1987), Musier and Evans (1990), Reklaitis (1991), and Rippin (1993). In previous works, many researchers have recognized the importance of on-line scheduling systems to deal with a wide variety of uncertainties, such as changes of production demands, inventory levels, resource availability, processing times, and so on, in modern batch process management. However, researches on on-line scheduling systems have been limited so far because of the difficulty of the problem.

Egli and Rippin (1986) state the importance of flexible and dynamic scheduling procedure in the face of changing conditions because it is essential for scheduling to add the data of changed scheduling situations to a computer, and then to produce a new schedule adapted to the changed conditions at short notice. They develop a computer program SRSBP (Short Range Scheduling for Batch Plants) which can restart from any intermediate state to handle changes of scheduling situations. However the detailed procedures to deal with

uncertainties in scheduling are not described. Cott and Macchietto (1989a) propose a computer-based system which integrates planning and plant control of batch processes. In the system, they propose an on-line scheduling structure consisting of POMA (Projected Operation Modification Algorithm) (Cott and Macchietto, 1989b) and a scheduling algorithm. The system employs POMA to deal with processing time variations. Other variations such as changes of production demand, resource availability, and so on, are assumed to be handled by a scheduling algorithm. (Although no concrete scheduling algorithm is described in the paper, they specify that the algorithm should generate schedules under realistic operational constraints including multiple resource usage, materials transfer times, and so on.) However, the detailed methodology for coordinating the POMA and the scheduling algorithm, such as timing of schedule modification, range of modification, and so on, is not described. Terpstra *et al.* (1994) propose a prototype of reactive scheduler, intended to handle a wide variety of uncertainties, for mixed-batch/continuous plants with a real-time expert system. The scheduler consists of three modules: Planner, Integer Scheduler (IS), and Non-Integer Scheduler (NIS). Planner generates all possible production paths to produce products required by production orders in a specific plant. IS searches optimal production paths and sequences by using a branch-and-bound method, and then NIS optimizes timing and flows within a path by NLP (Non Linear Programming). This approach, however, has not been well investigated yet. Moreover, it seems to be inflexible as far as changes of process configurations and problem sizes are concerned, because their basic technologies are mathematical programming which offers relatively limited modeling, and computational capabilities for general scheduling problems. For example, Ku *et al.* (1987) state that the size of problem that can be resolved using exact techniques is limited to about 12 products.

In previous works, on-line schedule modification algorithms to deal with processing time variations, which are the most essential variability in batch processes, have been proposed. The algorithms can be classified into two approaches, i.e., time shift approach and operating parameter tuning approach.

The time shift approach, such as Cott and Macchietto (1989b), Ishii and Muraki (1995b), minimizes the effect of processing time variations in multiproduct batch processes by dynamically shifting the start times of products to be scheduled forward or backward based on the prediction of the future process states. The operating parameter tuning approach, such as Onogi *et al.* (1986), Hvala *et al.* (1994), dynamically adjusts operating conditions, such as temperature, pressure, and so on, based on the prediction of future process states, to control the processing times of certain tasks, and to minimize the effect of processing time variations. This approach is tested for processes involving intermediate tanks.

Both the time shift approach and operating parameter tuning approach, however, are not effective in case of uncertainties other than processing time variations. The time shift approach, for example, is ineffective against a demand change which requires a schedule modification including a change of production sequence. In addition, application of the operating parameter tuning approach is limited within the constraints of product quality and process operating conditions in most batch processes. Therefore a more comprehensive approach for on-line scheduling systems to deal with a wide variety of uncertainties effectively is required.

Thus, previous researches on on-line scheduling systems in batch processes propose no comprehensive system structure to deal with a wide variety of uncertainties in relation to batch process management.

Scheduling systems which work under the dynamic changes of production information have been extensively studied in discrete part manufacturing systems like an FMS. For example, Yamamoto and Nof (1985), Wu and Wysk (1989), Ishii and Talavage (1991), and so on, propose real-time systems for FMS or job shop type production systems. In addition, Harmonosky and Robohn (1991) provide a comprehensive review on this subject.

The approach employed in these systems can be categorized as real-time task based scheduling which controls manufacturing operations on a task-by-task basis. (In this research, a set of serial operations within a certain process unit is called a task.) In this approach, the production sequence of each task is defined on a real-time basis according to

current process situations. For example, the next job to proceed in a machine is defined with a dispatching rule.

Such real-time task based scheduling decomposes a scheduling problem into several local problems, and therefore can handle practical size problems in discrete part manufacturing systems. In batch processes, however, real-time task based scheduling is not applicable because of the characteristics of batch processes, such as waiting time constraints between tasks, limited process configuration and intermediate tanks, and so on as described in Chapter 1. For example, it is impossible to satisfy the waiting time constraints between tasks using real-time task based scheduling because the schedules of tasks cannot be synchronized with each other. Therefore establishment of on-line scheduling systems based on batch process characteristics is required.

One of the most primitive structures for an on-line scheduling system in batch processes is to adopt complete re-scheduling based on the latest production information at regular short term intervals. It is obvious, however, this is infeasible in practice because of its cost and the computational times required. Therefore on-line scheduling systems, which can provide an appropriate levels for schedule modification space at appropriate timing according to variations of production information, and then takes corrective actions, are required in batch process management.

The detailed features of the on-line scheduling system highly depends on the characteristics of each batch process. For example, processes having unlimited waiting time constraint or handling stable products require no time shift mechanisms to modify schedules against processing time variations. In addition, processes producing a single product require no sequencing mechanisms. For this reason, it is impossible to apply a single on-line scheduling system for a wide variety of batch processes.

However, the lack of fundamental approach and scheme for guidelines for on-line scheduling systems in batch process management causes a long system development period, inconsistency of the system performance, and high development cost. In this chapter, therefore, a fundamental structure of on-line scheduling system (FOSS: Fundamental structure for On-line Scheduling System) (Ishii and Muraki, 1995a) applicable for a wide

variety of batch processes is proposed as one of the structures of the intelligent scheduling system. The structure specifies essential elements for on-line scheduling in batch processes. The structure is flexible for adapting to the requirements of a specific batch process. By establishing the structure, not only the reduction of system development period for on-line scheduling system for a specific batch process but also a certain quality level of the system are expected.

4.3 Fundamental Structure for On-line Scheduling System (FOSS)

4.3.1 *Scope of the System*

Figure 4.1 shows the system boundaries between the structure for on-line scheduling system (FOSS) and the systems in relation to batch process management. The corporate system provides production planning information and performance criteria for scheduling to FOSS at certain time intervals, for example, every week, except for urgent situations. The process control system provides on-line process data regarding the current situation of batch processes. Based on the production planning information, master data for scheduling, and the on-line process data, FOSS modifies or re-generates a short-term schedule which best suits performance criteria over the scheduling period, and then informs the process control system of the modified schedule, which specifies the following;

- Production sequence on production orders,
- Resource usage of each task,
- Start and complete times of each task.

The process control system executes unit control and sequence control according to the schedule. The corporate system also refers to the schedule as feedback information from FOSS, and then updates production information or performance criteria for scheduling, if the schedule provided by FOSS is unacceptable according to the corporate policy.

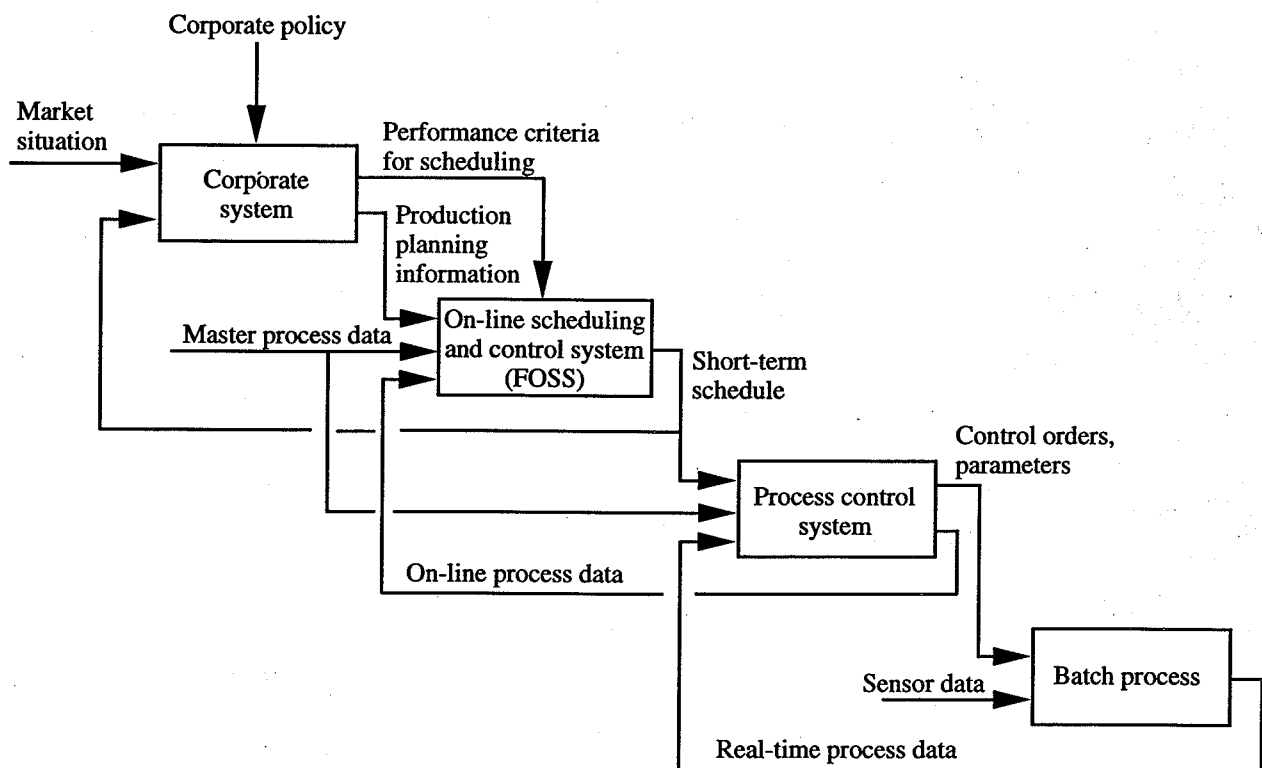


Figure 4.1 Boundaries of on-line scheduling system

Figure 1.5 in Chapter 1 describes data associated with FOSS in detail. The data in Figure 1.5 have quite different degrees of variability. Master process data, for example, are usually unchanged at least during a scheduling period. On the other hand, on-line process data are dynamically updated during production. FOSS takes corrective actions toward the current execution schedule (working schedule) during production to adapt to the changes of the widely variant data shown in Figure 1.5, and then improve performance according to the criteria given by the corporate system.

4.3.2 *Concept of FOSS*

FOSS, which provides a fundamental approach and scheme of on-line scheduling system as one of the structures of the intelligent scheduling system in batch process management, is constructed based on the following concepts.

- Provide essential mechanisms for on-line scheduling systems
- Utilize existing and emerging scheduling algorithms for customizing the system depending on the specific requirements of each batch process

The following describes the above concepts in detail.

(1) *Essential mechanisms*

The detailed features of on-line scheduling system will depend on the specific batch process. However, there are key elements in the structure for on-line scheduling systems which are commonly applicable for most batch processes. FOSS employs the following mechanisms to provide the elements required as a structure of on-line scheduling systems.

(a) *Definition of schedule modification timing using discrete-event simulation*

FOSS employs an event-driven approach to define schedule modification timing. This approach specifies the schedule modification timing according to the changes of production information, such as production orders, equipment status, inventory, and so on. The event-driven approach can provide high quality schedules with fewer

modification times than a periodic approach which modifies schedules at short time intervals (Church and Uzsoy, 1992). However, every event associated with changes of production information does not always affect the performance of a schedule. For example, change of utility availability would not affect a schedule if the utility is not critical in that schedule. To define adequate schedule modification timing, FOSS predicts disruptions or performance degradations of a schedule in the future when any event happens by discrete-event simulation. This mechanism reduces unnecessary schedule modifications and also the space of schedule modifications, and therefore requires fewer computational resources than a modification mechanism which considers complete re-scheduling at all the events.

(b) Definition of schedule modification space by schedule modification model

Figure 4.2 shows a schedule modification model consisting of three schedule modification spaces for the on-line scheduling system. The model considers schedule modifications as a problem which modifies production sequence, resource assignment of each product, or task start and completion times so as to optimize performance criteria under the current production information. In the model, schedule modifications in the larger space include modifications in the smaller spaces. For example, a schedule modification in the production sequence space (SP3) considers resource assignment (SP2), such as units, utilities, operators, and so on, and start and completion times of each task (SP1) together with the production sequence. On the other hand, schedule modifications in the task start and completion times space (SP1) modify start and completion times of each task under a fixed production sequence and resource assignment.

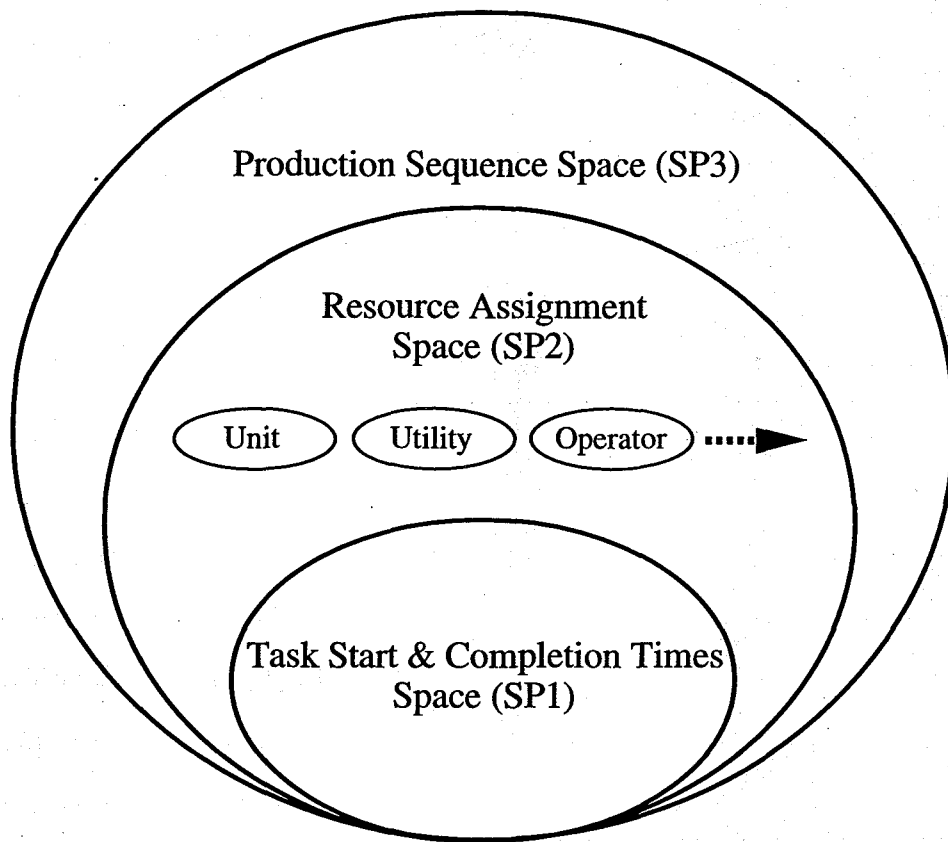


Figure 4.2 A schedule modification model
for on-line scheduling system

FOSS provides a mechanism which defines a suitable schedule modification space based on a schedule modification model according to the disruptions or performance degradations predicted by discrete-event simulation activated based on the changes of production information shown in Figure 1.5. For example, if FOSS predicts a disruption of task completion time due to a processing time variation, FOSS defines the schedule modification in the task start and completion times space (SP1).

A schedule modification in a modification space could fail in some cases because of constraints specified by the larger spaces in advance. In this case, FOSS takes a bottom-up modification approach which extends the schedule modification space in a step-by-step manner to the larger modification spaces in the model, to release the constraints.

This mechanism avoids unnecessary schedule modifications, such as complete re-scheduling at every modification, and therefore reduces computer resources and schedule modification time required.

(2) *Utilization of existing and emerging scheduling algorithms*

A large number of scheduling algorithms have been proposed both in research and practice. These algorithms, however, are developed either for a specific or abstracted batch process, because it is impossible to develop a single scheduling algorithm which can handle practical problems in a wide variety of process types.

Therefore on-line scheduling systems require a mechanism which dynamically selects a suitable algorithm from a library of scheduling algorithms according to the schedule modification space required. By establishing the mechanism, for example, heuristic algorithms, such as Kudva *et al.* (1994), Kuriyan and Reklaitis (1989), Ishii and Muraki (1995c), and so on, can be utilized for schedule modifications in the production sequence space. A flexible simulation algorithm proposed by Hasebe *et al.* (1992) can also be utilized in the resource assignment space.

This mechanism is effective for customizing an on-line scheduling system to the specific requirements of each batch process because establishment of a dedicated system by

selecting a set of suitable algorithms for each process is available. Evolution of the system will also be possible by employing emerging algorithms when they are developed.

4.3.3 *Overview of the Structure*

Figure 4.3 depicts a general structure of FOSS, consisting of three major modules, i.e., variability check module (VCM), action strategy generation module (ASGM), and corrective action module (CAM), to realize the above concepts of on-line scheduling systems in batch process management. VCM corresponds to Functions 1 and 2, and ASGM and CAM correspond to Function 3, within the framework for the intelligent scheduling system shown in Figure 1.6.

VCM is activated when production information; master process data, production planning information, performance criteria for scheduling, or on-line process data, are updated, and predicts scheduling disruptions or performance degradations in the future by simulation. After the simulation, VCM defines schedule modification timing, which is a trigger to activate ASGM, and a schedule modification space. ASGM selects the best suited action strategy among the scheduling algorithms prepared in FOSS to remove disruptions or performance degradations predicted in VCM. CAM modifies the schedule using an action strategy selected in ASGM, and then informs VCM of the modified schedule. VCM checks the feasibility of the modified schedule, and activates ASGM if the schedule is infeasible. Otherwise, VCM registers the modified schedule as the new working schedule which is referred to by the process control system.

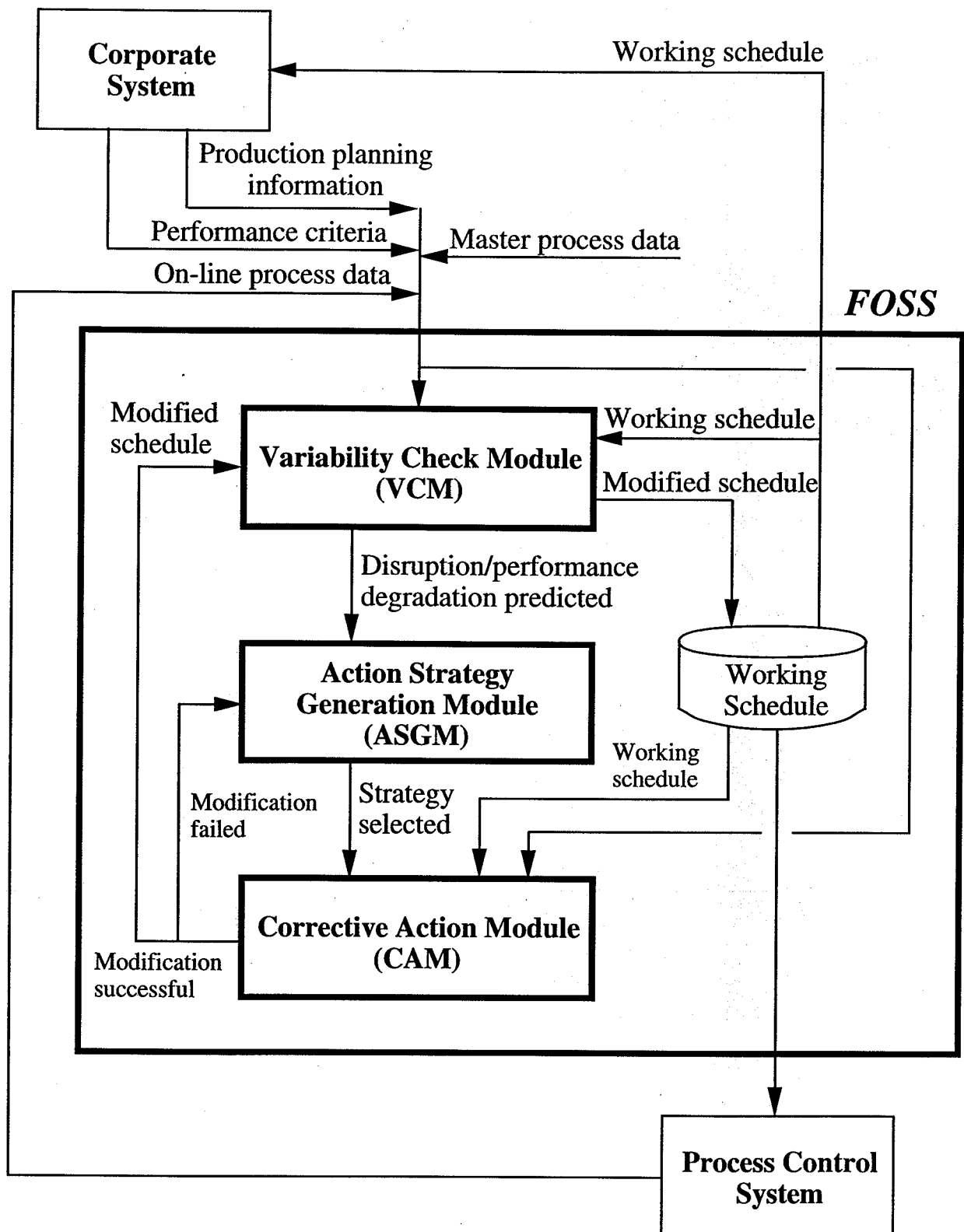


Figure 4.3 Overview of FOSS

4.3.4 Modules and Functions for FOSS

(1) Variability Check Module (VCM)

VCM consists of two mechanisms, i.e., main VCM and simulation mechanism, as shown in Figure 4.4.

(a) Main VCM mechanism

The main VCM mechanism (MVM) has three major functions, i.e., supervision of production information, activation of simulation mechanism, and analysis of simulation results.

MVM monitors changes of production information, such as on-line process data, production planning information, and so on. If MVM detects a change of production plan in production planning information, performance criteria for scheduling, or master process data, MVM immediately transfers the information to ASGM, and activates it. When MVM, on the other hand, detects any changes of on-line process data, it prepares data for a simulation, such as on-line process data, master process data, and the current working schedule. After the preparation of the simulation data, MVM transfers the data to the simulation mechanism, and activates the mechanism.

MVM analyzes the simulation results from the points of feasibility and performance of the simulated schedule. There are two reasons for checking the feasibility of a schedule by simulation. The first reason is to specify schedule modification timing by testing the feasibility of the current working schedule under the latest scheduling conditions, such as availability of raw materials, utilities, processing units, and so on. The second reason is to assure the feasibility of the schedule modified in CAM. As given in the concepts of FOSS, it modifies a schedule in a limited modification space. Algorithms in the smaller schedule modification space may not involve the constraints associated with the larger schedule modification spaces. For example, a simple time shift algorithm in the smallest space may not consider the availability of utilities, and therefore could violate utility constraints. For this reason, FOSS provides a simulation capability to check the feasibility of modified schedules in CAM. In addition, performance of a schedule is

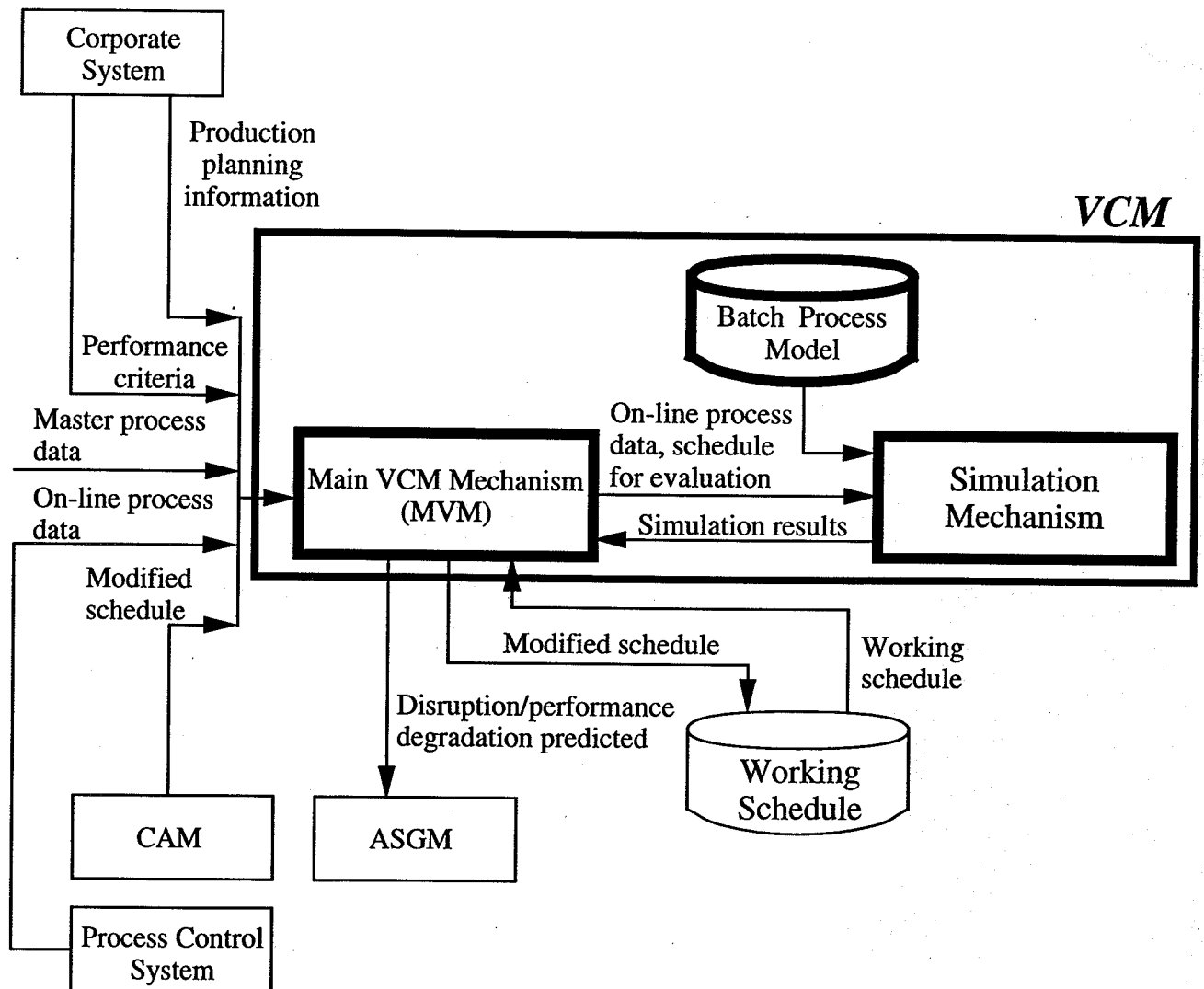


Figure 4.4 Overview of variability check module (VCM)

considered in VCM because changes of production information could increase idle times or cause performance degradations according to performance criteria.

If a simulation predicts disruptions or performance degradations in a schedule, MVM transfers analyzed data, which specify the disruptions or performance degradations, to ASGM, and activates ASGM to select a strategy for a schedule modification. Otherwise, MVM registers the schedule as the new working schedule which is used in the process control system.

(b) Simulation mechanism

The simulation mechanism is one of the key elements of FOSS. The major function of the simulation mechanism is to predict disruptions or performance degradations of a schedule in the future by discrete-event simulation. Discrete-event simulation is a powerful technique to evaluate or analyze the dynamic behavior of complex production systems (Pritsker, 1986). In addition, discrete-event simulation has been used in many ways within on-line scheduling systems, such as Davis and Jones (1988), Rogers and Flanagan (1991), Kim and Kim (1994), and so on.

In FOSS, the simulation mechanism uses deterministic simulation which simulates the effects of scheduled events based on the latest scheduling information. FOSS handles the unforeseeable events, such as sudden equipment breakdowns, sudden changes of resource availability, and so on, by responding VCM to any events, and activates the simulation mechanism based on the information on the events detected.

(2) Action Strategy Generation Module (ASGM)

ASGM specifies a schedule modification space, and selects a schedule modification strategy in response to the disruptions or performance degradations predicted in VCM. ASGM also selects a strategy based on a bottom-up modification approach, if a scheduling modification in CAM due to the strategy selected in ASGM is infeasible.

The selection of schedule modification strategy is essentially a decision making problem based on knowledge of on-line scheduling. There are a large number of potential

technologies to establish ASGM. Expert systems (Pedersen, 1989), for example, have been tested in industries on decision making problems which require human expertise to solve the problems. Wu and Wysk (1988) propose a Multi-pass Expert Control System which applies expert system technology and discrete-event simulation in FMS scheduling. In their system, an expert system generates potential scheduling alternatives based on real-time shop information and scheduling knowledge, and a simulation system evaluates the alternatives based on the performance. Neural network (Rumelhart and McClelland, 1986) which simulates the human brain's activities on a digital computer is also a potential technology applicable for ASGM. Cho and Wysk (1993), for example, apply a neural network technology in FMS scheduling. In their system, a neural network is used to dynamically select a set of promising dispatching strategies, which guides the future part movements in an FMS, based on shop information.

Both expert system and neural network are high potential technologies for ASGM. However, implementation of the technologies as a fundamental mechanism in FOSS is difficult at the present time because of their complexity and inflexibility. Further researches are required to establish a fundamental technology for decision making problems within on-line scheduling systems.

In this chapter, therefore, a simple approach for ASGM is employed to test the effectiveness of FOSS as described in Section 4.4. The approach uses two decision tables, i.e., space decision table and strategy decision table, to realize the function of ASGM which specifies a schedule modification space based on an on-line scheduling model and information on the disruptions or performance degradations in a schedule predicted in VCM.

(3) *Corrective Action Module (CAM)*

CAM consists of a main control component and a library of schedule modification strategies. The main control component prepares data for schedule modification using the strategy specified in ASGM, and activates the modification strategy in the library for a schedule modification. If the schedule modification is successful, the control component transfers the modified schedule to VCM as a candidate schedule to be executed in the process

control system. If the schedule modification is unsuccessful, the control component activates ASGM to obtain another schedule modification strategy within the wider schedule modification space.

4.4 Experiment for Evaluation

4.4.1 Example Problem for Experiment

(1) Overview of the problem

(a) General description of the problem

The example plant, as depicted in Figure 4.5, consists of two batch plants and storage facilities; warehouse and intermediate tanks. Raw materials received are stored in storage facilities, and delivered to each batch plant upon request for production. The products produced in the plants are either final products or intermediate products. The final products will be shipped to clients from storage facilities based on a shipping order issued by a corporate system. The intermediate products, on the other hand, will be stored in storage facilities, and used as raw materials. Therefore there is interdependency among products and intermediate products within this plant. The plant produces three final products and two intermediate products as follows;

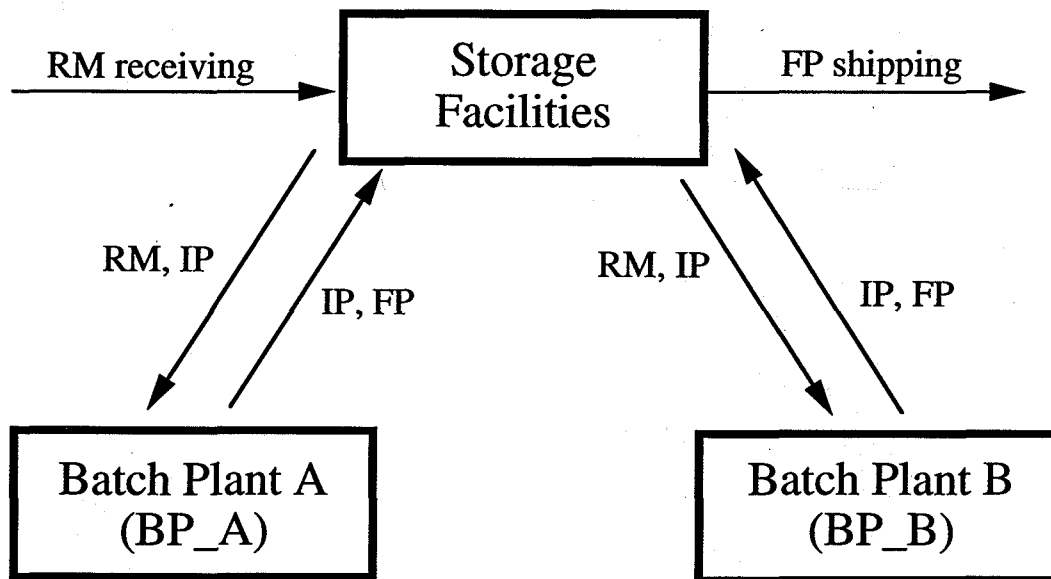
- Final products: FP1, FP2, FP3,
- Intermediate products: IP1, IP2.

(b) Equipment configuration and batch plant specifications

The batch plant A (BP_A) consists of two reactors, one filler, and one dryer, as depicted in Figure 4.6 (a). The batch plant B (BP_B) consists of two reactors, one filler, and two dryers, as depicted in Figure 4.6 (b).

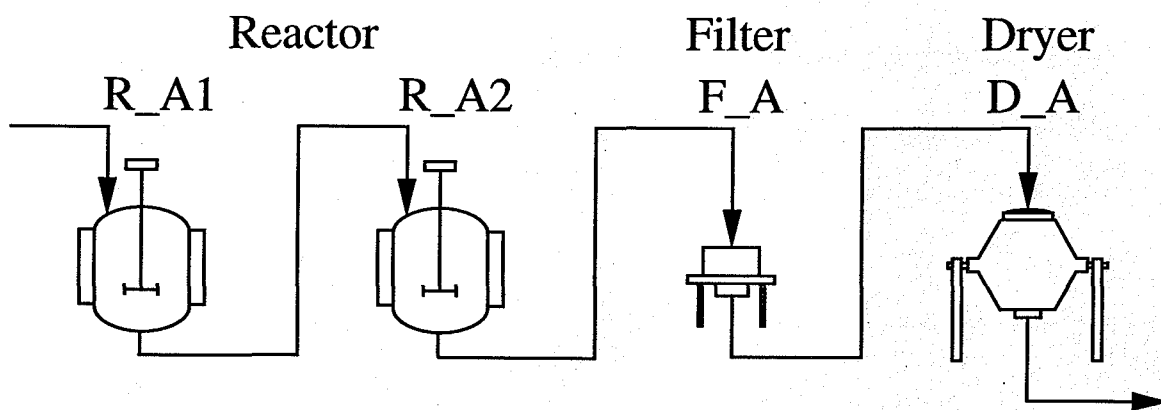
Each batch plant can produce limited products as specified in the following list;

- Product to be produced in BP_A: FP3, IP1, IP2,
- Product to be produced in BP_B: FP1, FP2.

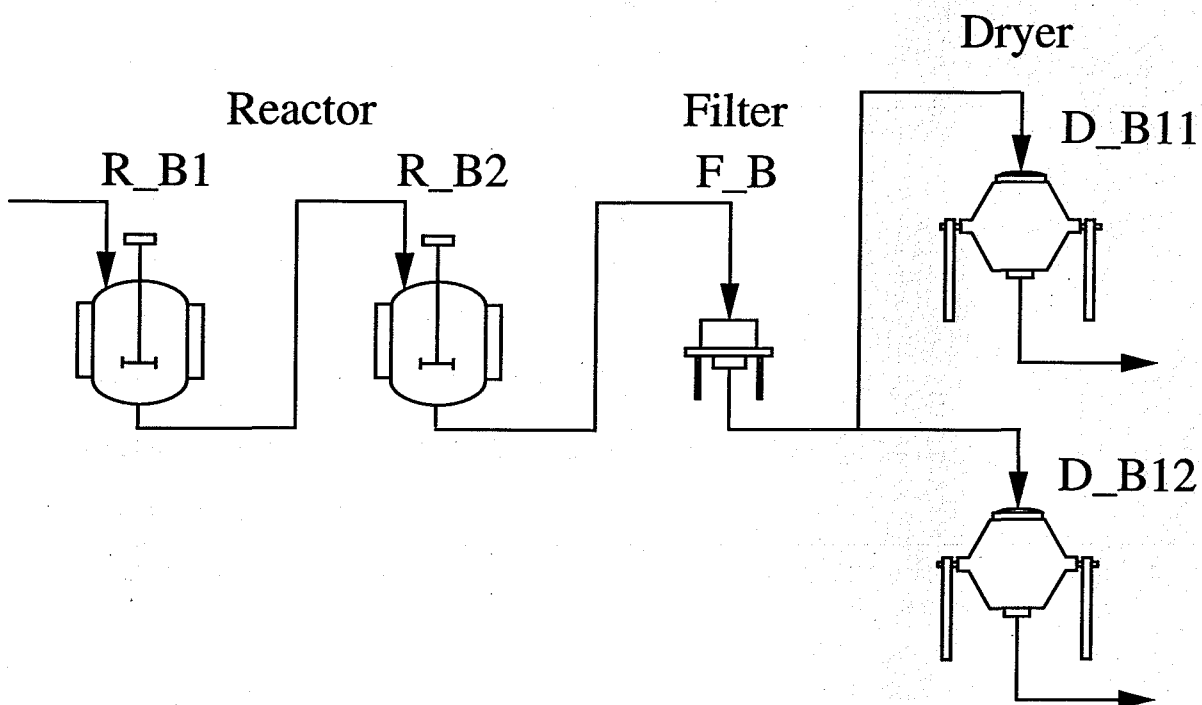


RM: Raw Materials, IP: Intermediate Products, FP: Final Products

Figure 4.5 Material flow in example plant



(a) Batch plant A (BP_A)



(b) Batch plant B (BP_B)

Figure 4.6 Equipment configuration

(c) Resource and operating conditions

BP_A and BP_B can share eight operators in 24 hours in this plant. A task cannot proceed if the number of operators specified in the recipe to execute the tasks is not available. Two dryers are equipped in parallel configuration which allow parallel dry processing in BP_B. Other resources, such as steam, electricity, and so on, are non-critical resources for production activities in this plant.

The plant is operated 24 hours a day, 7 days a week. Raw materials and intermediate products required for production are transported from storage facilities to each batch plant in advance, if they are stored.

Mixed campaign production which can produce different products in a sequence is available, however, changeover time conditions depending on production sequence is specified as shown in Table 4.1.

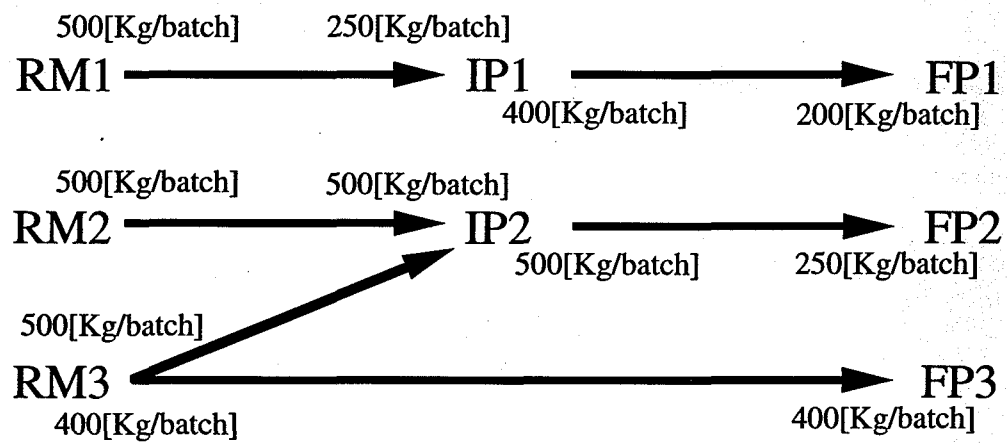
(d) Scheduling practice

Every five days, a corporate system generates new production planning information including a production plan and performance criteria for scheduling over the next ten days based on the sales plan and inventory information. The production plan involves product items, the number of batches of each product, and due date. FOSS generates or modifies a schedule to satisfy the new production plan and the performance criteria for the next ten days when the corporate system updates the information.

(2) Process data

(a) Product interdependency

Figure 4.7 shows product interdependency and stoichiometric relations per batch in this experiment. Because of the product interdependency, in which intermediate products are used as raw materials for other products produced within this plant, a production sequence must be defined in consideration of the future inventory level of each intermediate product.



[]: Raw material consumption or amount of product per batch.

Figure 4.7 Product interdependency and mass balance per batch

(b) Process time chart (recipe)

In this experiment, multiproduct processes as shown in Figures 4.8 (a) and 4.8 (b) which depict process time charts of each intermediate product and final product, respectively, are employed. The multiproduct process produces several products with similar recipes in a fixed equipment network.

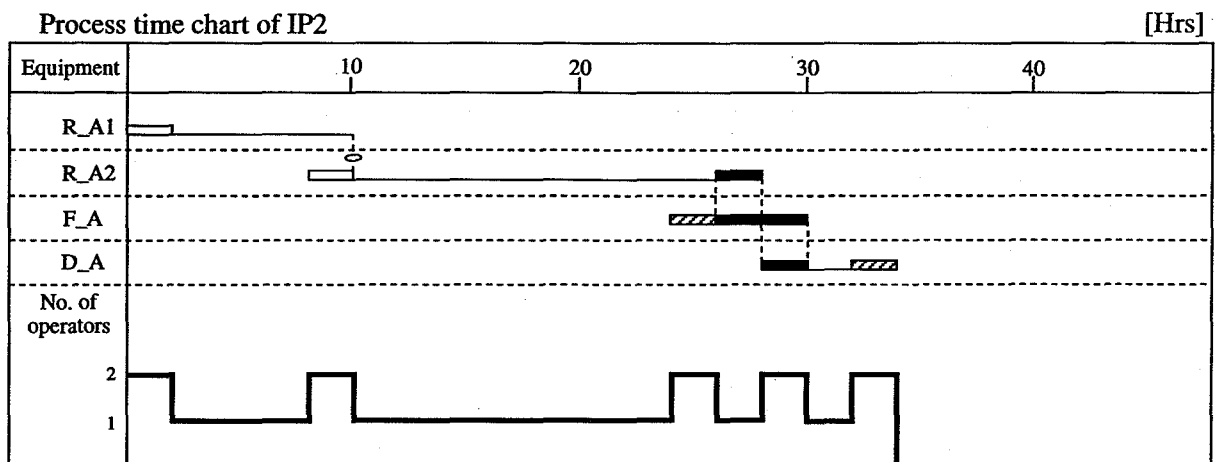
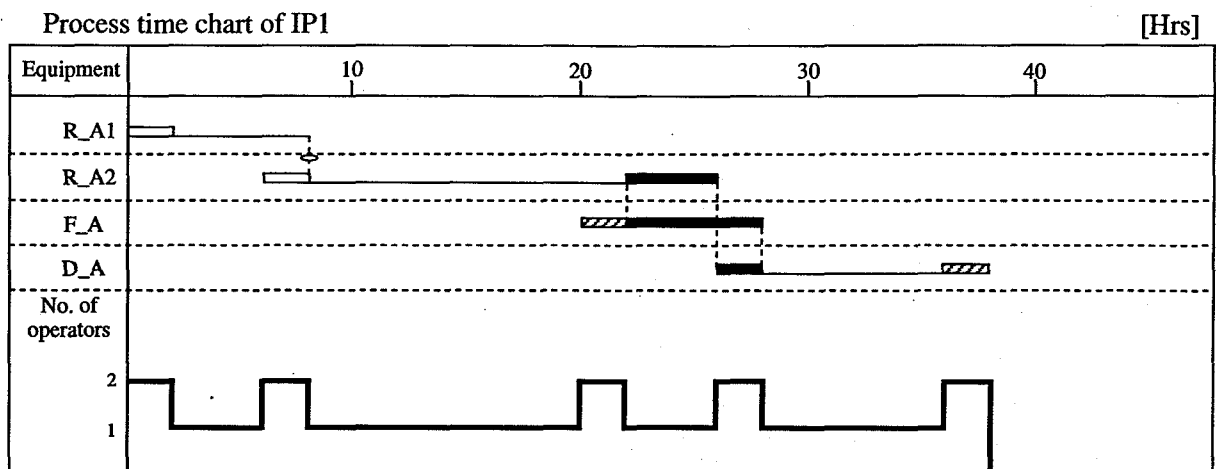
The time chart describes a task sequence, equipment requirement and average processing time of each task, material transfer conditions between tasks, and the number of operators required for each task. Material transfer between tasks is divided into ZW (Zero Wait) condition or UW (Unlimited Wait) condition. If the material is unstable, ZW condition where the material must be immediately processed in the succeeding task, is specified. On the other hand, if the material is stable, UW condition where the material is allowed to be held unconditionally is specified.

(c) Processing time variability

Each task has processing time variations within +/- one hour from average processing time. The processing time variability could affect ZW conditions on the material transfer between tasks.

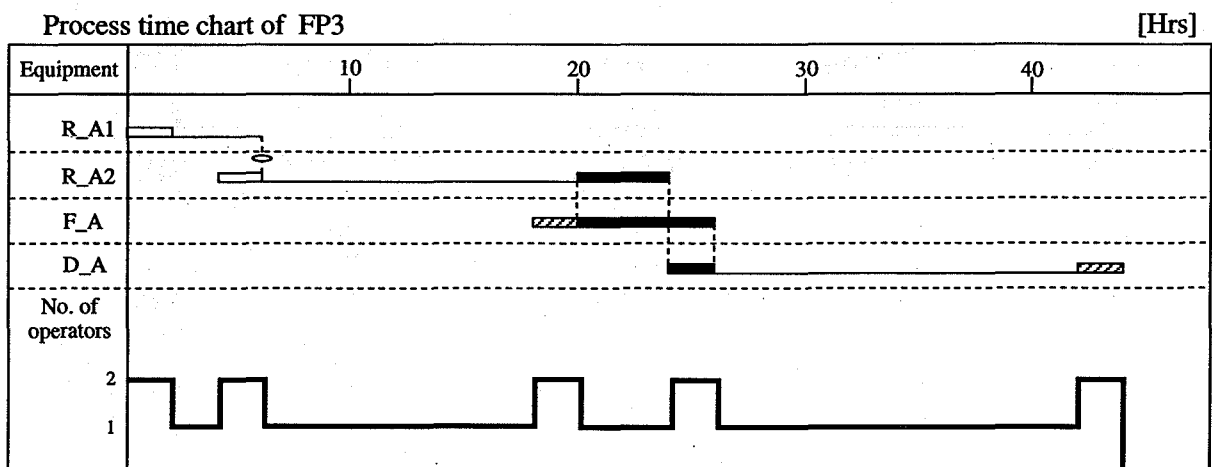
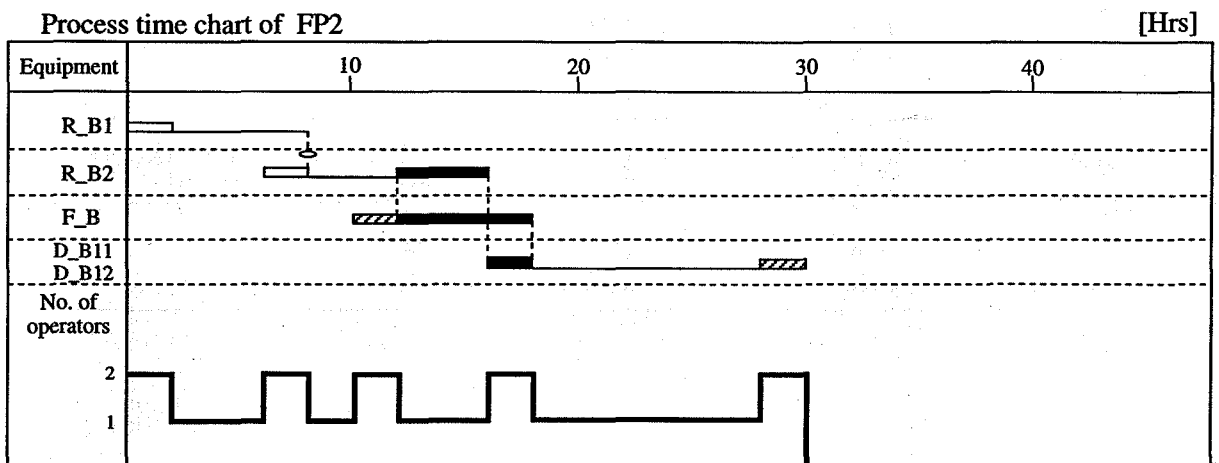
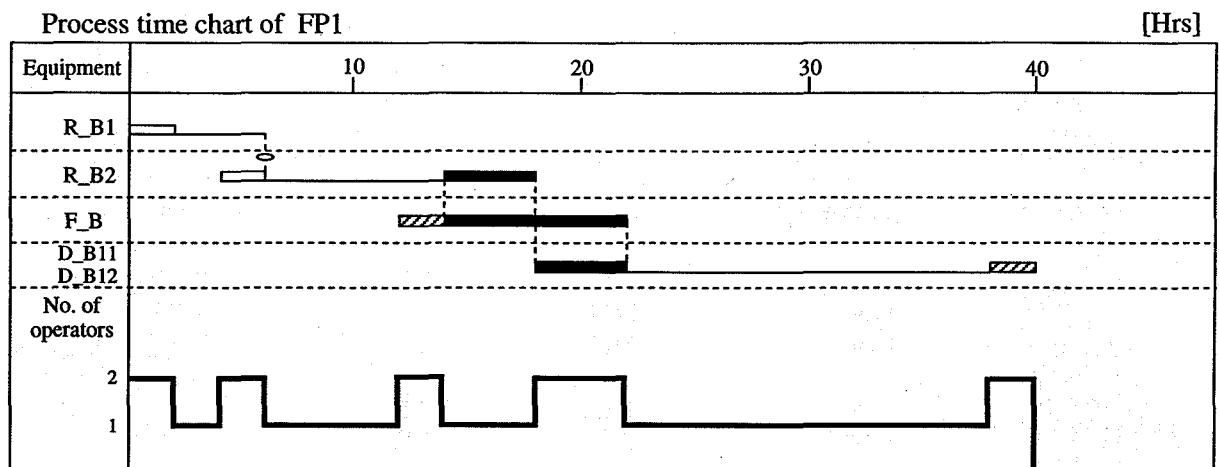
(3) Initial conditions

The experiment starts at 9:00 AM, September 1st by receiving new production planning information and performance criteria for scheduling. At that time, there are no executing or not-yet-executed products in either BP_A or BP_B. The inventory level of raw materials and intermediate products on September 1st is shown in Table 4.2.



[Feeding bar] : Feeding [Processing bar] : Processing [Material transfer with ZW condition symbol] : Material transfer with ZW condition
 [Set-up bar] : Set-up [Material transfer with UW condition bar] : Material transfer with UW condition

Figure 4.8 (a). Process time chart of IP1 and IP2



□ : Feeding — : Processing ⊕ : Material transfer with ZW condition
 ▨ : Set-up ■ : Material transfer with UW condition

Figure 4.8 (b). Process time chart of FP1, FP2, and FP3

Table 4.1 Changeover times between products

Plant	From	To	Time [Hr.]
Batch Plant A (BP_A)	IP1	IP1	2.0
	IP1	IP2	4.0
	IP1	FP3	4.0
	IP2	IP1	4.0
	IP2	IP2	2.0
	IP2	FP3	4.0
	FP3	IP1	4.0
	FP3	IP2	4.0
	FP3	FP3	2.0
Batch Plant B (BP_B)	FP1	FP1	2.0
	FP1	FP2	4.0
	FP2	FP1	4.0
	FP2	FP2	2.0

Table 4.2 Initial inventory level

Material	Inventory [Kg]
RM1	1200
RM2	400
RM3	600
IP1	1000
IP2	700

(4) *Production planning information and performance criteria*

On September 1st, FOSS receives production planning information and performance criteria for scheduling up to September 10th from the corporate system. The production plan shown in Table 4.3 requests the production of total twelve batches in BP_A and BP_B. (The order number in Table 4.3, such as OD1, OD2, will be referred to in the following section.) Performance criterion for scheduling is specified as total tardiness. (total tardiness = [sum of tardiness from due dates]) FOSS also receives raw materials receiving plans and a maintenance schedule of equipment as follows.

- Raw materials receiving plan:

- RM1: 1000Kg at 11:00, September 8th,
- RM2: 1000Kg at 11:00, September 3rd,
- RM3: 1500Kg at 11:00, September 2nd.

- Maintenance plan:

- Shutdown maintenance of DB_11 from 21:00, September 2nd to 9:00, September 3rd.

Table 4.3 Production plan received on September 1st

Product		Order No.	No. of Batch (Batch size [Kg/batch])	Due date
BP_A	IP1	OD1	1 (250)	SEP. 2nd
		OD2	1 (250)	SEP. 5th
	IP2	OD3	1 (300)	SEP. 4th
		OD4	1 (300)	SEP. 6th
	FP3	OD5	1 (400)	SEP. 3rd
		OD6	1 (400)	SEP. 8th
BP_B	FP1	OD7	1 (200)	SEP. 4th
		OD8	1 (200)	SEP. 4th
		OD9	1 (200)	SEP. 10th
	FP2	OD10	1 (250)	SEP. 3rd
		OD11	1 (250)	SEP. 6th
		OD12	1 (250)	SEP. 10th

(5) *Schedule modification strategies in FOSS for the experiment*

(a) *Strategy for production sequence space (SP3)*

The schedule modification strategy in SP3 must generate a complete schedule based on production information and performance criteria. In this experiment, EDR (Extended Dispatching Rule) search algorithm (Ishii and Muraki, 1994c) is employed for this schedule modification space. It generates promising dispatching rule combinations, and finds a near optimal schedule according to performance criteria by using a dispatching-simulation mechanism as described in Chapter 5. The mechanism generates a schedule by selecting products by a dispatching rule combination among candidate products to be scheduled under various scheduling constraints with a simulation mechanism. The algorithm works efficiently enough to be used within an on-line scheduling structure.

EDR search algorithm employs dispatching rules such as SLACK, FIFO, and so on. In this experiment, the following dispatching rules are employed for EDR search algorithm.

SLACK: Select the order which has the minimum slack time. (*Slack time = due date - present time - remaining processing time*)

SRPT: Select the order which has the shortest remaining processing time.

(b) *Strategy for resource assignment space (SP2)*

The schedule modification strategy in SP2 must assign resources for each task under a fixed production sequence. In this experiment, the following simple heuristic algorithm is employed as a strategy in this schedule modification space.

Resource assignment algorithm

Step 1: Suppose n denotes the production sequence number, and set $n = 1$.

Step 2: If n -th product has already assigned units, go to Step 5. Otherwise go to Step 3.

Step 3: Assign unit for each task in n -th product according to the following rules, and then go to Step 4.

- Assign unit for each task according to the task sequence of the n -th product. The unit which will become serviceable first for the task is assigned among a set of units for the task.

- If no unit is available, take the following action;

If the preceding task has UW condition: make the task time shift backward by inserting waiting time after the preceding task until the unit required is available*,

If the preceding task has ZW condition: make the product time shift backward until the unit required is available.

Step 4: If n -th product has already assigned operators, go to Step 6. Otherwise go to Step 5.

Step 5: Assign operators according to the following rules, and then go to Step 6.

- Assign operators required for each task according to the task sequence of the n -th product.

- If the number of operators required is not available for a task during T period, take the following action;

AC1: In preceding products according to the production sequence, find a task whose preceding task has UW condition, and make the task time shift backward by inserting waiting time after the preceding task until the number of operators required is available during T period. If all the preceding tasks have tested, go to AC3. Otherwise, go to AC2.

AC2: Check the feasibility of the time shift. If the time shift is feasible, terminate the step. Otherwise go to AC1.

AC3: Make n -th product time shift backward until the number of operators required is available.

Step 6: Set $n = n + 1$. If n is greater than the number of products in a schedule, terminate the algorithm. Otherwise, go to Step 2.

(c) Strategy for task start and completion times space (SP1)

The example process contains ZW conditions under processing time variations. Therefore PVSS (Ishii and Muraki, 1995b) which can control waiting times between tasks within the ranges specified in a recipe without substantial performance degradation under processing time variations is employed as a scheduling strategy in this schedule modification space. PVSS, as described in Chapter 6, specifies certain slack times between products to control waiting times between tasks under processing time variations based on current process state and possible future processing time variations, and then dynamically modifies the start and completion times of not-yet-finished-tasks to insert the slack times specified between products.

(6) Decision table approach in ASGM

The approach employed in this experiment for ASGM selects a schedule modification strategy via two steps corresponding to two kinds of decision tables as shown in Figure 4.9. The first step selects a scheduling modification space in an on-line scheduling model using the space decision table based on disruptions or degradations of a schedule identified in VCM. Matches are made between items in the left column in the space decision table and the variations or disruptions identified in VCM, and the corresponding spaces specified in the right column are identified as a set of potential schedule modification spaces. If VCM identifies more than two spaces, it selects the highest space among those selected for the next step.

The second step selects a schedule modification strategy which matches the schedule modification space identified in the first step and performance criteria from the strategy decision table. Performance criteria are considered because performance of scheduling strategy usually depends on the performance criteria for scheduling.

The schedule modification strategy selected in ASGM could fail to generate a feasible schedule in CAM. In this case, CAM notifies ASGM of the infeasibility as depicted in Figure 4.9, and ASGM selects another schedule modification strategy for CAM by

specifying a larger schedule modification space than previously specified in the first step, based on the bottom-up approach concept in FOSS. Because the larger schedule modification space can consider more decision variables for scheduling, there is more chance to generate a feasible schedule.

The decision table approach for ASGM described above is a simple and flexible method. It can be easily implemented in on-line scheduling systems. Extension of the approach is also possible by adding items to the tables according to the problem.

Tables 4.4(a) and 4.4(b) show the space decision table and the strategy decision table designed for this experiment. The tables are designed to utilize the algorithms specified for this experiment.

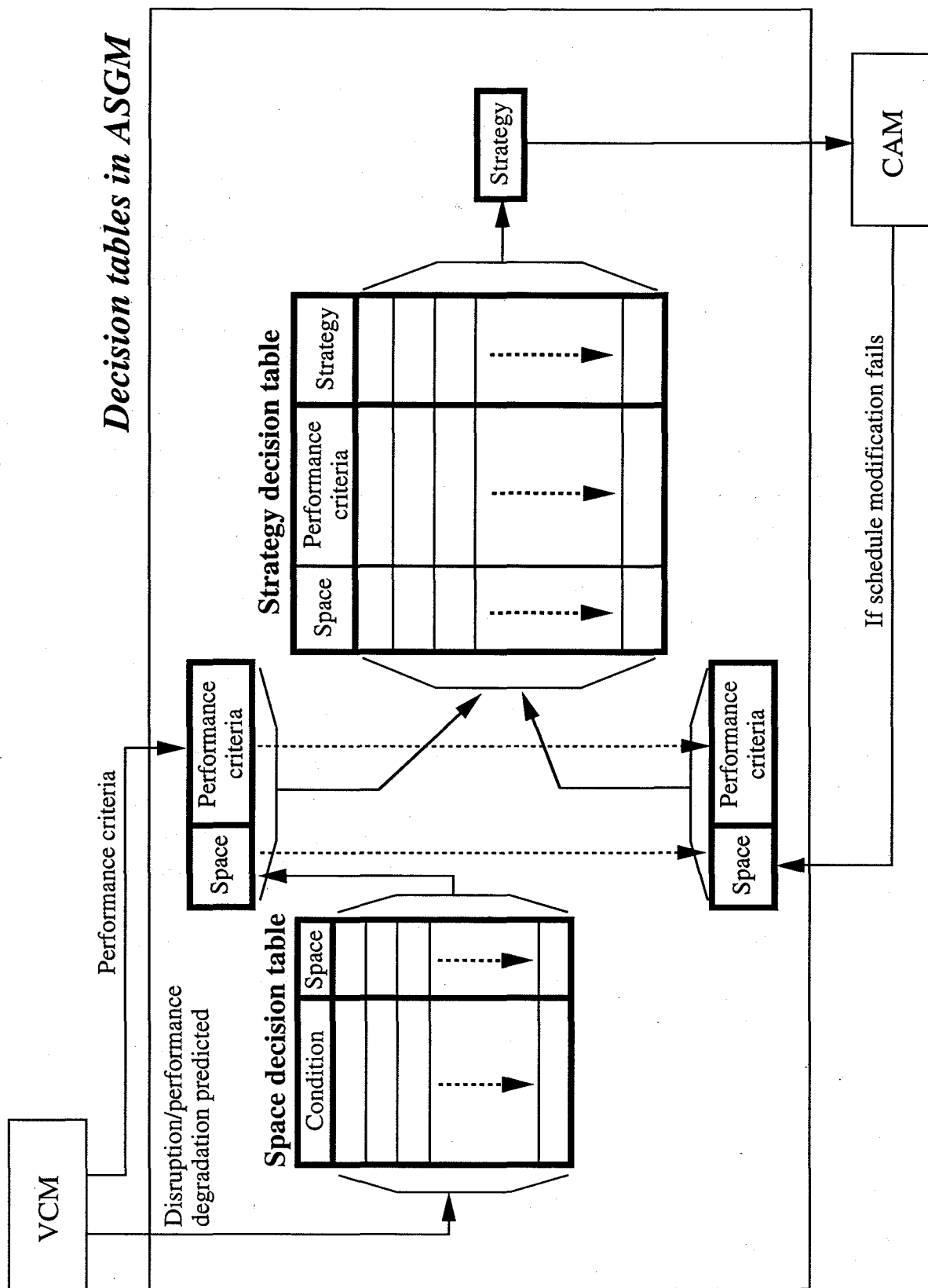


Figure 4.9 Decision table approach: mechanism to select corrective action strategy

Table 4.4(a) A space decision table

Condition	Modification space
Increasing idle times	SP1
Disruption of ZW condition	SP1
Disruption of unit assignment	SP2
Disruption of operator assignment	SP2
Shortage of raw material	SP3
Change of production plan	SP3
Change of performance criteria	SP3
Others	SP3

Table 4.4(b) A strategy decision table

Modification space	Performance criteria	Strategy
SP1	Total tardiness	PVSS
SP2	Total tardiness	Resource assignment algorithm
SP3	Total tardiness	EDR search algorithm

(7) Design of experimental scenarios

The following shows scenarios used in this experiment. The scenarios are designed to demonstrate the mechanism and effectiveness of FOSS in response to typical fluctuations in batch process operations, i.e., processing time variations, changes of unit availability and operator availability, and changes of raw material availability.

Experimental Scenarios

- SC0 (9:00, September 1st): FOSS receives new production planning information shown in Table 4.3 together with mean tardiness as a performance criterion for scheduling evaluation.
- SC1 (6:00, September 2nd): OD1 completes the reaction process in reactor R_A2 at 19:00. This is one hour shorter than the average processing time specified in the time chart.
- SC2 (9:00, September 2nd): The number of operators available between 9:00 September 2nd and 9:00 September 3rd is changed from eight operators to seven operators.
- SC3 (12:00, September 2nd): Maintenance schedule of D_B11 is changed to between 21:00 September 2 and 9:00 September 4.
- SC4: (10:00 September 3rd): The arrival schedule of RM2 is changed to 11:00 September 4th from 11:00, September 3rd.

4.4.2 Results of Experiment

The following describes the reactions and the results of scheduling modifications by FOSS for each experimental scenario.

(1) Schedule generation for SC0

For SC0, MVM in VCM detects the change in production planning information, and informs ASGM of the change. Because the change of production plan matches with the SP3 scheduling modification space in the space decision table, ASGM selects EDR search

algorithm for SP3 according to the strategy decision table in ASGM, and then CAM generates a schedule based on the new production planning information.

Figure 4.10 (SC0) depicts the first four days schedule generated by EDR search algorithm for the scenario SC0. The production sequence in each batch plant specified in SC0 is as follows;

- BP_A: OD1 -> OD5 -> OD3 -> OD2 -> OD4 -> OD6,
- BP_B: OD10 -> OD7 -> OD8 -> OD9 -> OD11 -> OD12.

The schedule generated has two tardy jobs (OD2 and OD4) and total tardiness (performance criterion) is 6 hours.

(2) Schedule modification for SC1

For SC1, MVM in VCM detects the processing time variability at R_A2 in OD1, and therefore activates the simulation mechanism. As a result, the simulation mechanism predicts increased idle time between OD1 and OD5 in BP_A, and MVM informs ASGM of the increase of idle time. Because the increased idle time matches with the SP1 scheduling modification space in the space decision table, ASGM selects PVSS strategy for SP1 according to the strategy decision table in ASGM, and then CAM modifies the schedule by PVSS based on the current on-line process data. CAM transfers the modified schedule to VCM to check the feasibility again by the simulation mechanism. Because the schedule is feasible, MVM registers the modified schedule as the new working schedule for the process control system.

Figure 4.10 (SC1) partially depicts the schedule modified by PVSS under the scenario SC1. As shown in the figure, the task start and completion times in F_A and D_A for OD1, and the schedule for OD5 are shifted forward one hour compared to the original schedule shown in Figure 4.11(SC0). However, OD3 makes no time shift because of the restriction of RM2 inventory level.

In this case, FOSS successfully modifies the schedule by applying a simple time shift algorithm (PVSS) for the variation, and minimizes the effect of processing time variations with less computational effort than complete re-scheduling.

(3) Schedule modification for SC2

For SC2, MVM in VCM detects the change of operator availability in the future, and therefore activates the simulation mechanism. The simulation mechanism predicts disruption of the current working schedule because of the decrease in number of available operators. MVM informs ASGM of the violation, and ASGM selects a resource assignment algorithm in FOSS for a schedule modification according to the decision tables. CAM modifies the schedule by the algorithm based on the revised operator resource availability, and transfers the modified schedule to VCM to check the feasibility of the schedule by the simulation mechanism. Because the schedule is feasible, MVM registers the modified schedule as the new working schedule for the process control system.

Figure 4.10 (SC2) partially depicts the schedule modified by the resource assignment algorithm in CAM under the scenario SC2. To adapt to the maximum available operators between 9:00 September 2nd and 9:00 September 3rd, in this case, FOSS inserts waiting times after processing operation at D_A for OD1 and DB_11 for OD10. As a result, schedule modification by FOSS causes no substantial performance degradation.

PVSS in the SC1 scheduling modification space can also remove the disruption, however, it increases idle time between OD10 and OD7, and will cause an increase of total tardiness in the future. In addition, FOSS successfully modifies a schedule with less computational effort than a complete re-scheduling. This shows that the mechanism, which defines an appropriate scheduling modification space according to the disruptions detected, works successfully under the scenario SC2.

(4) Schedule modification for SC3

For SC3, MVM in VCM detects the change of maintenance schedule which could affect the unit availability in the future, and activates the simulation mechanism. The

simulation mechanism predicts the disruption of the current working schedule because of the infeasible unit assignment of D_B11 for OD8 during the extended maintenance period. MVM informs ASGM of the disruption, and ASGM selects a resource assignment algorithm in the SP2 schedule modification space for schedule modification according to the decision tables. CAM modifies the schedule using an algorithm based on the revised maintenance schedule, and transfers the modified schedule to VCM to check the feasibility by the simulation mechanism. Because the schedule is feasible, MVM registers the modified schedule as the new working schedule for the process control system.

Figure 4.10 (SC3) partially depicts the schedule modified by the resource assignment algorithm in the SP2 schedule modification space. To adapt to the change of the maintenance schedule of D_B11, in this case, FOSS specifies unit re-assignment between D_B11 and D_B12 for OD8, OD9, OD11, and OD12 together with a backward time shift to avoid the unit's usage during the revised maintenance period. In addition, the modified schedule removes the waiting time in D_A for OD1 because of the reduced number of operators required for production by the backward time shift.

In this scenario, FOSS specifies a schedule modification in resource assignment space (SP2), and causes no substantial performance degradation by the modification. On the contrary, if the schedule is modified by time shift only in the SP1 schedule modification space, OD8 suffers seven hours tardiness.

(5) Schedule modification for SC4

For SC4, MVM in VCM detects the change of arrival schedule of RM2, and activates the simulation mechanism. The simulation mechanism predicts the violation of production start time of OD3 because of the shortage of RM2. MVM informs ASGM of the disruption of the production start time for OD3, and ASGM selects EDR search algorithm in the SP3 scheduling modification space according to the decision tables. CAM modifies the schedule using the algorithm based on the revised raw material arrival schedule, and transfers the modified schedule to VCM to check the feasibility of the schedule. Because the schedule is

feasible, MVM registers the modified schedule as the new working schedule for the process control system.

Figure 4.10 (SC4) partially depicts the schedule modified by EDR algorithm in CAM. In this case, FOSS specifies the new production sequence in BP_A by EDR search algorithm in the SP3 schedule modification space as follows;

- OD1 -> OD5 -> OD2 -> OD3 -> OD4 -> OD6.

If the schedule is modified by time shift only (PVSS) in the SP1 schedule modification space instead of EDR search algorithm, in this scenario, it causes 82 hours total tardiness. However, FOSS only allows 25 hours total tardiness by specifying a suitable scheduling modification space (SP3) and a strategy for the modification according to the changes of scheduling conditions.

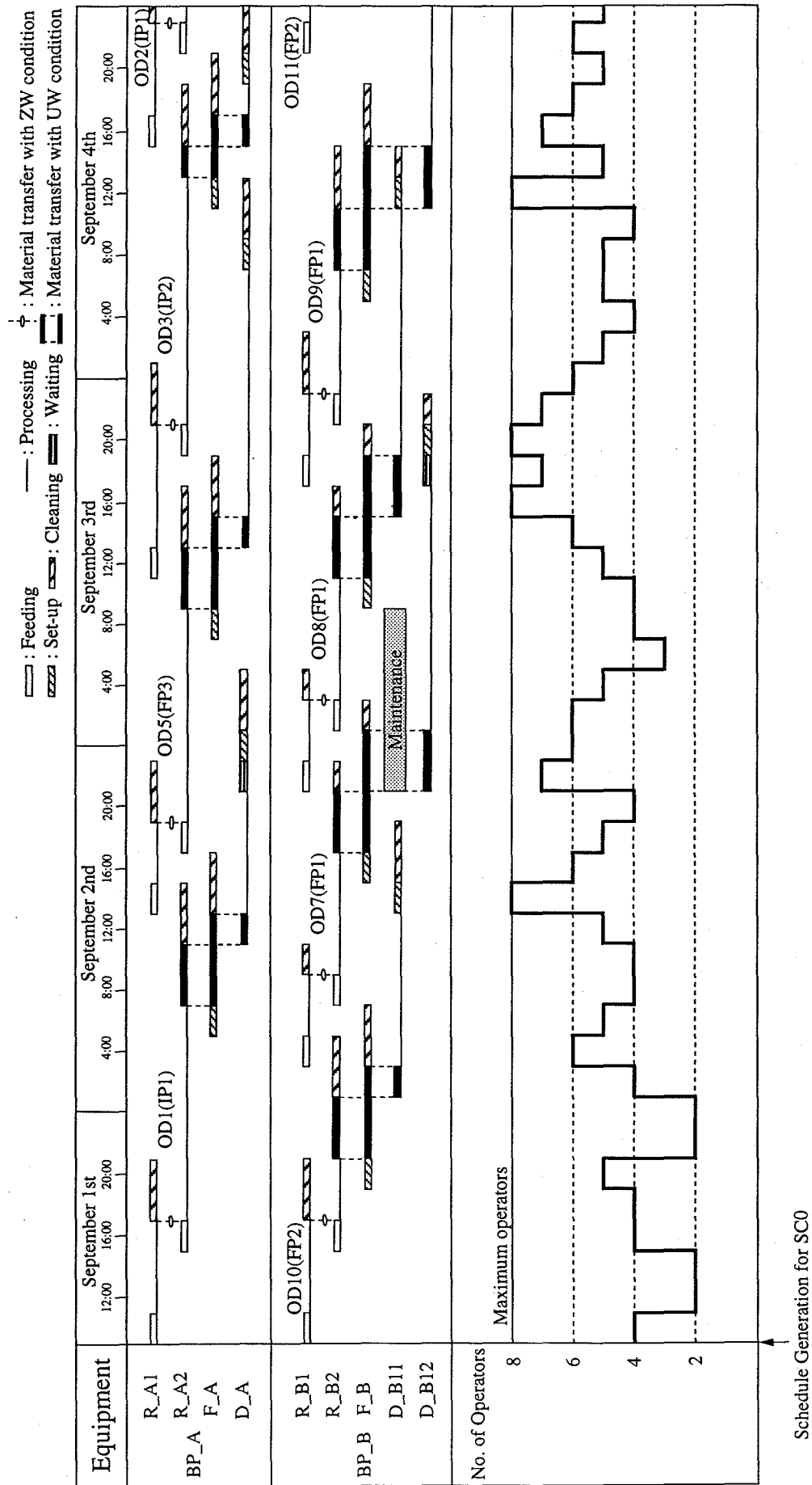


Figure 4.10 (SC0) Original schedule in SC0

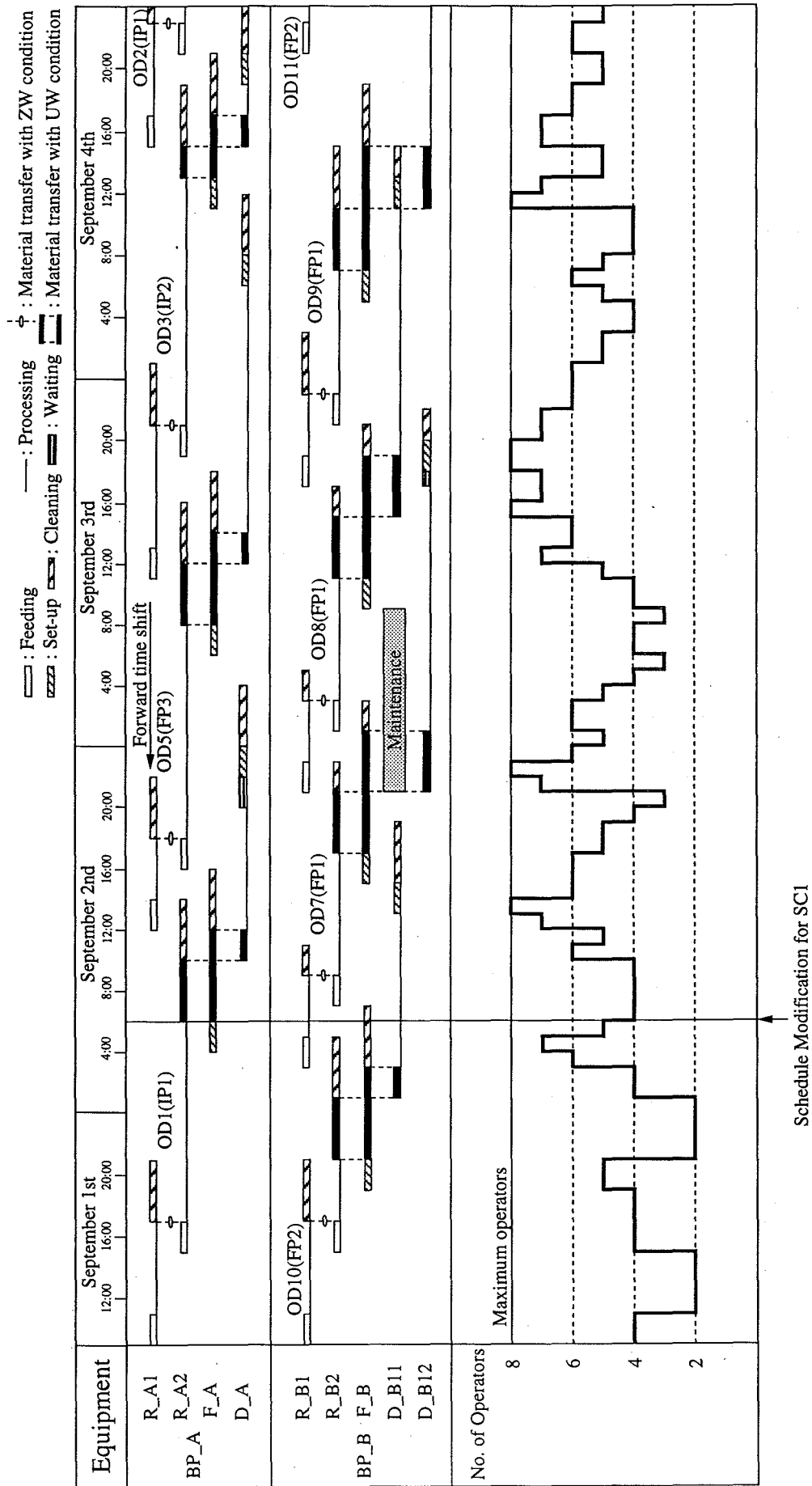


Figure 4.10 (SC1) Modified schedule in SC1

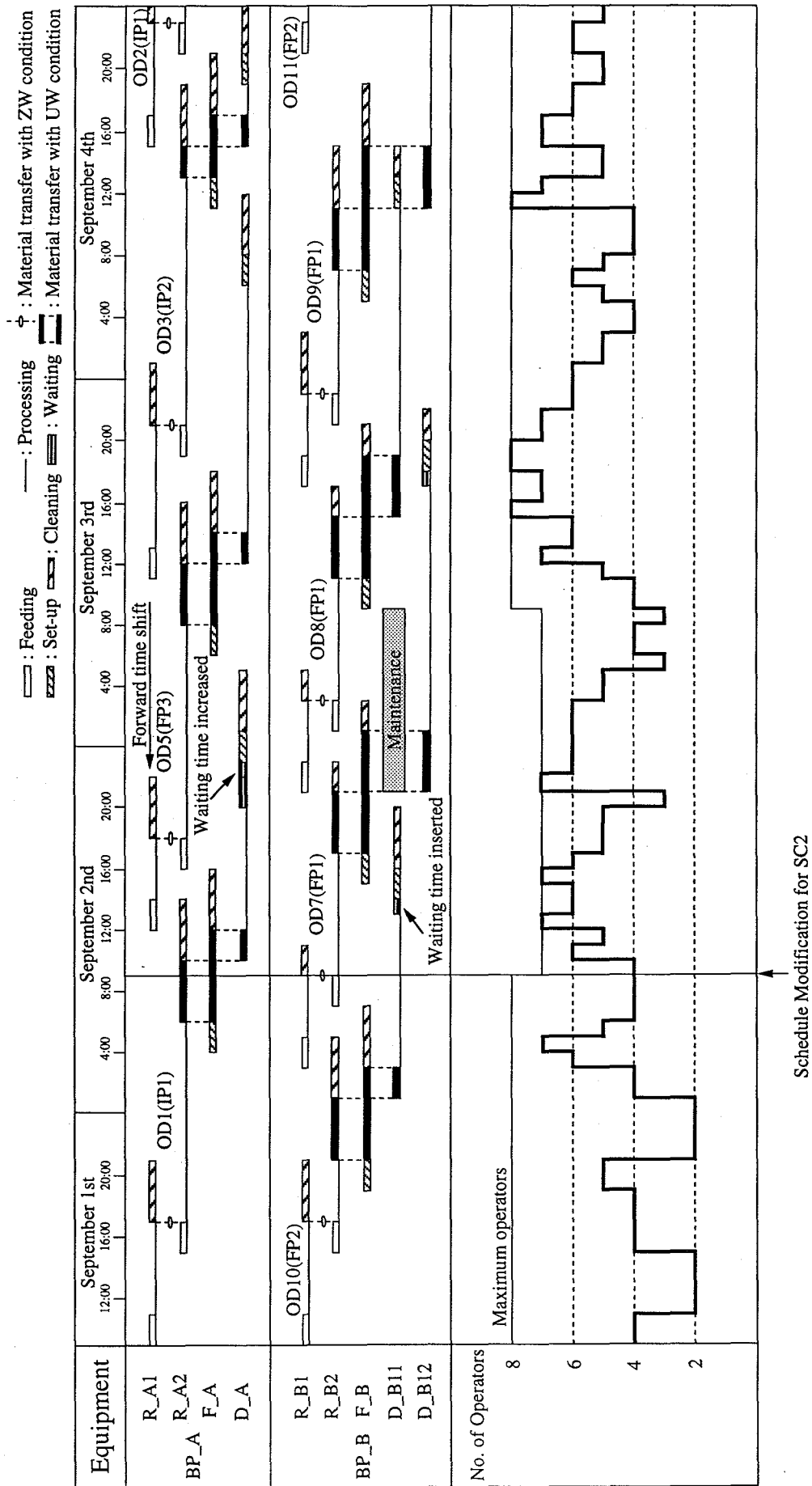


Figure 4.10 (SC2) Modified schedule in SC2

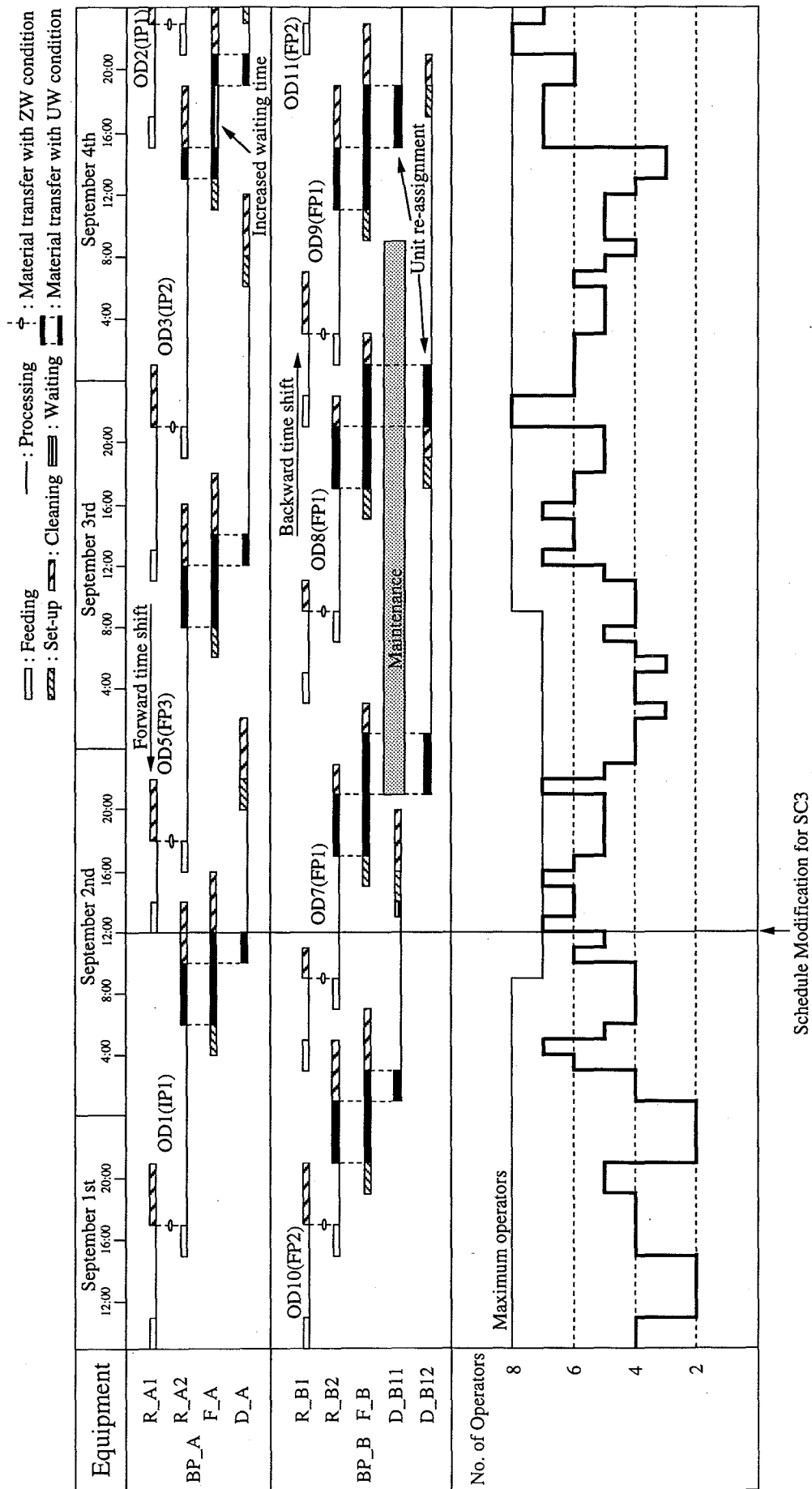


Figure 4.10 (SC3) Modified schedule in SC3

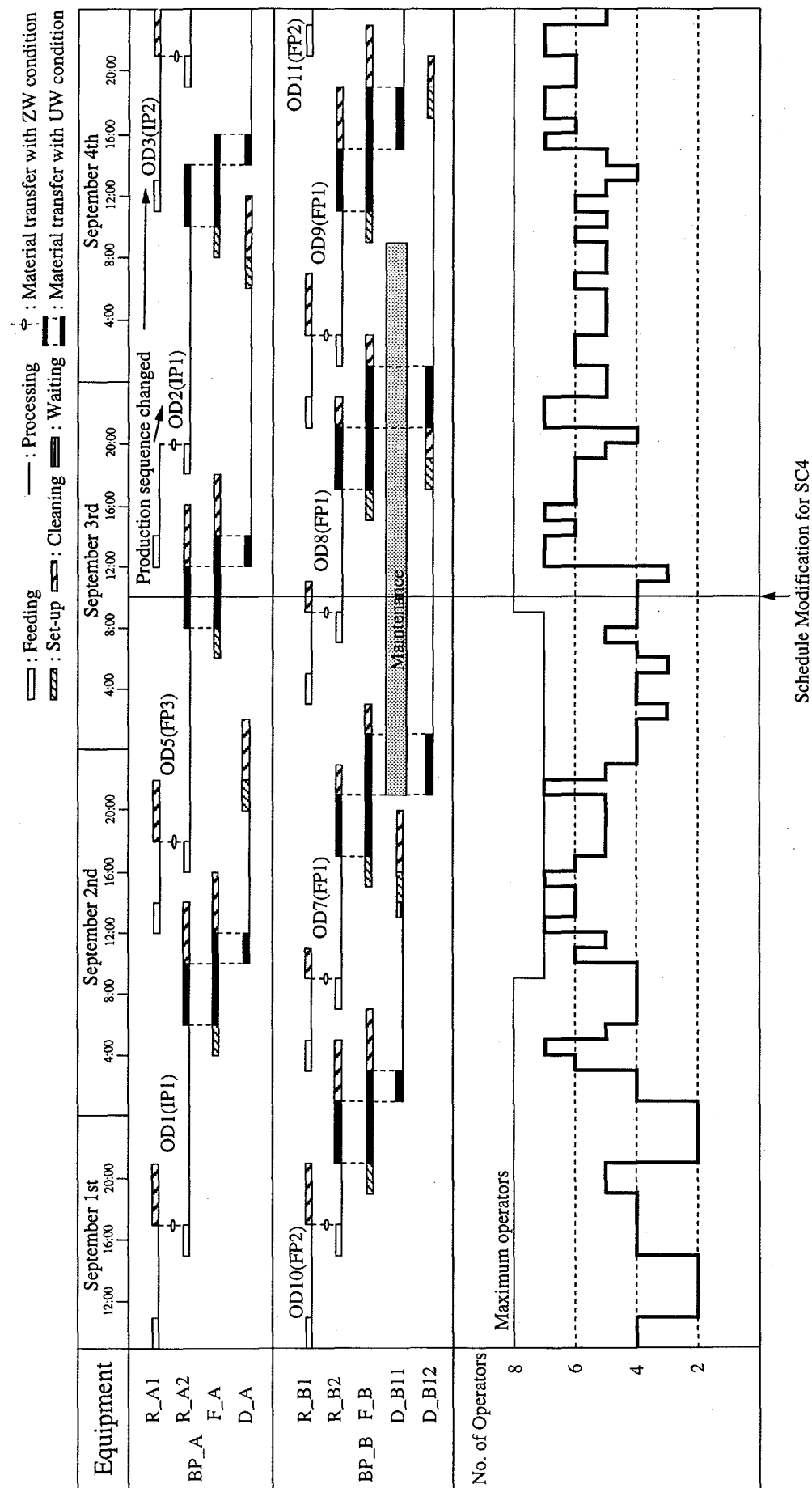


Figure 4.10 (SC4) Modified schedule in SC4

4.5 Chapter Summary

In this chapter, the needs and advantages of a fundamental structure for an on-line scheduling system as one of the structures of the intelligent scheduling system in batch processes is described. In addition, FOSS which provides a fundamental structure and mechanisms for an on-line scheduling system is proposed. The structure and mechanisms in FOSS are presented in detail, and also its performance against typical fluctuations in ordinary batch process operations is demonstrated via experiments with a multiproduct batch plant. The results of the experiments show that FOSS successfully defines schedule modification timing and a modification space for each timing according to changes of a wide variety of production information, and dynamically modifies a schedule to minimize the effect of the changes.

Because FOSS provides a flexible structure which can adapt to the specific requirements of each batch process, FOSS can reduce system development period for a specific batch process with maintaining a certain system quality level. Evolution of FOSS, according to changes of plant configuration or scheduling practice, is also possible by updating the components in FOSS, because of its modular structure.

Chapter 5

AN EXTENDED DISPATCHING RULE APPROACH WITHIN ON-LINE SCHEDULING STRUCTURE IN BATCH PROCESS

5.1 Chapter Overview

In Chapter 4, a fundamental structure of on-line scheduling system (FOSS) is proposed as one of the structures of the intelligent scheduling system in batch processes. The proposed structure consists of three modules, i.e., VCM (Variability Check Module), ASGM (Action Strategy Generation Module), and CAM (Corrective Action Module) as shown in Figure 4.3. VCM detects disruptions or performance degradations based on the latest production information and the current working schedule specified in advance to the process control system as an instruction. ASGM then selects a strategy to reduce the disruptions or increase the performance of the process, and CAM modifies the current working schedule according to the strategy selected in ASGM. FOSS, therefore, requires several schedule modification strategies to react to the wide variety of disruptions or performance degradations during production. One of the critical schedule modification strategies required in FOSS is a re-scheduling strategy which generates a short-term schedule rapidly under various scheduling constraints, such as product interdependency, resource availability, to replace the entire working schedule as the new instruction to the process control system. This strategy corresponds to the schedule modification for the production sequence space (SP3) depicted in Figure 4.2.

In this chapter, a dispatching-simulation mechanism is presented for a re-scheduling strategy within FOSS. Moreover, an extended dispatching rule (EDR) approach, which applies different dispatching rule combinations in the mechanism, and a simple search algorithm to find an appropriate dispatching rule combination with reasonable computational effort, are proposed. The effectiveness of the proposed approach and the search algorithm are demonstrated on an example multiproduct batch process model.

This chapter is organized as follows: Section 5.2 presents the background of the EDR approach. Section 5.3 describes the EDR approach and a search algorithm for the approach in detail. Section 5.4 demonstrates the effectiveness of the EDR approach and the performance of the search algorithm with a simple multiproduct batch process model.

5.2 Background

Rapid generation of a short-term schedule as a new working schedule to the process control system under various scheduling constraints, such as product interdependency, resource availability, and so on, is one of the essential schedule modification mechanisms within FOSS described in Chapter 4. Methodologies for short-term scheduling applicable within FOSS can be categorized into at least two approaches; i.e., mathematical programming approach and heuristic approach.

In the mathematical programming approach, mixed integer linear programming, branch and bound, and so on, are employed to generate a global optimal schedule. Several researchers (Fuchino *et. al.*, 1992, Kondili *et. al.*, 1993, and so on) propose short-term scheduling algorithms in batch processes based on the mathematical programming approach. In addition, Rich and Prokopakis (1986) propose a mixed integer programming approach for the scheduling problems involving product interdependency constraints. The mathematical programming approach provides a global optimum solution, however, it requires huge computational resources because scheduling problems, except a few exceptions, are classified in the NP-class problem from the computational complexity point of view (King *et*

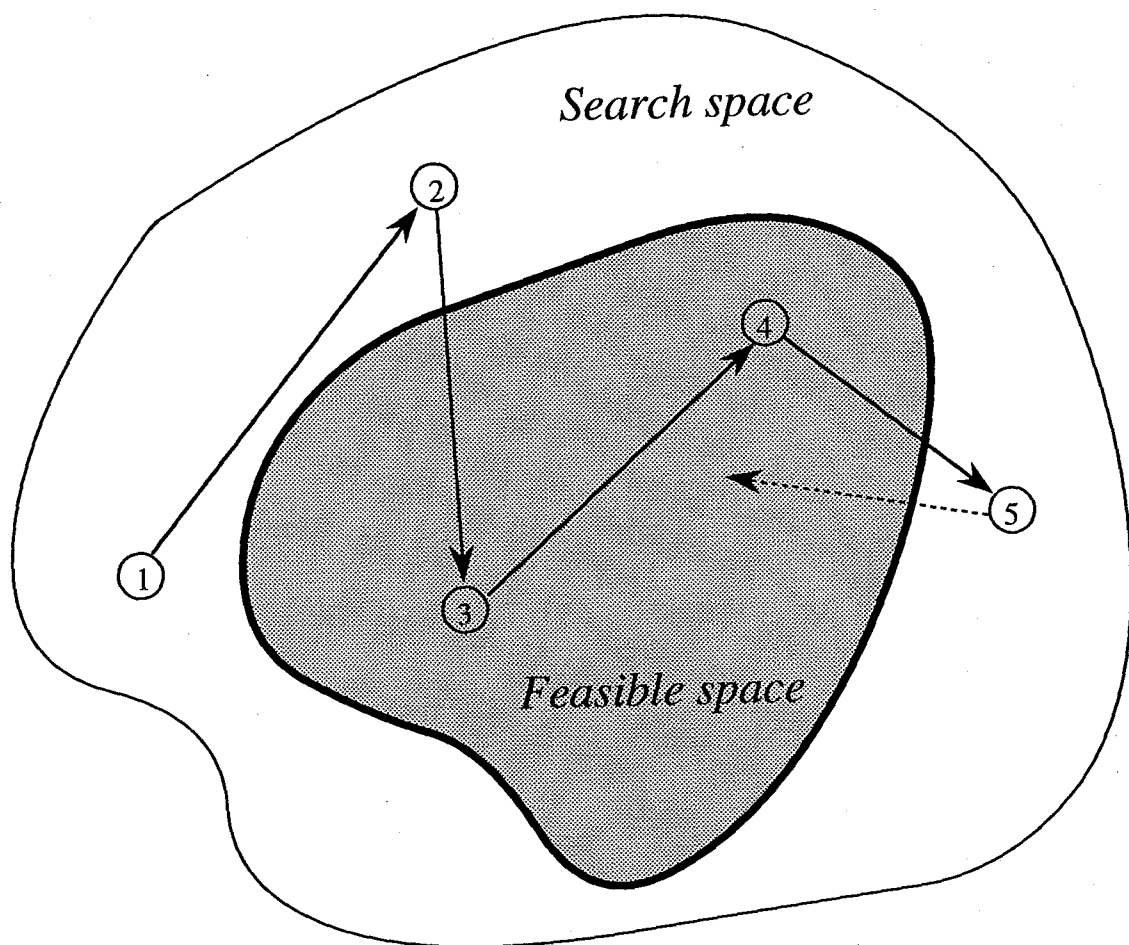
al., 1980). Therefore it is virtually impossible to solve a practical size problem by the mathematical programming approach.

The heuristic approach, on the contrary, has a high potential as methodology for the short-term scheduling within FOSS. In the heuristic approach, initial schedules are generated by a simple rule, and the schedules are evolved by a search technique. In previous works, several algorithms categorized in the heuristic approach have been proposed for batch process scheduling problems. For example, Kuriyan and Reklaitis (1989) propose the algorithms, such as dispatching rules, bottleneck sequencing, local search, best-fit sequencing, and investigate their performance for makespan criterion.

These algorithms generate a sub-optimal schedule more efficiently than the algorithms based on the mathematical programming approach. However, they are designed to work in simple process models with limited scheduling constraints and not assumed to be employed in the on-line scheduling structure. Therefore several algorithms cannot guarantee feasibility if the process has constraints such as product interdependency. In addition, they require much computational effort to obtain a solution because of the complexity of the algorithm.

Figure 5.1 conceptually shows a typical search process by a conventional search algorithm. In this case, the search algorithm can not control the feasibility of the schedules generated during a search process. Therefore the search algorithm may eventually select an infeasible schedule as the solution. In addition, the efficiency of the conventional search algorithm will be decreased because of the unnecessary searches outside the feasible space.

In the circumstances, development of a heuristic algorithm which generates a feasible short-term schedule with reasonable computational effort under various scheduling constraints is required for FOSS.



① i-th search point

Figure 5.1 A conceptual search process

5.3 Extended Dispatching Rule Approach

5.3.1 Dispatching-Simulation Mechanism

As mentioned in the above, FOSS requires a mechanism which generates a feasible short-term schedule rapidly under several practical scheduling constraints. From this point of view, employing dispatching rules combined with discrete event simulation, called dispatching-simulation mechanism, seems to be the most promising approach for FOSS. In the mechanism, dispatching rule and discrete event simulation are employed to reduce search space and to generate feasible short-term schedules, respectively.

Discrete event simulation is a powerful technique to evaluate or analyze complex production systems (Pritsker, 1986). It provides the capability to precisely predict the dynamic behavior of complex production systems. In addition, it has high potential to work within the on-line scheduling environment (Rogers and Flanagan, 1991). For these reasons, discrete event simulation seems to be an appropriate technique to be used for short-term scheduling in the on-line scheduling system. Dispatching rules are simple rules to select a suitable production order to proceed in a production system and have been widely applied in scheduling. Many researchers have been studying dispatching rules since the 1960's and propose a number of dispatching rules, such as SPT, SLACK, FIFO, and so on (Blackstone, *et al.*, 1982).

Figure 5.2 shows the basic mechanism of the dispatching-simulation mechanism. In this mechanism, a schedule is generated through several "order selection-simulation (O-S)" loops. In every O-S loop, a dispatching rule selects the next production order to enter a system from the feasible order list. The list indicates production orders which are currently available for production. The list is generated through O-S loops based on a production plan, scheduling conditions, and the process states predicted by a simulator. For example, only production orders having usable intermediate products enough to start production are registered in the list. The list is dynamically revised upon the changes of the process states, scheduling conditions, or production plan. After the selection of the next production order to be entered, the mechanism specifies the production start time of the selected order so as to

satisfy waiting time constraints associated with the waiting times between tasks for the production. PVSS (Ishii and Muraki, 1995b), for example, provides an algorithm to calculate the production start times which satisfy waiting time constraints under processing time variations. (PVSS, as described in Chapter 6, calculates the minimum slack times between neighboring production orders in a production sequence to avoid the violation of the waiting time constraints between tasks under processing time variations. The production start time of the first task in each production order is specified based on the slack times.) In each O-S loop, the simulator generates a schedule by assigning production orders to resources in a process according to a process model. The results of simulation which indicate process states in the near future are also used to update the feasible order list. The O-S loop is continued until all production orders proceed in a process. As the result, a complete short-term schedule is generated by the simulator.

Because the dispatching-simulation mechanism generates a short-term schedule based on the feasible order list in every O-S loop, the feasibility of the generated schedule is guaranteed. In addition, a dispatching rule selects a suitable production order from the feasible order list. Therefore rapid and high performance short-term scheduling by the dispatching-simulation mechanism can be expected.

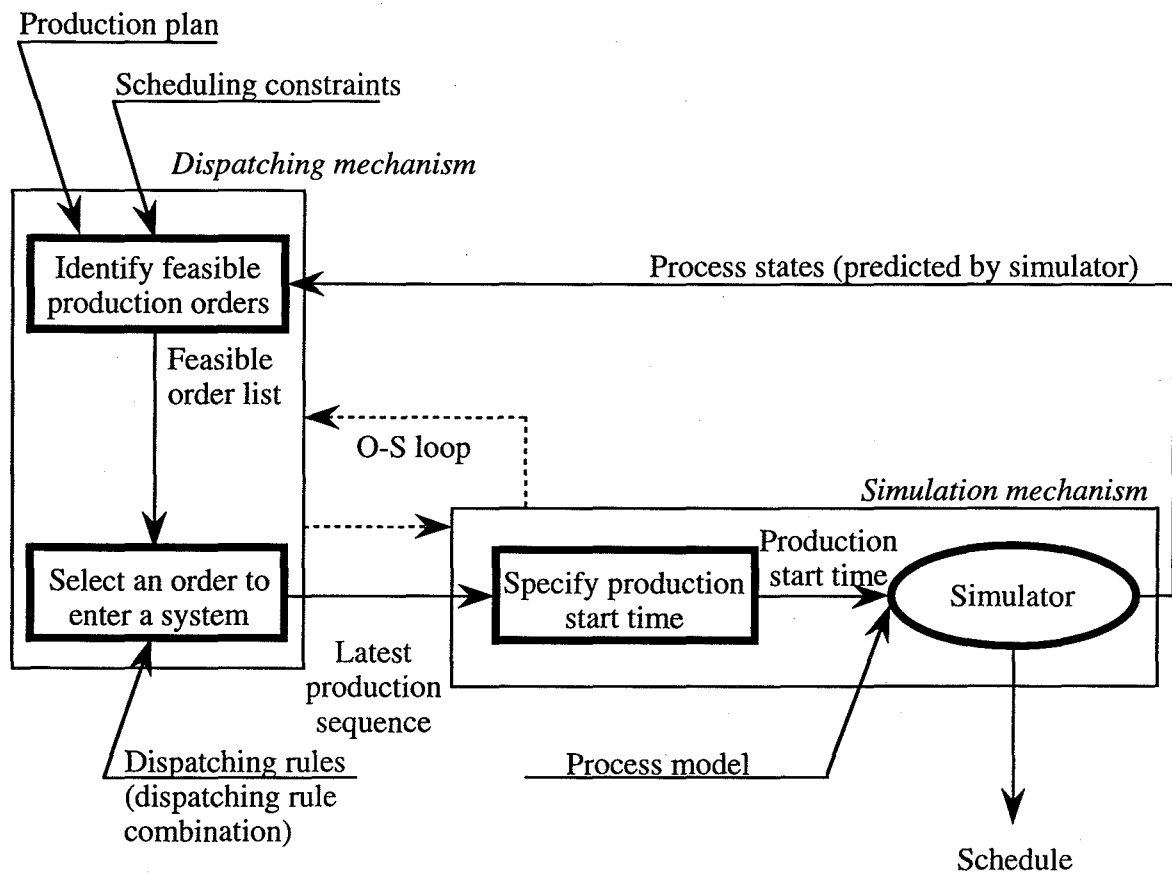


Figure 5.2 A schema of dispatching and simulation mechanism

5.3.2 Concept of EDR Approach

The dispatching-simulation mechanism can generate short-term schedules efficiently and be applicable within FOSS. However, the performance of the conventional dispatching rule approach which uses a single dispatching rule for all the process to select production orders is limited because no single dispatching rule has been shown to consistently generate better schedules than other rules under a variety of process and scheduling conditions (Blackstone *et al.*, 1982).

In this chapter, an Extended Dispatching Rule approach (EDR approach) is proposed to overcome the difficulty of the conventional dispatching rule approach. The EDR approach could apply different dispatching rules to select the next production order in each O-S loop depending on the process state. If a dispatching rule combination is represented according to the sequence to apply rules for the production order selection in O-S loops, the EDR approach could use, for example, [R1, R3, R2] combination for a problem consisting of three candidate dispatching rules (R1, R2, and R3) and three production orders to be scheduled. (A rule combination [R1, R3, R2] represents that rules R1, R3, and R2 will be used in O-S loops to specify 1st, 2nd, and 3rd production order, respectively.) In the conventional approach, on the contrary, one of the single rule combinations, i.e., [R1, R1, R1], [R2, R2, R2], or [R3, R3, R3], is used though the entire O-S loops. Because the EDR approach applies the best dispatching rule depending on process states, more performance improvement with the EDR approach can be expected than with the conventional approach.

5.3.3 EDR Search Algorithm

In the EDR approach, the number of candidate dispatching rules and production orders to be scheduled define the number of feasible dispatching rule combinations. Because it is unrealistic to try all the dispatching rule combinations to find the best one in practical size problems, the EDR approach requires an algorithm which can search an appropriate dispatching rule combination efficiently.

In this section, a simple algorithm, called the EDR search algorithm, to search an appropriate dispatching rule combination for the EDR approach is proposed. Although the

EDR search algorithm could have several forms, a primitive algorithm is employed in this thesis for the purpose of evaluation of the EDR approach.

(1) Overview of EDR search strategy

The EDR search algorithm proposed in this chapter, employs the following strategies to guide the search process.

Strategy 1: Identify the best single dispatching rule according to the performance criteria and use it as the initial current best dispatching rule combination (CBRC) in a search process.

Strategy 2: Replace the *i*-th dispatching rule in CBRC to an alternative dispatching rule and evaluate the performance with the dispatching-simulation mechanism. If the performance is improved, replace CBRC.

Strategy 3: Execute the replacement of dispatching rules one by one from the top rule in CBRC.

Figure 5.3 shows an overview of the EDR search algorithm consisting of two search stages.

Stage 1 identifies the initial dispatching rule combination to start a search process. Several single dispatching rule combinations, for example, the SPT (Shortest Processing Time) rule for all the processes of the production order selection, i.e., [SPT, SPT,.....], are evaluated by using the dispatching-simulation mechanism. As a result, the best single dispatching rule combination according to the performance criteria is selected as the initial combination for a search process. The selected dispatching rule combination is called the current best dispatching rule combination (CBRC).

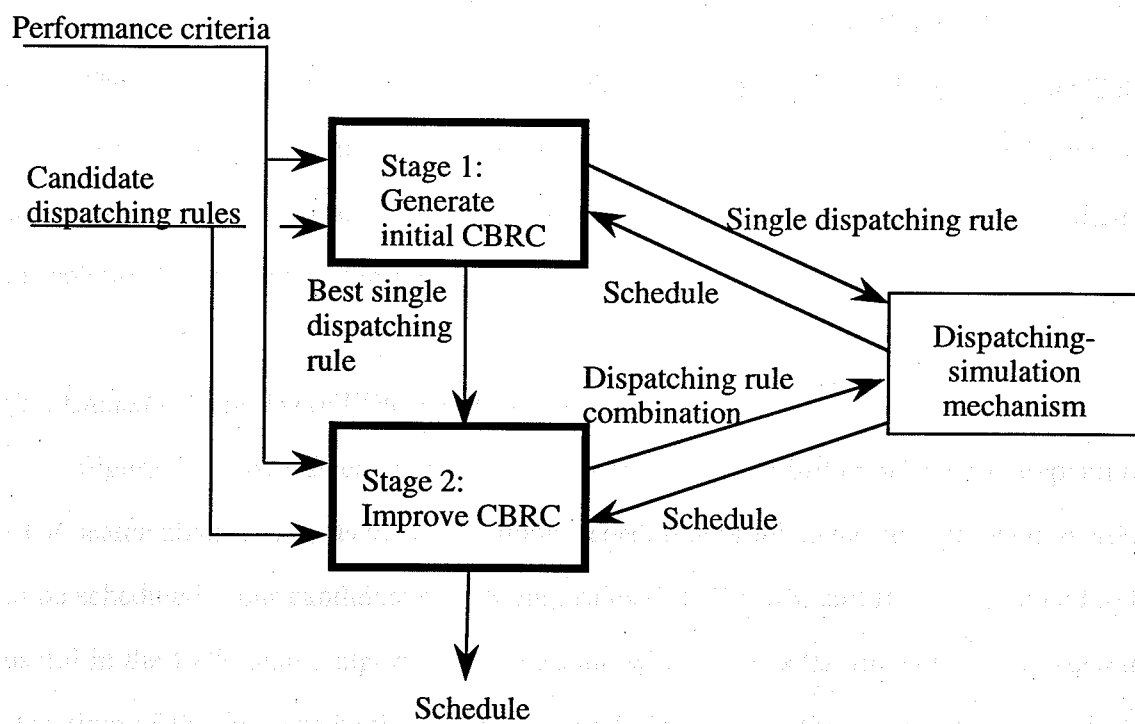


Figure 5.3 Overview of two-stage EDR search algorithm

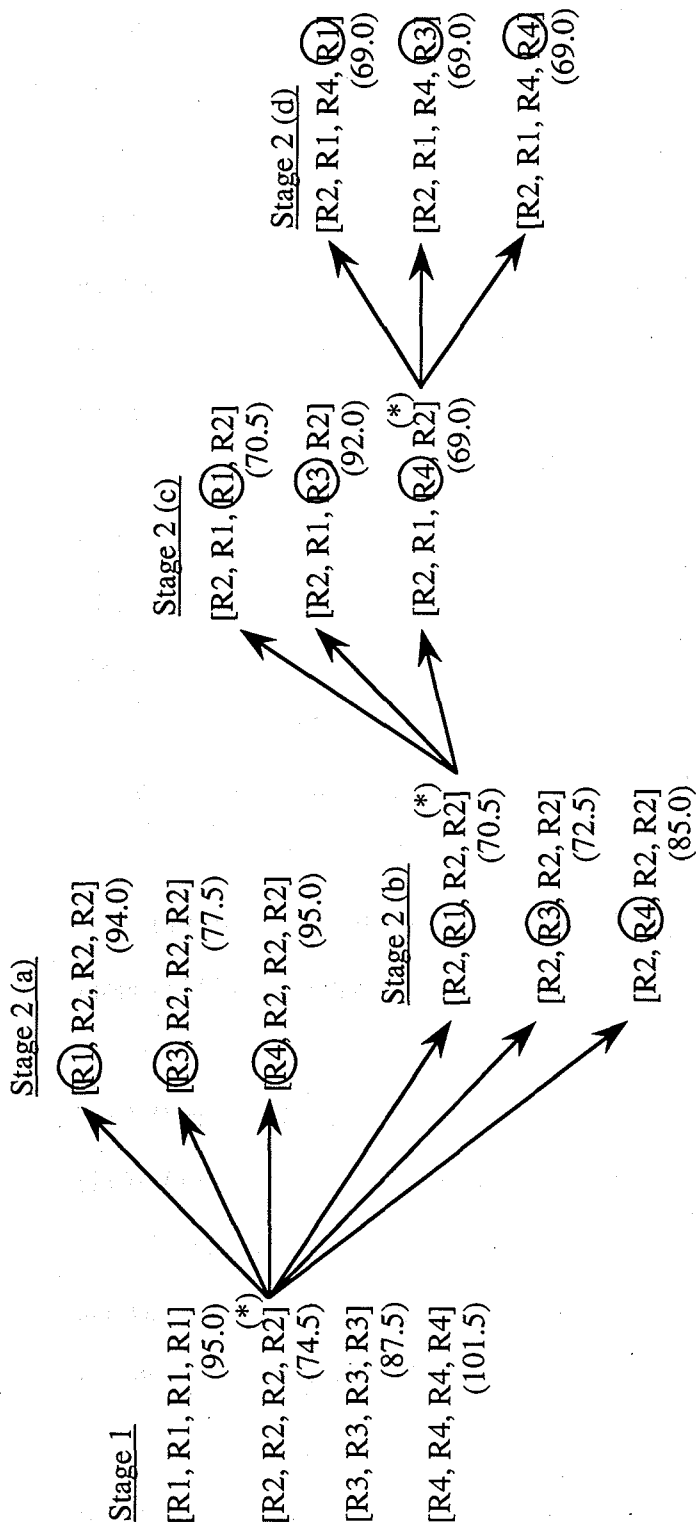
Stage 2 tries to improve CBRC by replacing dispatching rules in CBRC with alternative rules. The performance of the replaced dispatching rule combination is tested using the dispatching-simulation mechanism again, and CBRC is updated to the new one if the tested dispatching rule combination performs better than CBRC. For example, if a dispatching rule combination [SPT, SPT, SPT] is the CBRC, [SLACK, SPT, SPT] is generated by replacing the SPT at the first rule with SLACK rule and the performances of both dispatching rule combinations are compared according to the given performance criterion. As a result, the dispatching rule combination [SLACK, SPT, SPT] is recognized as the new CBRC, if it performs better than the [SPT, SPT, SPT] combination.

The replacement of dispatching rules is done sequentially from the top rule in CBRC. One dispatching rule is replaced at one time. The search process is terminated when the replacement reaches the last rule in CBRC and the schedule of the final CBRC is specified as the solution by the search algorithm.

(2) Detailed explanation of EDR search algorithm

Figure 5.4 shows a search process for an example scheduling problem to explain the EDR search algorithm in detail. The example scheduling problem has four production orders to be scheduled. Four candidate dispatching rules (R1, R2, R3, and R4) are assumed to be useful in the EDR search algorithm. Makespan, which shows the time elapse between the start time of the first production order and the finish time of the last production order, is specified as the performance criterion.

In stage 1, the algorithm evaluates the makespan of four alternative single dispatching rule combinations consisting of all R1, all R2, all R3, and all R4, respectively. As a result, all R2 combination [R2, R2, R2, R2] is recognized as the initial solution in the search process (CBRC) because it gives the best makespan (74.5) among the combinations in stage 1. Based on the CBRC selected in stage 1, the algorithm executes the searches in the order of (a), (b), (c), and (d) in stage 2.



Legend

Dispatching rules: R1, R2, R3, R4

○ : Dispatching rule replaced in the search process

Dispatching rule combination for selecting 1st, 2nd, 3rd, and 4th production order to enter a system.

[X, X, X, X]^(*) (ZZ.Z)
 ← Indication of current best dispatching rule combination (CBRC)

← Makespan from dispatching & simulation mechanism

Figure 5.4 An example of search process by EDR search algorithm

In stage 2(a), three alternative dispatching rule combinations which replace the top dispatching rule of CBRC with R1, R3, and R4 are evaluated by the dispatching-simulation mechanism. Because the alternatives perform worse than CBRC, CBRC is not changed in the stage 2(a).

In stage 2(b), three alternative dispatching rule combinations are also evaluated by replacing the second dispatching rule in CBRC with alternative rules. As a result, CBRC is updated to [R2, R1, R2, R2] combination because the combination performed better than the current CBRC ([R2, R2, R2, R2]). The search process in the following stage is done based on the new CBRC.

The search processes in stages 2(c) and 2(d) are executed by replacing the third and the fourth dispatching rule in CBRC, respectively. As a result, the dispatching rule combination [R2, R1, R4, R2] and the makespan (69.0) are specified as the solution by the EDR search algorithm.

5.4 Evaluation of EDR Approach

In this section, the effectiveness of the EDR approach and the performance of the proposed EDR search algorithm are tested on an example multiproduct batch process model.

5.4.1 Design of Experiments

(1) Multiproduct batch process model for experiments

In experiments, a simple multiproduct batch process model consisting of four tasks as shown in Figure 5.5 is employed. Each task is assumed to be operated in a different unit. (In the multiproduct process, the production of all products follows a fixed task network.)

(2) Product and product interdependency

Eight kinds of product which have the same task network, as shown in Figure 5.5, but have different processing times are specified. (The transfer times between tasks are assumed to be negligible.) The mean processing time of each task for a product is defined randomly in

each experiment based on an uniform distribution between 5 and 35 time units. The processing time of each task is assumed to have an uniformly distributed variability up to $\pm 10\%$ of the mean processing time. The processing time variabilities of all tasks are assumed as independent of each other.

Figure 5.6 shows the product interdependency and the stoichiometry data related to the experiments.

(3) *Production order and due date*

Table 5.1 shows the production order and the initial stock level of each raw material and intermediate product. The due date of each production order is specified randomly in each experiment based on uniform distribution between 200 and 600 time units. (The production orders for IP1 and IP2 are specified to replenish the stock to produce FP1, FP2, and FP3 in consideration of the product interdependency shown in Figure 5.6.)

(4) *Waiting time constraints (WTC)*

For waiting time constraints, the following three scenarios referred to as WTC1, WTC2, and WTC3 are specified.

WTC1: no waiting time constraint (UW) for any task

WTC2: task 2 has a ZW constraint

WTC3: all tasks except task 4 have a ZW constraint

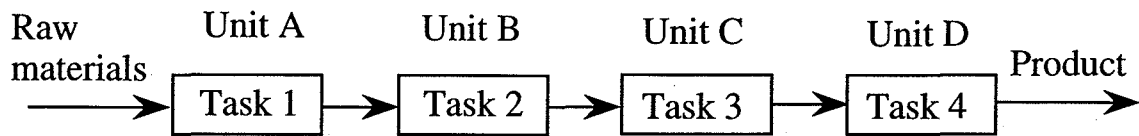


Figure 5.5 A multiproduct process model

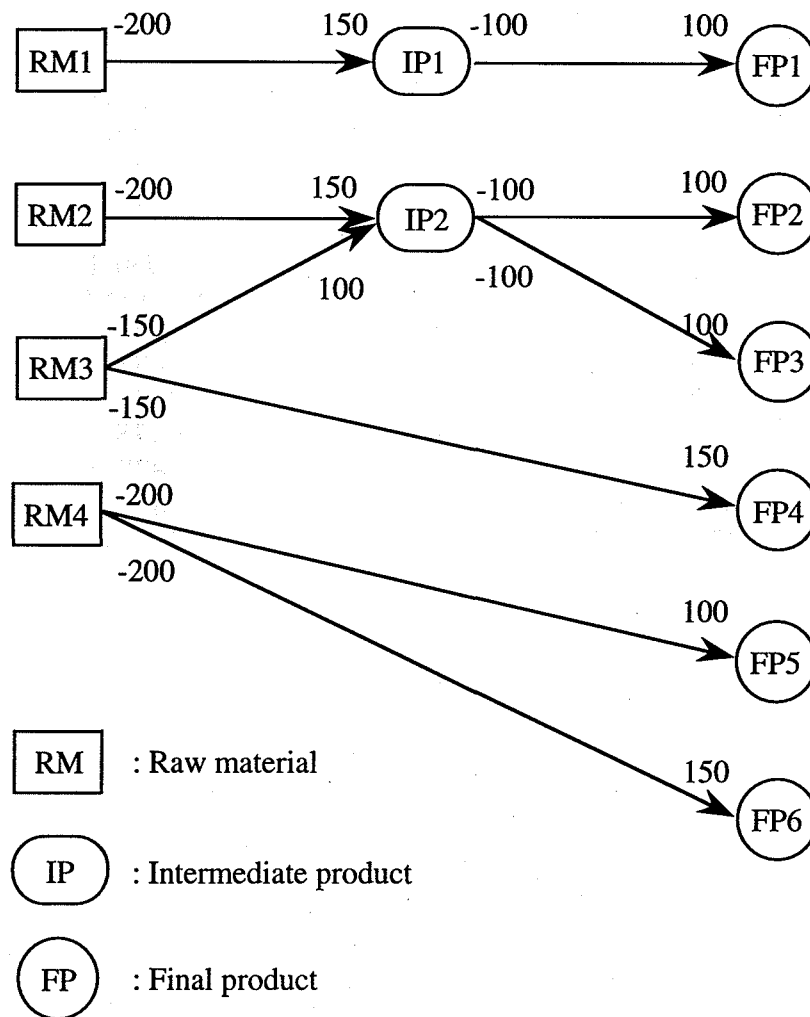


Figure 5.6 Product interdependency and materials balance

(The numbers at the left side and the right side of an arrow indicate the amount of materials consumption and production per batch, respectively.)

Table 5.1 Production order and inventory level

	Production order [Batches]	Initial stock level
Raw materials		
RM ₁	-----	500
RM ₂	-----	650
RM ₃	-----	1000
RM ₄	-----	900
Intermediate products		
IP ₁	2	150
IP ₂	3	300
Final products		
FP ₁	3	-----
FP ₂	3	-----
FP ₃	2	-----
FP ₄	3	-----
FP ₅	2	-----
FP ₆	2	-----

(5) Candidate dispatching rules

The following simple dispatching rules are employed for the EDR search algorithm.

SLACK: Select the order which has the minimum slack time.

(Slack time = due date - present time - remaining processing time)

EDD: Select the order which has the minimum due date.

SPT: Select the order which has the shortest processing time at the next task.

SRPT: Select the order which has the shortest remaining processing time.

LPT: Select the order which has the longest processing time at the next task.

(6) Dispatching-Simulation mechanism

A process model is developed by using a discrete event simulation technique. The feasible product list is generated based on the stock data and the production orders. In addition, the production start timing of each order selected from the feasible order list is specified by PVSS logic (Ishii and Muraki, 1995b) to assure waiting time constraints under processing time variability.

(7) Benchmark scheduling algorithm

The following benchmark scheduling algorithms are employed in the experiments.

- (a) Best dispatching rule (BDR): BDR generates schedules with candidate single dispatching rules by the dispatching-simulation mechanism and the best schedule according to the given performance criterion is taken as the solution.
- (b) Random sequencing (RS): RS generates a number of schedules by selecting a production order randomly from the feasible product list in the dispatching-simulation mechanism and the best schedule according to the performance criterion is taken as the solution. In the experiments, RS generates 1000 schedules randomly for each problem. In addition, the RS_AVE, which shows the average performance of randomly generated schedules in the RS process, is used to evaluate the effectiveness of BDR and single dispatching rules.

(c) EDR with random initial solution (EDR_R): EDR_R generates a number of schedules by the EDR search algorithm with randomly generated initial CBRCs, and the best schedule according to the performance criterion is taken as the solution. In each experiment, ten schedules are generated with different initial CBRCs. In addition, the EDR_R_AVE, which shows the average performance of the schedules generated during an EDR_R process, is used to evaluate the effectiveness of EDR.

(8) *Performance criteria*

The performance is based on the following four criteria in the experiments.

(a) *Make span (MS)*

$$\text{Max } [C_k] \\ k=1, NP$$

(b) *Mean flow time (MFT)*

$$\sum_{k=1}^{NP} (C_k - S_k) / NP$$

(c) *Mean tardiness (MT)*

$$\sum_{k=1}^{NP} \text{Max } [0, L_k] / NT$$

s.t.

$$L_k = C_k - D_k$$

(d) *Weighted mean flow time and mean tardiness (WPC)*

$$W * \sum_{k=1}^{NP} (C_k - S_k) / NP + (1-W) * \sum_{k=1}^{NP} \text{Max } [0, L_k] / NT$$

s.t.

$$L_k = C_k - D_k; W = 0.5$$

where C = completion time of a production, S = start time of a production, NT = the number of tardy orders of all production orders, NP = the number of production orders to be scheduled, W = weighting for mixed performance criteria.

The first and second performance criteria are processing-time-based measures and widely used in scheduling. The third performance criterion is to evaluate the performance of due date control by the EDR approach. The fourth performance criterion is to evaluate the performance of the EDR approach using mixed criteria.

(9) *The number of problems*

For the experiments, 15 problems are generated. Because there are three types of waiting time constraints and four performance criteria, total of 180 scheduling problems are solved by the proposed EDR search algorithm and by benchmark algorithms.

5.4.2 *Results of Experiments*

(1) *Effectiveness of dispatching rule*

Tables 5.2 and 5.3 summarize the performance of single dispatching rules and search algorithms employed in the experiments, respectively. In the Table 5.2, the numbers inside the parentheses show the average of relative ranking which ranks the performance of each single dispatching rule on a scale of 0-1 by the following equation.

$$RR_i = (S_i - S_{best}) / (S_{worst} - S_{best})$$

where RR_i = relative ranking of the schedule by a single dispatching rule i , S_i = performance of the schedule by a single dispatching rule i , S_{best} = the best performance among the schedules generated by candidate single dispatching rules, S_{worst} = the worst performance among the schedules generated by candidate single dispatching rules.

Table 5.2 Summary of experiments
(Single dispatching rules)

Performance criteria	Average MS	Average MFT	Average MT	Average WPC
WTC1				
<i>SLACK</i>	640.51 (0.813)	363.92 (0.844)	75.71 (0.212)	219.50 (0.324)
<i>EDD</i>	642.26 (0.847)	359.99 (0.802)	71.83 (0.189)	215.91 (0.228)
<i>SPT</i>	607.68 (0.341)	325.67 (0.251)	159.34 (0.842)	242.53 (0.652)
<i>SRPT</i>	594.59 (0.089)	313.99 (0.092)	122.14 (0.517)	218.06 (0.260)
<i>LPT</i>	631.89 (0.737)	367.20 (0.946)	139.05 (0.727)	253.12 (0.888)
WTC2				
<i>SLACK</i>	676.07 (0.830)	383.38 (0.843)	89.91 (0.169)	236.64 (0.285)
<i>EDD</i>	768.73 (0.912)	381.15 (0.881)	83.85 (0.085)	232.50 (0.224)
<i>SPT</i>	638.48 (0.280)	344.09 (0.292)	165.55 (0.862)	254.82 (0.618)
<i>SRPT</i>	633.98 (0.198)	325.02 (0.065)	139.63 (0.564)	240.48 (0.281)
<i>LPT</i>	665.02 (0.692)	382.30 (0.857)	149.55 (0.723)	265.93 (0.863)
WTC3				
<i>SLACK</i>	793.76 (0.837)	439.23 (0.857)	133.99 (0.095)	288.31 (0.226)
<i>EDD</i>	796.36 (0.853)	441.36 (0.878)	132.13 (0.048)	285.75 (0.197)
<i>SPT</i>	756.53 (0.292)	405.06 (0.406)	222.04 (0.841)	313.55 (0.652)
<i>SRPT</i>	742.06 (0.051)	379.52 (0.016)	208.72 (0.679)	294.12 (0.363)
<i>LPT</i>	783.12 (0.754)	440.75 (0.867)	206.96 (0.710)	323.85 (0.888)

(): Average of relative ranking against the best single dispatching rule.

Table 5.3 Summary of experiments
(EDR, BDR, RS, EDR_R)

Performance criteria	Average MS	Average MFT	Average MT	Average WPC
WTC1				
EDR	561.41	297.04	32.51	172.68
BDR	587.40	310.43	59.22	202.65
RS	571.17	311.14	56.14	195.78
RS_AVE	656.30	355.82	161.49	258.66
EDR_R	550.18	294.81	23.30	167.01
EDR_R_AVE	569.83	305.70	39.44	178.81
WTC2				
EDR	595.63	313.67	43.78	187.09
BDR	621.19	325.94	76.83	222.26
RS	601.83	327.44	75.93	212.83
RS_AVE	671.80	374.12	177.11	275.61
EDR_R	589.15	311.89	36.55	184.61
EDR_R_AVE	606.86	322.20	55.69	195.58
WTC3				
EDR	714.39	365.83	88.67	240.75
BDR	736.86	377.63	126.87	275.96
RS	725.59	382.76	128.08	267.44
RS_AVE	789.04	433.58	234.20	333.89
EDR_R	709.19	360.40	80.27	234.29
EDR_R_AVE	725.63	374.32	101.26	248.60

As shown in the Table 5.2, no single dispatching rule performed better than other single dispatching rules in all cases. For example, SRPT, which ranks 1st place (0.051) under makespan criterion and WTC3 constraint, ranks 3rd place (0.679) under mean tardiness criterion and WTC3 constraint. Table 5.3, however, indicates that BDR performed better than the RS_AVE. Moreover, BDR seems to perform better than RS in some cases.

To test the differences among RS, RS_AVE, and BDR statistically, paired comparisons (Hole, 1971) were employed based on the results of experiments. Table 5.4 shows the results of paired comparisons for each performance criterion and each waiting time constraint. As shown in the table, significant differences between RS_AVE and BDR are obtained in all experiment conditions. (The null hypothesis, H_0 : means of solutions by RS_AVE and BDR are identical, is rejected at less than 1% level.) In addition, differences are not obtained between RS and BDR in both MFT and MT performance criteria.

Therefore the results of experiments support the following.

- The performance of dispatching rules depends on scheduling conditions, however, dispatching rules perform better than random scheduling.
- Dispatching rules can reduce search space efficiently in the short-term scheduling if appropriate dispatching rules are employed in a search process depending on scheduling conditions.

(2) Effectiveness of EDR approach

Table 5.3 shows that the proposed EDR search algorithm (EDR) successfully improves the performance of the schedules generated by BDR. To validate the effectiveness of EDR against BDR, paired comparisons between the solutions by EDR and BDR were constructed as shown in Table 5.5. As a result, significant differences between EDR and BDR are obtained in all experiment conditions. (The null hypothesis, H_0 : means of solutions by EDR and BDR are identical, is rejected at less than 1% level.) Therefore it is possible to conclude that applying different dispatching rules depending on scheduling conditions in a search process, like an EDR approach, performs significantly better than applying a single dispatching rule for all search processes like BDR.

Table 5.4 Paired comparisons between RS, RS_AVE, and BDR

Performance criteria	MS	MFT	MT	WPC
WTC1				
<i>RS - BDR</i>				
t-Value	-2.72 ^(*)	0.16	-0.51	-2.25 ^(*)
<i>RS_AVE - BDR</i>				
t-Value	3.44 ^(**)	8.12 ^(**)	14.52 ^(**)	17.87 ^(**)
WTC2				
<i>RS - BDR</i>				
t-Value	-3.69 ^(**)	0.40	-0.13	-3.48 ^(**)
<i>RS_AVE - BDR</i>				
t-Value	8.07 ^(**)	8.58 ^(**)	13.67 ^(**)	15.55 ^(**)
WTC3				
<i>RS - BDR</i>				
t-Value	-2.30 ^(*)	1.36	0.34	-2.30 ^(*)
<i>RS_AVE - BDR</i>				
t-Value	7.21 ^(**)	9.01 ^(**)	24.90 ^(**)	15.88 ^(**)

Degree of freedom=14; $t(14, 0.05)=2.14$; $t(14, 0.01)=2.98$.

(*): Significant difference at 95% level.

(**): Significant difference at 99% level.

Table 5.5 Paired comparisons between BDR, RS, EDR_R, EDR_R_AVE, and EDR

Performance criteria	MS	MFT	MT	WPC
WTC1				
BDR - EDR				
t-Value	6.57(**)	6.48(**)	7.30(**)	9.12(**)
RS - EDR				
t-Value	2.14(*)	3.96(**)	4.06(**)	7.26(**)
EDR_R - EDR				
t-Value	-3.88(**)	-1.29	-2.24(*)	-2.86(*)
EDR_R_AVE - EDR				
t-Value	3.45(**)	5.06(**)	1.73	3.51(**)
WTC2				
BDR - EDR				
t-Value	6.66(**)	6.66(**)	4.79(**)	9.19(**)
RS - EDR				
t-Value	1.25	4.88(**)	6.01(**)	8.57(**)
EDR_R - EDR				
t-Value	-1.88	-1.08	-1.94	-1.40
EDR_R_AVE - EDR				
t-Value	3.09(**)	4.87(**)	3.10(**)	4.75(**)
WTC3				
BDR - EDR				
t-Value	6.93(**)	6.42(**)	6.63(**)	7.13(**)
RS - EDR				
t-Value	3.71(**)	5.91(**)	7.67(**)	6.58(**)
EDR_R - EDR				
t-Value	-2.20(*)	-1.17	-1.85	-3.66(**)
EDR_R_AVE - EDR				
t-Value	4.80(**)	4.72(**)	2.81(*)	3.77(**)

Degree of freedom=14; $t(14, 0.05)=2.14$; $t(14, 0.01)=2.98$.

(*): Significant difference at 95% level.

(**): Significant difference at 99% level.

Scheduling conditions could affect the performance of the EDR approach. Table 5.6 shows the ANOVA (Analysis of Variance) (Hole, 1971) tables constructed for each performance criterion to check the effects of waiting time constraints on the EDR approach. In the analysis, the percentage of performance improvement by EDR against BDR was used. (Percentage improvement = $[\text{Solution by BDR} - \text{Solution by EDR}] / \text{Solution by BDR} * 100$)

As shown in the table, no significant difference by the waiting time constraints was obtained except under the MT performance criterion. (The null hypothesis, H_0 : the percentage performance improvements in each waiting time constraint are identical, is rejected at 5% level in MT.) The result of analysis shows that the performance improvement for MT depends on the waiting time constraints.

Table 5.7 shows the results of analysis to test the effects of performance criteria on the EDR approach. As shown in the table, the percentage performance improvement in MT and WPC is significantly better than in MS and MFT.

The results of experiments indicate that EDR performs significantly better than BDR, however, scheduling conditions, such as waiting time constraints, performance criteria, and so on, could affect the degree of performance improvement by EDR against BDR.

(3) Performance of EDR search algorithm

The proposed EDR search algorithm (EDR) performs better than RS especially in MFT, MT, and WPC, as shown in Table 5.5. (The null hypothesis, H_0 : means of solutions by RS and EDR are identical, is rejected at 1% level in MFT, MT, and WPC.) In MS, the performance improvements by EDR against RS are inferior to the results for other performance criteria. However, EDR still performs much better than RS. In addition, the number of search iterations in EDR and RS for an example problem are 76 and 1000, respectively. The results of experiments indicate that the proposed EDR search algorithm generates significantly better schedules efficiently comparing to RS.

Table 5.6 ANOVA tables
(Effects of waiting time constraints)

Source of Variation	Degree of Freedom	Sum of Squares	Mean Square	F-Value
MS				
WTC	2	14.91	7.46	1.36
Residual	42	229.47	5.46	
MFT				
WTC	2	9.34	4.67	0.98
Residual	42	200.53	4.77	
MT				
WTC	2	3476.97	1738.48	3.32(*)
Residual	42	21983.03	523.41	
WPC				
WTC	2	75.59	37.79	1.10
Residual	42	1438.27	34.25	

$F(0.05) = 3.23$; $F(0.01) = 5.18$.

(*): Significant difference at 95% level.

(**): Significant difference at 99% level.

Table 5.7 ANOVA & mean tables
(Effects of performance criteria)

Source of Variation	Degree of Freedom	Sum of Squares	Mean Square	F-Value
<i>Performance criteria</i>	3	42382.64	14127.55	90.65(**)
<i>Residual</i>	176	27428.09	155.84	

$F(0.05) = 3.23$; $F(0.01) = 5.18$.

(*): Significant difference at 95% level.

(**): Significant difference at 99% level.

Performance criteria	Mean	Std. Dev.
MS	3.86	2.36
MFT	3.71	5.87
MT	41.31	24.06
WPC	14.21	5.87

As shown in Table 5.5, EDR performs significantly better than EDR_R_AVE, which shows the average performance of the schedules generated during an EDR_R process, in the most cases. In addition, the performance of EDR and EDR_R is identical in some cases. The results of experiments support the conjecture that the strategy 1 in the proposed EDR search algorithm works well to specify the initial CBRC in a search process. It seems, however, the possibility to improve the performance of the search algorithm exists by establishing a different strategy to specify the initial CBRC in the EDR search algorithm.

The computation time of the proposed EDR search algorithm depends on the computer used, the complexity of the process, and the problem size. In the experiments, an Apple Macintosh Quadra 950 which provides 2.0 mflops CPU capacity (Stadtherr *et al.*, 1993) was used. As a result, EDR spent 73 seconds on a four tasks and 20 orders scheduling problem. On the other hand, RS, which generates 1000 schedules randomly, spent 910 seconds. EDR_R spent 731 seconds.

In the proposed EDR search algorithm, the computation time (CT) will be proportional to the number of production orders to be schedules (NP), the number of candidate single dispatching rules to be used (NR), and the computational time to obtain a solution by the dispatching-simulation mechanism (P) as shown in the following equation. The equation shows that the search algorithm provides a solution with the computational time in the polynomial order.

$$CT = P \cdot (NR + NR \cdot NP - NP)$$

It is possible to judge from these experiments that the proposed EDR search algorithm generates short-term schedules for practical size problems rapidly enough to work within the on-line scheduling structure (FOSS).

The results of experiments indicate that the primitive search algorithm for the EDR approach presented in this chapter could work well to provide short-term schedules in the on-line scheduling structure. However, some alternative search algorithms seem to improve the performance of short-term scheduling. For example, a mechanism to test several initial

CBRC by the concept of simulated annealing algorithm (Manz *et al.*, 1989) could improve the performance significantly, as EDR_R performed well in the experiments. Moreover, a backtracking mechanism which could restart the search from alternative dispatching rule combinations during the stage 2 in the proposed EDR search algorithm could also improve the performance of the algorithm. However, any alternative search algorithm would likely increase the number of iterations of the search. The trade-off between the rank of solution and the CPU-time requirement must be considered since the algorithm is to be used in the on-line scheduling system.

5.5 Chapter Summary

In this chapter, a dispatching-simulation mechanism for a re-scheduling strategy within FOSS, described in Chapter 4, is presented. Moreover, an extended dispatching rule (EDR) approach in the dispatching-simulation mechanism, together with a simple search algorithm for the approach is proposed. The effectiveness of the EDR approach and the performance of the EDR search algorithm are tested on an example multiproduct batch process model.

The following conclusions can be drawn from the experiments:

- Dispatching rules can reduce search space for efficient short-term scheduling.
- Applying different dispatching rules depending on production information in a search process, as in the EDR approach, performs significantly better than applying a single dispatching rule for all the search processes, as in BDR.
- The proposed EDR search algorithm in the dispatching-simulation mechanism generates a short-term schedule rapidly enough to work within the on-line scheduling system for a practical size problem.
- The strategy 1 in the proposed EDR search algorithm works well to specify the initial CBRC of a search process. It seems, however, that the possibility to improve the performance of the search algorithm exists by establishing a different strategy to specify the initial CBRC in the algorithm.

Chapter 6

A PROCESS-VARIABILITY-BASED SCHEDULING SYSTEM WITHIN ON-LINE SCHEDULING STRUCTURE IN BATCH PROCESS

6.1 Chapter Overview

FOSS, proposed in Chapter 4 as a structure of the intelligent scheduling system in batch processes, requires several schedule modification strategies to react to the wide variety of disruptions or performance degradations during production. In the circumstances, Chapter 5 presents the EDR approach for a re-scheduling strategy within FOSS. One of the other critical strategies required in FOSS is a time shift strategy which modifies the current working schedule by shifting not-yet-started-tasks forward or backward to improve the performance of the system and maintain feasibility of waiting time constraints between tasks under processing time variations. This strategy corresponds to the schedule modification for the task start & completion times space (SP1) depicted in Figure 4.2.

In this chapter, a Process-Variability-based on-line Scheduling System (PVSS) for multiproduct batch processes is proposed for a time shift strategy within FOSS. The objective of PVSS is to maintain the performance of the working schedule and to control the waiting time of each task within the range specified in a recipe under processing time variations by shifting task start times, within FOSS proposed in Chapter 4. PVSS provides a new feedforward mechanism to control waiting times between tasks within the ranges specified by a recipe without substantial performance degradation under processing time

variations. It specifies certain slack times between products based on current process state and possible future processing time variations at short intervals and at the completion of each task, and then dynamically modifies the start and completion times of not-yet-finished-tasks to insert the slack times specified between products. The slack times inserted between products work to control waiting times between tasks under processing time variations. The insertion of slack times extends cycle times, therefore reduces the performance. To overcome this problem, PVSS employs two strategies to improve performance. The effectiveness of PVSS is demonstrated by simulation experiments on a multiproduct batch process model.

This chapter is organized as follows: Section 6.2 presents background and problem descriptions of PVSS. Section 6.3 describes PVSS in detail. Section 6.4 demonstrates the effectiveness of PVSS with a simple multiproduct batch process model.

6.2 Background and Problem Description

6.2.1 Background and Previous Works

In batch processes, major production information for short-term scheduling, such as production plan, resource availability, processing times, are usually changeable to some extent even in normal operations. In these circumstances, the performance of the conventional off-line-based scheduling systems is limited because they cannot follow the dynamic behavior of the production information. Therefore Chapter 4 provides a fundamental structure of the on-line scheduling system (FOSS), which modify a schedule dynamically based on the current production information to adapt to variations of the production information, as one of the structures of the intelligent scheduling system.

In batch processes, control of waiting times between tasks under processing time variations is critical because of the following reasons.

- High degree of processing time variations due to fluctuations of quality of raw materials or catalyst, fouling of equipment during production, and fluctuation of cleanness of equipment. (The degree of processing time variation depends

on the process. Cott and Macchietto (1989) state that variations in the processing time of 5% and upwards appear to be typical.)

- Constraints on the maximum waiting times between tasks specified by a batch recipe. (With an unstable intermediate product, ZW (Zero-Wait) constraint, where the intermediate must be immediately processed in the succeeding task, and FW (Finite-Wait) constraint, where the intermediate is allowed to be held for a specified period, are often seen in batch processes.)

The simple examples in Figure 6.1 show the possible effects of processing time variations with conventional off-line-based scheduling. As in Scenario 1, the processing time extension at unit B for product X will cause waiting time (WT) at unit A for product Y. The processing time extension at unit A for product Y, as in Scenario 2, will cause an increase of the idle time between products X and Y. In addition, as in Scenario 3, the processing time decrease at unit B for product Y will cause WT at unit B for product Y.

As per Scenarios 1 and 2 shown in Figure 6.1, processing time variations will cause low productivity because of increased waiting times and idle times. Furthermore, they will cause the violation of waiting time constraints in Scenarios 1 and 3 if tasks at units A and B of product Y have ZW constraints.

The violation of waiting time constraints could cause low product quality level because of increased by-products and low concentration of the desired product. In addition, it could require longer processing times for the rest of the task or even additional tasks to remove excess by-products and maintain desired product quality level. Therefore the management of the waiting time in each task under processing time variations is a vital issue in batch processes.

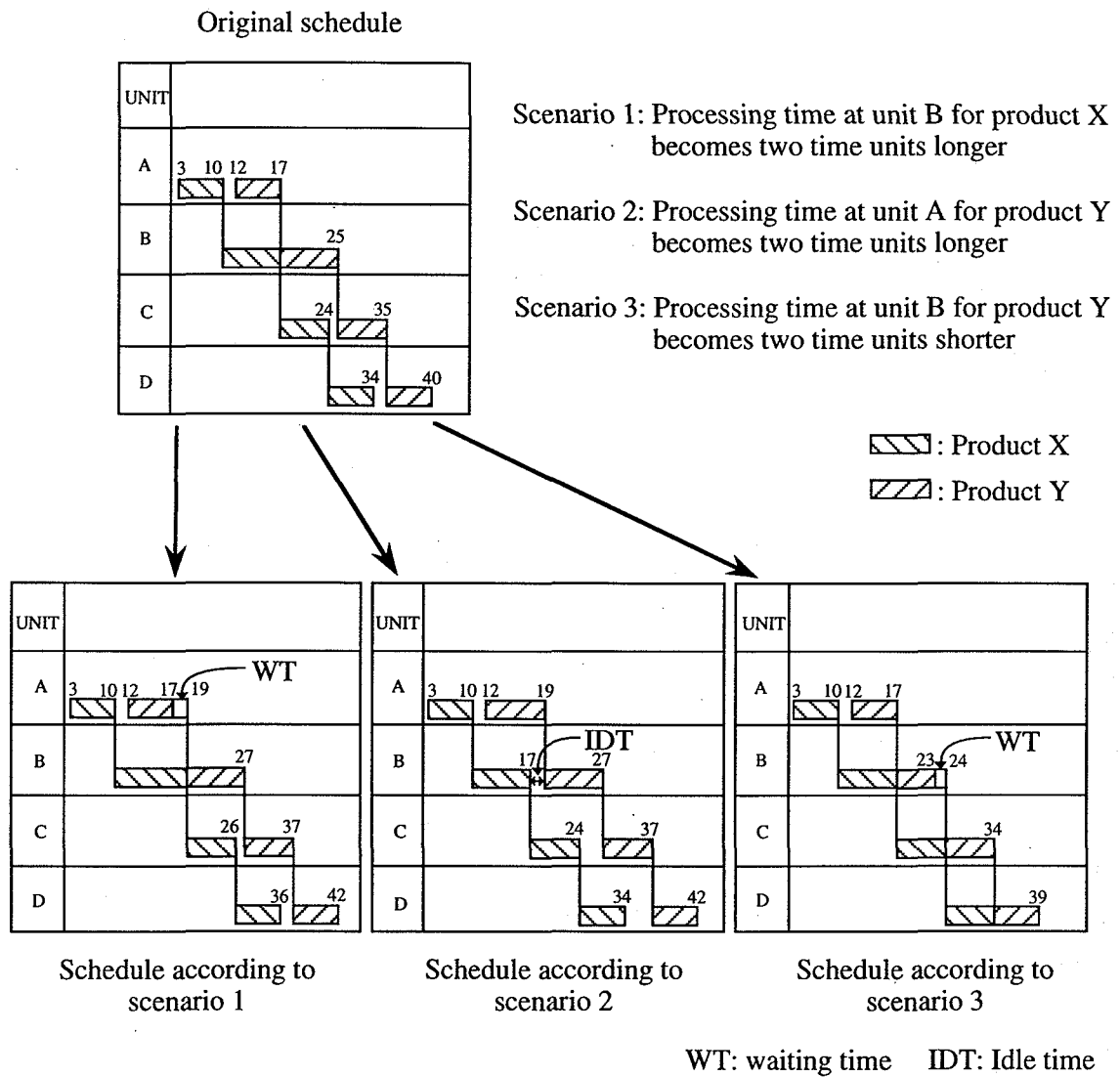


Figure 6.1 Effects of processing time variations

To manage processing time variations, at least three approaches are possible, as follows;

- Execute production according to a relaxed schedule by inserting too much slack time between products to absorb the variations,
- Prepare intermediate storage tanks between processing units,
- Control processing times by dynamically adjusting operational parameters.

However, the first approach needs a high investment cost because of low equipment utilization or over-capacity facilities. Shorter slack times between products seem to realize higher equipment utilization. However, such realization is impossible because of the disruption of waiting time constraints by processing time variations. The second approach also requires high investment cost. In addition, intermediate storage tanks are not applicable if the products are unstable and ZW constraints are specified. The third approach has been tested by several researchers, such as Onogi *et al.* (1986), and Hvala *et al.* (1993), for processes including intermediate storage tanks. This approach, however, has difficulties from the points of maintaining product quality level and process control technology, and therefore applicable batch processes are limited.

Terpstra *et al.* (1994) propose a prototype reactive scheduler for mixed batch/continuous plants with a real-time expert system. Their system is intended to deal with wide variety variability including processing time variations. However, their main concern is a framework of the scheduling system and no concrete mechanism to handle processing time variations has been described.

Cott and Macchietto (1989) studied the performance of on-line scheduling in multiproduct batch plants under processing time variations in detail. They investigated the performance of several simple time shift algorithms. In the algorithms, the deviations between the target schedule and the actual schedule are detected at short intervals and the schedule is shifted forward or backward as the deviations are detected to minimize the effects of processing time variations. In addition, they developed POMA (Projected Operation Modification Algorithm), which predicts the system states from the current operation and

takes corrective action by changing the start times of the not-yet-executed products. POMA performs better than the simple time shift algorithms, especially under ZW constraints. POMA shows that on-line scheduling by a time shift approach is a powerful technique to minimize the effects of processing time variations. However, its capability to control the waiting times between tasks under processing time variations is limited because POMA modifies a schedule without considering future processing time variations.

In any case, previous researches on on-line scheduling systems in batch processes have not focused on the control of waiting times between tasks within the ranges specified by a recipe under processing time variations, although it is one of the critical requirements in batch process management. For this reason, research into a new concept on-line scheduling system to deal with processing time variations is required.

6.2.2 Problem Description

In this research, the following assumptions are made.

(1) System boundaries

PVSS, proposed in this chapter, is assumed to be employed as a schedule modification mechanism at task start & completion times space (SP1), shown in Figure 4.2, within the on-line scheduling system (FOSS) described in Chapter 4. Upper level schedule modification mechanisms, such as re-scheduling (SP3), unit reassignment (SP2), to handle variability other than processing time variations are provided in FOSS.

(2) Scheduling model

The proposed schedule modification algorithm is assumed to be within the model of a multiproduct process without intermediate storage tanks. In the multiproduct process, several products with similar recipes are produced in a fixed equipment network. The multiproduct process is widely used in practice, for such as pharmaceuticals, paints, resins, and so on.

(3) Processing time variability

The average and range of processing time variation in each task under normal process operations are assumed to be given. These data are usually available from batch records of past operations or the analysis of experimental data.

6.3 Process-Variability-Based On-line Scheduling System

6.3.1 Overview of PVSS

The objective of PVSS is to control waiting times between tasks within the ranges specified by a recipe without substantial performance degradations under processing time variations, as a schedule modification mechanism within FOSS presented in Chapter 4. To achieve the objective, PVSS receives the following information within FOSS.

- Current working schedule to be modified,
- On-line process data, such as equipment status data, resource availability, inventory data, and so on.

Based on the information, PVSS modifies the working schedule to adapt to the current situation, and provides a modified schedule.

PVSS inserts certain slack times between products to control waiting times between tasks under processing time variations. To specify reasonable slack times, the algorithm predicts the earliest and latest completion times of each task under waiting time constraints between tasks, based on the current process state and possible future processing time variations. The earliest and latest completion times of each task predicted indicate the possible fluctuation ranges of task completion times in the future due to processing time variations. Based on the prediction, PVSS specifies reasonable slack times to satisfy the waiting time constraints between tasks under processing time variations. Because the slack times required to deal with processing time variations are subject to change during schedule execution, the schedule modification algorithm re-evaluates the reasonable slack times at short intervals and at the completion of each task. Then it dynamically modifies a currently

executing schedule by shifting the scheduled but not-yet-finished tasks forward or backward to insert revised slack times between products.

The upper level schedule modification strategies, to be installed within FOSS, as described in Chapter 4, provide corrective actions, such as resource reassignment, re-sequencing, or eventually complete re-scheduling, depending on the situation. One could argue that PVSS should include at least resource reassignment or local re-sequencing capability. In this research, however, PVSS is proposed as a simple and general form to deal with processing time variations which are the most essential and frequent in batch processes so that PVSS can be employed as a module for complex on-line scheduling systems.

6.3.2 Details of PVSS

(1) Strategies of PVSS

PVSS dynamically shifts the start and completion times of not-yet-finished tasks to insert reasonable slack times between products, required to control waiting times between tasks within the ranges specified by a recipe under processing time variations. The dynamic time shift in the algorithm is an effective approach not only to control waiting times between tasks but also to reduce idle times between products under processing time variations.

However, any slack time insertions cause performance degradation of a schedule because of increased batch cycle times. PVSS, therefore, employs the following two strategies to improve performance.

Strategies

S1 (Strategy 1): Take different control policies depending on the waiting time constraints of each task as follows;

- for the task with a waiting time constraint (ZW/FW): consider the slack time in advance between products to satisfy the constraint under processing time variations,
- for the task with no waiting time constraint (UW, Unlimited-Wait):
Create an IWT (intentional waiting time) dynamically to adjust the start times of the succeeding tasks, and then control the waiting times of the succeeding tasks within the ranges specified. IWT is created only when

the immediate succeeding tasks have waiting time constraints. IWT works to shorten slack times and improves performance of a schedule.

S2 (Strategy 2): Start tasks with no waiting time constraint which precede the bottleneck task as early as possible within the range of total processing time variations of the tasks to avoid the start time delay of the bottleneck task.

The advantage of the first strategy (S1) can be seen from Figure 6.2(a). Suppose the task at unit B of product Y (TBY) has ZW constraint and the task at unit A of product Y (TAY) has no waiting time constraint. In this case, S1 allows control of the start time of TBY by dynamically creating a certain IWT in TAY after T_y time. By doing so, a shorter slack time between TCX and TCY can be specified than the slack time to be specified at T_x time because there is less uncertainty of scheduling data after T_y than the data available at T_x . Therefore performance improvement in on-line scheduling is expected by implementing S1.

The second strategy (S2) could be categorized as a bottleneck oriented approach. Many researchers have been studying this approach from various points of view and report good results on the bottleneck oriented approach in scheduling. For example, the OPT scheduling system, developed based on the bottleneck oriented approach, is regarded as one of the high performance scheduling systems (Morton and Pentico, 1993). Ishii and Talavage (1994) propose a high performance algorithm called MDR (Mixed Dispatching Rule) approach for finding the best dispatching rule combination in FMS scheduling by focusing on the bottleneck machine.

In PVSS, the task having minimum slack time from the preceding product is identified as a bottleneck task. Because the start time of the bottleneck task dominates the performance of a schedule, the tasks preceding the bottleneck task should be controlled to maintain the start time of the bottleneck. In conventional scheduling, however, the start time of the bottleneck task is directly affected by processing time variations in the tasks preceding the bottleneck, because the schedule is generated with fixed processing times and has no accommodation for possible processing time extensions in the future. In S2, the tasks which precede the bottleneck task and have no waiting constraint are scheduled in consideration of

upper side processing time variations of the tasks and lower side processing time variations in the preceding products, to prepare the potential processing time variations from the average processing times and to keep the start time of the bottleneck task as scheduled. Figure 6.2(b) shows a simple example of how S2 works. In this example, the task at unit C of product Y (TCY) is identified as a bottleneck task in product Y. If the task at unit B of product Y (TBY) has no waiting time constraint and is expected to extend up to Z1 time units under normal operation, a schedule is developed as shown by the dotted line to absorb the upper side processing time variation of TBY and to keep the start time of TCY. Furthermore, if the task at unit C of product X (TCX) is expected to shorten up to Z2 time units, S2 shifts the start times of TAY and TBY to the left by Z2 time units in addition to the Z1 time units. If both TAY and TBY have ZW constraints, S2 is not applied because any forward time shift causes the violation of the ZW constraint.

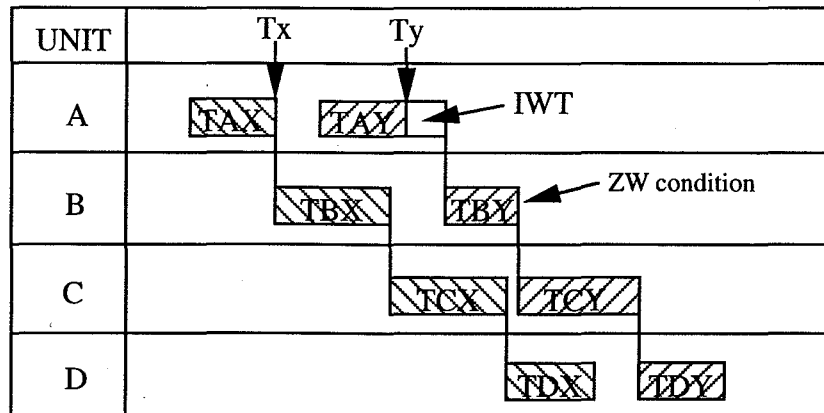
(2) Algorithm of schedule modification

Figure 6.3 shows the general flow of the schedule modification proposed in PVSS.

PVSS modifies the working schedule on a product-by-product basis in the order of product processing sequence specified by the working schedule. The working schedule is assumed to be provided. For example, if m products are scheduled by a working schedule and the first $n-1$ products have already been finished, the algorithm modifies the schedule of the remaining $m-n+1$ products beginning from n th product in the processing sequence.

In the algorithm, steps 1 and 8 specify the target product (n) for schedule modification. Step 8 also determines the termination of the algorithm. The algorithm is terminated when n becomes more than the number of products to be produced specified by a working schedule. Otherwise, the algorithm repeats the procedures from Step 2.

Steps 2, 3, and 4 are the procedures to obtain the shift time (S_{time}) for the target product based on the slack time required to satisfy the waiting time constraints between tasks in the product under processing time variations.



IWT: Intentional waiting time to control the start time of TBY

Figure 6.2(a) An example of schedule modification by Strategy 1 (S1)

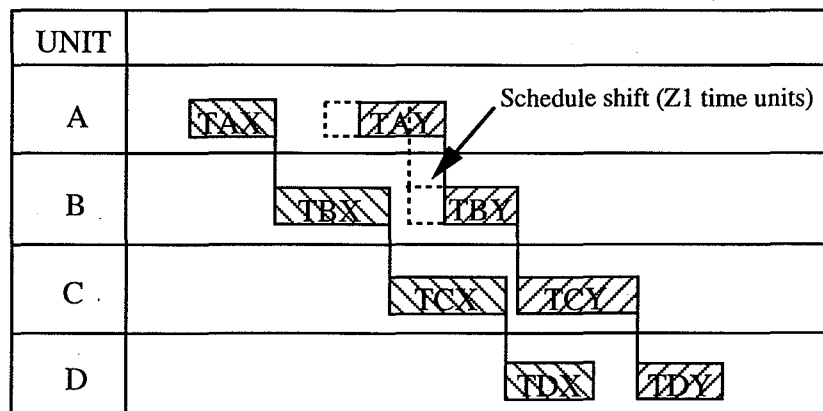


Figure 6.2(b) An example of schedule modification by Strategy 2 (S2)

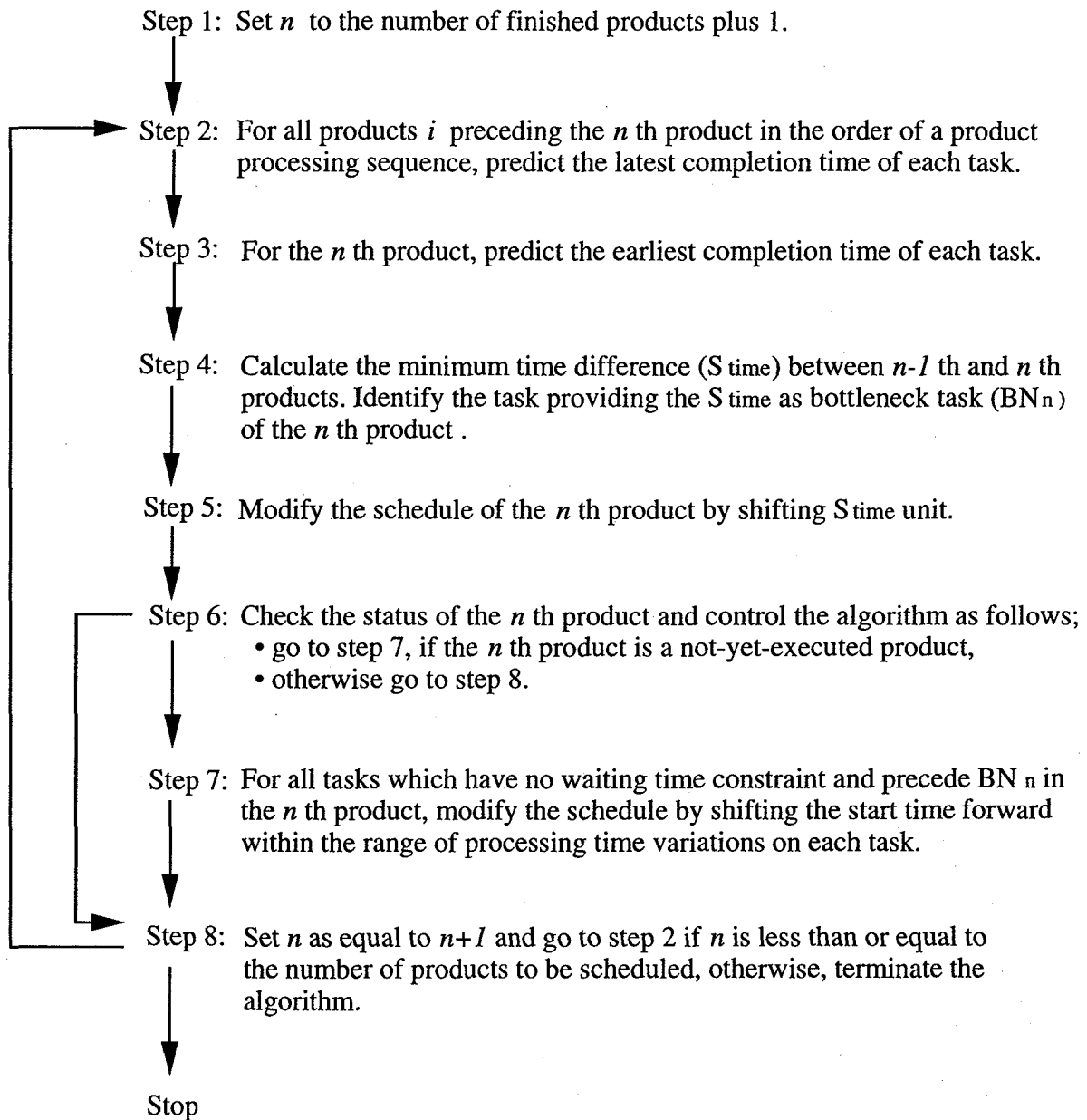


Figure 6.3 Schedule modification algorithm

The procedures to obtain S_{time} are quite simple in case of no waiting time constraints in all tasks, because no slack time evaluation is required. The S_{time} for the n th product in a product processing sequence can basically be obtained as the minimum time difference between scheduled or actual completion time (CT) of the $n-1$ th product and the scheduled start time (ST) of the n th product through all the tasks (NT) for a product. The following shows a basic equation to obtain the S_{time} by a simple procedure.

$$S_{time} = \min_{j=1,NT} [ST_{nj} - CT_{n-1 j}]$$

However, on-line scheduling systems are not effective for schedules modified by this simple procedure if tasks have processing time variations and waiting time constraints. As an example shown in Figure 6.4, the simple procedure defines the S_{time} for the n th product at T_x time and modifies the schedule of the product as shown in the figure. The modified schedule, however, will fail if all the tasks have ZW constraints and processing time variations. For example, a processing time increase in TCX or decrease in TBY from the scheduled times will cause a waiting time in TBY even when controlled by using on-line scheduling systems, because it is impossible to modify the schedule of the n th product after executing production at time T_y . This example indicates that even on-line scheduling systems cannot control a schedule to satisfy the waiting time constraints between tasks under processing time variations if the schedule is modified by a simple procedure which uses fixed processing times and no slack time insertion to obtain an S_{time} .

The procedures in Steps 2, 3, and 4 proposed in this algorithm are developed to obtain a reasonable S_{time} which satisfies the waiting time constraints between tasks in a target product under processing time variations without substantial performance degradation based on Strategy 1. Steps 2 and 3 project the latest completion time (LC) of each task for the $n-1$ th product and earliest completion time (EC) of each task for the n th product, respectively. Step 4 defines the S_{time} of the n th product as the minimum time difference between LC of the $j+1$ th task for the $n-1$ th product projected in step 2 and EC of the j th task for the n th

product projected in step 3. For example, in Figure 6.5, S_{time} for the n th product is defined by the time difference between EC of task TBY and LC of task TCX. The S_{time} obtained by Steps 2, 3, and 4 indicates the differences between the current and reasonable slack times to absorb processing time variations in the $n+1$ th and n th products in the future. Schedule modification by shifting the S_{time} unit, therefore, provides a schedule which satisfies waiting time constraints between tasks under processing time variations without substantial performance degradation.

The following describes the procedures of Step 2 through 7 in detail.

Step 2 projects LC of the j th task for the i th product (T_{ij}) which precedes the n th product based on the current process state and the upper processing time variation (UPT) from the normal processing time (NPT) of the task. The procedure is executed in the order of task sequence and product processing sequence. The algorithm applies different procedures to obtain LCs depending on the status of the tasks and the waiting time constraints specified in each task. However, the basic procedures to calculate LC for not-yet-executed tasks are summarized in the following two equations.

For tasks with waiting time constraint:

$$LC_{ij} = \text{MAX} [LC_{i-1, j-1} + NPT_{ij} + UPT_{ij}, LC_{i-1, j+1}]$$

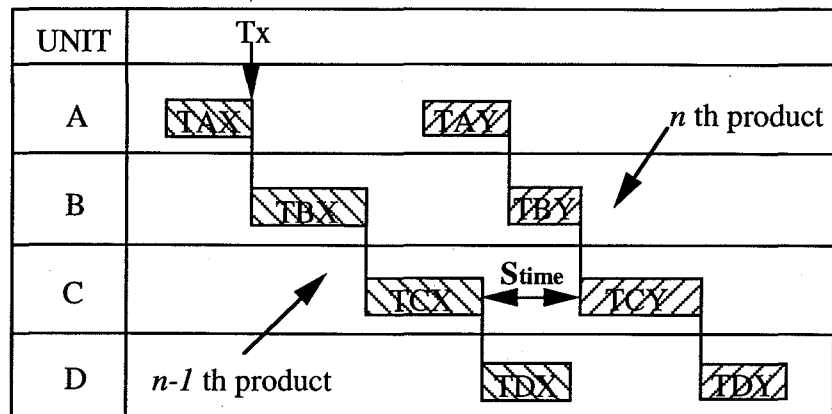
For tasks with no waiting time constraint:

$$LC_{ij} = \text{MAX} [LC_{i-1, j-1} + NPT_{ij} + \text{MAX} [IWT_{ij}, UPT_{ij}], LC_{i-1, j+1}]$$

The procedure, as shown in Figure 6.2(a), considers intentional waiting times (IWT) in the tasks with no waiting time constraint to control the waiting times of the immediately succeeding tasks according to strategy 1 proposed in this algorithm. The following shows the equation to obtain IWT_{ij} . (In the equation, LPT indicates the lower side processing time variation from NPT of each task.)

$$IWT_{ij} = \text{MAX} \left[LC_{i-1, m+1} - (LC_{i, j-1} + NPT_{ij} + \sum_{h=j+1}^m (NPT_{ih} - LPT_{ih})), 0 \right]$$

Schedule before modification



Time shift as S_{time} unit on
n th product

Schedule after modification

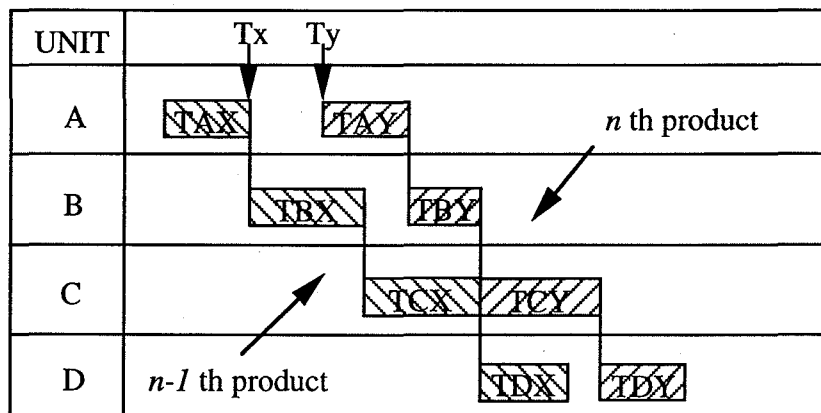


Figure 6.4 Shift time (S_{time}) and schedule modification with fixed processing times

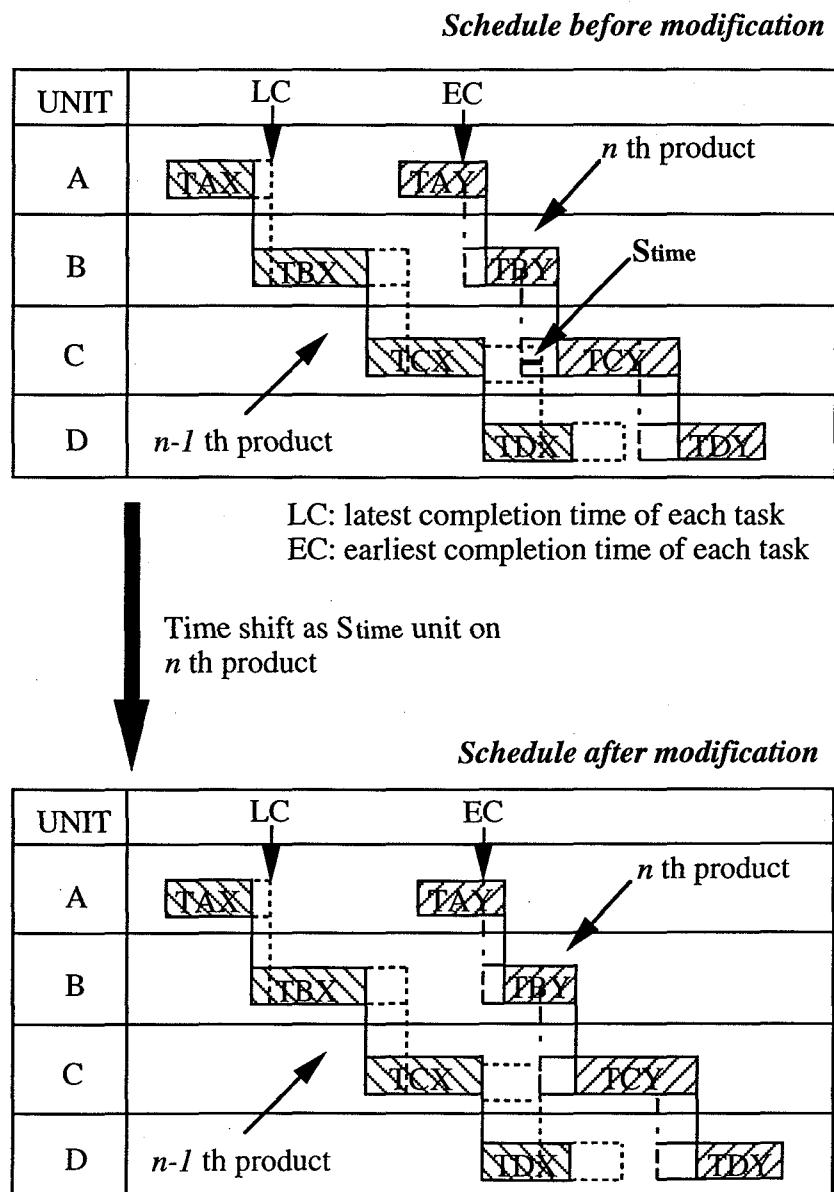


Figure 6.5 Shift time (S time) and schedule modification based on processing time variations by PVSS

In the above equation, IWT_{ij} is searched between the task $j+1$ and the task jx , which immediately succeeds the task $j+1$ and has waiting time constraint, but the next task $jx+1$ has no waiting time constraint because the task $jx+1$ possibly has another IWT for the tasks succeeding $jx+1$.

Step 3 projects EC of each task of the n th product based on LPT. The procedure considers no processing time variations in the tasks having no waiting time constraint. In Strategy 1, those tasks control their completion times by dynamically adjusting IWTs at short intervals based on the current process state. Therefore no consideration of processing time variations for those tasks are required in advance. The following shows the basic equations for not-yet-executed tasks, to calculate EC.

For a task j with a waiting time constraint:

$$EC_{nj} = EC_{n\ j-1} + NPT_{nj} - LPT_{nj}$$

For a task j with no waiting time constraint:

$$EC_{nj} = EC_{n\ j-1} + NPT_{nj}$$

Step 4 specifies S_{time} between the $n-1$ th product and the n th product in a product processing sequence. In addition, the task which provides the S_{time} is identified as a bottleneck task of the n th product (BN_n). S_{time} is basically obtained, as shown in Figure 6.5, as the minimum time difference between the latest completion time of the $n-1$ th product and the earliest completion time of the n th product projected at Steps 2 and 3, respectively. For the FW constraint, S_{time} is adjusted by the maximum waiting time (MWT) specified in each task. S_{time} should be zero or negative if the n th product is an executing-product or an executed-product because no forward time shifts are available in Step 5 in this case. Therefore the basic procedure to calculate S_{time} is summarized in the following two equations.

If n th product is a not-yet-executed product:

$$S_{time} = \text{MIN}[EC_{nj} - LC_{n-1\ j+1} + MWT_{nj}]$$

($j \in$ not-executed task except the last sequence task)

If n th product is an executed-product or executing-product:

$$S_{time} = \text{MIN}[EC_{nj} - LC_{n-1\ j+1} + MWT_{nj}, 0]$$

($j \in$ not-executed task except the last sequence task)

Step 5 is the procedure to modify the schedule of the n th product by shifting it S_{time} unit on the scheduled start time (ST) and completion time (CT) of each task of the product. For example in Figure 6.5, the schedule of the n th product is shifted backward by S_{time} unit. The basic procedures for such time shifts are summarized in the following equations.

For an executing task:

$$CT_{nj} = CT_{nj} - S_{time}$$

($j \in$ executing task in n th product)

For a not-yet-executed task:

$$ST_{nj} = ST_{nj} - S_{time}$$

$$CT_{nj} = CT_{nj} - S_{time}$$

($j \in$ not-executed task in n th product)

Step 7 is the procedure to modify a schedule based on Strategy 2 proposed in this chapter. The procedure tries to shift forward the ST of the tasks having no waiting time constraints and which precede the bottleneck task, within the range of S_{time} obtained among the tasks preceding the bottleneck, as shown in Figure 6.2(b). However, the shift time is limited to the time calculated by adding the upper processing time variation of the target task (T_{nj}) and the lower processing time variation of the next order task in the immediately preceding product ($T_{n-1\ j+1}$), to avoid the insertion of unnecessary waiting time for each task. Strategy 2 is not applied for tasks with waiting time constraints because forward time shifts on those tasks could cause additional slack times and cancel the effects of the time

shifts in Step 5. The procedures for Step 7 are basically the same as the procedures employed in Steps 4 and 5.

However, the algorithm applies the procedure for not-yet-executed product only, because no forward time shift is available for executed or executing products. Step 6 decides the applicability of S2 in this algorithm.

VCM in FOSS, described in Chapter 4, confirms the feasibility of the schedule modified by PVSS. The items to be considered, for example, are constraints on due dates and resource availability. If the modified schedule is feasible, VCM will register the schedule as the new working schedule, and each process control system will execute production based on the schedule. Otherwise, VCM notifies ASGM in FOSS to select a different schedule modification strategy, such as a re-sequencing strategy.

6.4 Evaluation by Simulation Experiments

In this section, the effectiveness of the PVSS is tested on a simple multiproduct process model by simulation. In addition, the results of experiments are evaluated and discussed.

6.4.1 Design of Experiments

(1) Multiproduct process model for experiments

In the experiments, a simple multiproduct process model consisting of three tasks is employed as shown in Figure 6.6. In the model, each task is divided into three processing operations, i.e., charging, chemical processing, and discharging. Each task is assumed to be processed on a different unit. For the model building, a discrete event simulation technique (Bratley *et al.*, 1987) is used.

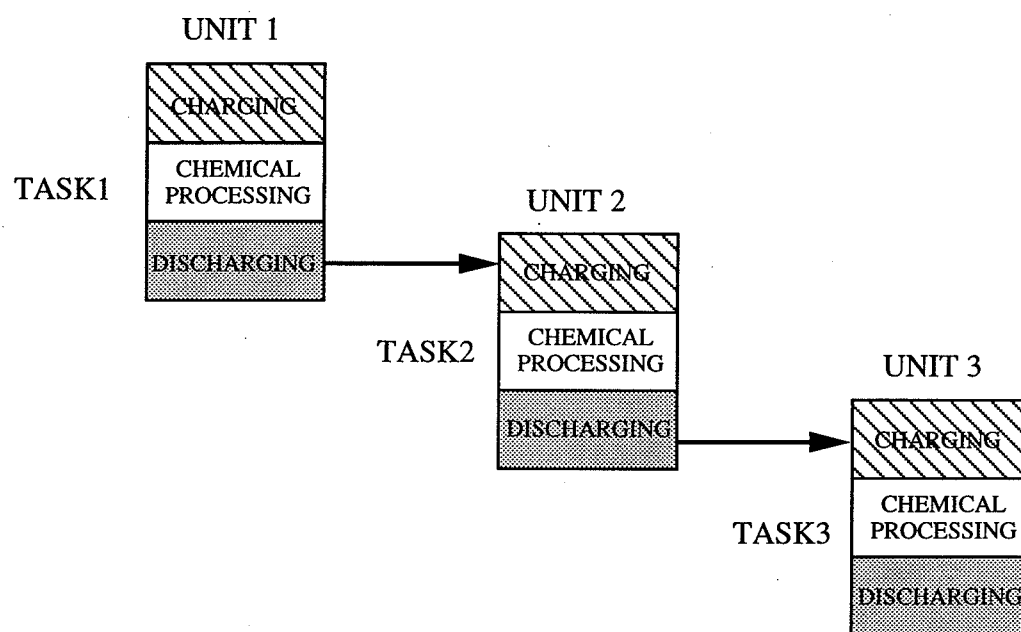


Figure 6.6 Task network for experiments

(2) Factors in experiments

There are five factors in the experiments, i.e., product type, product campaign, degree of processing time variation, waiting time constraint, and on-line scheduling system. The detailed conditions of the experimental factors are as follows.

(a) Products and processing time

Four products with the task network shown in Figure 6.6 but different processing times are defined. These products are called Product A, B, C, and D in the following sections. Table 6.1 describes the mean processing times of each product. Processing time includes charging and discharging times. In experiments, one time unit for charging and one time unit for discharging in all tasks are defined. For example, the chemical processing time is 20 time units and charging and discharging times are total two time units at task 1 for product A.

Table 6.1 Processing times

[Time unit]			
Task	Task1	Task2	Task3
Product			
• Product A			
Mean	22.00	22.0	22.00
PV1	21.00 - 23.00	21.00 - 23.00	21.00 - 23.00
PV2	19.00 - 25.00	19.00 - 25.00	19.00 - 25.00
• Product B			
Mean	32.00	17.0	17.00
PV1	30.50 - 33.50	16.25 - 17.75	16.25 - 17.75
PV2	27.50 - 36.50	14.75 - 19.25	14.75 - 19.25
• Product C			
Mean	17.00	32.0	17.00
PV1	16.25 - 17.75	30.50 - 33.50	16.25 - 17.75
PV2	14.75 - 19.25	27.50 - 36.50	14.75 - 19.25
• Product D			
Mean	17.00	17.0	32.00
PV1	16.25 - 17.75	16.25 - 17.75	30.50 - 33.50
PV2	14.75 - 19.25	14.75 - 19.25	27.50 - 36.50

(b) *Product campaign*

In experiments, the following four types of single product campaign, named SPC1, SPC2, SPC3, and SPC4, and one type of mixed product campaign named MPC1 are considered. In the single product campaign, the same product is repeatedly produced. On the other hand, more than one product is repeatedly produced in the mixed product campaign.

SPC1: Production of Product A is repeated 25 times

SPC2: Production of Product B is repeated 25 times

SPC3: Production of Product C is repeated 25 times

SPC4: Production of Product D is repeated 25 times

MPC1: Production of Products A, B, C, and D is repeated 5 times in a cycle of

A -> B -> C -> D

Each product campaign is designed to have different bottleneck tasks. More specifically, all tasks can be regarded as bottleneck tasks in SPC1. Task 1, task 2, and task 3 can be regarded as bottlenecks in SPC2, SPC3, and SPC4, respectively. On the other hand, MPC1 has bottlenecks in different tasks through the campaign because of the combination of different kinds of product.

PVSS can handle change-overs or clean-ups by regrading them with process operations of products or tasks. In the experiments, however, no change-overs or clean-ups in all product campaigns are considered to test the basic behavior of PVSS.

(c) *Processing time variation (PV)*

For processing time variations, the following two scenarios, named PV1 and PV2 are defined.

PV1: each processing time has uniformly distributed variations up to $\pm 5\%$ of the mean processing time

PV2: each processing time has uniformly distributed variations up to $\pm 15\%$ of the mean processing time

In the above scenarios, processing time variations of all tasks are assumed to be independent of each other. In addition, charging and discharging times are defined as having no processing time variation. For example, a task which has 22 time units for mean processing time and total 2 time units for charging and discharging operations will have processing time variations between 21 and 23 time units in the scenario of PV1. Table 6.1 describes the processing time variation of each task in PV1 and PV2.

(d) *Waiting time constraints (WTC)*

For waiting time constraints, the following four scenarios referred to as WTC1, WTC2, WTC3, and WTC4 are defined.

WTC1: no waiting time constraint for any task

WTC2: task 1 has a waiting time constraint

WTC3: task 2 has a waiting time constraint

WTC4: task 1 and task 2 have waiting time constraints

In each waiting time constraint except WTC1, three kinds of maximum waiting time (MWT) are specified for experiments, depending on the scenario of processing time variations as follows;

For PV1: 0.0 (ZW), 1.0, and 1.5 time units,

For PV2: 0.0 (ZW), 3.0, and 4.5 time units.

(e) *Benchmark scheduling system (SS)*

POMA (Cott and Macchietto, 1989) introduced in Section 6.2 is employed as a benchmark on-line scheduling system with PVSS. It has been proven that POMA reduces the effects of processing time variations significantly, especially under ZW constraints.

(3) *Performance criteria*

The following criteria are employed for evaluating the performance of PVSS and POMA.

(a) *Mean makespan (MS)*

$$\frac{NR}{\sum_{k=1} (C_k - S_k) / NR}$$

(b) *Variance of makespan*

$$\frac{NR}{\{ \sum_{k=1} (C_k - S_k)^2 - NR * MS^2 \} / (NR-1)}$$

(c) *Maximum waiting time in tasks with waiting time constraints*

$$\begin{aligned} \text{MAX } [WT_{k(i,j)}] \quad & 1 \leq i \leq NP \\ & 1 \leq j \leq NT \\ & 1 \leq k \leq NR \end{aligned}$$

where C : completion time of a product campaign, S : start time of a product campaign, WT: waiting time for a task with waiting time constraint, zero for others, NT: the number of tasks per a product, NP: the number of products in a product campaign, NR: the number of replications

The first and second performance criteria are processing-time-based measures and widely used in scheduling. It is possible to say that minimizing the first criterion is equivalent to minimizing the sum of equipment idle time, to maximizing average production

rate. The third performance criterion is to evaluate the performance of waiting time control of PVSS and a benchmark system.

(4) The number of replications

To reduce the effects of random numbers used to generate processing times in the simulations, 100 replications were run for each set of experimental factors.

The experimental results were compared with the results of 50 replications and no significant difference was observed among them. Therefore the experimental results obtained are considered reliable enough to evaluate the performance of PVSS.

6.4.2 Results of Experiments

Tables 6.2, 6.3, 6.4, 6.5, and 6.6 summarize the results of experiments in each product campaign scenario. In those tables, WTC, SS, PV, and MWT show, respectively, the scenario on waiting time constraint, on-line scheduling system, processing time variation, and maximum waiting time. Because the performance of POMA is independent of MWT constraints, only one result per processing time variation (PV) is shown in the tables.

(1) Performance on waiting time control

Throughout the experiments, PVSS successfully controls waiting times within the range specified as constraints under processing time variations. For example in Table 6.2, the maximum waiting times in a task in WTC2 under PV1 and PVSS scenarios were 0.0, 0.99, 1.49 against the allowable maximum waiting time constraint, respectively, 0.0 (ZW), 1.0, and 1.5.

On the other hand, POMA could not satisfy the waiting time constraints throughout the experiments. Figure 6.7, for example, shows maximum waiting time deviations by POMA from the ranges specified under PV2.

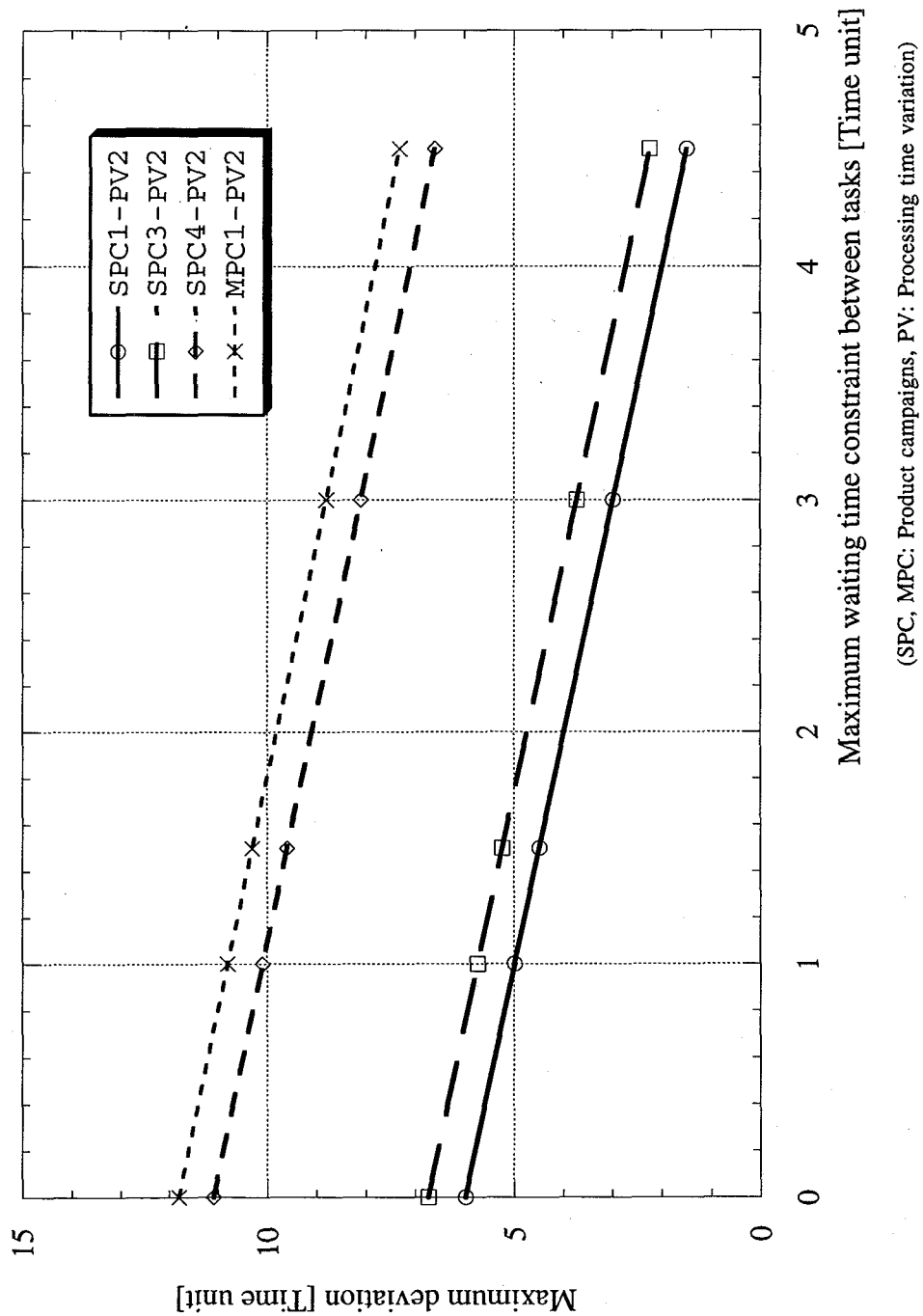


Figure 6.7 Maximum deviation from waiting time constraint under PV2 in case of no slack time insertion

POMA modifies a working schedule based on a process state predicted by using fixed processing times which are usually impossible to be specified in advance. In addition, POMA considers no slack time between products to absorb the schedule changes caused by processing time variations. POMA's technical basis as an on-line scheduling system can be regarded as a simple feedback control with no consideration of future processing time variations. Therefore it is obvious that POMA can satisfy the waiting time constraint of each task only incidentally.

For this reason, the results of experiments suggest that on-line scheduling systems should modify a working schedule in consideration of possible processing time variations in the future, like PVSS, if the target products have waiting time constraints.

(2) Effectiveness of Strategy 1 and Strategy 2

The effectiveness of Strategy 1 (S1) and Strategy 2 (S2) implemented in the schedule modification algorithm of PVSS was tested by comparing the experimental results on PVSS with the scenarios of WTC3 and WTC4.

In WTC3, the first task of each product has no waiting time constraint and the second task has a waiting time constraint. In WTC4, on the other hand, the first two tasks have waiting time constraints. S1 is applicable for the tasks with no waiting time constraint preceding the tasks with waiting time constraints and S2 is applicable for the tasks with no waiting time constraint. Therefore S1 and S2 are effective for WTC3 but not for WTC4. If S1 and S2 are not applied for WTC3, the experimental results of WTC3 and WTC4 on PVSS are the same. For this reason, the differences of the experimental results between WTC3 and WTC4 can be regraded as the effects of S1 and S2.

The comparison was made based on the percentage improvement of WTC3 against WTC4 on the mean makespans observed through experiments under the scenario of PVSS. (The percentage improvement by S1 and S2 = $[1.0 - (\text{mean makespan under WTC3 and PVSS}) / (\text{mean makespan under WTC4 and PVSS})] * 100.0$) For this comparison, SPC1, SPC3, SPC4, and MPC1 are selected. SPC2 was not selected because it has a bottleneck at the first task and offers no chance to apply S1 and S2. The results show performance

improvement throughout the experiments in comparison. Table 6.7 shows a summary of the percentage improvement in each scenario by S1 and S2. For example, the mean makespans in WTC3 and WTC4 are, respectively, 832.01 and 885.75 under PV1 and SPC3 scenario as shown in Table 6.4. Therefore the percentage improvement is 6.07% in this case. The overall performance improvements achieved by S1 and S2 were between 1.72% (MPC1 under PV1 and 1.5 MWT scenario) and 16.23% (SPC3 under PV2 and ZW scenario).

In addition, the effectiveness of S1 and S2 was statistically tested using Welch's method (Kleijnen, 1987) on the mean makespans of WTC3 and WTC4 under the scenario of PVSS. Welch's method is a statistical method to test the average of sampling data with possibly unequal variances. The results of the test show significant differences between WTC3 and WTC4 at 99% confidence level for all experimental scenarios. For example in Table 6.5, mean makespans in WTC3 and WTC4 under PV1 and ZW scenario are 886.02 and 904.25, respectively. In addition, variances of the mean makespan are 6.22 and 10.91, respectively. Therefore the statistical t' according to Welch's method is 43.82 with 184.2 degrees of freedom. Because the Student t with 184.2 degrees of freedom under 1% risk is approximately 2.57, the result shows a significant difference achieved by S1 under PV1 and ZW scenario.

In addition to the above observation, the effectiveness of PVSS was tested with POMA under scenario WTC1. Because no tasks have a waiting time constraint in WTC1, the differences between PVSS and POMA can be regarded as the effects of S2.

The comparison of PVSS with POMA under the scenario of WTC1 was made based on the percentage improvement on the mean makespans observed through the experiments. (The percentage improvement = $[1.0 - (\text{mean makespan under WTC1 and PVSS}) / (\text{mean makespan under WTC1 and POMA})] * 100.0$) For the comparison, SPC3, SPC4, and MPC1 are selected as product campaigns. In SPC1, all tasks can be regarded as bottleneck tasks. In SPC2, the first task is a bottleneck task. Because there is no task preceding a bottleneck task in SPC1 and SPC2, S2 has no opportunity to be applied. The results show a performance improvement by S2 between 0.84% (SPC4 under PV1) and 3.35% (SPC3 under PV2) throughout the experiments, as summarized in Table 6.8.

In addition, the effectiveness of S2 was statistically tested again by using Welch's method on the mean makespans by PVSS and POMA under the scenario of WTC1. The results of the test show significant differences between PVSS and POMA at 99% confidence level.

From the above observations, the results of experiments support the following.

- Applying a scheduling strategy which utilizes tasks with no waiting time constraint to dynamically adjust the start times of the succeeding tasks, as in S1, improves the performance of on-line scheduling systems significantly under processing time variations.
- Applying a scheduling strategy which generates a schedule to keep the scheduled start time of a bottleneck task under processing time variations, as in S2, improves the performance of on-line scheduling systems.

(3) Effects of intentional slack times by PVSS on schedule evaluation

Some product campaign structures require no intentional slack times to satisfy waiting time constraints under processing time variations by PVSS. For example, as shown in Table 6.3, no violation of waiting time constraints is observed in SPC2 without any intentional slack times. Neither is any violation observed in WTC3 of SPC3 as shown in Table 6.4. The schedules of the examples already have slack times enough to absorb processing time variations within the structure of product campaigns. For example, SPC2 consists of products whose bottlenecks are the first tasks. Because the processing time at the first task is much longer than the processing times at the succeeding tasks, the slack time sufficient to absorb processing time variations in the succeeding tasks are already embedded in the structure of the product campaign in SPC2.

Except for the above particular cases, however, intentional slack times are usually placed between products in a working schedule to satisfy waiting time constraints under processing time variations in PVSS. Although the minimum slack times are specified based on the current process state and the information on the products to be scheduled, PVSS would increase makespan to some extent. Table 6.9 shows the percentage increase on mean

makespan by each waiting time constraint against the mean makespan by WTC1. (Percentage increase = $\{(\text{mean makespan by each WTC and PVSS})/(\text{mean makespan by WTC1 and PVSS}) - 1.0\} * 100.0$) Because WTC1 specifies no waiting time constraints for all tasks, PVSS creates no intentional slack times. Therefore Table 6.9 shows the effects of intentional slack times by PVSS on mean makespan under each waiting time scenario. For example in MPC1, mean makespans by WTC1 and WTC2 under PV1 scenario are 566.66 and 589.63 as shown in Table 6.6. In this case, PVSS increases by 22.97 time units (4.05%) to satisfy ZW constraint of the first task in a task network under processing time variations of PV1. Through experiments, as shown in Table 6.9, up to 40.47% (SPC1 under PV2 and WTC4 scenario) increase in mean makespan is observed.

The schedule generated by conventional methods would tend to be a squeezed schedule which has fewer slack times than actually required to absorb future processing time variations. Therefore it would seem to perform better than a schedule containing adequate slack times. However, the schedule with fewer slack times cannot guarantee feasibility under processing time variations. The results of experiments suggest that a schedule should be evaluated in consideration of reasonable slack times to absorb possible processing time variations, although the insertion of slack times seems to inhibit the performance of a schedule.

Table 6.2 Results of experiments
(Product campaign scenario: SPC1)

				[Time unit]		
Criteria				Makespan		Maximum
				Mean	Variance	Waiting time(*)
Scenarios						
WTC	SS	PV	MWT			
WTC1	PVSS	PV1	---	603.25	4.85	---
		PV2	---	625.74	43.25	---
	POMA	PV1	---	603.25	4.85	---
		PV2	---	625.74	43.25	---
WTC2	PVSS	PV1	0.0	639.69	7.92	0.0
			1.0	617.06	5.92	0.99
			1.5	608.43	4.63	1.49
		PV2	0.0	735.08	72.04	0.0
			3.0	667.18	53.70	2.98
			4.5	641.29	42.38	4.48
	POMA	PV1	---	603.25	4.85	1.99
		PV2	---	625.74	43.25	5.98
WTC3	PVSS	PV1	0.0	641.81	9.82	0.0
			1.0	625.47	12.87	0.93
			1.5	616.60	8.55	1.43
		PV2	0.0	741.42	88.80	0.0
			3.0	692.40	117.17	2.78
			4.5	665.80	77.05	4.28
	POMA	PV1	---	603.25	4.85	1.93
		PV2	---	625.74	43.25	5.79
WTC4	PVSS	PV1	0.0	687.67	7.92	0.0
			1.0	663.67	8.04	0.78
			1.5	651.68	8.00	0.86
		PV2	0.0	879.00	72.49	0.0
			3.0	807.01	72.49	2.34
			4.5	771.02	72.44	3.84
	POMA	PV1	---	603.25	4.85	1.99
		PV2	---	625.74	43.25	5.98

WTC: scenario of waiting time constraint, SS: online scheduling system, PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task. (*) Maximum waiting time in a task with waiting time constraint throughout the experiment.

Table 6.3 Results of experiments
(Product campaign scenario: SPC2)

				[Time unit]		
Criteria				Makespan		Maximum
				Mean	Variance	Waiting time(*)
Scenarios						
WTC	SS	PV	MWT			
WTC1	PVSS	PV1	---	831.48	15.84	---
		PV2	---	830.44	145.90	---
	POMA	PV1	---	831.48	15.84	---
		PV2	---	830.44	145.90	---
WTC2	PVSS	PV1	0.0	831.48	15.84	0.0
			1.0	831.48	15.84	0.0
			1.5	831.48	15.84	0.0
		PV2	0.0	830.44	145.90	0.0
			3.0	830.44	145.90	0.0
			4.5	830.44	145.90	0.0
	POMA	PV1	---	831.48	15.84	0.0
		PV2	---	830.44	145.90	0.0
WTC3	PVSS	PV1	0.0	831.48	15.84	0.0
			1.0	831.48	15.84	0.0
			1.5	831.48	15.84	0.0
		PV2	0.0	830.44	145.90	0.0
			3.0	830.44	145.90	0.0
			4.5	830.44	145.90	0.0
	POMA	PV1	---	831.48	15.84	0.0
		PV2	---	830.44	145.90	0.0
WTC4	PVSS	PV1	0.0	831.48	15.84	0.0
			1.0	831.48	15.84	0.0
			1.5	831.48	15.84	0.0
		PV2	0.0	830.44	145.90	0.0
			3.0	830.44	145.90	0.0
			4.5	830.44	145.90	0.0
	POMA	PV1	---	831.48	15.84	0.0
		PV2	---	830.44	145.90	0.0

WTC: scenario of waiting time constraint, SS: online scheduling system, PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task. (*) Maximum waiting time in a task with waiting time constraint throughout the experiment.

Table 6.4 Results of experiments
(Product campaign scenario: SPC3)

				[Time unit]		
Criteria				Makespan		Maximum
				Mean	Variance	Waiting time(*)
Scenarios						
WTC	SS	PV	MWT			
WTC1	PVSS	PV1	---	832.01	21.70	---
		PV2	---	832.03	195.03	---
	POMA	PV1	---	841.63	7.84	---
		PV2	---	860.88	71.56	---
WTC2	PVSS	PV1	0.0	885.75	5.41	0.0
			1.0	862.69	4.57	0.99
			1.5	852.82	4.12	1.49
		PV2	0.0	993.25	48.65	0.0
			3.0	924.07	40.89	2.98
			4.5	894.46	39.60	4.48
	POMA	PV1	---	841.63	7.84	2.24
		PV2	---	860.88	71.56	6.73
WTC3	PVSS	PV1	0.0	832.01	21.70	0.0
			1.0	832.01	21.70	0.0
			1.5	832.01	21.70	0.0
		PV2	0.0	832.03	195.03	0.0
			3.0	832.03	195.03	0.0
			4.5	832.03	195.03	0.0
	POMA	PV1	---	841.63	7.84	0.0
		PV2	---	860.88	71.56	0.0
WTC4	PVSS	PV1	0.0	885.75	5.41	0.0
			1.0	862.69	4.57	0.99
			1.5	852.82	4.12	1.49
		PV2	0.0	993.25	48.65	0.0
			3.0	924.07	40.89	2.98
			4.5	894.46	39.60	4.48
	POMA	PV1	---	841.63	7.84	2.24
		PV2	---	860.88	71.56	6.73

WTC: scenario of waiting time constraint, SS: online scheduling system, PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task. (*) Maximum waiting time in a task with waiting time constraint throughout the experiment.

Table 6.5 Results of experiments
(Product campaign scenario: SPC4)

				[Time unit]		
Criteria				Makespan		Maximum
				Mean	Variance	Waiting time(*)
Scenarios						
WTC	SS	PV	MWT			
WTC1	PVSS	PV1	---	832.52	18.71	---
		PV2	---	833.54	168.69	---
	POMA	PV1	---	839.57	7.80	---
		PV2	---	854.72	72.04	---
WTC2	PVSS	PV1	0.0	833.02	16.53	0.0
			1.0	833.02	16.53	0.0
			1.5	833.02	16.53	0.0
		PV2	0.0	835.07	147.31	0.0
			3.0	835.07	147.31	0.0
			4.5	835.07	147.31	0.0
	POMA	PV1	---	839.57	7.80	0.0
		PV2	---	854.72	72.04	0.0
WTC3	PVSS	PV1	0.0	886.02	6.22	0.0
			1.0	862.86	5.21	0.96
			1.5	853.09	5.21	1.46
		PV2	0.0	994.05	58.34	0.0
			3.0	924.55	50.18	2.87
			4.5	895.27	45.66	4.37
	POMA	PV1	---	839.57	7.80	3.70
		PV2	---	854.72	72.04	11.09
WTC4	PVSS	PV1	0.0	904.25	10.91	0.0
			1.0	880.34	10.99	0.76
			1.5	868.89	9.74	1.12
		PV2	0.0	1048.75	98.18	0.0
			3.0	977.01	96.00	1.86
			4.5	942.66	88.16	3.36
	POMA	PV1	---	839.57	7.80	3.70
		PV2	---	854.72	72.04	11.09

WTC: scenario of waiting time constraint, SS: online scheduling system, PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task. (*) Maximum waiting time in a task with waiting time constraint throughout the experiment.

Table 6.6 Results of experiments
(Product campaign scenario: MPC1)

				[Time unit]		
Criteria				Makespan		Maximum
				Mean	Variance	Waiting time(*)
Scenarios						
WTC	SS	PV	MWT			
WTC1	PVSS	PV1	---	566.66	10.91	---
		PV2	---	572.98	90.42	---
	POMA	PV1	---	584.26	7.88	---
		PV2	---	592.61	66.75	---
WTC2	PVSS	PV1	0.0	589.63	6.12	0.0
			1.0	580.29	6.22	0.97
			1.5	576.67	6.28	1.47
		PV2	0.0	634.44	54.51	0.0
			3.0	599.97	49.15	2.92
			4.5	587.86	50.34	4.42
	POMA	PV1	---	584.26	7.88	2.22
		PV2	---	592.61	66.75	6.45
WTC3	PVSS	PV1	0.0	603.90	14.59	0.0
			1.0	597.87	7.66	0.91
			1.5	593.80	6.55	1.41
		PV2	0.0	656.81	112.89	0.0
			3.0	633.24	60.04	2.77
			4.5	620.18	56.42	4.27
	POMA	PV1	---	584.26	7.88	3.93
		PV2	---	592.61	66.75	11.80
WTC4	PVSS	PV1	0.0	624.00	6.38	0.0
			1.0	610.34	6.30	0.88
			1.5	604.19	6.26	1.38
		PV2	0.0	740.12	71.96	0.0
			3.0	691.63	62.83	2.65
			4.5	666.81	60.40	4.15
	POMA	PV1	---	584.26	7.88	3.93
		PV2	---	592.61	66.75	11.80

WTC: scenario of waiting time constraint, SS: online scheduling system, PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task. (*) Maximum waiting time in a task with waiting time constraint throughout the experiment.

Table 6.7 Percentage increase on mean makespan
by Strategies 1 and 2

		[%]			
Product campaign		SPC1	SPC3	SPC4	MPC1
PV	MWT				
PV1	0.0	6.67	6.07	2.02	3.22
	1.0	5.76	3.56	1.99	2.06
	1.5	5.38	2.44	1.82	1.72
PV2	0.0	15.65	16.23	5.22	11.26
	3.0	14.20	9.96	5.37	8.44
	4.5	13.65	6.98	5.03	6.99

PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task, Percentage increase = $\{1.0 - (\text{mean makespan under WTC3 and PVSS}) / (\text{mean makespan under WTC4 and PVSS})\} * 100.0$

Table 6.8 Percentage increase on mean makespan
by Strategy 2 under WTC1

		[%]		
Product campaign		SPC3	SPC4	MPC1
PV				
PV1		1.14	0.84	3.01
PV2		3.35	2.48	3.31

PV: scenario of processing time variability, WTC: scenario of waiting time constraint, Percentage increase = $\{1.0 - (\text{mean makespan under WTC1 and PVSS}) / (\text{mean makespan under WTC1 and POMA})\} * 100.0$

Table 6.9 Percentage increase on mean makespan by slack time insertion

			[%]				
Product campaign			SPC1	SPC2	SPC3	SPC4	MPC1
WTC	PV	MWT					
WTC2	PV1	0.0	6.04	0.0	6.46	0.06	4.05
		1.0	2.29	0.0	3.69	0.06	2.41
		1.5	0.86	0.0	2.50	0.06	1.77
	PV2	0.0	17.47	0.0	19.38	0.18	10.73
		3.0	6.62	0.0	11.06	0.18	4.71
		4.5	2.49	0.0	7.50	0.18	2.60
WTC3	PV1	0.0	6.39	0.0	0.0	6.43	6.57
		1.0	3.68	0.0	0.0	3.64	5.51
		1.5	2.21	0.0	0.0	2.47	4.79
	PV2	0.0	18.49	0.0	0.0	19.26	14.63
		3.0	10.07	0.0	0.0	10.92	11.05
		4.5	6.40	0.0	0.0	7.41	8.24
WTC4	PV1	0.0	13.99	0.0	6.46	8.62	10.12
		1.0	10.02	0.0	3.69	5.74	7.71
		1.5	8.03	0.0	2.50	4.24	6.62
	PV2	0.0	40.47	0.0	19.38	25.82	29.17
		3.0	28.97	0.0	11.06	17.21	20.71
		4.5	23.22	0.0	7.50	13.09	16.38

WTC: scenario of waiting time constraint, PV: scenario of processing time variability, MWT: allowable maximum waiting time in each task

Percentage increase = {(mean makespan by each WTC) / (mean makespan by WTC1) - 1.0} * 100.0

6.5 Chapter Summary

In this chapter, a process-variability-based on-line scheduling system (PVSS) designed for a scheduling modification strategy within a structure of the on-line scheduling system (FOSS), described in Chapter 4, is proposed. PVSS provides a new feedforward mechanism which inserts certain slack times between products to control waiting times between tasks within the ranges specified by a recipe without substantial performance degradation under processing time variations. It evaluates reasonable slack times to be inserted between products at short intervals and at the completion of each task, and then dynamically modifies the start and completion times of not-yet-finished tasks to insert those reasonable slack times. In addition, PVSS employs Strategy 1 and Strategy 2 to avoid substantial performance degradation due to the insertion of slack times.

The performance of PVSS and limitations of a conventional time shift approach (POMA) to waiting time constraints were tested with a simple multiproduct process model by using discrete event simulation. The results of experiments show that PVSS successfully controls the waiting times between tasks within the ranges specified by a recipe under processing time variations. On the contrary, a conventional time shift approach fails to control the waiting times. In addition, PVSS increases the performance of a schedule compared to a conventional time shift approach because the Strategies 1 and 2 in PVSS work well to reduce idle times between products caused by processing time variations.

From the results of experiments, the following observations can be made.

- A conventional simple feedback control logic cannot control the waiting times between tasks within the ranges specified by a recipe under processing time variations. To overcome the problem, on-line scheduling systems should employ a feedforward control mechanism, which predicts the possible waiting times between tasks in the future, for schedule modifications, as proposed in this chapter.

- Strategy 1, which utilizes tasks with no waiting time constraint to dynamically adjust the start times of the succeeding tasks, improves the performance of schedule modifications significantly under processing time variations.
- The bottleneck oriented approach, proposed as Strategy 2 in this chapter, is effective for schedule modifications within on-line scheduling systems to improve the performance of the system under processing time variations.
- To obtain a feasible schedule under processing time variations, the schedules should be evaluated in consideration of reasonable slack times to satisfy waiting time constraints.

Because the mechanism of PVSS is a simple and general form, it can be employed in practical multiproduct batch processes in many ways. For example, PVSS can be employed as a module to handle processing time variations of multiproduct batch processes within complex production systems.

PART 4:

CONCLUSIONS OF THE RESEARCH

Chapter 7

CONCLUSIONS

7.1 Summary and Conclusions of the Research

This research proposes a framework for the intelligent scheduling system which can effectively manage flexible production systems under the frequent changes of production information inherent in today's dynamic business environment. The framework is applied to establish intelligent scheduling systems in FMS and batch processes in the chemical industries based on the characteristics of each flexible production system. As a result, this research proposes the detailed structures of the intelligent scheduling system for FMS and batch processes within the framework.

For an intelligent scheduling system in FMS, a real-time scheduling system was investigated, and a transient based real-time scheduling system was developed. The system defines the next short period for an instruction revision based on the system transient state, and dynamically selects a dispatching rule as the instruction for the short period using discrete event simulation to respond to the changes of system state. Furthermore, a mixed dispatching rule approach, together with a search algorithm for the approach was proposed, to improve the performance of the real-time scheduling system. This approach can assign different dispatching rules for each machine. The search algorithm based on a bottleneck-oriented concept efficiently searches out an adequate mixed dispatching rule under the latest production information.

For an intelligent scheduling system in batch processes, an on-line scheduling system which modifies a schedule in a certain short period was investigated. As a result, a

fundamental structure for an on-line scheduling system within the framework for the intelligent scheduling system was proposed. The system defines timing for a schedule revision, and then specifies different schedule modification spaces according to the disruptions or performance degradations of a schedule by the changes of production information. The schedule modification space is specified to modify a schedule effectively by limiting the schedule modification space and employing a suitable schedule modification strategy depending on the situation. For schedule modifications at different modification spaces and modification strategies, this research proposed key schedule modification mechanisms within the structure. Namely, the EDR (Extended Dispatching Rule) approach corresponding to the re-scheduling strategy at the production sequence space (SP3) in Figure 4.3 was developed. The EDR approach can generate a complete schedule under various scheduling constraints rapidly by a dispatching-simulation mechanism. In addition, PVSS (Process-Variability-based on-line Scheduling System) corresponding to the time shift strategy at the task start & completion times space (SP1) was developed. PVSS modifies a schedule with waiting time constraints between tasks under processing time variations by a feedforward time shift mechanism.

The developed systems were evaluated by simulation with process models. As a result, the following observations are gained.

(1) Effectiveness of an intelligent scheduling system

- The framework proposed in Chapter 1 for the intelligent scheduling system works effectively for management of flexible production systems via several system structures within the framework. (Different structures could exist because of the diversity of flexible production systems.)
- In FMS, a real-time scheduling system which generates the instruction of suitable operation rules for the process control system in a short interval based on the latest production information is effective as a structure within the framework.

- In batch processes, an on-line scheduling system which generates the instruction of a complete schedule for the process control system in a short interval based on the latest production information is effective as a structure within the framework.

(2) *Effectiveness of a real-time scheduling system as an intelligent scheduling system in FMS*

- A real-time scheduling system which replaces dispatching rules as revisions of the instruction to a process control system at various short periods could perform better than a conventional single-pass scheduling system where a single dispatching rule is used throughout the entire production period.
- In a real-time scheduling system, defining the next scheduling interval based on system transient state makes the performance of the system much more stable than defining an interval by a constant short period.
- The problem of censored data strongly affects the selection of dispatching rules because of the short simulation window for evaluation. Methodologies designed to eliminate the problem, such as strategies 2, 3, and 4 proposed in Chapter 2, should be employed in a real-time scheduling system in FMS.
- The MDR approach within a real-time scheduling system improves the performance of FMS for the selected performance criteria compared with the conventional approach which uses a single dispatching rule for all machines.

(3) *Effectiveness of an on-line scheduling system as an intelligent scheduling system in batch processes*

- FOSS, presented in Chapter 4 as an on-line scheduling system in batch processes, successfully defines schedule modification timing and a modification space for each timing according to the changes of a wide variety of production information, and dynamically modifies a schedule to minimize the effect of the changes.
- The EDR approach generates a short-term schedule rapidly enough to work within an on-line scheduling system for a practical size problem.

- A conventional simple feedback control logic cannot control the waiting times between tasks within the ranges specified by a recipe under processing time variations. To overcome the problem, a schedule modification mechanism within an on-line scheduling system should have a feedforward control mechanism, which predicts the possible waiting times between tasks in the future and takes corrective actions at short intervals, as proposed in Chapter 6.
- To obtain a feasible schedule under processing time variations, the schedule should be evaluated in consideration of reasonable slack times to absorb the processing time variations and satisfy waiting time constraints.

7.2 Promising Areas for Further Study

The following shows a few promising areas for further studies in relation to this research.

(1) *Data management mechanism*

The data associated with the intelligent scheduling system are heterogeneous data. They are changed at various events and by various systems. The intelligent scheduling system requires a data management mechanism which realizes efficient data acquisition, storage, and on time data transfer to the correct modules within the system. The data transfer would be made in the distributed system environment where a wide variety of computer systems are connected via information networks. The data management mechanism under the circumstances should be studied.

(2) *Simulation mechanism*

Simulation is one of the critical techniques for the intelligent scheduling system because of its capability to predict the system situations in the future. However, simulation has basically been studied on an off-line basis. Therefore on-line simulation for the intelligent scheduling system should be studied in detail. Especially, research including

suitable simulation length depending on scheduling conditions, modeling structure to realize quick simulation execution, and methodology to evaluate the results of simulation are strongly required.

(3) Decision making mechanism

There are a large number of decision making problems in the intelligent scheduling system. For example, extension of the decision making mechanism to select a suitable schedule modification strategy in ASGM (Chapter 4) is required. Research on rule-based systems for ASGM, together with learning mechanisms to adapt to the dynamic scheduling environment are promising ways to maintain and improve the performance of FOSS. In addition, the problem of defining the timing of the instruction revision can be seen as a decision making problem. For example, a neural network technique by using on-line process data as input to a network could work effectively to resolve the problem.

REFERENCES

- Basnet, C. and J.H. Mize, Scheduling and control of flexible manufacturing systems: a critical review. *Int. J. Computer Integrated Manufacturing* **7**, 340-355 (1994).
- Bethea, R.M., B.S. Duran, and T.L. Boullion, *Statistical Methods for Engineers and Scientists*. Marcel Dekker, New York (1975).
- Blackstone, J. H., D. T. Phillips, and, G. L. Hogg, A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *Int. J. Prod. Res.* **20**, 27-45 (1982).
- Blazewicz, J., K.H. Ecker, G. Schmidt, and J. Weglarz, *Scheduling in computer and manufacturing systems*. Springer-Verlag, New York (1994)
- Bratley, P., B.L. Fox, and L.E. Schrage, *A Guide to Simulation*. Springer-Verlag, New York (1987).
- Cho, H. and R.A. Wysk, A robust adaptive scheduler for an intelligent workstation controller. *Int. J. Prod. Res.* **31**, 771-789 (1993).
- Church, L.K. and R. Uzsoy, Analysis of periodic and event-driven rescheduling policies in dynamic shops. *Int. J. Computer Integrated Manufacturing* **5**, 153-163 (1992).
- Cott, B.J. and S. Macchietto, An integrated approach to computer-aided operation of batch chemical plants. *Comput. Chem. Engng.* **13**, 1263-1271 (1989a).
- Cott, B.J. and S. Macchietto, Minimizing the effects of batch process variability using online schedule modification. *Comput. Chem. Engng.* **13**, 105-113 (1989b).
- Davis, W.J. and A.T. Jones, A real-time production scheduler for a stochastic manufacturing environment. *Int. J. Computer Integrated Manufacturing* **1**, 101-112 (1988).
- Drucker, P.F., The emerging theory of manufacturing. *Harvard Business Review* **68**, 94-102 (1990).
- Egli, U.M. and D.W.T. Rippin, Short-term scheduling for multiproduct batch chemical plants. *Comput. Chem. Engng.* **10**, 303-325 (1986).

- Fuchino, T., M. Muraki, and T. Hayakawa, Integer programming model for the globally optimal schedule of multipurpose batch plants. *J. Chem. Eng. Jpn.* **25**, 250-256 (1992).
- Harmonosky, C.M. and S.F. Robohn, Real-time scheduling in computer integrated manufacturing: a review of recent research. *Int. J. Computer Integrated Manufacturing* **4**, 331-340 (1991).
- Hasebe, S., Y. Murakami, and I. Hashimoto, A flexible simulation algorithm for scheduling of batch/semicontinuous plants. presented at *AIChE Annual Meeting*, Miami Beach (1992).
- Hatono, I., K. Tachibana, M. Umamo, and H. Tamura, Distributed real-time scheduling for flexible manufacturing. *Trans. Soc. Instr. Control Eng.* **31**, 108-115 (1995).
- Hole, P.G., *Introduction to Mathematical Statistics*. John Wiley & Sons, New York (1971).
- Hormozi, A.M., Agile manufacturing. *37th international conference proceedings, American Production and Inventory Control Society*, San Diego (1994).
- Hvala, N., S. Strmcnik, and J. Cernetic, Scheduling of batch digesters according to different control targets and servicing limitations. *Comput. Chem. Engng.* **17**, 739-750 (1993).
- Ishii, N., New flexible manufacturing for bulk production. presented at *Sixth International Congress of Pharmaceutical Engineering*, Philadelphia (1994).
- Ishii, N. and M. Muraki, A generic framework for on-line scheduling and control system in batch process management. *Submitted to Comput. Chem. Engng.* (1995a).
- Ishii, N. and M. Muraki, A process-variability-based online scheduling system in multiproduct batch process. *Comput. Chem. Engng.*, to be published (1995b).
- Ishii, N. and M. Muraki, An extended dispatching rule approach in on-line scheduling framework for batch process management. *Int. J. Prod. Res.*, to be published (1995c).
- Ishii, N. and J.J. Talavage, A transient-based real-time scheduling algorithm in FMS. *Int. J. Prod. Res.* **29**, 2501-2520 (1991).
- Ishii, N. and J.J. Talavage, A Mixed Dispatching Rule Approach in FMS Scheduling. *Int. J. Flexible Manufacturing Systems* **6**, 69-87 (1994).
- Johnson, S.M., Optimal two- and three-stage production scheduling with setup times included. *Naval Research Logistics Quarterly* **1**, 61-68 (1954).
- Kim M.H. and Y-D Kim, Simulation-based real-time scheduling in a flexible manufacturing system. *Journal of Manufacturing Systems* **13**, 85-93 (1994).

- King, J.R. and A.S. Spachis, Scheduling: bibliography and review. *International Journal of Physical Distribution and Materials Management* **10**, 105-132 (1980).
- Kiran, A. S., and M.L. Smith, Simulation studies in job shop scheduling - I & II. *Computers and Industrial Engineering* **8**, 87-105 (1984).
- Kleijnen, J.P.C., *Statistical Tools for Simulation Practitioners*. Marcel Dekker, New York (1987).
- Kondili, E., C.C. Pantelides, and R.W. Sragent, A general algorithm for short-term scheduling of batch operations-I. MILP formulation. *Comput. Chem. Engng.* **17**, 211-227 (1993).
- Ku, H.M., D. Rajagopalan, and I. Karimi, Scheduling in batch processes. *Chem. Engng. Prog.* **83**, 35-45 (1987).
- Kudva, G., A. Elkamel, J.F. Pekny, and G.V. Reklaitis, Heuristic algorithm for scheduling batch and semi-continuous plants with production deadlines, intermediate storage limitations and equipment changeover costs. *Comput. Chem. Engng.* **18**, 859-875 (1994).
- Kuriyan, K. and G.V. Reklaitis, Scheduling network flowshops so as to minimize makespan. *Comput. Chem. Engng.* **13**, 187-200 (1989).
- Lozinski, C. and C.R. Glassey, Bottleneck Starvation Indicators for Shop Floor Control. *IEEE Transactions on Semiconductor Manufacturing* **1**, 147-153 (1988).
- Manz, E.M., J. Haddock, and J. Mittenthal, Optimization of an Automated Manufacturing System Simulation Model Using Simulated Annealing. *Proceedings of the 1989 Winter Simulation Conference*, 390-395, Washington, D.C. (1989).
- Mason, R.L., R.F. Gunst, and J. L. Hess, *Statistical design and analysis of experiments*. John Wiley & Sons, New York (1989).
- Morton, T.E. and D.W. Pentico, *Heuristic Scheduling Systems: with Applications to Production Systems and Project Management*. John Wiley & Sons, New York (1993).
- Musier, R.F.H. and L.B. Evans, Batch process management. *Chem. Engng. Prog.* **86**, 66-77 (1990).
- Okino, N., A prototyping of bionic manufacturing systems. *Proc. of object-oriented manufacturing systems*, 297-302 (1992).
- Onogi, K., Y. Nishimura, Y. Nakata, and T. Inomata, An on-line operating control system for a class of combined batch/semi-continuous process. *J. Chem. Eng. Jpn.* **19**, 542-548 (1986).

- Parsaei, H.R. and W.G. Sullivan, *Concurrent engineering: contemporary issues and modern design tools*. Chapman & Hall, London (1993).
- Pedersen, K., *Expert Systems Programming*. John Wiley & Sons, New York (1989).
- Pritsker, A. A. B., *Introduction to Simulation and SLAM II*. Halsted Press, New York (1986).
- Procyk, L.M., Batch process automation. *Chem. Engng.* **98**, 110-117 (1991).
- Prosser, P. and I. Buchanan, Intelligent scheduling: past, present and future. *Intelligent Systems Engineering* **3**, 67-78 (1994).
- Reklaitis, G.V., Perspectives on scheduling and planning of process operations. *Fourth International Symposium on Process Systems Engineering*, Montebello Canada (1991).
- Rich, S.H. and G.J. Prokopakis, Scheduling and sequencing of batch operations in a multipurpose plant. *Ind. Eng. Chem., Process Des. & Dev.* **25**, 979-988 (1986).
- Rippin, D.W.T., Batch process systems engineering: a retrospective and prospective review. *Comput. Chem. Engng.* **17** Suppl., S1-S13 (1993).
- Rogers, P. and M.T. Flanagan, On-line simulation for real-time scheduling of manufacturing systems. *Ind. Eng.* **23**, 37-40 (1991).
- Rumelhart, D.E., J.L. McClelland, and the PDP Research Group, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press, Massachusetts (1986).
- Savage, C.M., *Fifth generation management: integrated enterprises through human networking*. Asahi Shimbun, Tokyo (1990).
- Snedecor, G.W. and W.G. Cochran, *Statistical methods*. Iowa State University Press, Ames (1989).
- Stadtherr, M.A., H.N. Cofer, and K.V. Camarda, Chemical engineering computing 2001: Hardware: Developing the metacomputer. *Chem. Engng. Prog.* **89**, 27-37 (1993).
- Talavage, J.J. and R.G. Hannam, *Flexible Manufacturing Systems in Practice*. Marcel Dekker, New York (1988).
- Terpstra, V.J., R.M. de Bruijkere, and H.B. Verbruggen, A prototype of a reactive scheduler for mixed batch/continuous plants. *ESCAPE 4: Fourth European Symposium on Computer Aided Process Engineering*, Rugby U.K. (1994).

- Thesen, A. and L. Lei, An expert system for scheduling robots in a flexible electroplating system with dynamically changing workloads. *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, 555-566 (1986).
- Vollmann, T.E., W.L. Berry, and D.C. Whybark, *Manufacturing Planning and Control Systems*. Irwin, Illinois (1988).
- Wu, S.-Y.D. and R.A. Wysk, Multi-pass expert control system - a control/scheduling structure for flexible manufacturing cells. *Journal of Manufacturing Systems* **7**, 107-120 (1988).
- Wu, S.-Y.D. and R.A. Wysk, An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. *Int. J. Prod. Res.* **27**, 1603-1623 (1989).
- Yamamoto, M. and S.Y. Nof, Scheduling/rescheduling in the manufacturing operating system environment. *Int. J. Prod. Res.* **23**, 705-722 (1985).