1-1-2005

Development of a dynamic scheduling system for flexible manufacturing systems

Wence Sui

Ryerson University

Follow this and additional works at: <http://digitalcommons.ryerson.ca/dissertations>

Part of the [Mechanical Engineering Commons](#)

Recommended Citation

Sui, Wence, "Development of a dynamic scheduling system for flexible manufacturing systems" (2005). *Theses and dissertations*. Paper 391.

DEVELOPMENT OF A DYNAMIC SCHEDULING SYSTEM FOR FLEXIBLE MANUFACTURING SYSTEMS

by

Wence Sui

BEng in Mechanical Engineering, Xi'an Institute of Technology, China, 1986

A thesis

presented to Ryerson University

In partial fulfillment of the
requirements for the degree of
Master of Applied Science

In the Program of
Mechanical Engineering

Toronto, Ontario, Canada, 2005

© Wence Sui 2005

PROPERTY OF
RYERSON UNIVERSITY LIBRARY

UMI Number: EC53766

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI[®]

UMI Microform EC53766
Copyright 2009 by ProQuest LLC
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

Author's Declaration Page

I hereby declare that I am the sole author of this thesis.

I authorize Ryerson University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Signature:

I further authorize Ryerson University to reproduce this thesis by photocopying or by other means, in total or in part, at the request other institutions or individuals for the purpose of scholarly research.

Signature:

Borrower's Page

Ryerson University requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

Abstract

Development of a Dynamic Scheduling System for Flexible Manufacturing Systems

Wence Sui

Master of Applied Science

Mechanical Engineering

Ryerson University 2005

A dynamic scheduling system for flexible manufacturing systems (FMSs) is developed. The system consists of an artificial intelligence scheduler (AIS), a simulation model, a database, and a user interface. The AIS is used to generate candidate schedules according to specified dispatching rules and to control the search depth based on the status of an FMS. The simulation model is utilized to evaluate candidate schedules. The user interface is used to manually generate candidate schedules and to facilitate communication between different modules so as to realize automatic scheduling. Schedule related data are stored in the database. A dispatching-rule-based approximation search method is employed, which can quickly converge on good areas of the solution space, and the search method is suitable for dynamic scheduling in FMSs. A hypothetical FMS is designed and a sample scheduling problem is used to demonstrate the mechanism of the scheduling system.

Acknowledgement

I would like to express my sincere thanks to my supervisor, Dr. Liping Fang, for his invaluable support, guidance, patience, encouragement and understanding during my graduate program. Were it not for his support and guidance, I could not have completed my study and this thesis. I always feel honored and fortunate to have him as my supervisor.

I would also like to thank Dr. Greg Kwall, Dr. Jeff Xi, and Dr. Saeed Zolfaghari for serving as my thesis committee members and providing their precious time, advice and guidance. In particular, Dr. Saeed Zolfaghari's help in building the simulation model of the hypothetical flexible manufacturing system is greatly appreciated.

Dedication

To my wife and my son

To my parents in the heaven

To my brothers and their families

Table of Contents

Abstract.....	iv
Acknowledgement.....	v
Dedication.....	vi
Table of Contents.....	vii
List of Tables.....	x
List of Figures.....	xi
Nomenclature.....	xiii
Chapter 1: Introduction.....	1
1.1 Manufacturing.....	1
1.2 Flexibility in Manufacturing.....	2
1.3 Flexible Manufacturing Systems.....	6
1.4 Scheduling in Manufacturing.....	8
1.5 Dynamic Scheduling in FMSs.....	10
1.6 Objectives of the Research.....	10
1.7 Organization of the Thesis.....	11
Chapter 2 Scheduling in Manufacturing Systems.....	13
2.1 Introduction.....	13
2.2 Complexity of Scheduling Problems in FMS.....	13
2.3 Methodologies for Solving FMS Scheduling Problems.....	16
2.3.1 Optimization-Based Techniques.....	17
2.3.2 Approximation Methods.....	20
2.3.2.1 Dispatching Rules.....	21
2.3.2.2 Artificial Intelligence Approaches.....	23
2.4 Simulation and Scheduling.....	27

2.5 Scheduling System Incorporating Artificial Intelligence and Simulation Techniques.....	29
Chapter 3 A Proposed Approach for Dynamic Scheduling in Flexible Manufacturing Systems.....	31
3.1 Introduction.....	31
3.2 Selection of a Scheduling System for Dynamic Flexible Manufacturing Systems.....	31
3.3 A Proposed Approach for Dynamic Scheduling.....	37
3.4 Proposed Search Technique.....	39
3.5 Software Structure for Realizing the Proposed Approach.....	48
3.5.1 Simulation Module.....	48
3.5.2 Artificial Intelligent Scheduling Module.....	49
3.5.3. SCADA system.....	49
3.5.4. Central Database.....	50
3.5.5. VB Interface.....	50
Chapter 4 Simulation Model of a Hypothetic FMS.....	51
4.1 A Hypothetic FMS.....	51
4.2 Simulation Model Development Environment — AweSim!.....	55
4.3 AweSim Simulation Model of the Hypothetic FMS.....	57
4.3.1 Parts Entry.....	59
4.3.2 Part Routing Strategy.....	59
4.3.3 Transforming Orders into Lists of Components and Products.....	59
4.3.4 Recording and Storing Completion Times.....	60
4.3.5 Control File and Attribute Index.....	60

Chapter 5 Implementation of the Dynamic Scheduling System.....	62
5.1 Introduction.....	62
5.2 Inter-Modular Communication — Dynamic Data Exchange (DDE).....	62
5.3 VB Interface and Manual Operation.....	65
5.3.1 Extracting Orders from the Database.....	66
5.3.2 Generating Schedules Manually.....	67
5.3.3 Evaluating Candidate Schedules.....	67
5.4 Realization of Automation of the Scheduling System.....	68
5.4.1 Expert System Development Environment — CLIPS.....	69
5.4.1.1 CLIPS Features.....	69
5.4.1.2 CLIPS Windows Interface.....	70
5.4.2 Objects in the Artificial Intelligent Scheduler (AIS).....	71
5.4.2.1 The “Order” Object.....	72
5.4.2.2 The “Candidate_Schedule” Object.....	72
5.4.2.3 The VB-Interface Object.....	73
5.4.3 Modelling Artificial Intelligent Scheduling System in CLIPS.....	73
5.4.3.1 Programming the AIS and a Case Illustration.....	75
 Chapter 6 Conclusion and Future Work.....	 82
6.1 Summary of the Contributions.....	82
6.2 Future Work.....	83
 References.....	 84

List of Tables

Table 3.1 Base Set of Orders for Permutation and Testing.....	40
Table 3.2 Similar Schedules in Six Groups.....	42
Table 3.3 Base Schedule according to Due Date.....	44
Table 3.4 Representative Schedules within Each Group.....	45
Table 3.5 Representative Schedules for Sub-groups within Group 2.....	47
Table 4.1 Processing Time for Each Component.....	55
Table 4.2 Attribute Index.....	61
Table 5.1 A Hypothetic Scheduling Problem with Four Orders.....	76
Table 5.2 Candidate Schedules at Level 0.....	78
Table 5.3 Candidate Schedules with Results at Level 1.....	79
Table 5.4 Candidate Schedules with Results at Level 2.....	80
Table 5.5 “Best” Schedule of the Example.....	81

List of Figures

Figure 1.1 CIM Technology (Parrish, 1993).....	5
Figure 1.2 A Typical Flexible Manufacturing System.....	6
Figure 1.3 Hierarchical Control Structure of an FMS.....	7
Figure 2.1 Branch and Bound Tree Structure (Baker, 1974).....	18
Figure 2.2 Computational Cost versus Solution Quality.....	21
Figure 2.3 Structure of a Rule-Based Expert System (Negnevitsky, 2002).....	25
Figure 2.4 Architecture of the Real-time Scheduling Mechanism (Jeong and Kim, 1998).....	29
Figure 3.1 Scheduling Techniques: Quality versus Time.....	33
Figure 3.2 A Multi-Pass Expert Control System (MPECS) for Flexible Manufacturing Systems (Wu and Wysk, 1989).....	36
Figure 3.3 A Proposed Approach for Dynamic Scheduling.....	37
Figure 3.4 Entire Set of Solutions of Sample Schedules.....	43
Figure 3.5 Results of Similar Schedules in Each Group.....	44
Figure 3.6 Results of Representative Schedules Superimposed on Results of Entire Set.....	46
Figure 3.7 Software Structure.....	48
Figure 3.8 Schedule Evaluation Loop.....	49
Figure 4.1 Layout of the Hypothetic FMS.....	53
Figure 4.2 The AweSim Architecture (Pritsker and O'Reilly, 1999).....	56
Figure 4.3 AweSim Executive Window.....	57
Figure 4.4 Simulation Network of the Hypothetic FMS.....	58
Figure 4.5 Sample Group of Orders.....	60
Figure 5.1 Inter-Module Communication.....	64
Figure 5.2 Scheduling System Interface.....	66

Figure 5.3 Expert System (AIS) Development Environment — CLIPS.....	71
Figure 5.4 Complete AIS Tree Structure for a Scheduling Problem with Four Orders.....	74
Figure 5.5 Illustration of Search Strategy of the Four Order Scheduling Problem.....	75
Figure 5.6 Level 0 of the Decision Tree.....	78
Figure 5.7 Level 1 of the Decision Tree.....	80
Figure 5.8 Level 2 of the Decision Tree.....	81
Figure 5.9 Complete Decision Tree of the Example.....	81

NOMENCLATURE

Symbols

b	Lead time estimation parameter
C	Cost function value of a schedule
d	Due time of a job
J	Finite set of jobs
j	Product factor
k	Adjustment multiplier
M	Finite set of machines
m	Number of machines
n	Number of jobs
P	Total number of orders in a schedule
p	Processing time of an operation or a component
q	Number of operators
S	Total number of components in an order
t	Completion time of an order
w	Remaining work of an operation. It is the sum of processing times of successor operations including itself.
Σ	Summation

Abbreviations and Acronyms

AC	Area Controller
AGV	Automated Guided Vehicle
AI	Artificial Intelligence

AIS	Artificial Intelligence Scheduler
AMT	Advanced Manufacturing Technology
CAD	Computer Aided Design
CADD	Computer Aided Design and Drafting
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CAP	Computer Aided Planning
CAPP	Computer Aided Processing Planning
CAQ	Computer Aided Quality Control
CC	Cell Control
CIM	Computer Integrated Manufacturing
CLIPS	C Language Integrated Production System
CNC	Computer Numerical Control
COOL	CLIPS Object-Oriented Language
COVERT	Cost OVER Time
DAS	Data Acquisition System
DDE	Dynamic Data Exchange
DNC	Direct Numerical Control
FAS	Flexible Assembly System
FCFS	First Come First Served
FMS	Flexible Manufacturing System
FMC	Flexible Manufacturing Cell
IC	Integrated Circuit
LAN	Local Area Network
MDA	Machine Data
MPECS	Multi-Pass Expert Control System
MRP	Materials Resource Planning

NC	Numerical Control
OR	Operations Research
PDA	Production Data
PLC	Programmable Logic Controller
RDO	Remote Data Objects
SCADA	Supervisory Control and Data Acquisition
SOPT	Shortest Operation Processing Time
SJPT	Shortest Job Processing Time
VB	Working In Process

Chapter 1 Introduction

1.1 Manufacturing

Nowadays, because of fierce competition in the global customer-driven market, companies find, not only to gain a competitive advantage over their competitors, but often merely to survive, they have to operate in a more flexible and dynamic mode, offering more customer options in products with greatly reduced life cycles and far great product complexity. They have to be able to supply “Just in Time” and introduce a range of new products very rapidly. As a result, companies are forced to adopt computer controlled, fully integrated and automated facilities, among which flexible manufacturing systems have become increasingly popular since the late 1970s.

A flexible manufacturing system (FMS) is defined as “an integrated manufacturing system that consists of numerically controlled machines equipped with tool magazines and connected by a material handling system, where all system components are under computer control” (Askin and Standridge, 1993). FMSs can provide high automation and massive alternatives in parts processing, parts routing, tool delivering and capacity planning. These automation and alternatives forge the foundation for the flexibility of FMSs. Although the capability to manufacture a variety of different part types on a versatile automated system is attractive, the capital investment for such a system usually is very considerable. Hence, the design, scheduling, and control of an FMS are very critical in order for production needs to be satisfied in an optimal way. However, on the other hand, these automation and alternatives also make it extremely difficult to study FMSs using analytical methods. This thesis discusses one problem that has been compounded by the increasing emphasis on automation—that of scheduling.

Scheduling plays a vital role in the production control of an FMS, which involves several real-time decisions, such as part type and machine selection, resource allocation, machine allocation, and tool loading (Reddy et al., 2004). The primary objective of an effective scheduling system is to produce the right parts, at the right time, at a competitive cost, by minimizing overhead and operating costs, subject to satisfying demand for the enterprise's products. Traditionally this task has been seen as notoriously difficult to solve even in relatively static manufacturing environments. The solution space explodes factorially in size with increasing problem dimension. Even the addition of a small number of variables to the problem makes it unsolvable in reasonable time. Consequently, the trend towards high levels of system automation and the significant increase in system complexity has posed serious questions to the traditional scheduling methodologies (Cowling et al., 2003). Furthermore, a dynamically changing manufacturing environment makes schedules predetermined in a static environment impractical. Schedulers for highly automated FMSs are required to dynamically respond to the occurrence of unexpected events (e.g., machine breakdown or tool breakage). In most cases, the response time is quite imminent. Therefore, dynamic scheduling is a challenging new research area (Dorn, 1995; Tang et al., 2001). In recent years, a lot of research projects have been carried out on this topic. Several methods have been proposed (Chan and Chan, 2002; Shen, 2004). This research tries to integrate two advanced computing techniques, Discrete Event Simulation and Expert System, to handle more computationally complex scheduling problems in FMSs. To develop a dynamic scheduling system for FMSs, it is necessary to introduce a number of concepts on FMSs and scheduling.

1.2 Flexibility in Manufacturing

Generally speaking, flexibility means a high degree of dexterity or agility in handling unexpected changes and anomalies. Flexibility in manufacturing can appear in various

forms. Flexible manufacturing involves using general-purpose machines in the production of a variety of products in a varied production combination. Flexibility can also be incorporated at all levels of the planning and decision-making hierarchy of an enterprise, from the design of the equipment at the corporate level to its daily operation on the shop floor. In the design of the facility and equipment, planners need to make use of resources and structure them in a way that can quickly and reliably adapt to project variations and changes. For example, the facility can be designed to use general-purpose re-programmable machines. Flexibility can also be displayed at the floor level by developing a dynamic control system that allows the manufacturing system to respond to changes in the system on a real-time basis. The control system must be able to cope with the dynamic nature of the shop floor and deal with exceptions caused by external and uncontrollable variations resulted from customers and suppliers, etc. The control system must also be able to handle unpredictable fluctuations in resource levels such as machine breakdowns, or unexpected tooling and labor shortages.

In general, the globalization and increased competition are the main reason why flexibility has become essential for manufacturing companies to survive. The globalization of markets has given customers more choices. With more producers in competition for a larger share of the market, customers have found themselves in a very favorable position. "Customers can now demand high product variety, high quality, frequent product design changes, variable batch sizes, shorter lead-times, and more competitive prices" (Parrish, 1993). However, traditional manufacturing, featured in mass production and dedicated facilities, does not offer anything in flexibility. Once a product design is changed or an entirely new product is demanded in the markets, then some or all of the dedicated facility must be abandoned. The significance of flexibility is becoming more and more important while product life cycles become shorter and shorter. Good adaptability and agility for the markets brought about by FMS technology have not

only become a source of competitive edge but in some circumstances can be vital for a company to survive in a situation where stagnation directly leads to death.

Recent technological advances have proved instrumental in facilitating the introduction of flexibility, not only on the factory floor but also at all levels throughout entire corporations. It is quite agreeable that the single most important development to have a significant impact on the manufacturing industry in the last century has been the development of the digital computer. The subsequent progress in integrated circuits (ICs), micro-controllers and programmable logic controllers (PLCs) resulted in the localized automation of manufacturing activities. For example, the introduction of numerical control (NC) and later computer numerical control (CNC) machines has led to the complete but maybe single and stand-alone automation within manufacturing enterprises.

Naturally, the next task was to focus on the problem of linking and co-coordinating these individual automation systems to form a systematic automation. This complete integration of individual functions is sometimes called computer-integrated manufacturing (CIM). The ultimate goal of CIM is to realize the most efficient use of resources in a facility by automating and co-coordinating information flows between existing entities of automation. Figure 1.1 from Parrish (1993) outlines the different elements involved in a CIM system. CIM incorporates automated functions from both organizational and operational aspects of the manufacturing company. Organizational activities include functions such as finance and marketing. These supporting functions are indirectly related to the manufacturing process. The operational functions are directly involved in the manufacturing process in terms of both the planning (computer aided engineering (CAE)) and real manufacturing. It can be seen that the flexible manufacturing system forms an integral part of CIM technology.

Figure 1.1 CIM Technology (Parrish, 1993)

1.3 Flexible Manufacturing Systems

A flexible manufacturing system is usually considered as a computer controlled and integrated configuration of numerically controlled (NC) machining centers with automated material handling systems, comprising conveyors, robots, and/or automated guided vehicles (AGV) (Thomson, 1995). In other words, an FMS is a group of general-purpose and re-programmable machines, each of which is capable of performing multiple tasks. A typical FMS is illustrated in Figure 1.2.

Figure 1.2 A Typical Flexible Manufacturing System
(Courtesy of General Dynamics, Fort Worth Division)
(Downloaded from <http://www.generaldynamics.com>)

In typical FMSs, machines are linked by an automated transport system such as an automated guided vehicle system (AGVS), gantry style cranes or conveyor systems. The transport system handles the transfer of resources in an FMS. For example, it transfers raw material from storage to the CNC machines, partially manufactured product between machines, complete components to assembly robots and tooling to the machines.

Figure 1.3 shows a hierarchical control structure of an FMS. The host computer oversees activities in the entire area of the cell. The host can distribute control commands and transfer CNC programs to local controllers at machine level across a local area network (LAN). Distributed control in this method is a practical way of dealing with the high level of control complexity associated with FMSs.

Figure 1.3 Hierarchical Control Structure of an FMS

In a common sense, unique features of an FMS can be summarized as follows: (1) each machine in an FMS is quite versatile and capable of performing many different operations; (2) the system can machine several part types simultaneously; and (3) individual parts may follow different routes through the system.

Combining the merits of job shop production and flow shop production, an FMS is most suitable for mid-volume and mid-variety customized production. Due to the high-level distributed processing and material flow, an FMS inherits complicated design, planning, and operational problems. Solving flexible manufacturing related problems is relatively complex compared to that of traditional manufacturing systems where lead times are longer, inventory levels are higher, and the utilization rate is much lower. In general, the features of an FMS, while making the system flexible, increase the difficulty of decisions associated with all phases of the life cycle of an FMS—design, planning, scheduling and control.

1.4 Scheduling in Manufacturing

To remain competitive in the market, manufacturers must strive to continually improve their processes. Production scheduling that assigns manufacturing resources and time to the manufacturing jobs is one of the most important processes in manufacturing systems. A good production schedule can provide such benefits as increased shop throughput, enhanced customer satisfaction, lower inventory levels, and increased utilization of resources. Therefore, there is a great need for good scheduling strategies.

Simply speaking, scheduling problems essentially involve completing a set of jobs with a limited number of manufacturing resources under a number of constraints to optimize a particular objective function. More specifically, according to Brucker (2001), a

scheduling problem consists of three components: a machine environment, specific job characteristics, and one or more optimality criteria. The machine environment represents the type of the manufacturing system that executes the developed schedule. The manufacturing system may be a job shop system, flexible manufacturing system, cellular manufacturing system, transfer line, etc. Job characteristics represent such factors as the number of operations, the precedence relations among operations, and the possibility of preemption (whether the job can be split). Optimality criteria are the objectives to pursue when scheduling the jobs. Common objectives include minimizing makespan, mean flow time, mean lateness, the number of tardy jobs, and mean tardiness. All the three components mentioned above specify the variety and complexity of each scheduling problem.

A scheduling problem may be comprised of two sub-problems: job routing and job sequencing problems. A job routing problem involves assigning the operations of jobs to the specific machines. Such problems result from the allowance of routing flexibility. Routing flexibility depends on the capability of the machines. A versatile machine is capable of performing different operations. The versatility of the various machines in a shop essentially supports the possibility for the existence of alternative process plans for a job. Routing flexibility is a key issue that has increasingly attracted attention in modern manufacturing systems. FMSs provide routing flexibility due to the capability of NC machines. Once the route of a job is specified, decision makers must determine the production sequence of the jobs awaiting their next process in the machine queue.

These scheduling problems are known to be hard and usually belong to the NP-complete class of problems (Morton and Pentico, 1993; Pinedo, 1995), which means that there is no polynomial-time solution known to solve this problem. Research in production scheduling has been conducted for many decades and a large number of algorithms and

heuristics have been developed for various scheduling problems (Chan and Chan, 2002).

1.5 Dynamic Scheduling in FMSs

One of the most notoriously difficult systems for the scheduling community is FMS. The strategy of FMS is based on producing a wide variety of products in very low volumes. Producing such variable products requires different sequences. In a real-world environment of FMS, ever-changing task requirements, occurrence of a variety of unexpected events, complicated and dynamic resource constraints, uncertain processing time, and multiple or even conflicting scheduling objectives, complicate the scheduling problem. One way to deal with this problem is to make several assumptions to simplify the problem such as that resources are always available, all the jobs are known in advance, all the operation processing times are known and constant, transportation times are ignored, etc. However, applying too many assumptions may result in the treatment of scheduling problems that would be considered unrealistic. The other way is to make real-time scheduling decisions, concurrently and dynamically, taking the real-time manufacturing environment into account. Naturally, this furthermore increases the complexity of the FMS scheduling problem. But because dynamic scheduling is more realistic and more applicable, it has attracted much attention over many decades.

1.6 Objectives of the Research

The main objective of this research is to develop a practical solution to the dynamic scheduling problem in FMSs by using simulation and expert systems techniques. In striving towards this primary objective, the following objectives and tasks are to be accomplished:

- To conduct an in-depth investigation into the complexity of the dynamic scheduling problem in flexible manufacturing systems.
- To investigate various methodologies that have been employed in developing practical solutions for dynamic scheduling in FMSs, with focus on simulation and artificial intelligence approaches.
- To develop a search method, which can be implemented in dynamic (real-time) scheduling.
- To build a hypothetical FMS layout and to develop a simulation model of this hypothetical FMS.
- To develop an expert system as an intelligent decision maker and to combine it with the simulation model to form an automated and intelligent scheduling system.
- To solve the problems concerning the functioning of the proposed scheduling system, such as selecting appropriate development platforms, realizing communication between different modules, etc.

1.7 Organization of the Thesis

This thesis is organized as follows: Chapter 1 contains an overview of flexible manufacturing systems, scheduling problems in manufacturing, and dynamic scheduling in flexible manufacturing systems. Chapter 2 presents an in-depth review of some scheduling methods, which have influences directly or indirectly on the dynamic scheduling system proposed in this thesis. Integration of simulation and scheduling is

addressed. In Chapter 3, a practical approach for dynamic scheduling in flexible manufacturing systems is proposed. A simple and practical search technique is developed and discussed. The software structure is described. In Chapter 4, a hypothetical flexible manufacturing system is designed and a simulation model for this FMS is developed in AweSim. Chapter 5 discusses the communication between relevant modules, the VB interface used to manually conduct scheduling and to manage data, the artificial intelligence scheduler developed in CLIPS, and the automatic scheduling process. A sample case is used for demonstration. The main contributions of this thesis are summarized in Chapter 6, along with some possible directions of future work.

Chapter 2 Scheduling in Flexible Manufacturing Systems

2.1 Introduction

Scheduling of flexible manufacturing systems (FMSs) has been one of the most attractive areas for both researchers and practitioners because of its potential to dramatically reduce costs and increase efficiency. A considerable volume of literature has accumulated in this area since the late 1970s when the first batch of papers was published (Chan and Chan, 2002). A number of approaches have been adopted to schedule FMSs, including heuristics, constraint propagation techniques, simulation techniques, constraint satisfaction problem formulations, Tabu search, simulated annealing, genetic algorithms, artificial intelligence (AI) approaches (expert system, neural networks, fuzzy logic), and agent-based approaches (Distributed AI). Numerous articles can be found on each of these approaches. Although this chapter cannot cover all methods ever proposed in the literature, it seeks to provide a brief but in-depth review, explanation and analysis of some techniques and concepts concerned, particularly those having an impact on the method proposed in this thesis — approximation methods, simulation techniques, dispatching rules, and expert systems.

2.2 Complexity of Scheduling Problems in FMS

In order to focus research activities on a common set of objectives, it is necessary to reiterate the definition of the scheduling problem. According to Jain and Meeran (1998), the general scheduling problem can be defined as follows.

The deterministic general scheduling problem consists of a finite set J of n jobs that must be carried out on a finite set M of m machines. To unify the problem and make it more

acceptable, in the most common formulation of the general scheduling problem, the number of operations in each job is m . In other words, each job must be processed on each of m machines (Baker, 1974). Each job J_i , $i = 1, 2, \dots, n$, follows a series of operations that are performed in a predetermined order, subject to certain precedence constraints. Also, the problem is subject to resource capacity constraints or separate constraints, namely, that each machine can process only one operation on every job and that at any time each operation can be processed by only one machine. Moreover, each job is processed without pre-emption (i.e., uninterrupted for its assigned processing time).

The purpose is to find a best schedule that optimizes certain pre-determined objectives. In this context, a schedule is actually defined as the starting and completion times for each individual job on a group of machines that satisfy all problem constraints. According to Baker (1974), the following three types of objectives are commonly seen in scheduling:

- efficient utilization of resources,
- quick response to demands, and
- close conformance to prescribed deadlines.

But actually, in most cases, to be straightforward, the task is to find a schedule that has the minimum total duration required to complete all the jobs.

The classical scheduling problem is non-deterministic polynomial (NP) – hard. With n jobs and m machines, the difficulty is not in modeling but in computation since there is a total of $(n!)^m$ possible schedules (Rembold et al., 1993). In fact, once the problem dimension reaches three jobs and m machines then it becomes difficult to solve in polynomial time (Gonzalez and Sahni, 1978; Lenstra and Kan, 1979).

The problem becomes even more complex in following circumstances (Shen, 2004):

- When other manufacturing resources such as operators and tools are also considered during scheduling. For example, if q operators are considered, the total number of possible solutions could be $\left((n!)^m\right)^q$.
- When both process planning and manufacturing scheduling are to be carried out simultaneously. Traditional approaches simplify the optimization problem by separating planning and scheduling. Theoretically, an integrated optimization problem subject to both domains can obtain a global optimal solution. But it increases the solution space dramatically.
- When unforeseen dynamic situations are considered. In real manufacturing situations, things rarely go as expected. Additional urgent tasks may be added and a certain job may be neglected. The available resources are also subject to changes, such as machine breakdown, operator absence, and unavailability of tools and material. So an optimal schedule obtained from considerable efforts may rapidly become unacceptable because of unanticipated dynamic situations. Therefore, a new schedule has to be generated to restore best performance. This kind of rescheduling problem is just so-called dynamic scheduling or real-time scheduling.

It is generally accepted that the job shop scheduling problem represents the general scheduling problem in the manufacturing systems (Pinedo, 1995). The general scheduling problem differs from the flow shop problem in that product flows through the manufacturing systems are not unidirectional. The job shop scheduling problem represents the scheduling of a group of products on a group of workstations employing resources that are completely unconstrained. In other words, any product or group of jobs can be processed in any sequence on any machine. These characteristics very closely match those associated with scheduling of a flexible manufacturing system. Therefore, solutions proposed to the general scheduling problem can be directly applied to FMSs.

However, it is necessary to be aware that scheduling problems in an FMS are more complicated than ones in job shops because of several reasons: firstly, each machine is versatile and capable of holding different tools to perform different operations. Therefore, different part types can be manufactured at any given time. Secondly, material handling systems such as AGVs, jigs, fixtures, and pallets must also be scheduled. In other words, the number of decision points, where scheduling or operation rules can be varied, is greater in FMSs than one in job shops. In addition, because of the large set-up times required for an alternative operation in job shops, if an exception occurs, such as a rapid change of demand, or random entering of the new products with high priority, or machine breakdown, production would be interrupted. But, in FMSs, production is continuous even during unexpected events such as machine breakdowns because programmability of the instructions for operations dramatically reduces set-up times. An operation can be performed on a number of alternative machines with possibly different process times.

2.3 Methodologies for Solving FMS Scheduling Problems

Scheduling of FMSs has been extensively investigated over the last three decades and it continues to attract the interest of both the academic and industrial sectors. The literature on FMSs is abundant with papers on scheduling. It is not possible to review all methodologies proposed so far in the thesis. So only those having influences directly or indirectly on the scheduling system developed in this thesis are discussed.

There are many different ways to categorize all the approaches concerned. According to Kusiak and Ahn (1992), generally, there are three kinds of approaches:

- Operations research (OR).
- Artificial intelligence (AI)-based.
- Combination of OR and AI-based methods.

Spano et al. (1993) divide the scheduling research into two major approaches:

- Traditional.
- Artificial intelligence (AI)-based methods.

The traditional approach can be further divided into two categories:

- Theoretical research dealing with optimization procedures.
- Computational research dealing with dispatching rules.

For the literature review in this thesis, the approaches are divided into the following two categories:

- Optimization-based techniques.
- Approximation methods.

2.3.1 Optimization-Based Techniques

Optimization techniques focus on the development of mathematical models and optimal or sub-optimal algorithms, using integer, mixed integer and linear programming formulations. The branch and bound approach is often used in solution procedures. Mathematical programming models, which are based on simplified assumptions for the systems under study, are specific to individual manufacturing enterprises and processes. These models also require a high degree of accuracy in the data used. These theoretical results have not been widely used in industry because of the associated high computational complexity (Chan and Chan, 2002). Since the search method proposed in this thesis has similarity in structure to the branch and bound method, so the branch and bound method is introduced from the perspective of scheduling.

The branch and bound algorithm is one of useful enumerative techniques for solving combinatorial problems. Two basic operations, branching and bounding are used in order to avoid complete enumeration of the solution space, and hence reducing the

computational complexity of the algorithm. The branching divides the overall problem into a series of sub-problems in a tree-like structure. The bounding procedure is used to find the lower bound of each “active” sub-problem in order to separate out particular sub-problems that might contain the optimum solution. In this way a decision tree is created dynamically level-by-level, as shown in Figure 2.1. At each of these levels branching and bounding are used until the possible set of nodes on that level can be determined. As the complete problem is divided at each level, estimation of lower bounds for each sub-problem allows a large number of branches of the solution space to be removed from investigation as the search is guided towards the base of the tree, hence finding the optimal solution.

Figure 2.1 Branch and Bound Tree Structure (Baker, 1974)

In terms of a scheduling problem, the number of levels in the decision tree represents the number of operations in the problem. A schedule is any arrangement of these operations

that satisfies all problem constraints. At level 0, the entire set of unscheduled jobs is represented (i.e., the complete problem). At level 1, the entire problem is subdivided into $n-1$ sub-problems (where n is the original number of operations to be scheduled). Each sub-problem is identical to the problem at the preceding level with the exception of a single operation that has been fixed in the last available position in the emerging partial schedule. Repeating this process down through the decision tree and across its full width, one would construct the complete enumeration of the problem i.e., $(n!)^m$ nodes at level n . Obviously, evaluation of all possible permutations would reveal the optimal schedule but it would take too long when faced with any problem of practical dimensions. Branch and bound strategies seek to remove large groups of schedules that have been proven to lead to sub-optimal results. As the search advances through each level of the tree, the most likely candidates for optimality begin to emerge. The optimum schedule is pieced together level-by-level until all jobs have been given appropriate positions in that schedule.

Branch and bound and other enumerative algorithms have been studied over the years. Initially Balas (1969) applied some developments in attempting to solve the general scheduling problem. Some authors, such as McMahon and Florian (1975), suggested the decomposition of a problem into a series of single machine problems. Unfortunately, however, even with tight bounding techniques, these methods require excessive computational time when faced with complex practical problems. Even though success has been somewhat limited to problems of small to medium dimensions, many of these strategies have been embedded in modern approximation techniques that at present appear to be the most practical approach to solving combinatorial problems such as the general scheduling problem.

2.3.2 Approximation Methods

In many situations, an optimum solution to a problem is not required and a sub-optimal solution is satisfactory, as long as it is close to the optimum. Unlike optimization techniques, approximation methods do not guarantee that the solution generated optimizes whatever objective is concerned. However, by using aggressive elimination strategies, approximation methods can very quickly arrive at a solution that is adequately closely to the optimum.

Figure 2.2 illustrates the explosive nature of computational costs versus quality of solution. As a search technique approaches the optimum, the costs increase tremendously while gains are diminished in correspondence with the law of diminishing returns. Therefore, the cost for getting a sub-optimal solution is much less than the cost of computing the optimum. This is the fundamental basis for accepting approximation algorithms as promising approaches. So now researchers from many different backgrounds emphasize developing the most efficient approximation algorithm, that is, the algorithm that gets the closest to the optimum in the shortest possible time. A wide variety of approaches have been proposed from backgrounds like neurophysiology and genetics to solve the general scheduling problem. For example, simulated annealing is analogous to the annealing process in the heat treatment of metals, and genetic algorithms are analogous to mechanisms involved in natural evolution. In the following, only those that are closely related to the search technique proposed in this thesis are discussed.

Figure 2.2 Computational Cost versus Solution Quality

2.3.2.1 Dispatching Rules

Dispatching rules are the simplest form of approximation techniques. A dispatching rule is a priority rule used to determine the order in which the jobs waiting in the machine queue are to be processed as soon as a machine becomes available. The terms such as decision rules, priority rules and dispatching rules are often used interchangeably with scheduling rules. Dispatching rules are useful for finding a reasonably good schedule very quickly. In general, scheduling rules are widely used in practice ranging from direct applications as a stand-alone scheduling scheme to indirect application as part of complicated scheduling systems. They are attractive because of their simplicity, ease of implementation, and effectiveness in a highly dynamic and stochastic manufacturing environment. A variety of dispatching rules have been proposed. Panwalkar and Iskander (1977) identify the existence of more than 100 distinct rules. Wild (1995) lists some typical dispatching rules. A small sample of the rules are presented here:

- FCFS (First Come First Served): Select an operation that arrived earliest at the queue

for the machine.

- SOPT (Shortest Operation Processing Time): Select an operation with the shortest operation processing time.
- SJPT (Shortest Job Processing Time): Select an operation with the shortest job processing time (the sum of processing times of the operations for the job).
- SLACK: Select an operation with the least slack $(d_i - w_i - t)$, where d_i is the due time of the job in which operation i is included; w_i is the remaining work of operation i (the sum of processing times of the successor operations including itself); and t is the time at which operation i is to be selected for an available machine.
- COVERT (Cost OVER Time): Select an operation with the largest ratio of expected delay penalty to the processing time, i.e., select an operation with the maximum

value of $\left[\left\{ 1 - \frac{(d_i - w_i - t)^+}{k \cdot b \cdot w_i} \right\} / p_i \right]^+$, where p_i is processing time of operation i ; b is the lead time estimation parameter that takes into account the waiting time between operations; k is the adjustment multiplier which adjusts the expected waiting time to the worst case; and x^+ is the $\max(0, x)$. d_i , w_i , and t are the same as defined in the SLACK rule.

Because there is no iterative procedure involved, dispatching rules can generate a reasonably good schedule very quickly. Therefore, they are suitable for dynamic (real-time) scheduling. However, the dispatching rules have the following shortcomings:

- A dispatching rule always blindly pursues a single objective at the expense of other performance measures (Chandra and Talavage, 1991). But, in reality, a set of objectives may be important simultaneously.
- No single dispatching rule can be really dominant across all possible scenarios (Chiu and Yih, 1995; Kouliss et al., 1997; Pierrval and Mebarki, 1997; Subramaniam et al.,

2000). Therefore, the choice of rule is system dependent and often determined by the objective.

- A dispatching rule does not take in account the status of the other resources.

The rules themselves have been studied extensively, usually under the controlled conditions of a simulation model. The simulation model allows the performance of different rules to be compared and contrasted in different scenarios (Montazer and Van Wassenhove, 1990). Sabuncuoglu (1998) examines the effects of scheduling rules on the performance of flexible manufacturing systems. Several machine and AGV scheduling rules are tested against the mean flowtime criterion. It should be noticed that one valuable application of dispatching rules is they can be combined with other techniques such as artificial intelligence and neighborhood search techniques to provide valuable domain knowledge relating to a problem.

2.3.2.2 Artificial Intelligence Approaches

Artificial intelligence techniques appear to be particularly suited to solving operation problems of FMSs because AI techniques are developed to emulate human-like attributes such as intelligent decision-making and learning and to solve similar problems—problems involving a large search space, and where human expertise can find reasonable solutions pretty fast. Many researchers have sought to utilize this similarity. Several AI techniques have found use in the FMS literature: Expert Systems, Neural Networks, Fuzzy Logic, and Multi-Agent systems. In this section, expert systems are discussed in the context of developing an automated and intelligent, dynamic scheduling system.

“An expert system is a computing system capable of representing and reasoning about some knowledge domain with a view to solving problems and giving advice” (Jackson,

1990). The three basic components of an expert system are a knowledge base (where the knowledge is stored), an inference engine (which controls reasoning with the knowledge), and a user interface. Most commonly used expert systems are PC based and built with commercial shells (pre-fabricated expert systems without domain knowledge). Some systems are written in an AI language, such as PROLOG or LISP, while some are written in languages such as C.

Production control and scheduling has become one of the primary areas of use of expert systems in manufacturing. Expert system technology works best in domains where there is a substantial body of empirical knowledge connecting situations to actions. Obviously, this kind of situations exists when dealing with production scheduling.

Instead of attempting to develop a complex algorithm to represent the overall behavior of the system, the model of the system is broken down into a series of simple rules whose individual behavior is combined to effectively model the behavior of the entire system. An expert system attempts to emulate the decision-making capability of an expert by capturing his/her knowledge and representing it in rules of the form:

IF <antecedent>
THEN <consequent>

The structure of a rule-based expert system is fundamentally different from the algorithmic structure of procedural programs. The most striking difference is that the expert system separates knowledge and facts. Knowledge (i.e., the rules) is stored in a knowledge base, whereas the facts (i.e., the variable data that describe the state of the system and concerning which the rules reason) are stored in a separate database. The structure of a rule-based expert system is described in Figure 2.3. The reasoning core of the expert system is its inference engine. It uses the fact-base to assess the status of the

system and consequently invokes rules in the knowledge base. The explanation facility provides a human machine interface that helps developers and users alike in understanding exactly what the system is doing and more importantly, why it is doing.

Figure 2.3 Structure of a Rule-Based Expert System (Negnevitsky, 2002)

According to Buchanan and Smith (1988), in developing an expert system the following phases are important:

- Knowledge Acquisition — is the transfer and transformation of potential problem solving expertise from some knowledge sources to a program.
- Knowledge Representation — is the encoding of knowledge in data structure for symbolic (non-numeric) data, such as rule sets, generalized graphs and predicate logic.
- Knowledge Application — deals with how and when to use the knowledge.

Giarratano (1998) describes a number of attractive features of expert systems, such as, increased availability, reduced cost, permanence of expertise, increased reliability, fast response, steady at all times and intelligent database, etc. Some limitations of expert systems, such as lack of causal knowledge and limited expertise to the knowledge domain, are also discussed by Giarratano (1998).

Actually, expert systems can be seen as an extension of the use of heuristics. Selection of the rules generated by an expert system is based on encoding of an expert domain specific knowledge. Although heuristics, that is, rules of thumb or empirical knowledge gained from experience, cannot guarantee an exact solution to a problem, they can provide valuable shortcuts that can reduce both time and cost in searching for a satisfactory solution. This advantage is fully displayed when dealing with a dynamic scheduling problem where a balance is needed between the quality of solution and responsive time. Modern scheduling approaches often use expert systems linked to possibly more conventional scheduling heuristics or algorithms for developing detailed schedules. The expert systems may often act as an expert scheduler assistant rather than a stand-alone expert scheduler. The systems are used to suggest search heuristics to schedulers (Toal et al., 1994).

2.4 Simulation and Scheduling

Simulation is certainly more tractable than mathematical programming formulations of FMS operating problems. With simulation, models can be constructed to reflect real world systems more accurately. When using data from an actual FMS, the simulation model will be a “real production system”. In fact, a simulation model can be built as close to reality as one needs to do so. Many “what if ” questions can be answered. Therefore, simulation can normally work as decision support tools when there is a need to test different decision alternatives.

Discrete event simulation has long been used effectively in design, implementation and operation of FMSs, including real time scheduling. Basically, simulation is used as a test bed for evaluating schedules or scheduling methods on various system environments. Since advancement of computer technology has significantly reduced computational time for this evaluation, simulation has now become a popular tool for real-time evaluation in many circumstances.

It is most commonly seen that discrete event simulation is used to evaluate dispatching rules. Tang (1993) describes a study of decision rules used for scheduling an FMS. Bengu (1994) uses simulation to determine values of parameters in the COVERT (cost over time) rule, since the values that give the best performance vary due to changes in system environments. Sabuncuoglu (1998) compares rules under various experimental conditions by using an FMS simulation model. The objective is to measure sensitivity of the rules to changes in processing time distributions, breakdown rates, and AGV priority schemes. Jeong and Kim (1998) present a simulation-based real-time scheduling mechanism in which job dispatching rules vary dynamically based on information from discrete event simulation that is used for evaluating candidate dispatching rules. See Figure 2.4 for the

architecture of the scheduling mechanism. Major components of the scheduling mechanism are a simulation mechanism and a real-time control system. The simulation mechanism evaluates dispatching rules and selects the best dispatching rule for a given criterion, while the real-time control system periodically monitors the shop floor and checks the performance value of the system. The selected dispatching rule is used until the difference between the actual performance value and the value estimated by the simulation exceeds a given limit and/or a major disturbance occurs, and then a new simulation is performed with the remaining operations to select a new dispatching rule.

Figure 2.4 Architecture of the Real-Time Scheduling Mechanism (Jeong and Kim, 1998)

2.5 Scheduling Systems Incorporating Artificial Intelligence and Simulation Techniques

In recent times, systems incorporating artificial intelligence and simulation techniques have been used within the domain of production scheduling. These kinds of hybrid

systems combine the strengths of technologies in solving complex scheduling problems. For example, expert systems are usually rule-based and help in the selection of possible scheduling heuristics, while simulation can then test in detail the effect of a certain scheduling decision. Therefore, it is quite natural to integrate these two technologies together to build a more capable scheduling system.

Wu and Wysk (1989) present a Multi-Pass Expert Control System (MPECS) in which an expert system generates candidate dispatching rules and a simulator evaluates the rules. The detailed structure for this system is shown in Figure 3.2. Generally, expert systems require large volumes of domain knowledge. Because scheduling knowledge tends to be system specific, it has proven very difficult to develop a generic knowledge base for use across various scheduling applications. However, the MPECS system is capable of generating and analyzing its own system data. This effectively reduces the substantial work needed in collection and collation of system knowledge and increases the generic applicability of the MPECS model.

One of the typical features in dynamic scheduling is rescheduling, which means a scheduling system must test the performance of candidate scheduling solutions before choosing the best one for implementation. Undoubtedly, adding simulation models into scheduling systems as test beds is a good choice.

Chapter 3 A Proposed Approach for Dynamic Scheduling in Flexible Manufacturing Systems

3.1 Introduction

From the literature review, it can be seen that a lot of effort has been made in proposing not only theoretically sound solutions but also more importantly, practical solutions. This is because only optimal or “good” scheduling solutions can take advantage of the production potential of highly automatic and costly FMSs, hence making enterprises more agile and more competitive in the ever-changing markets. Moreover, a practical solution to the general scheduling problem can also lead to solving those optimization problems which exhibit similar characteristics and have huge solution spaces to search.

A simple but practical approach is proposed in this chapter for solving the dynamic scheduling problem in flexible manufacturing systems. The factors affecting the selection of an appropriate scheduling technique for flexible manufacturing systems are analyzed. The proposed approach for solving the dynamic scheduling problem is developed. A detailed discussion of the proposed search technique is presented. The software structure used in implementing the proposed scheduling approach is discussed.

3.2 Selection of a Dynamic Scheduling System for Flexible Manufacturing Systems

The literature review in Chapter 2 shows that there are many possible scheduling approaches available for FMSs. But there is no one approach absolutely overwhelming others. Different approaches may be applicable in different situations. The key for choosing an appropriate scheduling technique is to take into account the characteristics of an actual FMS. Since, in FMSs, operations are controlled by computers, setups between

consecutive operations are conducted automatically, and most of the operations processed by NC machines, it can be assumed that processing times are almost deterministic. This means that if there are no system disturbances, the result of a schedule can be predicted in advance. Therefore, in some cases, a predetermined, off-line schedule may be enough, although the problem is difficult to deal with analytically. However, actual states of FMSs may not be predictable because of unexpected events such as new orders arrivals, machine states (breakdowns), tool damages and urgent jobs. Because of such a dynamic and uncertain nature of the system states, pre-fixed, off-line scheduling may be impractical for FMSs. "In general, FMSs are more sensitive to system disturbances than conventional manufacturing systems because of a tighter synchronization, system integration and inter-dependencies among automated components" (Jeong and Kim, 1998). Therefore, a quick response to changes in system states is required, and this can be achieved only by real-time (dynamic) scheduling. Real-time (dynamic) scheduling means that scheduling or rescheduling actions must respond to actual system state changes in real time. A system's performance is closely related to the speed at which the control system makes scheduling decisions.

Obviously, this immediate response requirement makes optimization techniques impractical in dynamic scheduling because optimization techniques, such as linear programming and branch and bound algorithms, try to find the optimum solution for a scheduling problem by searching large portions of the solution space. Inevitably, it takes a relatively long time. When the problem dimension increases, optimization techniques rapidly become ineffective. On the other hand, approximation methods are not driven by an overriding objective to find an optimum solution. They try to achieve an acceptable balance between the quality of the solution and the time taken to find that solution. In order to select an appropriate scheduling technique for FMSs, it is more meaningful to abandon optimization methods and turn to approximation methods because the evaluation

time involved in dealing with disturbances is shorter, therefore providing dynamic scheduling capability.

From Figure 3.1 it can be seen that optimization techniques can guarantee the best results for an infinitely long evaluation period. On the other extreme, there are dispatching rules. Dispatching rules are heuristic approaches. Although for certain performance measures, dispatching rules can produce optimal solutions, they cannot guarantee optimality. However, they can produce acceptable solutions “instantaneously”. Some elaborate dispatching rules can even produce good results for certain performance measures. Between the two extremes, there are typical iterative neighborhood search techniques, i.e., approximation methods, which might produce very good solutions in a reasonable period of time. Some well-designed techniques can generate multiple search start points so as to improve the quality of the solution.

Figure 3.1 Scheduling Techniques: Quality versus Time

Since approximation methods can only produce “good” rather than guarantee optimal results, so the more time is spent on searching, i.e., the more portions of solution spaces are searched, the more likely the results become better. Therefore, a system should be designed to be able to test as many alternatives as possible until the next decision is

required (i.e., maximizing the use of the scheduling system). However, the decision time requirement varies under different circumstances. Even the same exception, machine breakdown, or new order arrival may lead to different time requirements. For example, if a new order arrives that is considered to be more important than an order that is currently in production or about to be produced, then the decision of whether or not to exchange the two jobs may be of critical importance. Conversely, if an order arrives that is less important than all other scheduled orders, then this scheduling decision is not important at this time. Therefore, the scheduling system is required to be capable of dynamically and intelligently tailoring itself to suit actual conditions in the FMS.

Generic (general) approximation techniques described in Chapter 2 can provide “good” solutions to various kinds of problems. Because of ignoring domain knowledge related to a concrete problem, they are usually required to search far more solution space before locating an “acceptable” solution. Therefore, to increase the efficiency of the search process, it is better to integrate scheduling domain knowledge, such as dispatching rules, priority rules and other problem specific knowledge, which results in so-called hybrid techniques. Furthermore, to take advantage of domain knowledge and to allow the scheduling system to respond differently according to the type and the urgency of each scheduling request, intelligence capability should be added to the dynamic scheduling system. To some extent, adding artificial intelligence ability, such as expert system, within the scheduling system, can be seen as an extension of the use of heuristics where the selection of rules to apply are suggested by the expert system. This allows the system to automatically apply the heuristics like an expert scheduler.

To choose the most suitable dispatching rule or the best schedule among candidates in varying system situations, their performance must be evaluated first. This involves huge requirements of computation if traditional analytical computational methods are used.

Thanks to the advancement of computer technology, simulation is often used for evaluating candidate schedules. Therefore, this advanced decision support tool for real time scheduling of manufacturing systems should be utilized.

In fact, as described in the literature review, some researchers have combined AI and simulation techniques to pursue a richer modelling capability for scheduling and control. Wu and Wysk (1989) present a multi-pass scheduling approach in which an expert system generates candidate dispatching rules and a simulator evaluates the rules, as shown in Figure 3.2.

O'Keefe and Rao (1988) propose methodologies for determining parts input sequences using simulation and a fuzzy rule base. These approaches have similar characteristics that come close to meeting the requirements of an FMS dynamic scheduling system. Therefore, they inspire the approach proposed in this thesis.

Figure 3.2 A Multi-Pass Expert Control System (MPECS) for Flexible Manufacturing Systems (Wu and Wysk, 1989)

3.3 A Proposed Approach for Dynamic Scheduling

Comprehensively considering various factors involved in dynamic scheduling for flexible manufacturing systems, an approach for dynamic scheduling is proposed, as shown in Figure 3.3. The approach consists of an AI scheduler (AIS), a simulator and a Supervisory Control and Data Acquisition (SCADA) system.

Figure 3.3 A Proposed Approach for Dynamic Scheduling

The AI scheduler is a knowledge and rule based expert system to drive the scheduling process. It is activated by a scheduling request to choose a subset of rules. Each prospective schedule that results from the scheduling rules is passed directly to the

simulator for evaluation and the simulation results of each candidate schedule are sent back to the AIS. Instead of sending the whole group of schedules at once, the AIS firstly arranges the schedules in decreasing order of priority based on the conditions of the FMS. Then the AIS passes the schedules one by one to the simulator for evaluation. After the results of a schedule are sent back to the AIS, the AIS compares the results of this particular schedule to the results of other schedules that have been tested before. By observing trends in results, the AIS can decide whether or not to keep testing more candidate schedules further. If the information is sufficient to conclude that it has already located the “best schedule” within the defined time limit, then the scheduling process in this period of time is complete.

The simulator is a discrete event simulation model of the FMS. This model simulates the interaction of objects in the FMS. It is used as a support tool by the AIS to evaluate the performance of different schedules.

As soon as a “best schedule” is selected, it is sent to the FMS’s control or configuration module so that it can be implemented. In most real world FMSs, the control module is actually a Supervisory Control and Data Acquisition (SCADA) system. Generally, the SCADA system is in complete control of the machine controllers and any feedback of information from the FMS to the AIS is executed and managed by the SCADA system. A further additional feedback loop from the SCADA system to the simulation model is also created. This loop is used to update the simulation model, making it reflect changes and variations in the FMS, thus ensuring the effectiveness of the simulation tests and the results produced. Once a schedule is decided, the SCADA system issues appropriate control commands to the machine controllers and responds to the feedback signals.

3.4 Proposed Search Technique

For any scheduling system, the ultimate goal is to optimize a cost function. The neighbourhood search (approximation) methods can be roughly divided into three categories based on their respective likelihood to find the optimum results (Flanagan and Walsh, 2003). The first one, the simplest one, uses a conventional dispatching rule as a starting point and then searches for a local minimum point in the solution space. The second and more complicated category is to allow the search process to move from local minimum points and go towards a global optimum. This is realized by allowing the search process to temporarily accept some worse results of the objective function because these worse results might lead to overall improvements in the long run. The third and the most elaborate approach uses multiple starting points to search many areas of the solution space, locating a few of troughs and crests (local optimum), therefore increasing the possibilities of finding a global optimum, or at least finding a better results than those obtained from a single starting point.

No matter what methods are adopted, they need to search quite large portions of the solution space, especially for those iterative search methods designed to be generally applicable to many problem types. Obviously, this time-consuming process contradicts the requirements of FMS dynamic scheduling that response time should be immediate. That is why dispatching rules are often used when meeting with disturbances such as machine breakdowns where immediate response is absolutely vital. However, simple dispatching rules usually result in sub-optimal solutions. Therefore, in some circumstances, it may be better to spend a little more time testing a little more alternatives which might lead to significant improvements in the objective function. In other words, sometime a trade-off exists between machine-idle-time and the number of possible schedules that can be evaluated and this might lead to better quality of results. This

trade-off necessitates the need for flexibility and intelligence in the scheduling system.

In order to make this kind of flexibility and intelligence possible, a simple search technique which can be incorporated into an intelligent rule-based system is needed. The intelligent system should allow the search process to respond differently according to different scheduling requests.

Testing every permutation of a sample schedule may appear to be clumsy, but it does reveal a clear trend or pattern to some extent. A group of six orders is selected randomly with respect to the hypothetical FMS (discussed in Chapter 4). The complete enumeration of the six orders equates to the evaluation of 6! or 720 schedules. Even if the evaluation time of a schedule in the simulation model is only 7 seconds, the total testing time is 5040 seconds or 1 hour 24 minutes. Adding a single order could increase the testing time to 9.8 hours. Table 3.1 is a sample set of orders containing order ID, product name, quantity, due date and number of components. Here, Product 1 consists of three components and Product 2 has two components. The due date is given as the time periods from the present.

Table 3.1 Base Set of Orders for Permutation and Testing

Order I.D.	Product	Quantity	Due Date	Order Size (Total Number of Components)
101	P1	11	8	33
102	P2	14	8	28
103	P1	9	17	27
104	P2	8	25	16
105	P2	5	26	10
106	P1	12	32	36

The objective of this scheduling system is to optimize a cost function which is a measure of the cost of the deviation from the due date. Actually, the cost function can be very complex and might include factors such as a customer's or an order's importance to the manufacturer, etc. However, because the FMS here is a hypothetical one lacking details to develop a precise cost function, a more general cost function (i.e. the performance measure) is proposed:

$$C = \sum_{i=1}^P S_i \times |t_i - d_i| \times j_i$$

where C is the total cost of a schedule; t_i is the completion time of order i ; d_i is the due date of order i ; S_i is the total component number of order i ; and j_i is the product factor for order i . j_i is selected as follows: $j = 0.125$ when the order is Product 1 and $j = 0.15$ when the order is Product 2.

It should be pointed out that the specifics here have no significant meaning because the hypothetical FMS is not a real manufacturing system affected by a lot of external factors such as customers, suppliers and products, etc.

To find the pattern of the whole solution space, all schedules are divided into six groups by arranging each of six orders as the leading order within a group. Within each group, the leading order is fixed and all other orders are permuted. Each group contains the same number of schedules (i.e., 120). Each schedule in a group has a similar schedule in each of the other groups. For example, a set of similar schedules is listed in Table 3.2. They are within six groups respectively. These similar schedules can be obtained from a base schedule. In this case, the base schedule is Schedule 1. Each of the similar schedules is obtained by moving one of the other orders to the top of the schedule and keeping the remaining orders in the same sequence.

Table 3.2 Similar Schedules in Six Groups

Schedule 1 in Group 1	Schedule 2 in Group 2	Schedule 3 in Group 3	Schedule 4 in Group 4	Schedule 5 in Group 5	Schedule 6 in Group 6
101	102	103	104	105	106
102	101	101	101	101	101
103	103	102	102	102	102
104	104	104	103	103	103
105	105	105	105	104	104
106	106	106	106	106	105

The results in terms of the objective function for the entire set of permutations of these six orders are shown in Figure 3.4. Each point represents the cost value of a unique schedule. The results are plotted with increasing cost on the Y-axis. The X-axis defines different groups with different leading orders. As can be imagined, similar schedules in different groups produce similar results. This is because that for the similar schedules, most sequences remain unchanged except for the leading order. Therefore, relative interaction of orders is quite similar. The different results for similar schedules can be mostly due to the effects of different leading orders.

Figure 3.4 Entire Set of Solutions of Sample Schedules

Figure 3.5 shows the results of the similar schedules. It can be seen that the results of similar schedules relative to one another are actually good representations of how entire groups perform relative to one another. This trend becomes more striking for the optimum point in each group. It can be imagined that it would be better to make similar schedules close to the optimum point of each group. Therefore, the base schedule used to generate the other similar schedules must itself be close to the optimum. In other words, the starting point of the search should be close to the optimum solution. To make this possible, it is necessary to be aware of two points. One is that dispatching rules can be used to produce good quality results. Therefore, it is useful to use certain dispatching rules to prioritize each order. The other is that the performances of dispatching rules are

closely related to the objective function, i.e. the performance measure. Because the objective function here is to minimize the deviation from the due date in the system, a due date based dispatching rule is employed. According to this rule, orders should be arranged in order of due date. As for the orders with the same due date, they are arranged in order of decreasing product quantity. Prioritizing the sample orders in this way, the base schedule is shown in Table 3.3.

Figure 3.5 Results of Similar Schedules in Each Group

Table 3.3 Base Schedule according to Due Date

Order I.D.	Product Quantity	Due Date
102	14	8
101	11	8
103	9	17
104	8	25
105	5	26
106	12	32

Using this base schedule, similar schedules in each group can be generated in the way mentioned before. Each of similar schedules can be taken as a representative for its own group. Table 3.4 shows the representative schedule in each group.

Table 3.4 Representative Schedules within Each Group

Group 1	Group 2 (Base Schedule)	Group 3	Group 4	Group 5	Group 6
101	102	103	104	105	106
102	101	101	101	101	101
103	103	102	102	102	102
104	104	104	103	103	103
105	105	105	105	104	104
106	106	106	106	106	105

As shown in Figure 3.6, by superimposing the results of the representative schedules on the results of the entire set, it can be found that the pattern of the results of the representative schedules is similar to the pattern of the results of the entire set and the results of representative schedules are close to the optimum values in each group. Naturally, by comparing the results of group representatives, it is possible to separate a group or groups most likely to contain the optimum schedule from other groups with poor performances. If one group representative is outstanding against all others, then the search can be focused on this group and the other groups can be disregarded. Once only one group is focused on, the leading order in this group is fixed and can be implemented in the manufacturing system. This is quite useful in dynamic situations. In this example, the order 102 is fixed and the search should be focused within Group 2. The search process repeats the above procedure except that this time the leading order is fixed and

only five orders need to be considered. See Table 3.5 for representative schedules of sub-groups within Group 2. The search process continues in this way until the last order is fixed.

Figure 3.6 Results of Representative Schedules Superimposed on Results of Entire Set

Table 3.5 Representative Schedules of Sub-groups within Group 2

Schedule 1	Schedule 2	Schedule3	Schedule 4	Schedule 5
102	102	102	102	102
101	103	104	105	106
103	101	101	101	101
104	104	103	103	103
105	105	105	104	104
106	106	106	106	105

Sometimes it is difficult to distinguish between groups when the results of the representative schedules are close to each other. In this situation, the Artificial Intelligent Scheduler is utilized to make the decision. As a matter of fact, the AIS is not only responsible for selecting dispatching rules but also for controlling the depth of search, in order to match the search time as closely as possible to the time available for scheduling. That means, if given enough time, the search can traverse the entire solution set completely until finding the optimum solution. If the given time is limited, i.e., the responsive time is required to be immediate, the search can go extremely to the dispatching rules. It is the flexibility, i.e., to adapt dynamically to changing situations that makes this search technique suitable for dynamic scheduling.

Sometimes if the AIS is forced to adopt an aggressive strategy, i.e., converging on a group too fast, it risks making an incorrect decision which could later compromise the search by trapping it in a local minimum point. But even in this circumstance, the search process must be subject to the responsive time requirements because, otherwise, the benefits brought by an optimum schedule might not outweigh the damage incurred by the machine downtime.

3.5 Software Structure for Realizing the Proposed Approach

To realize the proposed dynamic scheduling approach for flexible manufacturing systems, the software structure as shown in Figure 3.7 is designed.

Figure 3.7 Software Structure

3.5.1 Simulation Module

As a support tool, the simulation model plays a key role in the realization of the proposed dynamic scheduling approach, as shown in Figure 3.8. An accurate simulation model is vital to the successful evaluation of manufacturing schedules. A detailed description of the development of the simulation model of a hypothetical FMS is given in Chapter 4. The AweSim!3.0 is chosen to develop the simulation model.

Figure 3.8 Schedule Evaluation Loop

3.5.2 Artificial Intelligent Scheduling Module

The artificial intelligent scheduler (AIS) is a rule-based expert system. CLIPS is chosen to develop the AIS. CLIPS, an acronym for C Language Integrated Production System, is an expert system development platform, designed using the C programming language at NASA Johnson Space Center with the specific purpose of providing high portability, low cost, and easy integration with external systems. It is a multi-paradigm programming language that provides support for rule-based, objective-oriented, and procedural programming.

3.5.3. SCADA System

Normally, a Supervisory Control And Data Acquisition or SCADA system is available in FMSs, along with machines and controllers. Since there is no real FMS concerned in this project, the SCADA part is excluded. Attention is focused on other modules except for the SCADA system.

3.5.4. Central Database

The central database is important in this scheduling system. It helps to prevent data redundancy and inconsistency among different modules. To keep the database updated, each of the other software modules should be able to read and write to this database. In this thesis, ACCESS 2000 is chosen for its ease of application.

3.5.5. VB Interface

The VB interface is developed in Visual Basic 6.0. The main task of the interface is to facilitate communication between each software model, to manage and control data transfer, to manually generate candidate schedules and send them to the simulation model for testing, and to provide automatic control of the simulation model. The detailed functionalities of the interface are described in Chapter 5.

Chapter 4 Simulation Model of a Hypothetic FMS

A dynamic scheduling approach for flexible manufacturing system is proposed in Chapter 3. In the proposed approach, a simulation model is used by the Artificial Intelligent Scheduler (AIS) as a decision support tool. Specifically, according to the overall scheduling system described in Chapter 3, the FMS simulation model should be capable of doing the following:

- Receiving candidate schedules from the AIS.
- Resolving each schedule into the list of components and simulating the schedule.
- Feeding back the results of each simulation run to the AIS.

To meet above requirements, the simulation model development environment must be capable of :

- Reading information from and writing information to an Access database.
- Communicating with all other software modules.
- Communicating with the scheduling system interface.

4.1 A Hypothetic FMS

Since there is no existing FMS available which can be used to realize the proposed dynamic scheduling system, a hypothetic FMS must be designed. This hypothetic FMS must possess main characteristics of a typical FMS, which can be summarized as follows.

- High Degree of Functional Integration. The integration of an FMS can be summarized in two aspects, namely, integration among machining centers and integration between machines and the transportation subsystem. The integration between machines and the transportation subsystem is realized by adopting

automated material handling equipments, such as robots and AGVs.

- Arbitrary Operational Sequences with Negligible Setup Delays. When machining centers are capable of performing the same kind of operations, there exist alternative paths to route parts, especially when the optimal system performance of the FMS is pursued (Caprihan and Wadhwa, 1997).
- Complex Tool Management. If part types to be processed within a certain time period are changed and tool requirements are different, tools have to be exchanged. Therefore, tool management is so critical to an FMS that it requires a complex tool supply subsystem, similar to the material handling subsystem in an FMS. The tool management subsystem transports, prepares and stores tools required for the processing of workpieces between the local tool magazines and the central tool storage. Ideally, tool exchange should not interrupt the current running process so that the tool exchange effect is negligible. Therefore, it can be ignored in the simulation models. Unfortunately, in practice, the tool management problem may become the bottleneck of the whole FMS. But in this research, tool management issues are not addressed.
- Complicated Control Software. The highly automated nature of the FMS is achieved by a complicated system operating on a cell controller computer. The control system supervises most of the activities within an FMS, such as, downloading NC programs to machining centers, moving parts and tools around using transportation equipment (e.g., a robot or an AGV), synchronizing the connection between a machine and the transportation system, and issuing commands to each individual machine. All components in an FMS must be linked with the cell controller. The cell controller has to respond quickly to the request from any equipment. The data collection and status

check, often neglected in other conventional production systems, are indispensable requirements for the operation of an FMS.

Based on aforementioned characteristics of FMSs, a hypothetical FMS is proposed, containing the following components:

- 1 PC cell controller.
- 2 CNC lathes.
- 2 CNC mills.
- 2 material handling robots.
- 3 index tables (WIP storage).
- 4 belt conveyors (with built-in micro-controllers).
- 2 automatic part dispensers (computer controlled).

The layout of the hypothetical FMS is shown in Figure 4.1.

Figure 4.1 Layout of the Hypothetic FMS

One unique feature of the FMS is that the robots are seated on a slideway instead of a fixed location. The slideway provides an additional axis of movement to the robot so that each robot can move between two machines and is capable of performing part load/unload service to these two machines. Another special feature of the FMS lies in its symmetrical equipment layout. Along the central line, the machines on both sides are identical, with the ability to provide alternative part routing when the other is down. Robots are not able to cross sub-cells, thus the index tables are used as the buffers to connect the two sub-cells, in addition to its basic function as the working-in-process storage. With the help of the index tables, the system enables more complex material flow within the FMS so as to better emulate real world manufacturing systems.

There are two kinds of product to be processed in this FMS, Product 1 and Product 2. The basic operating sequence of the FMS is described below. Products 1 and 2 enter the FMS from the two part dispensers respectively and follow through the loading (input) conveyors connected to the dispensers. Parts wait at the end of the loading conveyors for a robot to carry them to dedicated machines to be processed. The robot performs the material handling function. Thus, if a part needs further operation on another machine, the robot picks up the in-progress part and delivers it to the destination machine. When all operations of a part are completed, the robot transports the part out of the FMS by one of two unloading conveyors.

Product 1 contains 3 components while Product 2 has 2 components. Each component requires operations on milling and lathe once. For processing time on each machine, see Table 4.1. To make the simulation running time not too long, short processing times are assumed. Generally, it is assumed that Mill 1 is better than Mill 2, whilst Lathe 2 is better than Lathe 1.

Table 4.1 Processing Time for Each Component

Component I.D.	11	12	13	21	22
Mill 1	5	6	8	3	7
Mill 2	6	7	10	5	9
Lathe 1	10	7	9	8	8
Lathe 2	8	5	6	7	6

4.2 Simulation Model Development Environment — AweSim!

AweSim is a general-purpose simulation system that supports a range of tasks required to perform a simulation project. It uses the Visual SLAM simulation language to build network, sub-network, discrete event, and continuous models. AweSim also provides integrating capabilities to store, retrieve, browse and communicate with externally written software applications. The architecture of AweSim is shown in Figure 4.2. The most fundamental feature of the AweSim architecture is its openness and interconnectivity to databases, spreadsheets and word processing programs such as Microsoft Office. AweSim is built in Visual Basic and C/C++, so programs written in these languages are easily incorporated into its architecture. The details on the capabilities of AweSim can be found in the AweSim User's Guide (Pritsker and O'Reilly, 1999). All AweSim features are accessible from the AweSim Executive Window through pull-down menus and dialog boxes, as shown in Figure 4.3.

Figure 4.2 The AweSim Architecture (Pritsker and O'Reilly, 1999)

Figure 4.3 AweSim Executive Window

4.3 AweSim Simulation Model of the Hypothetic FMS

One of important factors to the success of a simulation model is the detail level at which the model is built. For a real FMS, it is important to model the flow of parts through the system as precisely they would go through in real situation. But for this hypothetical, simplified FMS, those factors that have little effect on the flow of parts, for example, the control system, have already been neglected. Therefore, the most important factors left are the different processing times of the same type of machines because they contribute the most to the scheduling results. The Visual SLAM model of the FMS is given in Figure 4.4.

4.3.1 Parts Entry

Since, for the simulation system, schedules consisting of groups of orders are given, the input for each simulation run is fixed. The information about all these orders is available in the Access database before the initiation of the scheduling procedure. The entry of parts orders into the FMS is modeled as a queue of products awaiting manufacturing in the order that they are scheduled. A READ node is used to open the database file and read variables of part type and part quantity.

4.3.2 Parts Routing Strategy

The proportion of parts processed by each of two identical machines can be significant in terms of overall lead-time for a particular order. This is mainly because of the difference in performance of the two machines. The processing time of different components is different, but the processing time of each component on each machine is assumed to be fixed. The strategy adopted here is to balance the total processing time for each individual machine of the same kind, not the total components assigned to each machine. That is why QUEUE and SELECT pair nodes are used.

4.3.3 Transforming Orders into Lists of Components and Products

A schedule is a group of orders arranged in a particular order for manufacturing. But in this FMS simulation model, the basic entities are components to be manufactured individually, so UNBATCH nodes are used to do this transformation. Figure 4.5 shows a Sample Group of Orders. It should be pointed out that, because the AweSim!3.0 used is a student version with a limitation of a total of 300 entities, the product quantity is less than 10, in order for the total entities in the model not to exceed the limit.

	OrderID	Product	Quantity	DueDate	Total Component
1	102	1	9	16	45
2	101	2	4	16	12
3	103	2	6	24	18
4	104	2	2	24	06
5	105	1	5	32	25
6	106	1	8	32	40
7					
8					
9					

Figure 4.5 Sample Group of Orders

4.3.4 Recording and Storing Completion Times

As mentioned before, the objective function has been established as the cost of the deviation from the due date. The lateness of a particular order equals the difference between the order due date and the order completion time. The order due date is a customer related requirement, whilst the order completion time is determined by its position in the schedule and the general performance of the schedule. Moreover, in this dynamic scheduling system, a schedule, even if it is the final best one, needs to remain open to changes even after manufacturing has begun. Therefore, the completion time of each order must be recorded. The WRITE node is utilized to write data into a spreadsheet file. The cost function values are calculated in the related spreadsheet.

4.3.5. Control File and Attribute Index

The processing times for the mills and lathes are stored in a set of arrays within the control file. The simulation control file is as follows:

“ GEN, “Wence”, “FMS”, 2005/08/20,, YES, YES;
 LIMITS,,,,8;
 ARRAY,1,5,{5,6,8,3,7};
 ARRAY,2,5,{6,7,10,5,9};
 ARRAY,3,5,{10,7,9,8,8};
 ARRAY,4,5,{8,5,6,7,6};
 NET;
 FIN;”

The attribute index is shown in Table 4.2.

Table 4.2 Attribute Index

Attributes	Descriptions
ATRI[0]	Starting Time of a Schedule
ATRI[1]	Product Type
ATRI[2]	Product Quantity
ATRI[3]	Component I.D.
ATRI[4]	Processing Time of Mill 1
ATRI[5]	Processing Time of Mill 2
ATRI[6]	Processing Time of Lathe 1
ATRI[7]	Processing Time of Lathe 2
LL[1]	Number of Value Read

Chapter 5 Implementation of the Dynamic Scheduling System

5.1 Introduction

Chapter 4 discusses the development of simulation model which is used in the scheduling system. This chapter describes the actual realization and implementation of the intelligent scheduling system proposed in Chapter 3. Firstly, the issue of communication among each individual module is addressed. The inter-modular communication, data transfer, and data storage are the primary prerequisites for implementation of the scheduling system. Secondly, with the introduction of functions of the scheduling system interface, the manual scheduling process is discussed. Through the interface, sample schedules can be manually generated and then sent to the simulation model for evaluation. By comparing the results, the best schedule can be determined by the person who operates the system. Finally, the automation of the scheduling system is detailed with a case study. For the automation of the system, the human decision-maker is replaced by an Artificial Intelligent Scheduler — an expert system. The AIS undertakes the control of the system and executes the search strategy described in Chapter 3. Based on the real situation and using the search strategy, the AIS recommends a few candidate schedules. Those candidate schedules are sent to the simulation model for evaluation automatically, so that the “best” schedule is selected accordingly. In Section 5.4, the development environment for the AIS is introduced, and an example is used to illustrate the automatic scheduling mechanism and to explain the programming process.

5.2 Inter-Modular Communication — Dynamic Data Exchange (DDE)

The scheduling system proposed involves several software modules; so, inevitably, the smooth communication among these modules is critical to the functioning of the system.

Communication between modules includes data transfer and remote control. If all modules are running within one computer, the situation would be very simple. The communication would be simply calling external functions or methods. Fortunately, all the developing platforms here, such as AweSim and CLIPS, are able to call externally written software applications or to be embedded into other applications and used as subroutines. However, in real situation, different modules are more likely to be located in different machines, communicating with each other around a database. Therefore, an appropriate communication protocol, which supports local communication on the same machine, as well as remote communication via a network, must be chosen. For this purpose, a general-purpose communication protocol — Dynamic Data Exchange (DDE) is used. The DDE can be extended to Network DDE, so that applications can be operated on different computers.

DDE is a data exchange protocol supported by most Windows based applications. It employs a message-based mechanism, not functions or libraries, to pass values from a sending application (client) to the receiving application (server). It is most appropriate for data exchanges that do not require ongoing user interaction. Usually, an application provides a method for the user to establish the link between the applications exchanging data. Once that link is established, however, the applications exchange data without further user involvement, which paves the way for the automation of the scheduling system.

The general form for DDE conversation is a three-level hierarchy of application, topic, and item names, as follows:

`\\NodeName\ApplicationName|topicname`

Once a conversation has been built, data query can be performed between the remote applications about a specified DDE topic.

The VB interface is designed not only as a user interface but also as a hub through which messages can be passed between different modules as shown in Figure 5.1. It must be able to receive messages from external applications, recognize them, and interpret them. If necessary, it can transfer these messages to another application.

Figure 5.1 Inter-Modular Communication

For the VB interface to establish a DDE link with an application, its LinkMode, LinkItem and LinkTopic properties must be set up correctly. The LinkTopic property consists of a string that supplies part of the information necessary to set up either a destination link or a source link. The string used depends on whether it is working with a destination control or a source form. Each string corresponds to one or more elements of the standard DDE syntax, which includes application, topic, and item. For example, after changing LinkTopic for a destination control, LinkMode must be set to 1 (Automatic), 2 (Manual), or 3 (Notify) to establish a conversation with the new topic.

The relay role played by the VB interface in data transferring makes it possible that almost any software can access the database as long as it has a DDE ability, because even

if a software module does not have a database access ability, it can still use the interface's database access ability to access the database. This is very useful in communication.

However, the VB interface's function is not just to relay simple messages among modules. It can identify the message, for example, a string, to launch procedures embedded in the interface so that the remote activation of a procedure can be realized. These procedures include those that read from or write information to the database, control the simulation model remotely, and provide feedback to the interface.

5.3 VB Interface and Manual Operation

Since Visual Basic provides a powerful programming ability with easy learning and usage, as well as a capability to realize communication and data transfer, it is selected to develop the scheduling system interface. This interface has the following functions:

- Extracting orders from the database (allowing orders to be removed manually and their status in database to be reset).
- Generating candidate schedules manually.
- Activating the simulation model to evaluate candidate schedules.
- Feeding back simulation results and passing them to the Artificial Intelligent Scheduler (AIS).
- Allowing the above functions to be executed automatically at the request of the AIS, if a real FMS control system is incorporated.
- Making sure harmoniously integration of each software module involved.

The interface shown in Figure 5.2 allows a user to interact with the scheduling system and to realize the aforementioned tasks. Actually, manual interaction can be developed without involving the AI section and other automated functions. It clearly demonstrates the interface's capabilities. To make a good schedule, a human expert is needed to use

this manual function well. When this human expert is replaced by an AI system, the AI system acts the role played by the human expert in the manual system. Therefore, developing manual functions clearly shows inherent mechanisms in software modules and their relationships. This provides a foundation for realization of automated functions when the AI system is incorporated. To be specific, the AI system is intended to perform the search strategy discussed in Chapter 3 and to act like a human user in testing, evaluating, and comparing schedules, and in sending the orders into the manufacturing system (if a control system is incorporated).

Figure 5.2 Scheduling System Interface

5.3.1 Extracting Orders from the Database

The orders and related data are stored in the database. In the VB interface, it is designed so that the order information is picked up from the database to be shown in the order list

box at start-up using the form's load event code. Each time "ExtractOrders" command button is clicked, one unscheduled order from the database in descending order of importance is picked up. This arrangement is actually the starting point of the search and the base schedule for developing other candidate schedules. Through establishing an RDO (Remote Data Objects) connection, the interface accesses the database. It should be noticed that each time the search chooses an important order, only those un-transferred orders are considered, so as to prevent repeated references to the same order in the database. The second button of "Clear Orders" is to clear orders and importance value in the list boxes.

5.3.2 Generating Schedules Manually

After the orders are extracted from the database, a user can manually test any schedule combination. It is designed so that, by just clicking on any order in the order list, this order is added to the schedule list box. In order to facilitate the evaluation of a schedule using the simulation model in the next step, the interface takes this opportunity to pass each order name to the spreadsheet for the simulation model. Of course, it should be made certain that no order has been repeatedly added to the schedule list box. The "Clear Schedule" button allows a user to clear the schedule at any time. This clear button also clears the text box used to display the simulation results. In addition to clearing the above, the button also clears any schedules that have been created previously in the spreadsheet for the simulation model.

5.3.3 Evaluating Candidate Schedules

One of the main functions of the interface is to evaluate the candidate schedules using the simulation model of the FMS. The purpose of the simulation model is to evaluate and

compare the performance of each candidate schedule. Actually, when the orders are passed to the schedule list box, they are also passed to the spreadsheet for the simulation model, as mentioned before. This could be one of the steps that contribute to the automatic evaluation of schedules directly from the VB interface. The order list is stored in the spreadsheet for the simulation model in a sequence in which the orders are entered into the simulation model during a simulation run. Once the proper schedule is generated, what remains to do is to “Run” the simulation model. Clicking the “Test Schedule” button remotely activates the simulation process from the VB interface. This is realized by calling a user-written function. When the simulation is complete, the “Cost of Deviation” value is returned to the text box in the interface.

Once a satisfactory solution is located, the leading job of this schedule can be executed by the manufacturing system. Since there is no real control system involved in the model, though on the interface a “sending order” button is designed, it cannot be functioned at this time. It is here for the purpose of illustration.

5.4 Realization of Automation of the Scheduling System

The ultimate goal of this research is to develop an intelligent automated scheduling system using the approach proposed in the thesis. This section discusses the designing and implementing process, with focus on the Artificial Intelligent Scheduler. The AIS is an expert system based application developed in an expert system development environment called CLIPS. In order to successfully achieve the automation, the AIS should be able to act like a human scheduling expert to follow certain procedures and to make appropriate decisions. In other words, it should be able to control the functions automatically. To be precise, it should be able to arrange orders, to select candidate schedules for evaluation according to the search method proposed, to pass these

schedules to the simulation model, to analyze the results, to decide whether a schedule is outstanding or whether more tests are needed, and finally, if linked with an FMS control system, to transfer orders to the control system and to activate their production.

5.4.1 Expert System Development Environment — CLIPS

Selecting an appropriate expert system development environment is important for the successful realization of the automation function of the AIS. For this thesis, an expert system development environment called CLIPS is selected.

5.4.1.1 CLIPS Features

CLIPS is selected as the platform not only because of its free availability as an open source software but also because of its following features (Giarratano, 2004):

- Knowledge Representation: Rule-based, object-oriented and procedural programming are supported.
- Portability: CLIPS is portable with an ANSI compliant C or C++ compiler. All source code are available.
- Integration/Extensibility: CLIPS can be embedded within procedural codes and integrated with various languages.
- Interactive Development: An interactive, text oriented development environment, including debugging aids, on-line help, and an integrated editor, is provided.
- Verification/Validation: A number of features can support the verification and validation of expert systems.
- Fully Documented: Extensive documentation is available.
- Low Cost: CLIPS is a public domain software.

It is necessary to emphasize the CLIPS' integration ability because integration with other software packages is critical to realization of automation of the schedule system. CLIPS is designed to allow two types of integration:

- Embedding CLIPS into other systems. In these situations, CLIPS can be called as a subroutine and information may be passed to and from CLIPS.
- Calling external functions from CLIPS. It also may be useful to call external functions while executing a CLIPS construct or from the top-level of the interactive interface. CLIPS variables or literal values may be passed to an external function, and functions may return values to CLIPS. The easy addition of external functions allows CLIPS to be extended or customized in almost any way.

5.4.1.2 CLIPS Windows Interface

The CLIPS 6.2 Windows interface, as shown in Figure 5.3, is straightforward, following Windows interface standards. Commands can be entered directly in the dialog window or can be edited in other text editing programs and loaded into the CLIPS interface. The integrated editor is similar to that of other Windows editors. The execution menu not only includes running commands but also most commonly used debugging commands.

Figure 5.3 Expert System Development Environment — CLIPS

5.4.2 Objects in the Artificial Intelligent Scheduler (AIS)

One of the important programming elements in CLIPS is the CLIPS Object-Oriented Language (COOL). COOL includes elements of data abstraction and knowledge representation. This allows programmers to model a system by separately representing certain aspects of the system, physical or conceptual, using a series of user-defined objects. Objects are described in two basic parts: properties and behaviors. Therefore, the condition of the system at any time can be described by the properties and behaviors.

User-defined objects in CLIPS can be referred to uniquely by their addresses. Within the scope of a module they can also be uniquely referred to by their names. User-defined objects can only be manipulated via message-passing, which can be achieved conveniently through communication between different modules.

In order to represent the scheduling system in the CLIPS environment and to program the

AIS, it is necessary to define the following objects:

- Order.
- Candidate_Schedule.
- VB-Interface.

5.4.2.1 The “Order” Object

Originally, the information about orders is stored in the database. In order for the inference engine to reason about order information, the information must first be part of the knowledge base (in the form of rules). So the “Order” object is designed to store information concerning orders. The “Order” object’s properties includes an order’s ID, product ID, quantity, and due date.

5.4.2.2 The “Candidate_Schedule” Object

The main task of the AIS is to control the execution of the scheduling search strategy discussed in Chapter 3. This involves testing candidate schedules and evaluating the performances of these schedules as potential optimum solutions. Therefore, the AIS inference engine must be capable of reasoning about the performances of candidate schedules. So, the information concerned must also be contained in the application’s knowledge base. The “Candidate_Schedule” object is designed to represent the performance of an individual candidate schedule in the AIS. Therefore, in addition to the “Order” object’s properties, “performance” and “search level” are added to the properties of the “Candidate_Schedule” Object.

5.4.2.3 The VB-Interface Object

The VB-interface object is designed to allow the AIS to communicate with the VB interface.

5.4.3 Modeling Artificial Intelligent Scheduling System in CLIPS

The scheduling problem is represented in the AIS as a group instance of “Order” objects, each of which represents an individual order to be released for manufacturing. To automate the scheduling system, it is necessary to execute the scheduling strategy described in Chapter 3. A number of candidate schedules are sent to the simulation system and then tested. From these limited solutions, estimation can be made upon the position of the good schedule in the whole solution space. Therefore, the tasks of the AIS are:

- Choosing candidate schedules according to certain dispatching rules.
- Passing a candidate schedule to the simulation model for evaluation and testing it automatically.
- From the evaluation results, deciding candidate schedules for the next round of testing so as to control the search depth and width until drawing the final conclusion about the “optimum” or “best” schedule.

Since the search strategy here is somewhat similar to the branch and bound method in structure, the decision tree structure is a good way to illustrate the mechanism and functions of the AIS. Figure 5.5 depicts a complete tree structure of a scheduling problem involving four orders. 24 possible schedules can be extracted by tracing from level 0 to the maximum level. Actually, the AIS search strategy is to remove as many branches of the tree structure as possible in order to shorten the search time. The benefits become

obvious when the problem dimension is increased.

Figure 5.4 Complete AIS Tree Structure for a Scheduling Problem with Four Orders

As mentioned before, evaluated candidates are returned to the AIS and represented as “Candidate_Schedule” objects. Grouping these “Candidate_Schedule” objects graphically forms a tree structure that represents the search process. In actual implementation of the AIS, candidate schedules are those schedule solutions that have been highlighted as group representatives in Figure 3.6. The performance of each representative candidate is compared in order to ascertain which group or which candidate will eventually result in the “best” schedule. For this four-order scheduling example, level 0 of the decision tree means four candidate schedules. They are chosen by the search strategy as representatives respectively of a group of schedules with the same leading order. Level 1 of the tree involves groups of schedules with the same order in the second place, and so on.

In most cases, it is difficult to separate candidates with close results. If so, the search should continue at the next level in two or more groups, as shown in Figure 5.6.

Figure 5.5 Illustration of Search Strategy of the Four-Order Scheduling Problem

For this four-order scheduling example, the first round is to test four candidate schedules. After comparing the results, assume two results are better. Therefore, the other two branches at level 1 are pruned, leaving six candidate schedules to be tested further. The second round is to test these six candidates and to compare results. Suppose at this level, three results are better and they cannot be discriminated. Therefore, three branches are cut off, leaving three other branches. Therefore the third round is to test six more candidates, at level 2 of the tree, as shown in Figure 5.6. At level 3 of the tree, only one result can be picked up because it is the end of the iteration, even if two results are too close. Obviously, the search depth and width depend on the time available to respond.

5.4.3.1 Programming the AIS and a Case Illustration

The practical implementation of the AIS by coding is quite complex because of many programming considerations such as data transfer and calling external functions. To better explain the programming logic, a hypothetical case with four orders is used in the following.

- Task of the problem: Arrange the four orders shown in Table 5.1 in a schedule to minimize the objective of the cost of deviation. For the purpose of illustration, fewer

orders are used here.

Table 5.1 A Hypothetic Scheduling Problem with Four Orders

Order I.D.	Product	Quantity	Due Date
101	2	4	15
102	1	9	15
103	2	6	23
104	2	2	23

- Step 1: Extracting Orders

Originally, the order information is stored in the “Order” table of the central database. When all the applications are started, the VB interface begins to use its “ExtractOrderForClips()” subroutine to extract the order related data, such as Order ID, Customer ID, Product, Quantity, Due Date and Status, from the database and then to transfer these data to the AIS. Only those unscheduled orders are extracted.

- Step 2: Building the “Order” Object in the AIS

The record of each order is used to create an “Order” object in the AIS. A remote procedure call to the “Build_Orders” subroutine in the AIS is used to create these objects. The order data is passed as arguments of this call. The entire set of “Order” objects can be created by applying the aforementioned procedure through the database.

- Step 3: Starting Search Process

Once the problem is represented in the knowledge base, the AIS can use its rules to make

inferences for the problem. The scheduling search process is executed by the AIS and the VB interface and guided by three integer variables: LevelNo, SubgroupNo and SchedulNo. The LevelNo denotes the decision tree level. It ranges from 0 to $n-1$, where n is the number of orders. The SubgroupNo is the subgroup number. For this four-order example, level 0 is the whole problem and there is no subgroup. Level 1 involves 4 subgroups. The ScheduleNo defines the individual candidate schedule within a subgroup. These three variables can locate any node (any “Candidate_Schedule” object) in the decision tree. The search is implemented in the VB interface procedure. This procedure is called by the AIS.

Since there is only one group on level 0 for this example, there is no subgroup number. The search picks up an order according to the ScheduleNo variable and then adds the remaining orders in the order of decreasing priority to form a schedule. For this example, the first schedule, the base schedule (arranged in the descending order of priority), is 102,101,103,104. This schedule is passed to the simulation model.

- Step 4: Evaluation by Simulation

When a schedule is completed, the “TestSchedule_Click” is called to evaluate the schedule. The evaluation results are then returned to the VB interface and further relayed to the AIS.

- Step 5: Building “Candidate-Schedule” Objects

This step involves two tasks for the AIS. The first task is to generate all possible candidate schedules by iteration with respect to the three variables, i.e., LevelNo, SubgroupNo and SchedulNo. To be specific, the ScheduleNo is incremented by one after

each iteration. If the search reaches the boundary of a subgroup, then the SubgroupNo is incremented by one and the ScheduleNo is reset to its initial value. When the search reaches the end of a particular level, then the LevelNo is incremented by one and the other two variables are reset.

For this example, at level 0, LevelNo=0 and SubgroupNo=0, after the base schedule of 102,101,103, 104 is tested, the ScheduleNo is incremented by one at a time. Therefore, each of the schedules, as shown in Table 5.2 is generated as a candidate schedule and is tested. If illustrated by the decision tree, they form level 0 of the tree, as shown in Figure 5.7.

Table 5.2 Candidate Schedules at Level 0

Schedule 1 (Base Schedule)	Schedule 2	Schedule 3	Schedule 4
102	101	103	104
101	102	102	102
103	103	101	101
104	104	104	103
Cost Value	Cost Value	Cost Value	Cost Value
77.86	108.45	155.34	198.93

Figure 5.6 Level 0 of the Decision Tree

The second task in this step is to decide which portions of the schedule groups are more likely to contain the better schedules when the search reaches the end of a level. The selected groups are kept and all other groups are pruned. The search continues at one level higher. To be specific, at the end of each level, the “PruningRule” is activated by the AIS. This rule is to start a single instance of the “NotGood_Candidates()” procedure for each of the “Candidate_Schedule” objects at this search level. If the performance of a particular candidate is not within a specified value, this candidate is eliminated. For this example, the AIS decides that the groups as represented by schedule 1 and schedule 2 are kept for further testing.

- Step 6: Go Back to Step 3 to Continue the Search

The search continues at level 1. The search generates the candidate schedules, as shown in Table 5.3, which must be evaluated.

Table 5.3 Candidate Schedules with Results at Level 1

Schedule1	Schedule 2	Schedule 3	Schedule 4	Schedule 5	Schedule 6
102	102	102	101	101	101
101	103	104	102	103	104
103	101	101	103	102	103
104	104	103	104	104	102
Cost Value	Cost Value	Cost Value	Cost Value	Cost Value	Cost Value
77.86	119.07	90.32	108.45	134.76	155.21

As before, the candidates are sent to simulation for testing, and the results are also shown

in Table 5.3. If illustrated by the decision tree, they form level 1 of the tree, as shown in Figure 5.8.

Figure 5.7 Level 1 of the Decision Tree

The “PruningRule” is executed at the end of this level, terminating poorly performing schedule groups. This time, the schedule groups as represented by schedules 1,3 and 4 are retained. Continuing the search process, the candidate schedules are generated as shown in Table 5.4 and their evaluation results are shown as well. This time, the decision tree is shown in Figure 5.9.

Table 5.4 Candidate Schedules with Results at Level 2

Schedule1	Schedule 2	Schedule 3	Schedule 4	Schedule 5	Schedule 6
102	102	102	102	101	101
101	101	104	104	102	102
103	104	101	103	103	104
104	103	103	101	104	103
Cost Value	Cost Value	Cost Value	Cost Value	Cost Value	Cost Value
77.86	87.79	90.32	126.29	108.45	131.22

Figure 5.8 Level 2 of the Decision Tree

Again, the “PruningRule” is executed at the end of this level. Since this is a four-order scheduling problem, at level 2 the search is completed and the best schedule emerges as shown in Table 5.5. The complete decision tree is shown in 5.10.

Table 5.5 “Best” Schedule of the Example

Best Schedule
102
101
103
104

Figure 5.9 Complete Decision Tree for the Example

Chapter 6 Conclusions and Future Work

A dynamic scheduling system for flexible manufacturing systems (FMSs) is developed. The system consists of a rule-based Artificial Intelligence Scheduler (AIS), a simulation model, a database and a user interface. The AIS is used to generate candidate schedules and to control the search process based on the status of the FMS. Candidate schedules are evaluated by the simulation mode. The user interface is utilized to manually generate candidate schedules and facilitate communication between different modules so as to realize automatic scheduling. The schedule related data are stored in the database. The search strategy that uses a dispatching rule as the start point can find a good quality solution relatively quickly. This strategy is practical and suitable for dynamic scheduling in flexible manufacturing system. A hypothetical FMS is designed to demonstrate the dynamic scheduling system. The scheduling system is developed using different platforms around a central database.

6.1 Summary of the Contributions

The main contributions of this thesis can be summarized as follows:

1. A practical dynamic scheduling system for FMSs is developed.
2. A simple and practical search method is presented, which can be implemented in dynamic scheduling.
3. A hypothetical FMS is designed, which features all basic characteristics of a real world FMS. The simulation model of this hypothetical FMS is built.
4. An expert system as an intelligent scheduler is developed and the automation of the proposed dynamic scheduling system is realized.
5. Problems concerning the functioning of the proposed scheduling system, such as communication between different modules, user interface design, etc., are solved,

paving the way for extension of other modules in the future.

6.2 Future Work

A practical dynamic scheduling approach for flexible manufacturing systems is proposed in this thesis. Some future research topics are:

1. Based on a real world FMS, a detailed simulation and an expert system should be built. Using this real world FMS, the proposed approach can be implemented and tested.
2. A scheduling system based on multiple scheduling rules can be developed so that the AIS can use a library of scheduling rules based on the status of the FMSs. Therefore, the system can be more flexible and robust in dealing with dynamic conditions in the FMSs. To realize this, a control system must be incorporated.
3. Emulating unexpected situations, such as machine breakdowns, after incorporating a real control system, so that the advantages of this dynamic scheduling system can be fully displayed.

References

- Askin, R.G. and C.R. Standridge (1993), *Modeling and Analysis of Manufacturing Systems*, Wiley, New York.
- Baker, K.R. (1974), *Introduction to Sequencing and Scheduling*. Wiley, New York.
- Balas, E. (1969), Machine Scheduling via Disjunctive Graphs: An Implicit Enumeration Algorithm, *Operations Research*, Vol. 17, pp. 941-957.
- Banks, J., J.S. Carson II, and B.L. Nelson (1996), *Discrete-Event System Simulation, 2nd Ed.*, Prentice-Hall, Englewood Cliffs, New Jersey.
- Bengu, G. (1994), A Simulation-based Scheduler for Flexible Flowlines, *International Journal of Production Research*, Vol. 32, pp. 321-344.
- Brucker, P. (1988), An Efficient Algorithm for the Job-Shop Problem with Two Jobs, *Computing*, Vol. 40, pp. 353-359.
- Brucker, P. (2001), *Scheduling Algorithms*, Springer Verlag.
- Buchanan, B.G. and R. Smith (1988), Fundamentals of Expert Systems, *Annual Review of Computer Science*, Vol.3, pp. 23-58.
- Caprihan, R. and S. Wadhwa (1997), Impact of Routing Flexibility on the Performance of An FMS — A Simulation Study, *The International Journal of Flexible Manufacturing Systems*, Vol. 9, pp. 273-298.
- Cassell, P. (2000), *Sams Teach Yourself Microsoft Access 2000 in 21 Days*, Macmillan Computer Publishing, Indiana, USA.
- Chandra, J. and J. Talavage (1991), Intelligent Dispatching for Flexible Manufacturing. *International Journal of Production Research*, Vol.29, No.11, pp. 2259-2278.
- Chan, F. T. S. and H.K. Chan (2002), The State of the Art in Simulation Study on FMS Scheduling: A Comprehensive Survey. *International Journal of Advanced Manufacturing Technology*, Vol.19, pp. 830-849.
- Chiu, C. and Y. Yih (1995), A Learning-Based Methodology for Dynamic Scheduling in Distributed Manufacturing Systems. *International Journal of Production Research*,

Vol.33, 11, pp. 3217-3232.

Conway, R.W., W.L. Maxwell and L.W. Miller (1967), *Theory of Scheduling*, Addison-Wesley, Reading, MA.

Cowling, P.I., D. Ouelhardj and S. Petrovic (2003), A Multi-agent Architecture for Dynamic Scheduling of Steel Hot Rolling, *Journal of Intelligent Manufacturing*, Vol.14, pp. 457-470.

Davis, L. (1985), Job-Shop Scheduling with Genetic Algorithm, in Grefenstette J. J. (ed), *Proceedings of the 1st International Conference on Genetics Algorithms and their Applications*, Pittsburgh, PA, USA, Lawrence Erlbaum, pp. 136-140.

Dorn, J. (1995), Reactive Scheduling Improving the Robustness of Schedules and Restricting the Effects of Shop Floor Disturbances by Fuzzy Reasoning, *International Journal of Human Computer Studies*, Vol. 42, pp. 687-704.

Falkenauer, E. and S. Bouffouix (1991), A Genetic Algorithm for the Job-Shop, *Proceedings of the IEEE International Conference on Robotics and Automation*, Sacramento, California, USA, pp. 824-829.

Flanagan, B. and D. Walsh (2003), The Role of Simulation in the Development of Intelligence for Flexible Manufacturing Systems, *Proceedings of the 20th Conference of the Irish manufacturing Committee*, C.I.T., Cork, Ireland.

French, S. (1982), *Sequencing and Scheduling — an Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, Wiley, New York.

Gere, W.S. Jr. (1996), Heuristics in Job Shop Scheduling, *Management Science* Vol.13(3), pp. 167-190.

Giarratano, J. (1998), *Experts Systems —Principles and Programming*, 3rd edition, PWS Publishing Co, Boston, MA.

Giarratano, J. (2004), *CLIPS User's Guide*, Downloaded from CLIPS Download Area, <http://www.ghg.net/clips/download.html>, accessed in Dec, 2004

Giarratano J, (2005) *CLIPS Reference Manual*, Vol.1, 2, & 3, Downloaded from CLIPS Download Area, <http://www.ghg.net/clips/download.html>, accessed in Jan, 2005.

Gonzalez, T. and S. Sahni (1978), Flowshop and Jobshop Schedules: Complexity and

Approximation, *Operations Research*, Vol. 20, pp. 36-52.

Greenwood, N. R. (1988), *Implementing Flexible Manufacturing Systems*, MacMillan Education Ltd., London, UK.

Ignall, E. and L. Schrage (1965), Application of the Branch and Bound Technique to Some Flow-Shop Scheduling Problems, *Operations Research*, Vol.13, pp. 400-412.

Jackson, P. (1990), *Introduction to Expert Systems, 2nd edition*, Addison-Wesley, Boston, MA.

Jain, A.S. and S. Meeran (1998), A State-of-the-Art Review of Job-shop Scheduling Techniques, *Technical Report*, Department of Applied Physics, Electronics and Mechanical Engineering, University of Dundee, Scotland.

Jeong, K., and Kim Y.D. (1998), A Real-time Scheduling Mechanism for a Flexible Manufacturing System: Using Simulation and Dispatching Rules. *International Journal of Production Research*, Vol. 36, no. 9, pp. 2609-2626.

Kouliss, K, H. Pierreval and N. Mebarki (1997), Using Multi-agent Architecture in FMS for Dynamic Scheduling, *Journal of Intelligent Manufacturing*, Vol.8, pp. 1-47.

Kusiak, A. and J. Ahn (1992), Intelligent Scheduling of Automated Machining Systems, *International Journal of Computer Integrated Manufacturing Systems*, Vol.5(1), pp. 3-14.

Lenstra, J.K. and A.H.G. Rinnoy Kan (1979), Computational Complexity of Discrete Optimisation Problems, *Annals of Discrete Mathematics*, Vol. 4, pp. 121-140.

Luggen, W. W. (1991), *Flexible Manufacturing Cells and Systems*, Prentice-Hall, Englewood Cliffs, New Jersey.

McMahon, G. B. and M. Florian (1975), On Scheduling with Ready Times and Due Dates to Minimize Maximum Lateness, *Operations Research*, May-June, Vol. 23(3), pp. 475-482.

Montazer, M. and L. Van Wassenhove (1990), Analysis of Scheduling Rules for an FMS, *International Journal of Production Research*, Vol.28, pp. 785-802.

Morton, T. E. and D.W. Pentico (1993), *Heuristic Scheduling Systems With Applications to Production Systems and Project Management*, Wiley, New York.

MSDN Library (2005), *Dynamic Data Exchange*, Downloaded from <http://www.msdn.microsoft.com/library>.

Negnevitsky, M. (2002), *Artificial Intelligence – A Guide to Intelligent Systems*, Addison-Wesley, Harlow, England.

O’Keefe, R.M. and R. Rao (1988), Part Input into a Flexible Flow System: An Evaluation of Look-ahead Simulation and a Fuzzy Rule Base, *International Journal of Production Research*, Vol.26, pp.1739-1748.

Ouelhadj, D. and C. Hanachi (1999), A Multi-agent Net Protocol for Dynamic Scheduling in Flexible Manufacturing Systems, *ICRA ’99, IEEE International Conference on Robotics and Automation*, Detroit, Michigan, USA.

Ouelhadj, D. and C. Hanachi (2000), Multi-agent Architecture for Distributed Monitoring in Flexible Manufacturing Systems. *ICRA ’2000, IEEE International Conference on Robotics and Automation*, San Francisco, USA.

Panwalkar, S. S. and W. Iskander (1977), A Survey of Scheduling Rules, *Operations Research*, Vol.25, pp.45- 61.

Parrish, D. (1993), *Flexible Manufacturing*, Butterworth-Heinemann Ltd, Oxford, UK.

Parunak, H. V. (1996), Applications of Distributed Artificial Intelligence in Industry, *Foundation of Distributed Artificial Intelligence*, Wiley, New York.

Pierreval, H. and N. Mebarki (1997), Dynamic Selection of Dispatching Rules for Manufacturing System Scheduling, *International Journal of Production Research*, Vol.35, 6, pp.1575-1591.

Pinedo, M. (1995), *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, Englewood Cliffs, NJ.

Pritsker, A.A.B. and J.J. O’Reilly (1999), *Simulation with Visual SLAM and AweSim*, Wiley, New York.

Rachamadugu, R. and K. E. Stecke (1994), Classification and Review of FMS Scheduling Procedures, *Production Planning and Control*, Vol. 5(1), pp. 2–20.

- Reddy, K. R. B., Xie N., and V. Subramaniam (2004), Dynamic Scheduling of Flexible Manufacturing Systems, *Innovation in Manufacturing System and Technology*, <http://hdl.handle.net/1721.1/3903>
- Rembold, U., B.O. Nnaji and A. Storr (1993), *Computer Integrated Manufacturing and Engineering*, Addison-Wesley, MA.
- Sabuncuoglu, I. and D.L. Hommertzheim (1989), Expert Simulation Systems —Recent Developments and Applications in Flexible Manufacturing Systems, *Computers in Industrial Engineering*, Vol.16, pp.575-585.
- Sabuncuoglu, I. (1998), A Study of Scheduling Rules of Flexible Manufacturing Systems: A Simulation Approach, *International Journal of Production Research*, Vol.36(2), pp. 527–546.
- Schneider, D. I. (1999), *An Introduction to Programming Using Visual Basic 6.0*, Prentice Hall, New York.
- Sotskov, Y.N. (1991), The Complexity of Shop-Scheduling Problems with Two or Three Jobs, *European Journal of Operational Research*, Vol. 53, pp. 326-336.
- Shen, W., D. H. Norrie and J.P. Barthes (2001), *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*, Taylor and Francis, London, UK.
- Shen, W. (2004), Agent-Based Integration of Manufacturing Process Planning and Scheduling: A Review, *Proceedings of the 14th International Conference on Flexible Automation and Intelligent Manufacturing*, July, 2004, Ryerson University, Toronto, Canada.
- Spano, M. R., P. J. O’Grady and R. E. Young (1993), The Design of Flexible Manufacturing Systems, *Computers in Industry*, Vol.21, pp. 185–198.
- Subramaniam, V., G. K. Lee, G. S. Hong, Y. S. Wong and T. Ramesh (2000). Dynamic Selection of Dispatching Rules for Job Shop Scheduling, *Production Planning & Control*, Vol. 11, 1, pp.73-81.
- Tang, L.-L. (1993), A Study on Decision Rules of a Scheduling Model in an FMS, *Computer in Industry*, Vol.22, pp. 1-13.
- Tang, L., J. Liu, A. Rong and Z. Yang (2001), A Review of Planning and Scheduling Systems and Methods for Integrated Steel Production, *European Journal of Operational*

Research, Vol.133(1), pp. 1-20.

Thomson, N. (1995), *Simulation at Manufacturing*, Wiley, New York.

Toal, D.J.F., T. Coffey and P. Smith (1994), *Expert Systems and Simulation in Scheduling*, IMC-11, Belfast.

Wild, R. (1995), *Production and Operations Management*, 5th ed., Cassell, London, UK.

Wu, S.Y.D. and R. A. Wysk (1989), An Application of Discrete-Event Simulation To On-line Control and Scheduling in Flexible Manufacturing, *International Journal of Production Research*, Vol.23, pp.705-722.