FUZZY MULTI OBJECTIVE MACHINE RELIABILITY BASED HYBRID FLOW SHOP SCHEDULING

BY

TAHMINA FERDOUSI LIPI

DEPARTMENT OF INDUSTRIAL AND PRODUCTION ENGINEERING

BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY

DHAKA, BANGLADESH

JANUARY 2007

CERTIFICATE OF APPROVAL

The thesis titled "**Fuzzy Multi Objective Machine Reliability Based Hybrid Flow Shop Scheduling**" submitted by Tahmina Ferdousi Lipi Roll No: 040408017P, Session: April, 2004 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of **Master of Science in Industrial and Production Engineering** on January 10 , 2007.

Board of Examiners

[illegible]

Dr. M. Ahsan Akhtar Hasin Professor Department of Industrial and Production Engineering BUET, Dhaka.	Chairman (Supervisor)
[illegible] Dr. A. K. M. Masud Assistant Professor Department of Industrial and Production Engineering BUET, Dhaka.	Member
[illegible] Dr. Abdullahil Azeem Assistant Professor Department of Industrial and Production Engineering BUET, Dhaka.	Member
[illegible] Dr. Nikhil Ranjan Dhar Professor and Head Department of Industrial and Production Engineering BUET, Dhaka.	Member (Ex-officio)
[illegible] Dr. Md. Mostofa Akbar Associate Professor Department of Computer Science & Engineering BUET, Dhaka.	Member (External)

Department of Industrial and Production Engineering
Bangladesh University of Engineering and Technology

January, 2007

CANDIDATE'S DECLARATION

It is hereby declared that this thesis or any part of it has not been submitted elsewhere for the award of any degree or diploma.

Tahmina

Tahmina Ferdousi Lipi

To the Almighty

To my Parents

ACKNOWLEDGEMENT

All praises to Allah, the most benevolent and the Almighty, for guiding, providing and sustaining me through all my life and specially, success full completion of this thesis.

I would like to express my sincere respect and gratitude to my thesis supervisor, Dr. M. Ahsan Akhtar Hasin, Professor of the Department of Industrial and Production Engineering (IPE), Bangladesh University of Engineering & Technology (BUET), Dhaka, for his constant support, encouragement, invaluable inputs and thoughtful suggestions to my work. It is a blessing for me to get him as my thesis supervisor

I would like to express my intense gratefulness & thanks to Dr. Nikhil Ranjan Dhar, Professor and Head, Department of Industrial and Production Engineering, BUET for his continuous support, encouragement to perform this thesis work successfully.

I also express my sincere gratitude and thanks to Dr. A. K. M. Masud, Assistant Professor, Department of Industrial and Production Engineering, BUET, Dr. Abdullahil Azeem, Assistant Professor, Department of Industrial and Production Engineering, BUET, Dr. Md. Mostofa Akbar, Associate Professor, Department of Computer Science & Engineering, BUET, for their constant guidance and suggestions.

I would like to extend my thanks to Mr. Atique for helping computer coding to develop the software according to my algorithm.

I thank my parents and sisters and also my husband for their unconditional love, sacrifice, support and encouragement to complete the thesis successfully.

ABSTRACT

Many real-world scheduling problems are multi objective and complex in nature. That is, there exist several criteria that must be taken into consideration when evaluating the quality of the proposed solution or schedule. On the other hand consideration of machine reliability is very important during job allocation in each stage to get a realistic hybrid flow shop schedule. This research aims to develop two fuzzy inference systems (FIS) for the hybrid flow shop problem. First FIS is used to get the priority of each job considering multiple objectives of processing time, due date and cost over time. Second FIS is used to get machine reliability and availability based priority using the information of mean time to failure (MTTF) & mean time to repair (MTTR) of each individual machine of each stage. Then the total load is balanced depending on their reliability and availability i.e., maximum utilization target are determined. An algorithm is developed for grouping, sequencing & allocating the jobs to the machines at every stage in such a way that total percentage of over utilization is minimum. According to this algorithm a computing tool is developed and with a case study the entire process is explained.

TABLE OF CONTENTS

Acknowledgement		vi
Abstract		vii
Table of Contents		viii
List of Tables		xi
List of Figures		xii
List of abbreviation		xiv
CHAPTER 1	**INTRODUCTION**	1-2
1.1	General Introduction	1
1.2	Rationale of the Study	1
1.3	Objectives of the Study	2
1.4	Methodology	2
CHAPTER 2	**LITERATURE REVIEW**	3 -23
2.1	Introduction	3
2.2	Simulated Annealing	3
2.3	Tabu Search	7
2.4	Genetic Algorithm	9
2.5	Neural Network	11
2.6	Fuzzy Logic	14
2.7	Summarized Literature Review	19
CHAPTER 3	**PROBLEM IDENTIFICATION**	24-25
3.1	Introduction	24
3.2	Over all problem definition	24

CHAPTER 4	**FUZZY SCHEDULING PRELIMINARIES**	26-37
4.1	Introduction	26
4.2	Fuzzy Logic in Scheduling	26
4.2.1	Fuzzy Logic	27
4.2.2	Membership Functions	27
4.2.3	Logical Operations	30
4.2.4	If-Then Rules	32
4.2.5	Fuzzy Inference System	32
4.2.6	The Fuzzy Inference Diagram	36
4.3	Reliability and Availability based Scheduling	37
CHAPTER 5	**RELIABILITY & AVAILABILITY BASED HYBRID FLOW SHOP SCHEDULING**	38-49
5.1	General Introduction	38
5.2	Proposed Method	38
5.2.1	Fuzzy Model identification	39
5.2.2	FIS for job Priority determination	40
5.2.3	FIS for machine's Priority determination	43
5.3	Grouping & Sequencing algorithm	46
5.3.1	Grouping	47
5.3.2	Sequencing	48
CHAPTER 6	**RESULT ANALYSIS**	50-55
6.1	Introduction	50
6.2	Case Study	50
6.3	Computational Result	55

CHAPTER 7 CONCLUSION AND RECOMMENDATIONS 56-57

7.1 Conclusion 56
7.2 Recommendations 57

References 58

Appendices

Appendix A: Input Data 63
Appendix B: Result 66
Appendix C: Computer Code 68

LIST OF TABLES

Table No.	Title	Page No.
2.1	Summarized Literature Review	19
6.1	Information of jobs in stage 1, stage2, stage3	51
6.2	Priority of Job in stage 1, stage2, stage3	52
6.3	Information & priority of machines in stage 1	53
6.4	Information & priority of machines in stage 2	54
6.5	Information & priority of machines in stage 3	54
6.6	Final Schedule	54

LIST OF FIGURES

Figure No.	Title	Page No.
3	Typical Scenario of hybrid flow shop	25
4.1	Triangular & Trapezoidal membership function	28
4.2	Different Gaussian membership function	29
4.3	Sigmoidal membership function	30
4.4	Polynomial membership function	30
4.5	Explanation of logical operation	31
4.6	Numerical Explanation of logical operation	31
4.7	Graphical Explanation of logical operation	32
4.8	Fuzzy Inference Diagram	36
5.1	Reliability and Availability based Multi Objective Scheduling Frame Work	39
5.2	FIS for job priority calculation	41
5.3	Membership function of Processing Time	41
5.4	Membership function of Due Date	42
5.5	Membership function of COVERT	42
5.6	Membership function of Priority of Job	42
5.7	FIS for machine priority calculation	44
5.8	Membership function of MTTF	45

5.9	Membership function of MTTR	45
5.10	Membership function of Priority of Machine	45
5.11	Relationship Diagram of Machine's Priority, MTTR, MTTF	46
5.12	Grouping, Sequencing Algorithm	49
6.1	3 stage hybrid flow shop	50
6.2	Job Priority Calculation using FIS	52
6.3	Machine Priority Calculation using FIS	53

LIST OF ABBREVIATION

Ant Colony System (ACS)

Artificial Neural Networks (ANN)

Cost Over Time (COVERT)

Due Date (DDate)

Flexible Machining Workstation (FMW)

Flow-Shop Scheduling Problem (FSSP)

Fuzzy Logic (FL)

Fuzzy Inference System (FIS)

Genetic Algorithm (GA)

Job Shop Schedules (JSS)

Master Production Schedule (MPS)

Membership Function (MF)

Mean Time To Failure (MTTF)

Mean Time To Repair (MTTR)

Multi-Objective Simulated Annealing (MOSA)

Random Key Genetic Algorithm (RKGA)

Simulated Annealing (SA)

Total Processing Time (TPT)

Triangular Fuzzy Numbers (TFN)

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Scheduling is the process of organizing, choosing and timing resource usage to carry out all the activities necessary to produce the desired outputs of activities and resources. In flow shop there is more than one machine and each job must be processed on each of the machines – the number of operations for each job is equal with the number of machines, the jth operation of each job being processed on machine j.

1.2 Rationale of the Study

Flow shop problem concerns the sequencing of a given number of jobs through a series of machines in the exact same order on all machines with the aim to satisfy a set of constraints as much as possible, and optimize a set of objectives. The commonly studied objectives include: make span, mean flow time, tardiness etc. Among those objectives, the make span, defined as the time when the last job completes on the last machine, is the most frequently studied one. A large number of deterministic scheduling algorithms have been proposed in last decades to deal with flow shop scheduling problems with various objectives and constraints. However, it is often difficult to apply those algorithms to real-life flow shop problems. For example, in practice the processing times of jobs could be uncertain due to incomplete knowledge or uncertain environment which implies that there exist various external sources and types of uncertainty. Fuzzy sets and logic can be used to tackle uncertainties inherent in actual flow shop scheduling problems. Majority of approaches consider fuzzy processing time and/or fuzzy due dates.

1.3 Objectives of the Study

Objectives of this research are as follows:

- Incorporate uncertainty of processing time, cost over time, and earliest due date during hybrid flow shop scheduling.
- Incorporate multi objectives in Scheduling.
- Incorporate influence of machine reliability during scheduling to realize the integration between maintenance and production planning.

1.4 Methodology

- Identify multi objective scheduling parameter & construct their appropriate membership functions & fuzzy rules.
- Identify machine reliability & availability affected variables & construct their appropriate membership functions & fuzzy rules.
- Develop a fuzzy inference system to identify priority of each job & other fuzzy inference system to identify priority of each machine for each individual stage of multi processor flow shop using MATLAB Fuzzy Logic Toolbox.
- Develop an algorithm for grouping, sequencing & allocating the jobs to the machines at every stage in such a way that total percentage of over utilization is minimum.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Real-world scheduling problems are multi objective and complex combinatorial nature. That is, there exist several criteria that must be taken into consideration when evaluating the quality of the proposed solution or schedule. It is common that some or all of these criteria are conflicting, perhaps incommensurable and set by more than one decision-maker. Among these criteria there are for example: processing time, critical ratio, earliest due date, utilization of resources. Traditionally, these problems have been tackled as single objective optimization problems after combining the multiple criteria into a single scalar value. On the other hand, inherent complexity and uncertainty makes the scheduling process complicated. This imposes a pressure upon the researcher to implement an appropriate and realistic scheduling process. The interest of this research lies with hybrid flow shop scheduling which is an important field of scheduling.

2.2 Simulated Annealing

Loukil et al. (2005) described the analysis of the performance of a schedule often involves more than one aspect and therefore requires a multi-objective treatment. In this paper they presented the general context of multi-objective production scheduling, analyzed briefly the different possible approaches and define the aim of this study i.e. to design a general method able to approximate the set of all the efficient schedules for a large set of scheduling models. Then they introduced the models they want to treat one machine, parallel machines and permutation flow shops and the corresponding notations The method used called multi-objective simulated annealing is described is devoted to extensive numerical experiments and their analysis.

Cho et al. (2005) described that the effectiveness of the solution method based on simulated annealing (SA) mainly depends on how to determine the SA-related parameters. A scheme as well as parameter values for defining an annealing schedule should be appropriately determined, since various schemes and their corresponding parameter values have a significant impact on the performance of SA algorithms. That's why they, based on robust design, have proposed a new annealing parameter design method for the mixed-model sequencing problem which is known to be NP-hard. To show the effectiveness of the proposed method, extensive computation experiments were conducted.

Andreas and Nearchou (2004) articulated that advances in modern manufacturing systems such as CAD/CAM, FMS, CIM, have increased the use of intelligent techniques for solving various combinatorial and NP-hard sequencing and scheduling problems. Production process in these systems consists of workshop problems such as grouping similar parts into manufacturing cells and proceeds by passing these parts on machines in the same order. This paper presented a new hybrid simulated annealing algorithm (hybrid SAA) for solving the flow-shop scheduling problem (FSSP); an NP-hard scheduling problem with a strong engineering background. The hybrid SAA integrates the basic structure of a SAA together with features borrowed from the fields of genetic algorithms (GAs) and local search techniques. Particularly, the algorithm works from a population of candidate schedules and generates new populations of neighbor schedules by applying suitable small perturbation schemes. Further, during the annealing process, an iterated hill climbing procedure is stochastically applied on the population of schedules with the hope to improve its performance. The proposed approach is fast and easily implemented. Computational results on several public benchmarks of FSSP instances with up to 500 jobs and 20 machines showed the effectiveness and the high quality performance of the approach. In comparison to the performance of previous SA and GA methods, the performance of the proposed one was found superior.

Mansouri (2006) proposed a multi-objective simulated annealing (MOSA) solution approach to a bi-criteria sequencing problem to coordinate required set-ups between two successive stages of a supply chain in a flow shop pattern Each production batch has two distinct attributes and a set-up occurs in each stage when the corresponding attribute of the two successive batches are different. There are two objectives including: minimizing total set-ups and minimizing the maximum number of set-ups between the two stages that are both NP-hard problems.

Ying and Liao (2004) described Ant colony system (ACS) which is a novel meta-heuristic inspired by the for aging behavior of real ant. This paper is the first to apply ACS for the $n/m/P/C_{max}$ problem, an NP-hard sequencing problem which is used to find a processing order of n different jobs to be processed on m machines in the same sequence with minimizing the make span. To verify the developed ACS algorithm, computational experiments are conducted on the well-known benchmark problem set of Taillard. The ACS algorithm is compared with other mata-heuristics such as genetic algorithm, simulated annealing, and neighborhood search from the literature. Computational results demonstrate that ACS is a more effective mata-heuristic for the $n/m/P/C_{max}$ problem.

Fink and Vob (2003) described Continuous flow-shop scheduling problems which circumscribed an important class of sequencing problems in the field of production planning. The problem considered here is to find a permutation of jobs to be processed sequentially on a number of machines under the restriction that the processing of each job has to be continuous with respect to the objective of minimizing the total processing time (flow -time). This problem is NP-hard. they considered the application of different kinds of meta heuristics from a practical point of view, examining the trade-off between running time and solution quality as well as the knowledge and efforts needed to implement and calibrate the algorithms. Computational results show that high quality results can be obtained in an efficient way by applying meta- heuristics software components with neither the need to understand their inner working nor the necessity to manually tune parameters.

Gupta et al.(2002) developed and compared different local search heuristics for the two-stage flow shop problem with make span minimization as the primary criterion and the minimization of either the total flow time, total weighted flow time, or total weighted tardiness as the secondary criterion. They investigated several variants of simulated annealing, threshold accepting, tabu search, and multi-level search algorithms. The influence of the parameters of these heuristics and the starting solution are empirically analyzed. The proposed heuristic algorithms are empirically evaluated and found to be relatively more effective in finding better quality solutions than the existing algorithms.

Tian et al.(1999) focused on the generation mechanism of random permutation solutions, this paper investigated the application of the Simulated Annealing (SA) algorithm to the combinatorial optimization problems with permutation property. Six types of perturbation scheme for generating random permutation solutions are introduced. They are proved to satisfy the asymptotical convergence requirements. The results of experimental evaluations on Traveling Salesman Problem (TSP), Flow-shop Scheduling Problem (FSSP), and Quadratic Assignment Problem (QAP) reveal that the efficiencies of the perturbation schemes are different in searching a solution space. By adopting a proper perturbation scheme, the SA algorithm has shown to produce very efficient solutions to different combinatorial optimization problems with permutation property.

Ishibuchi et al.(1994) formulated a fuzzy flow shop scheduling problem where the due-date of each job is given as a fuzzy set. The membership function of the fuzzy due-date corresponds to the grade of satisfaction of a completion time. The objective function of the formulated problem is to maximize the minimum grade of satisfaction over given jobs. Several local search algorithms including multi-start descent, simulated annealing and taboo search algorithms are applied to the problem. The performance of each algorithm is compared with one another by computer simulations on randomly generated test problems. It is shown by simulation results that some algorithms do not work well for the fuzzy flow shop scheduling problem. Thus, a new approach is proposed by changing the objective function. The effectiveness of this approach is demonstrated by computer simulations.

2.3 Tabu Search

Garcia et al. (2005) dealt with the problem of selecting and scheduling the orders to be processed by a manufacturing plant for immediate delivery to the customer site. Among the constraints considered were the limited production capacity, the available number of vehicles and the time windows within which orders must be served. They first described the problem as it occurs in practice in some industrial environments, and then presented an integer programming model that maximizes the profit due to the customer orders to be processed. A tabu search-based solution procedure to solve this problem was developed and tested empirically with randomly generated problems. Comparisons with an exact procedure show that the method finds very good-quality solutions with small computation requirements.

Nowicki and Smutnicki (2006) dealt with the flow-shop scheduling problem with the makespan criterion is a certain strongly NP-hard case from the domain of OR. This problem, having simple formulation contrasting with its troublesome, complex and time-consuming solution methods, is ideal for testing the quality of advanced combinatorial optimization algorithms. Although many excellent approximate algorithms for the flow-shop problem have been provided, up till now, in the literature, some theoretical and experimental problems associated with an algorithm's activity still remain unexamined. In this paper, they provide a new view on the solution space and the search process. Relying upon it, we are proposing the new approximate algorithm, which applies some properties of neighborhoods, refers to the big valley phenomenon, uses some elements of the scatter search as well as the path re linking technique. This algorithm shows up to now unprecedented accuracy, obtainable within a short time on a PC, which has been confirmed in a wide variety of computer tests. Good properties of the algorithm remain scalable if the size of instances increases.

Janiak et al. (2005) studied the flow-shop scheduling problem with parallel machines at each stage (machine center). For each job its release and due date as well as a processing time for its each operation are given.The scheduling criterion consists of three parts: the total weighted earliness, the total weighted tardiness and the total weighted waiting time. The criterion takes into account the costs of storing semi-manufactured products in the course of production and ready-made products as well as penalties for not meeting the deadlines stated in the conditions of the contract with customer. To solve the problem, three constructive algorithms and three meta heuristics (based one Tabu Search and Simulated Annealing techniques) are developed and experimentally analyzed. All the proposed algorithms operate on the notion of so-called operation processing order, i.e. the order of operations on each machine. They showed that the problem of schedule construction on the base of a given operation processing order can be reduced to the linear programming task. They also proposed some approximation algorithm for schedule construction and show the conditions of its optimality.

Liu et al.(2005) described three types of shop scheduling problems, the flow shop, the job shop and the open shop scheduling problems, have been widely studied in the literature. However, very few articles address the group shop scheduling problem introduced in 1997, which is a general formulation that covers the three above mentioned shop scheduling problems and the mixed shop scheduling problem. In this paper, they applied tabu search to the group shop scheduling problem and evaluate the performance of the algorithm on a set of benchmark problems. The computational results show that our tabu search algorithm is typically more efficient and faster than the other methods proposed in the literature. Furthermore, the proposed tabu search method has found some new best solutions of the benchmark instances.

Grabowski and Pempera (2005) developed and compared different local search algorithms for the no-wait flow-shop problem with make span criterion (C_{max}). We present several variants of descending search and tabu search algorithms. In the algorithms the multi moves are used that consist in performing several moves simultaneously in a single iteration of algorithm and allow us to accelerate the convergence to good solutions. Besides, in the tabu search algorithms a dynamic tabu list is proposed that assists additionally to avoid trapped at a local optimum. The proposed algorithms are empirically evaluated and found to be relatively more effective in finding better quality solutions than existing algorithms. The presented ideas can be applied in any local search procedures.

Grabowski and Wodecki (2004) dealt with a classic flow-shop scheduling problem with make span criterion. Some new properties of the problem associated with the blocks have been presented and discussed. These properties allow us to propose a new very fast local search procedure based on a tabu search approach. Computational experiments (up to 500 jobs and 20 machines) are given and compared with the results yielded by the best algorithms discussed in the literature. These results show that the algorithm proposed solves the flow-shop instances with high accuracy in a very short time. The presented properties and ideas can be applied in any local search procedures.

Ben-Daya and Al-Fawzan (1998) proposed a tabu search approach for solving the permutation flow shop scheduling problem. The proposed implementation of the tabu search approach suggests simple techniques for generating neighborhoods of a given sequence and a combined scheme for intensification and diversification that has not been considered before. These new features result in an implementation that improves upon previous tabu search implementations that use mechanisms of comparable simplicity Also, better results were obtained than those produced by a simulated annealing algorithm from the literature.

2.4 Genetic Algorithm

Wang et al. (2006) explained as a typical manufacturing and scheduling problem with strong industrial background, flow shop scheduling with limited buffers has gained wide attention both in academic and engineering fields. With the objective to minimize the total completion time (or make span), such an issue is very hard to solve effectively due to the NP-hardness and the constraint on the intermediate buffer. In this paper, an effective hybrid genetic algorithm (HGA) is proposed for permutation flow shop scheduling with limited buffers. In the HGA, not only multiple genetic operators based on evolutionary mechanism are used simultaneously in hybrid sense, but also a neighborhood structure based on graph model is employed to enhance the local search, so that the exploration and exploitation abilities can be well balanced. Moreover, a decision probability is used to control the utilization of genetic mutation operation and local search based on problem-specific information so as to prevent the premature convergence and concentrate computing effort on promising neighbor solutions. Simulation results and comparisons based on benchmarks demonstrate the effectiveness of the HGA. Meanwhile, the effects of buffer size and decision probability on optimization performances are discussed

According to Chang et al. (2006) the sub-population genetic algorithm (SPGA) is effective in solving multi objective scheduling problems. Based on the pioneer efforts, this research proposes a mining gene structure technique integrated with the SPGA. The mining problem of elite chromosomes is formulated as a linear assignment problem and a greedy heuristic using threshold to eliminate redundant information. As a result, artificial chromosomes are created according to this gene mining procedure and these artificial chromosomes will be reintroduced into the evolution process to improve the efficiency and solution quality of the procedure. In addition, to further increase the quality of the artificial chromosome, a dynamic threshold procedure is developed and the flow shop scheduling problems are applied as a benchmark problem for testing the developed algorithm. Extensive tests in the flow-shop scheduling problem show that the proposed approach can improve the performance of SPGA significantly.

Zandieh et al. (2006) explained that much of the research on operations scheduling problems has either ignored setup times or assumed that setup times on each machine are independent of the job sequence. This paper deals with the hybrid flow shop scheduling problems in which there are sequence dependent setup times, commonly known as the SDST hybrid flow shops. This type of production system is found in industries such as chemical, textile, metallurgical, printed circuit board, and automobile manufacture. With the increase in manufacturing complexity, conventional scheduling techniques for generating a reasonable manufacturing schedule have become ineffective. An immune algorithm (IA) can be used to tackle complex problems and produce a reasonable manufacturing schedule within an acceptable time. This paper described an immune algorithm approach to the scheduling of a SDST hybrid flow shop. An overview of the hybrid flow shops and the basic notions of an IA are first presented. Subsequently, the details of an IA approach are described and implemented. The results obtained are compared with those computed by Random Key Genetic Algorithm (RKGA) presented previously. From the results, it was established that IA outperformed RKGA.

Chan,W and Hu,H.(2001) described that there are two alternatives for production organization in pre cast factories, namely the comprehensive method and the specialized method. Production scheduling under the specialized alternative has been found to be a difficult optimization problem if heterogeneous elements are involved. A flow shop sequencing model is developed for this kind of production scheduling ,that considers the constraints encountered in actual practice. The model is optimized using a genetic algorithm (GA) approach. The results are compared with those obtained using classical heuristic rules in two examples that involve the objective of minimizing the make span or the total tardiness penalty. The comparison shows that the GA can obtain good schedules for the model, giving a family of solutions that are at least as good as those produced by the heuristic rules.

2.5 Neural Network

Anilkumar and Tanprasert (2006) developed the design and implementation of a neural network-based job priority assigner system for a job-scheduling environment. This paper reports on research that established that a back propagation neural network-based priority procedure would recognize jobs from a job queue by estimating each job's priority.

Solimanpur et al. (2004) dealt with the sequencing of different jobs of flow shop scheduling that visit a set of machines in the same order. A neural networks-based tabu search method, namely EXTS, is proposed for the flow shop scheduling. Unlike the other tabu search-based methods, the proposed approach helps diminishing the tabu effect in an exponential way rather than most commonly used way of diminishing it in a sudden manner. On the basis of the conducted tests, some rules are evolved to set the values for different parameters. The effectiveness of the proposed method is tested with 23 problems selected from literature. The computational results indicate that the proposed approach is effective in terms of reduced make span for the attempted problems.

Lee and Shaw (2000) considered the classic problem of sequencing a set of jobs that arrive in different combinations over time in a manufacturing flow-shop. They focus on the development of a two-level neural network that incrementally learns sequencing knowledge. Based on the knowledge gained from learning using a set of training exemplars, the neural network makes real time sequencing decisions for a set of jobs that arrive in different job combinations. In addition to explain the details regarding the workings of the neural network, they also evaluate its performance for flow-shop sequencing problems. The practical benefit of the neural-net approach is that the neural network incrementally learns the sequencing knowledge and can apply the knowledge for sequencing a set of jobs on a real time basis. They showed that the neural network can be used to develop hybrid genetic algorithms. The experimental results demonstrate that (1) the neural-net approach produces consistently superior solution quality (i.e., make span) with significantly less computational time than the traditional heuristic approaches; (2) when compared to genetic algorithms the neural-net approach's performances are within 3.4% of those of genetic algorithms, but using only less than 0.2% of the computational time needed by genetic algorithms; and (3) the neural-net approach further improves solution quality and computational time by combining it with genetic algorithms. These results support the efficacy of using the neural-net approach for real time flow-shop sequencing

Sabuncuoglu and Gurgun (1996), have successfully applied artificial neural networks (ANNs) to solve a variety of problems. They proposed a new neural network approach to solve the single machine mean tardiness scheduling problem and the minimum make span job shop scheduling problem. The proposed network combines the characteristics of neural networks and algorithmic approaches. The performance of the network is compared with the existing scheduling algorithms under various experimental conditions.

Watanabe et al. (1993) Job-shop scheduling cannot easily be analytically accomplished, so, it is done by computer simulation using heuristic priority rules. The SLACK rule for calculating the margins of jobs to their due-dates is effective in meeting the due-dates. However, the calculated margins are not precise because the actual margin is shortened due to conflicts with other jobs. The authors propose a method for estimating the margins by using a neural network. It is found that the method is effective for improving the average lateness to due-dates but not the maximum lateness. This paper proposes a method for adding a second neural network for judging the reliability of the estimated margins composed to the first one and for switching to the margins calculated by the SLACK rule when the reliability is low. The proposed method is verified by scheduling simulations to be effective in decreasing the maximum lateness to due-dates as much as the average lateness.

Dagli and Sittisathanchai (1993) described a hybrid approach between two new techniques, Genetic Algorithms and Artificial Neural Networks, for generating Job Shop Schedules (JSS) in a discrete manufacturing environment based on non-linear multi-criteria objective function. Genetic Algorithm (GA) is used as a search technique for an optimal schedule via a uniform randomly generated population of gene strings which represent alternative feasible schedules. GA propagates this specific gene population through a number of cycles or generations by implementing natural genetic mechanism (i.e. reproduction operator and crossover operator). It is important to design an appropriate format of genes for JSS problems. Specifically, gene strings should have a structure that imposes the most common restrictive constraint; a precedence constraint. The other is an Artificial Neural Network, which uses its highly connected-neuron network to perform as a multi-criteria evaluator. The basic idea is a neural network evaluator which maps a complex set of scheduling criteria (i.e. flow time, lateness) to evaluate values provided by experienced experts. Once, the network is fully trained, it will be used as an evaluator to access the fitness or performance of those stimulated gene strings.

The proposed approach was prototyped and implemented on JSS problems based on different model sizes; namely small, medium, and large. The results are compared to the Shortest Processing Time heuristic used extensively in industry.

2.6 Fuzzy Logic

Fuzzy logic, which was introduced by Zadeh (1965), has been applied to various industrial problems including production systems [Zadeh 1965]. Recently, there has been significant attention given to modeling scheduling problems within a fuzzy framework. The advantage of the fuzzy logic system approach is that it incorporates both numerical results from a previous solution or simulation and the scheduling expertise from experiences or observation, and it is easy to implement.

Grabot and Geneste (1994) proposed a way to use fuzzy logic in order to build aggregated rules allowing obtaining a compromise between the satisfactions of several criteria. When the criteria of performance change with the evolution of the production environment, these aggregated rules can be parameterized in order to modify the respective influence of the elementary rules they are composed of.

Hierarchical planning, scheduling and control in flexible manufacturing systems (FMS) provide a systematic way to effectively allocate resources along different time horizons. Keung et al. (2003) described the design and development of an intelligent hierarchical control model based on a proposed tool management method. The control model consists of four levels: the process plan selection, the master scheduling, the job sequencing and the control level. The model is developed to optimize the machine utilization and balance tool magazine capacity of a flexible machining workstation (FMW) in a tool-sharing environment. Problems are identified and modeled in the level of process plan selection, master scheduling, and job sequencing. A genetic-based algorithm was developed to solve the problem domains throughout the hierarchical planning and scheduling model. Fuzzy logic technique could also be incorporated into the master production schedule (MPS) level to allow for a more realistic result in the presence of uncertainty and impreciseness in order to fit the realistic nature of actual industrial environments.

When scheduling jobs in a flow shop no single dispatching rule works best for all performance criteria. Hence, it becomes paramount to assess which rule is more balanced in terms of different conflicting achievements. An alternative to the simulation based comparison of different dispatching rules can be represented by a linguistic based decision making method. Petroni and Rizzi (2002) presented a fuzzy logic based tool intended to rank flow shop dispatching rules under multiple performance criteria. This tool is detailed with reference to a significant industrial case of a major company operating in the boilermaker industry. The results show that the approach is robust and effective in providing a practical guidance to scheduling practitioners in choosing priorities dispatching rules when there are multiple objectives. Finally, the benefits and the shortcomings of the approach are discussed.

Allet (2003) dealt with a particular scheduling problem inspired by a practical case coming from a Belgian Pharmaceutical Company. This problem has some particularities with respect to the 'classical job shop problem'. The principal one is the existence of a delay between the end of an operation and the start of the next operation of the same job. This delay is not fixed but belongs generally to an interval range of possible values. A resolution method (the horizon method) has been proposed in the deterministic case (fixed data and strict constraints). In this method, the horizon time of each machine is discretised by unit times and at each unit time a binary value is associated; all the operations applied on the machines horizons are logic operations (OR, AND, right or left shifting and so on). Unfortunately, this method does not take into account several aspects of the problem: the existence of a preference relation on the possible values of the delay between successive operations and of flexible due dates.

In this work, he modeled such flexible parameters using the fuzzy logic. They propose a new method generalizing the horizon method. This new method try to find (for the realization of the jobs) a compromise between the choice of good values for the delay between couples of successive operations of the same job and good values for the completion time of the jobs.

Most scheduling problems are complex combinatorial problems and very difficult to solve. That is why, lots of methods focus on the optimization according to a single criterion (make span, workloads of machines, waiting times, etc.). The combining of several criteria induces additional complexity and new problems. Kacem et al. (2002) proposed a Pareto approach based on the hybridization of fuzzy logic (FL) and evolutionary algorithms (EAs) to solve the flexible job-shop scheduling problem (FJSP). This hybrid approach exploits the knowledge representation capabilities of FL and the adaptive capabilities of EAs. The integration of these two methodologies for the multi-objective optimization has become an increasing interest. The objective considered is to minimize the overall completion time (make span), the total workload of machines and the workload of the most loaded machine. Many examples are presented to illustrate some theoretical considerations and to show the efficiency of the suggested methodology.

Yun, Y. S. (2002) proposed a new genetic algorithm (GA) with fuzzy logic controller (FLC) for dealing with preemptive job-shop scheduling problems (p-JSP) and non-preemptive job-shop scheduling problems (np-JSP). The proposed algorithm considers the preemptive cases of activities among jobs under single machine scheduling problems. For these preemptive cases, he first use constraint programming and secondly develop a new gene representation method, a new crossover and mutation operators in the proposed algorithm. How ever, the proposed algorithm, as conventional GA, also has a weakness that takes so much time for the fine-tuning of genetic parameters. FLC can be used for regulating these parameters. FLC is used to adaptively regulate the crossover ratio and the mutation ratio of the proposed algorithm. To prove the performance of the proposed FLC, we divide the proposed algorithm into two cases: the GA with the

FLC (pro-fGA) and the GA without the FLC (pro-GA).In numerical examples, he applied the proposed algorithms to several job-shop scheduling problems and the results applied are analyzed and compared. Various experiments show that the results of pro-fGA outperform those of pro-GA.

Flexible flow shops can be thought of as generalizations of simple flow shops. In the past, the processing time for each job was usually assumed to be known exactly, but in many real-world applications, processing times may vary dynamically due to human factors or operating faults. Felix and Abhary (1997) generalized it to continuous fuzzy domains. They use triangular membership functions for flexible flow shops with two machine centers to examine processing-time uncertainties and to make scheduling more suitable for real applications. They first used triangular fuzzy LPT algorithm to allocate jobs, and then use triangular fuzzy Johnson algorithm to deal with sequencing the tasks. The proposed method thus provides a more flexible way of scheduling jobs than conventional scheduling methods.

In job sequencing for a flow shop, processing times are frequently not known exactly and only estimated intervals are given. Fuzzy numbers are ideally suited to represent these intervals. McCahonE and Lee (1992) described the process of the Campbell, Dudek and Smith (CDS) job sequencing algorithm is modified to accept trapezoidal fuzzy processing times. Deterministic sequences result, but the sequence performance measurements of makespan and job mean flow time are fuzzy, having been calculated using fuzzy arithmetic. The use of possibility theory and the fuzzy integral enables the schedular to meaningfully interpret these fuzzy results. Deterministic approximations to this fuzzy approach are also investigated.

Tsujimura et al.(1993) showed that fuzzy set theory can be useful in modeling and solving flow shop scheduling problems with uncertain processing times and illustrates a methodology for solving job sequencing problem which the opinions of experts greatly disagree in each processing time. Triangular fuzzy numbers (TFNs) are used to represent the processing times of experts. And the comparison methods based on the dominance property is sued to determine the ranking of the fuzzy numbers. By the dominance criteria, for each job, a major TFN and a minor TFN are selected and a pessimistic sequence with major TFNs and an optimistic sequence with minor TFNs are computer. Branch and bound algorithm for make span in three-machine flow shop is utilized to illustrate the proposed methodology.

Hong and Wang (2000) articulated that flexible flow shops can be thought of as generalizations of simple flow shops. In the past, the processing time for each job was usually assumed to be known exactly, but in many real-world applications, processing times may vary dynamically due to human factors or operating faults. They have demonstrated how discrete fuzzy concepts could easily be used in the Sriskandarajah and Sethi's algorithm for managing uncertain flexible- flow-shop scheduling In this paper, they generalize it to continuous fuzzy domains. They use triangular membership functions for flexible flow shops with two machine centers to examine processing-time uncertainties and to make scheduling more suitable for real applications. They first use triangular fuzzy LPT algorithm to allocate jobs, and then use triangular fuzzy Johnson algorithm to deal with sequencing the tasks. The proposed method thus provides a more flexible way of scheduling jobs than conventional scheduling methods.

Cheng et al. (2001) considered the three-machine permutation flow-shop scheduling problem with release times where the objective is to minimize the maximum completion time. A special solvable case is found for the $F2/r_j/C_{max}$ problem, which sharpens the boundary between easy and hard cases and can be used to compute a tight lower bound for our problem. Two dominance rules are generalized and applied to generating initial schedules, directing the search strategy and decomposing the problem into smaller ones. The branch and bound algorithm proposed here combines an adaptive branching rule with a fuzzy search strategy to narrow the search tree and lead the search to an optimal solution as early as possible. Our extensive numerical experiments have led to a classification of 'easy' vs. 'hard' problems, dependent only on the relative size of the release times. The algorithm has quickly solved approximately 90% of the hardest test problem instances with up to 200 jobs and 100% of the large problems classified as easy.

2.7 Summary of Literature Review

The above literatures are summarized in a tabular form according to the following table as per their research field:

Table 2.1: Summarized Literature Review

Field	Reference	Description of the work
Simulated Annealing Based Scheduling	Loukil et al. (2005)	Introduced multi-objective simulated annealing based scheduling.
	Cho et al. (2005)	Described that the effectiveness of the solution method based on simulated annealing (SA) & determined the SA-related parameters.
	Andreas and Nearchou (2004)	Presented a new hybrid simulated annealing algorithm (hybrid SAA) for solving the flow-shop scheduling problem (FSSP).
	Mansouri (2006)	Proposed a multi-objective simulated annealing (MOSA) solution approach including: minimizing total set-ups and minimizing the maximum number of set-ups between the two stages.
	Ying and Liao (2004)	Described Ant colony system (ACS) which is a novel meta-heuristic to solve NP-hard sequencing problem.
	Fink and Vob (2003)	Described the process of how continuous flow-shop scheduling finds a permutation of jobs to be processed sequentially on a number of machines for minimizing the total processing time.

	Gupta et al.(2002)	Developed and compared different local search heuristics for the two-stage flow shop problem with make span minimization.
	Tian et al.(1999)	Described that by adopting a proper perturbation scheme, the SA algorithm has shown to produce very efficient solutions to different combinatorial optimization problems with permutation property.
Tabu Search Based Scheduling	Garcia et al. (2005)	Developed a tabu search-based solution procedure to solve this problem and tested empirically with randomly generated problems.
	Nowicki and Smutnicki (2006)	Flow-shop scheduling problem with the make span criterion is solved using tabu search algorithm.
	Janiak et al. (2005)	Three constructive algorithms and three meta heuristics (based one Tabu Search and Simulated Annealing techniques) are developed to solve scheduling problem and experimentally analyzed.
	Liu et al.(2005)	They showed that tabu search algorithm is typically more efficient and faster than the other methods to solve scheduling problem
	Grabowski and Pempera (2005)	Developed and compared different local search algorithms for the no-wait flow-shop problem with make span criterion (C_{max}).
	Grabowski and Wodecki (2004)	They dealt with a classic flow-shop scheduling problem with make span criterion and proposed a way to solve with tabu search algorithm.
	Ben-Daya and Al-Fawzan (1998)	Proposed a tabu search approach for solving the permutation flow shop scheduling problem.

Genetic Algorithm Based Scheduling	Wang et al. (2006)	Proposed an effective hybrid genetic algorithm (HGA) for permutation flow shop scheduling with limited buffers.
	Chang et al. (2006)	Described sub-population genetic algorithm (SPGA) is effective in solving multi objective scheduling problems.
	Zandieh et al. (2006)	Dealt with the hybrid flow shop scheduling problems in which there are sequence dependent setup times, commonly known as the SDST hybrid flow shops.
	Chan,W. and Hu,H.(2001)	described the GA can obtain good schedules for their proposed model, giving a family of solutions that are at least as good as those produced by the heuristic rules.
Neural Network Based Scheduling	Anilkumar and Tanprasert (2006)	Formulated a back propagation neural network-based priority procedure to solve scheduling problem.
	Solimanpur et al. (2004)	A neural networks-based tabu search method, namely EXTS, is proposed for the flow shop scheduling.
	Lee and Shaw (2000)	They proposed the neural network based hybrid genetic algorithms to solve scheduling problem.
	Sabuncuoglu and Gurgun (1996),	Proposed a new neural network approach to solve the single machine mean tardiness scheduling problem and the minimum make span job shop scheduling problem.
	Watanabe et al. (1993)	The authors propose a method for estimating the margins by using a neural network. It is found that the method is effective for improving the average lateness to due-dates but not the maximum lateness.
	Dagli and	Described a hybrid approach between two new

	Sittisathanchai (1993)	techniques, Genetic Algorithms and Artificial Neural Networks, for generating Job Shop Schedules (JSS).
Fuzzy Logic Based Scheduling	Zadeh (1965)	Articulated that the advantage of the fuzzy logic system approach is that it incorporates both numerical results from a previous solution or simulation and the scheduling expertise from experiences or observation, and it is easy to implement.
	Grabot and Geneste (1994)	Proposed a way to use fuzzy logic in order to build aggregated rules to solve the multi objective scheduling problem.
	Keung et al. (2003)	Described the design and development of an intelligent hierarchical control model to solve the scheduling problem.
	Petroni and Rizzi (2002)	Presented a fuzzy logic based tool intended to rank flow shop dispatching rules under multiple performance criteria.
	Allet (2003)	Proposed a resolution method (the *horizon method*) for the deterministic case (fixed data and strict constraints). In this method, the horizon time of each machine is discretised by unit times and at each unit time a binary value is associated.
	Kacem et al. (2002)	Proposed a Pareto approach based on the hybridization of fuzzy logic (FL) and evolutionary algorithms (EAs) to solve the flexible job-shop scheduling problem (FJSP).
	Yun, Y. S. (2002)	Described a new genetic algorithm (GA) with fuzzy logic controller (FLC) for dealing with preemptive job-shop scheduling problems.

	Tsujimura et al.(1993)	Showed that fuzzy set theory can be useful in modeling and solving flow shop scheduling problems with uncertain processing times & Triangular fuzzy numbers (TFNs) are used to represent the processing times of experts.
	Cheng et al. (2001)	Considered the three-machine permutation flow-shop scheduling problem with release times where the objective is to minimize the maximum completion time.

In this research the literature is reviewed on local search methods for flow shop and job shop scheduling and adapts these to the hybrid flow shop scheduling. The above proposed methods did not consider the fuzzy multi objective parallel flow shop problem and machine reliability based utilization during scheduling. If machine reliability is not considered during the scheduling process then the schedule may not be a realistic schedule due to the uncertain shutdown of the machine.

To address the uncertainty & satisfy the multi objectives, fuzzy rule-based system is developed. This system provides the priority of each job by taking the information of processing time, cost over time, earliest due date etc. as an appropriate fuzzy membership function. On the other hand other fuzzy inference system (FIS) provides the machine priority based on reliability and availability of each stage based on the information about mean time to failure (MTTF) & mean time to repair (MTTR) of each individual machine. A maximum utilization target of each machine is calculated using the reliability of each machine found from the second FIS. Based on the priority calculation for each job and machine for each stage, an algorithm is developed to grouping, sequencing & allocating the jobs to the machines at every stage in such a way that total percentage of over utilization is minimized.

CHAPTER 3

PROBLEM IDENTIFICATION

3.1 Introduction

Scheduling problems consist of the allocation of tasks to scare resources over time. The hybrid flow shop is an extension of the flow shop; at every stage a number of identical machines are available that can operate in parallel. The difference with the flow shop is that at each stage it has to assign the jobs to one of the several available machines. In essence, the hybrid flow shop problem consists of two sub problems: assigning jobs to machines, i.e., a grouping problem, and sequencing operations on the machine, i.e., a scheduling problem. Both problems should be solved in such a way that the given objectives are satisfied and optimized.

3.2 Over all problem definition

In hybrid flow shop there may be a numbers of stages of processor and each stage has more then one identical machine. The machines are identical in a sense that, for a given stage the jobs need the same time to be processed on each machine. But the reliability and availability characteristics i.e., mean time to failure (MTTF) and mean time to repair (MTTR) is different for each machine in a single stage. So though these machines in an individual stage are identical in a sense of processing time, there different values of mean time to failure (MTTF) and mean time to repair (MTTR) makes them having different priority.

Each job has to be processed on every stage. The priority of the jobs could be appraised by the values of there processing time, due date (Due Date) and cost over time (COVERT) etc. In each stage the identical machine's priority is determined based on the information of mean time to failure (MTTF) and mean time to repair (MTTR).

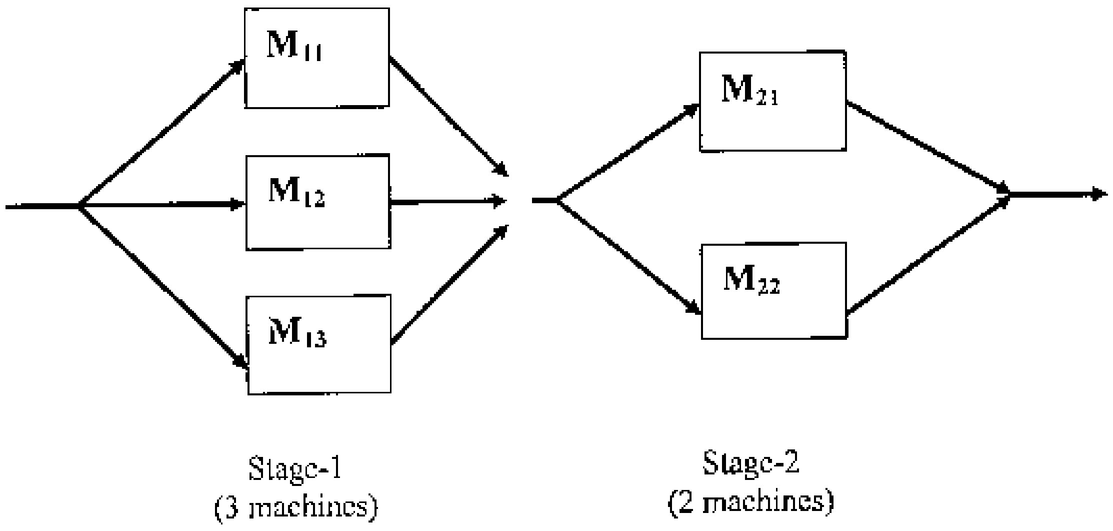


Figure 3: Typical Scenario of hybrid flow shop

So this problem involves determining the mechanism of priority determination of the jobs and machines in an individual stage and also determine the way for grouping, sequencing & allocating the jobs to the machines at every stage in such a way that total percentage of over utilization will be minimum and consequently the top priority jobs will be processed on the top priority machines.

CHAPTER 4

FUZZY SCHEDULING PRELIMINARIES

4.1 Introduction

The flow shop scheduling problems exist naturally in many real-life situations, since there are many practical as well as important applications for a job to be processed in series with more than one-stage in industry. An ordinary flow shop is a multi-stage production process with the property that all products have to pass through a number of stages in the same order. The multiprocessor flow shop is an extension of the flow shop: at every stage a number of identical machines are available that can operate in parallel.

4.2 Fuzzy Logic in Scheduling

Fuzzy set theory has been used to model systems that are hard to define precisely. As a methodology, fuzzy set theory incorporates imprecision and subjectivity into the model formulation and solution process. Lotfi Ashker Zadeh, a logician of human cognition, introduced some formal settings for dealing with uncertainty]Zadeh 2005]. The main theme of his philosophy is that the human cognition has two main facets: Partiality and Granularity. Partiality means tolerance to partial truth—truth value of a proposition or an event is not only true and false, but also partially true or false. Granularity means formation of granules (words or phrases) assigned un-sharply to a set of values or attributes. It has been applied to various industrial problems including production systems [Zadeh 1965]. Recently, there has been significant attention given to modeling scheduling problems within a fuzzy framework. The advantage of the fuzzy logic system approach is that it incorporates both numerical results from a previous solution or simulation and the scheduling expertise from experiences or observation, and it is easy to implement.

4.2.1 Fuzzy Logic

Fuzzy Logic is a problem-solving control system methodology that lends itself to implementation in systems ranging from simple, small, embedded micro-controllers to large, networked, multi-channel PC or workstation-based data acquisition and control systems. It can be implemented in hardware, software, or a combination of both FL provides a simple way to arrive at a definite conclusion based upon vague, ambiguous, imprecise, noisy, or missing input information.

Fuzzy logic has two different meanings. In a narrow sense, fuzzy logic is a logical system, which is an extension of multi-valued logic. But in a wider sense, which is in predominant use today, fuzzy logic (FL) is almost synonymous with the theory of fuzzy sets, a theory which relates to classes of objects with un sharp boundaries in which membership is a matter of degree.

4.2.2 Membership Functions

A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the universe of discourse. It is a graphical representation of the magnitude of participation of each input. It associates a weighting with each of the inputs that are processed. Membership functions can be like triangular, trapezoidal, Gaussian membership functions, sigmoidal membership function etc. Use of membership function type depends on the problem situation.

The only condition a membership function must really satisfy is that it must vary between 0 and 1. The function itself can be an arbitrary curve whose shape can be defined as a function that suits us from the point of view of simplicity, convenience, speed, and efficiency.

A classical set might be expressed as

A = {x | x > 6}

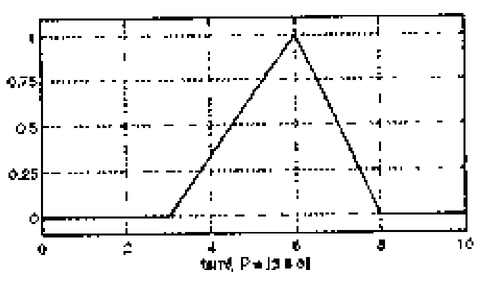
A fuzzy set is an extension of a classical set. If X is the universe of discourse and its elements are denoted by x, then a fuzzy set A in X is defined as a set of ordered pairs.

A = {x, μA(x) | x X}

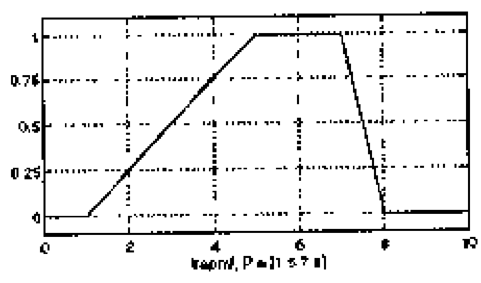
μA(x) is called the membership function (or MF) of x in A. The membership function maps each element of X to a membership value between 0 and 1.

The Fuzzy Logic Toolbox includes 11 built-in membership function types. These 11 functions are, in turn, built from several basic functions: piecewise linear functions, the Gaussian distribution function, the sigmoid curve, and quadratic and cubic polynomial curves. By convention, all membership functions have the letters mf at the end of their names.

The simplest membership functions are formed using straight lines. Of these, the simplest is the triangular membership function, and it has the function name trimf. it is nothing more than a collection of three points forming a triangle. The trapezoidal membership function, trapmf, has a flat top and really is just a truncated triangle curve. These straight line membership functions have the advantage of simplicity.



trimf



trapmf

Figure 4.1 : Triangular & trapezoidal membership function

Two membership functions are built on the Gaussian distribution curve: a simple Gaussian curve and a two-sided composite of two different Gaussian curves. The two functions are gaussmf and gauss2mf

The generalized bell membership function is specified by three parameters and has the function name gbellmf. The bell membership function has one more parameter than the Gaussian membership function, so it can approach a non-fuzzy set if the free parameter is tuned. Because of their smoothness and concise notation, Gaussian and bell membership functions are popular methods for specifying fuzzy sets. Both of these curves have the advantage of being smooth and nonzero at all points.

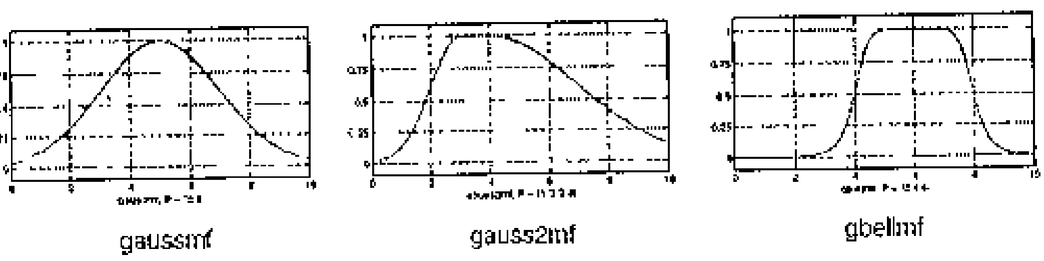


Figure 4.2: Different Gaussian membership function

Although the Gaussian membership functions and bell membership functions achieve smoothness, they are unable to specify asymmetric membership functions, which are important in certain applications. Next the sigmoidal membership function, which is either open left or right. Asymmetric and closed (i.e. not open to the left or right) membership functions can be synthesized using two sigmoidal functions, so in addition to the basic sigmf, we also have the difference between two sigmoidal functions, dsigmf, and the product of two sigmoidal functions psigmf.

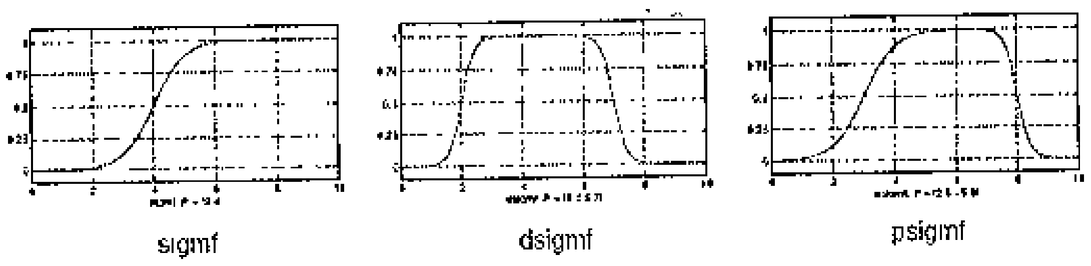


Figure 4.3: Sigmoidal membership function

Polynomial based curves account for several of the membership functions in the toolbox. Three related membership functions are the *Z*, *S*, and *Pi curves*, all named because of their shape. The function zmf is the asymmetrical polynomial curve open to the left, smf is the mirror-image function that opens to the right, and pimf is zero on both extremes with a rise in the middle.

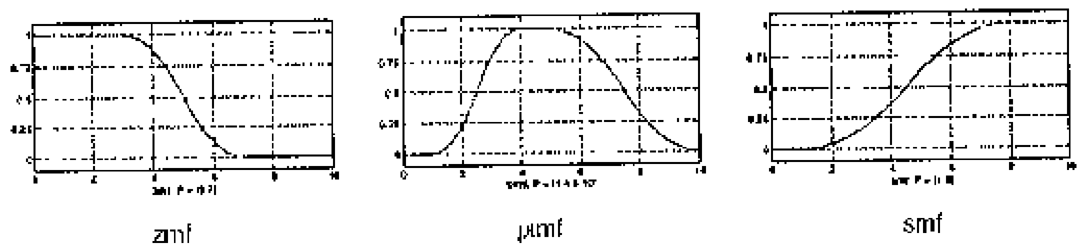


Figure 4.4: Polynomial membership function

4.2.3 Logical Operations

The most important thing to realize about fuzzy logical reasoning is the fact that it is a superset of standard Boolean logic. In other words, if we keep the fuzzy values at their extremes of 1 (completely true), and 0 (completely false), standard logical operations will hold. As an example, consider the standard truth tables below:

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	not A
0	1
1	0

NOT

Figure 4.5: Explanation of logical operation

In fuzzy logic the truth of any statement is a matter of degree. The input values can be real numbers between 0 and 1. Which function will preserve the results of the AND truth table (for example) and also extend to all real numbers between 0 and 1 is the min operation. That is, resolve the statement A AND B, where A and B are limited to the range (0,1), by using the function min(A,B). Using the same reasoning, the OR operation can be replaced with the max function, so that A OR B becomes equivalent to max(A,B).Following table described how the truth table above is completely unchanged by this substitution.

A	B	min(A,B)
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	max(A,B)
0	0	0
0	1	1
1	0	1
1	1	1

OR

A	1 - A
0	1
1	0

NOT

Figure 4.6: Numerical Explanation of logical operation

Moreover, since there is a function behind the truth table rather than just the truth table itself, now it is considered values other than 1 and 0. The next figure uses a graph to show the same information. It is converted the truth table to a plot of two fuzzy sets applied together to create one fuzzy set. The upper part of the figure displays plots corresponding to the two-valued truth tables above, while the lower part of the figure displays how the operations work over a continuously varying range of truth values *A* and *B* according to the fuzzy operations.

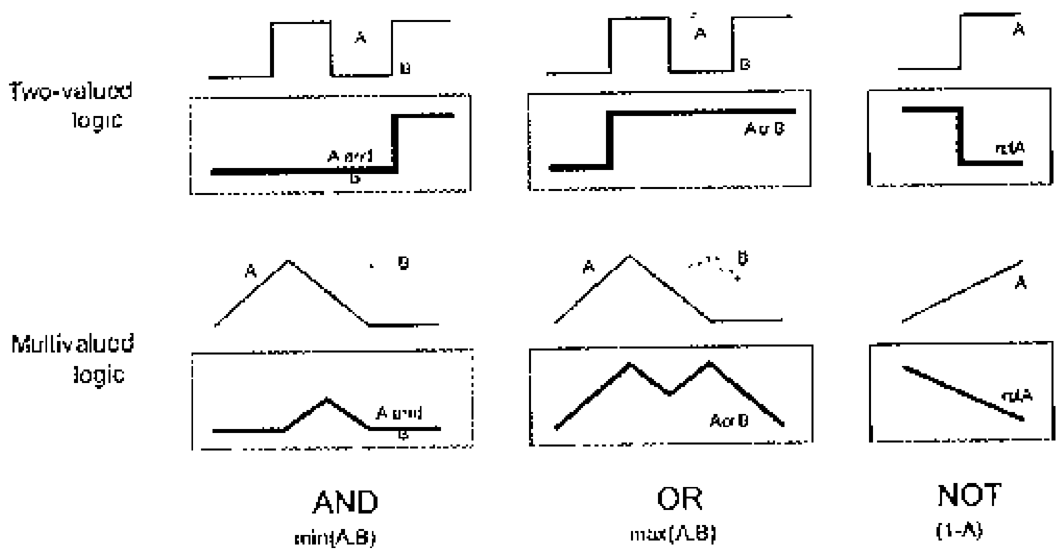


Figure 4.7: Graphical Explanation of logical operation

4.2.4 If-Then Rules

Fuzzy sets and fuzzy operators are the subjects and verbs of fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic. A single fuzzy if-then rule assumes the form if x is A then y is B where A and B are linguistic values defined by fuzzy sets on the ranges (universes of discourse) X and Y, respectively. The if-part of the rule "x is A" is called the antecedent or premise, while the then-part of the rule "y is B" is called the consequent or conclusion.

4.2.5 Fuzzy Inference System

Fuzzy Inference Systems is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. The process of fuzzy inference involves all of the pieces that are described in the previous sections:

membership functions, fuzzy logic operators, and if-then rules. There are two types of fuzzy inference systems that can be implemented in the Fuzzy Logic Toolbox: Mamdani-type and Sugeno-type. These two types of inference systems vary somewhat in the way outputs are determined. Fuzzy inference systems have been successfully applied in fields such as automatic control, data classification, decision analysis, expert systems, and computer vision. Because of its multidisciplinary nature, fuzzy inference systems are associated with a number of names, such as fuzzy-rule-based systems, fuzzy expert systems, fuzzy modeling, fuzzy associative memory, fuzzy logic controllers, and simply (and ambiguously) fuzzy systems. Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani's effort was based on Lotfi Zadeh's 1973 paper on fuzzy algorithms for complex systems and decision processes . Mamdani-type inference, as it has been defined it for the Fuzzy Logic Toolbox, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. It is possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. This is sometimes known as a singleton output membership function, and it can be thought of as a pre-defuzzified fuzzy set. It enhances the efficiency of the defuzzification process because it greatly simplifies the computation required by the more general Mamdani method, which finds the centroid of a two-dimensional function. Rather than integrating across the two-dimensional function to find the centroid, weighted average method of a few data points is used. Sugeno-type systems support this type of model. In general, Sugeno-type systems can be used to model any inference system in which the output membership functions are either linear or constant.

In the Fuzzy Logic Toolbox, there are five parts of the fuzzy inference process: fuzzification of the input variables, application of the fuzzy operator (AND or OR) in the antecedent, implication from the antecedent to the consequent, aggregation of the consequents across the rules, and defuzzification. These sometimes cryptic and odd names have very specific meaning that we'll define carefully as we step through each of them in more detail below.

4.2.5.1 Step 1. Fuzzify Inputs

The first step is to take the inputs and determine the degree to which they belong to each of the appropriate fuzzy sets via membership functions. In the Fuzzy Logic Toolbox, the input is always a crisp numerical value limited to the universe of discourse of the input variable and the output is a fuzzy degree of membership in the qualifying linguistic set (always the interval between 0 and 1). Fuzzification of the input amounts to either a table lookup or a function evaluation. Each input is fuzzified over all the qualifying membership functions required by the rules.

4.2.5.2 Step 2. Apply Fuzzy Operator

Once the inputs have been fuzzified, the degree to which each part of the antecedent has been satisfied for each rule is kown. If the antecedent of a given rule has more than one part, the fuzzy operator is applied to obtain one number that represents the result of the antecedent for that rule. This number will then be applied to the output function. The input to the fuzzy operator is two or more membership values from fuzzified input variables. The output is a single truth value.As is described in the section on fuzzy logical operations, any number of well-defined methods can fill in for the AND operation or the OR operation. In the Fuzzy Logic Toolbox, two built-in AND methods are supported: min (minimum) and prod (product). Two built-in OR methods are also supported: max (maximum), and the probabilistic OR method probor. The probabilistic OR method (also known as the algebraic sum) is calculated according to the equation

probor(a,b) = a + b – ab

In addition to these built-in methods, It is possible to create own methods for AND and OR by writing any function and setting that by the choice.

4.2.5.3 Step 3. Apply Implication Method

Before applying the implication method, we must take care of the rule's weight. Every rule has a weight (a number between 0 and 1), which is applied to the number given by the antecedent. Generally this weight is 1 (as it is for this example) and so it has no effect at all on the implication process. From time to time you may want to weight one rule relative to the others by changing its weight value to something other than 1. Once proper weighting has been assigned to each rule, the implication method is implemented. A consequent is a fuzzy set represented by a membership function, which weights appropriately the linguistic characteristics that are attributed to it. The consequent is reshaped using a function associated with the antecedent (a single number). The input for the implication process is a single number given by the antecedent, and the output is a fuzzy set. Implication is implemented for each rule. Two built-in methods are supported, and they are the same functions that are used by the AND method. min (minimum), which truncates the output fuzzy set, and prod (product), which scales the output fuzzy set.

4.2.5.4 Step 4. Aggregate All Outputs

Since decisions are based on the testing of all of the rules in an FIS, the rules must be combined in some manner in order to make a decision. Aggregation is the process by which the fuzzy sets that represent the outputs of each rule are combined into a single fuzzy set. Aggregation only occurs once for each output variable, just prior to the fifth and final step, defuzzification. The input of the aggregation process is the list of truncated output functions returned by the implication process for each rule. The output of the aggregation process is one fuzzy set for each outpnt variable. Notice that as long as the aggregation method is commutative (which it always should be), then the order in which the rules are executed is unimportant. Three built-in methods are supported: max (maximum), probor (probabilistic OR), and sum (simply the sum of each rule's output set).

4.2.5.5 Step 5. Defuzzify

The input for the defuzzification process is a fuzzy set (the aggregate output fuzzy set) and the output is a single number. As much as fuzziness helps the rule evaluation during the intermediate steps, the final desired output for each variable is generally a single number. However, the aggregate of a fuzzy set encompasses a range of output values, and so must be defuzzified in order to resolve a single output value from the set. Perhaps the most popular defuzzification method is the centroid calculation, which returns the center of area under the curve.

4.2.6 The Fuzzy Inference Diagram

The fuzzy inference diagram is the composite of all the smaller diagrams we've been looking at so far in this section. It simultaneously displays all parts of the fuzzy inference process we've examined. Information flows through the fuzzy inference diagram as shown below:

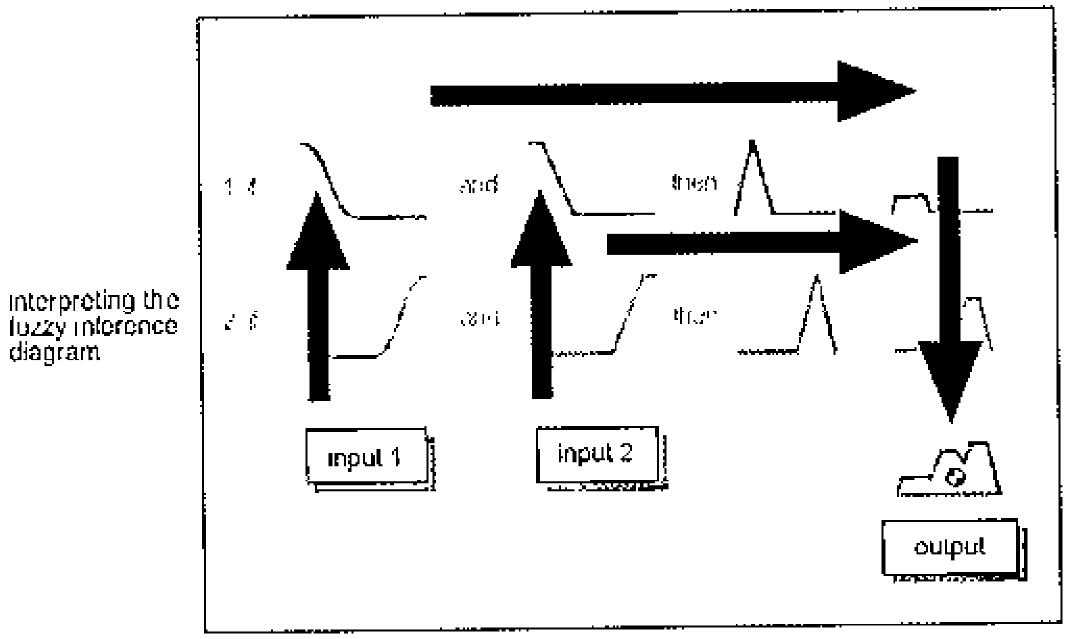


Figure 4.8: Fuzzy Inference Diagram

Notice how the flow proceeds up from the inputs in the lower left, then across each row, or rule, and then down the rule outputs to finish in the lower right. This is a very compact way of showing everything at once, from linguistic variable fuzzification all the way through defuzzification of the aggregate output.

4.3 Reliability And Availability Based Scheduling

Reliability is the probability that a plant or component will not fail to perform within specified limits in a given time while working in a stated environment. For the case of mechanical systems the following definition can be used to define reliability:

Mechanical **Reliability** is the probability that a spare, item, or unit will perform its prescribed duty without failure for a given time when operated correctly in a specified environment.

Availability is the measure of how often the system is available for use (such as a system's up-time percentage).Availability and reliability may sound like the same thing, but it is worth noting that a system can have great Availability but no Reliability. An internet router is a good example of this; it stores no state data. It is one of the few systems where in data loss is acceptable, as long as high availability is maintained.

If machine reliability and availability is not considered during the scheduling process then the schedule may not be a realistic schedule due to the uncertain shutdown of the machine. So instead of choosing alternative machine randomly, in order to incorporate the machine's reliability and availability in scheduling process the priority of the machines should be determined by their failure and repair characteristics. These types of characteristics are usually determined by the mean time to failure (MTTF) and mean time to repair (MTTR).

CHAPTER 5

RELIABILITY & AVAILABILITY BASED HYBRID FLOW SHOP SCHEDULING

5.1 General Introduction

Flow shop is a multi-stage production process with the property that all products have to pass through a number of stages in the same order. The multiprocessor flow shop is an extension of the flow shop: at every stage a number of identical machines are available that can operate in parallel. Scheduling problems may deal with multi objectives and then the scheduling becomes more complicated. In this research the multi criteria of the jobs like the processing time, due date , cost over time and the multi criteria of the machine like the reliability and availability are used to have an realistic schedule.

5.2 Proposed method

Proposed approach of this research is to identify multi objective scheduling parameter (processing time, due date, cost over time), machine reliability- and availability affected variables (MTTR, MTTF) and construct their appropriate membership functions & fuzzy rules. Using these membership functions & fuzzy rules developed a fuzzy inference system (FIS) to identify priority of each job & other fuzzy inference system to identify priority of each machine for each individual stage of hybrid flow shop using MATLAB Fuzzy Logic Toolbox. To process the top priority job in the top priority machines and minimize the make span lastly an algorithm is developed for grouping, sequencing & allocating the jobs to the machines at every stage using the priority in such a way that total percentage of over utilization is minimized. Proposed method is summarized as the following frame work:

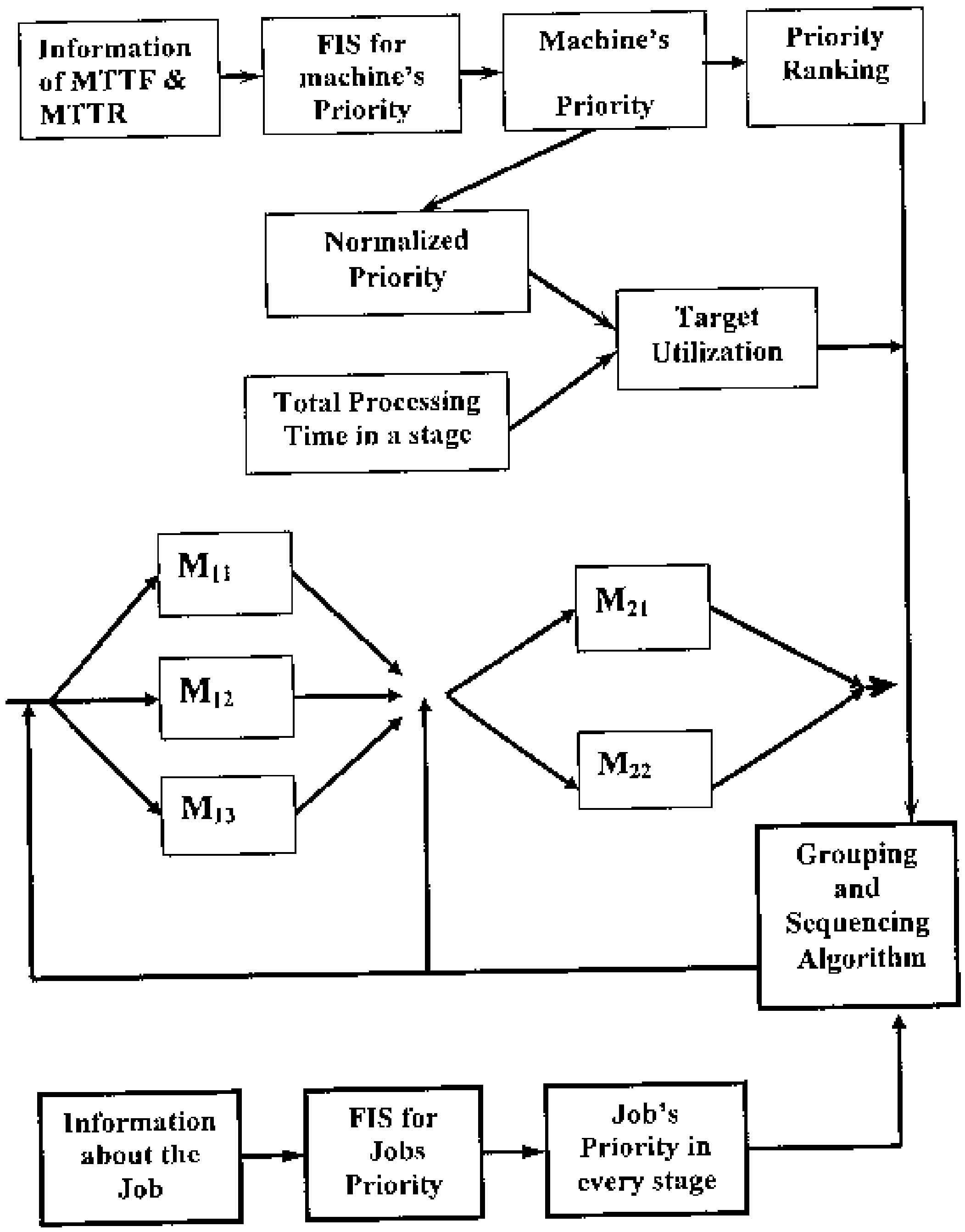

Information of MTTF & MTTR
FIS for machine's Priority
Machine's Priority
Priority Ranking
Normalized Priority
Target Utilization
Total Processing Time in a stage
M_{11}
M_{12}
M_{13}
M_{21}
M_{22}
Grouping and Sequencing Algorithm
Information about the Job
FIS for Jobs Priority
Job's Priority in every stage

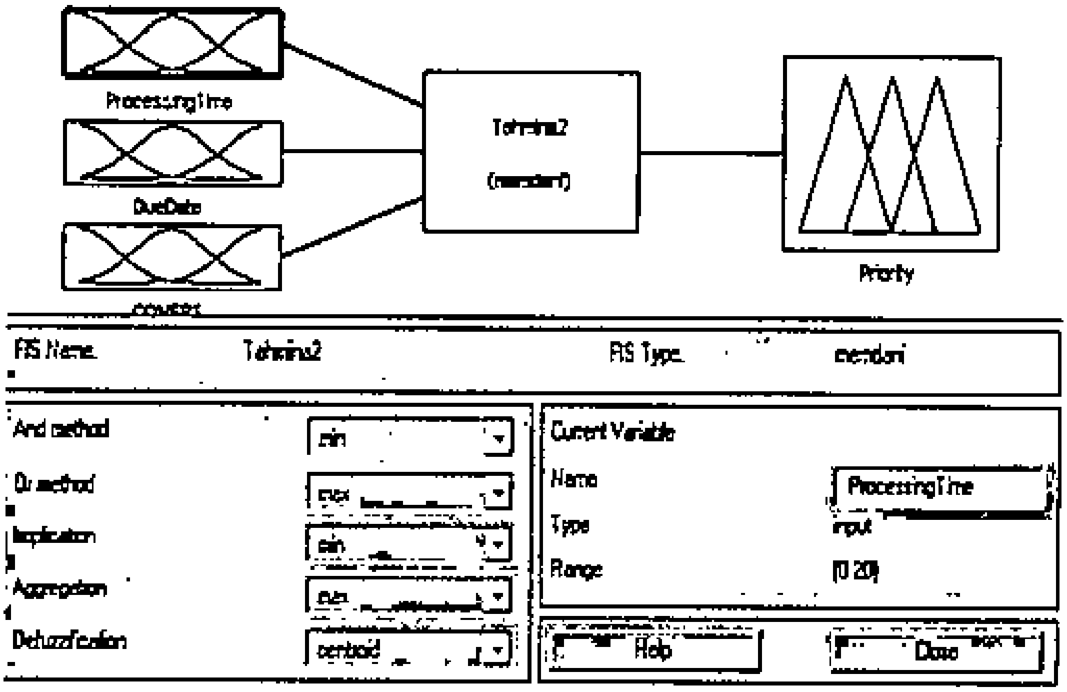
Figure 5.1: Reliability and Availability based Multi Objective Scheduling Frame Work.

5.2.1 Fuzzy model identification

Mamdani's fuzzy inference method is the most commonly seen fuzzy methodology. Mamdani's method was among the first control systems built using fuzzy set theory. It was proposed in 1975 by Ebrahim Mamdani as an attempt to control a steam engine and boiler combination by synthesizing a set of linguistic control rules obtained from experienced human operators. Mamdani type inference, as we have defined it for the Fuzzy Logic Toolbox, expects the output membership functions to be fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification. it is possible, and in many cases much more efficient, to use a single spike as the output membership function rather than a distributed fuzzy set. In this study Mamadani type fuzzy inference method is used because it is intuitive, well suited to human input and nonlinear system. Here all the variables are expressed as linguistic variables. In this model 'minimum' is used for implication stage, 'maximum' is used for aggregation stage and 'centroid' is used for defuzzification.

5.2.2 FIS for job priority determination

To incorporate multi objective scheduling fuzzy priority is calculated by developing a fuzzy inference system using MATLAB fuzzy logic tool box. Three input variables processing time, due date, cost over time are used in this FIS and out put of this FIS is priority value of the job. These input and output values should be expressed as a membership function to develop the FIS. A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. The input space is sometimes referred to as the universe of discourse, a fancy name for a simple concept. There are many ways to assign values or function to fuzzy variables. This assignment process can be intuitive process performed by the scheduling knowledge. In this research Gaussian membership function is chosen for the processing time and triangular membership function is chosen for other variables and three membership functions for inputs and five membership functions for output is used because the more the number of membership function, the more the number of rule needs. The following figure shows the membership functions used in this FIS for job priority determination.



FIS Name: Tehsina2
FIS Type: mamdani

And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid

Current Variable
Name: ProcessingTime
Type: input
Range: [0 20]

Figure 5.2: FIS for job priority calculation

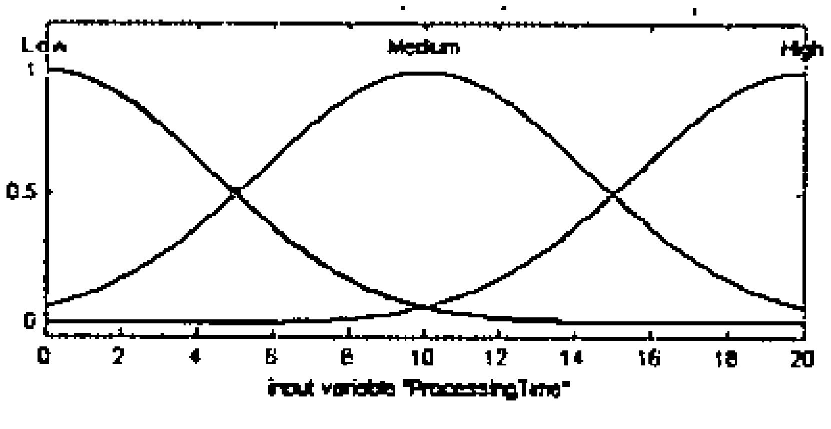


Figure 5.3: Membership function for Processing Time

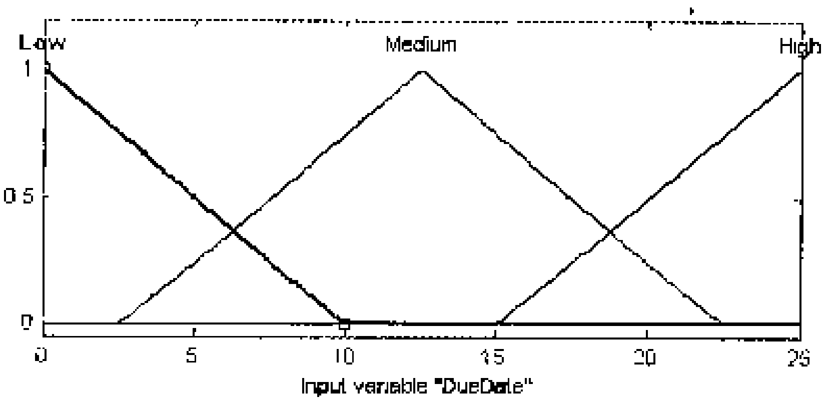


Figure 5.4: Membership function for Due Date

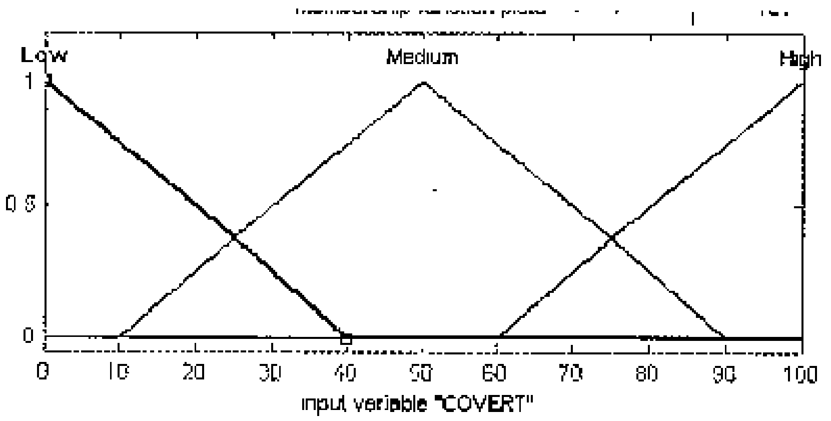


Figure 5.5: Membership function for COVERT

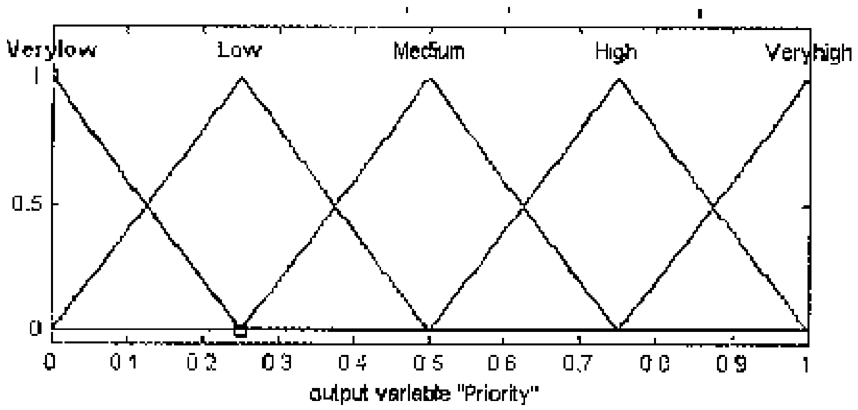


Figure 5.6: Membership function for Priority of Job

Processing time has close relation to average flow time, WIP inventory and capacity Utilization. In case there is more than one job to be processed, it is preferred to process the job first which have lowest processing time to reduce the average flow time, WIP inventory level and increase capacity utilization. Other important parameter is due date which is considered significantly to complete the job by its due date and thus tardiness can be minimized. Cost over time (COVERT) is also incorporated to prioritize the job because the expected per unit delay cost varies and it significantly affects the production cost.

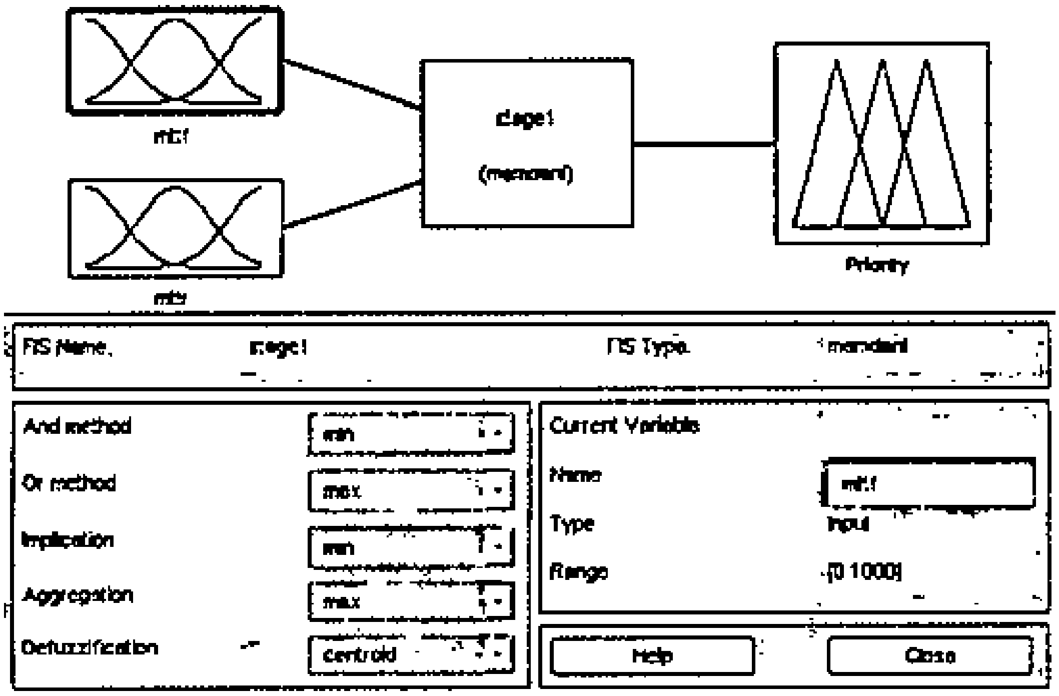
Taking all these above information in mind the following 9 rules are constructed in the fuzzy inference system to determine the priority of the jobs on each machine at every stage. Among them the top 3 rules have weight 0.5 and rest of the 6 rules has weight 1 because Due date and Cost Over Time (COVERT) are considered more important than the processing time.

If (processing time is low) then (priority is high)
If (processing time is medium) then (priority is medium)
If (processing time is high) then (priority is low)
If (Due date is low) then (priority is very high)
If (Due date is medium) then (priority is medium)
If (Due date is high) then (priority is very low)
If (COVERT is low) then (priority is low)
If (COVERT is medium) then (priority is high)
If (COVERT is high) then (priority is very high)

5.2.3 FIS for machine's priority determination

In hybrid flow shop scheduling, machine's priority is very important because the highly reliable and available machine should get the high priority during allocation of the top priority job. Reliability is a broad term that focuses on the ability of a product to perform its intended function. Reliability can be defined as the probability that an item will continue to perform its intended function without failure for a specified period of time under stated conditions. To determine the priority of the each machine in every stage fuzzy inference system is developed which take the mean time to failure(MTTF) and mean time to repair (MTTR) as input & machine's priority as a output. In machine reliability, mean time to failure (MTTF) is a measure of expected time where the machine will fail. In a case of an exponential distribution, the MTTF value is the reciprocal of the failure rate. On the other hand mean time to repair (MTTR) is also important parameter which affects the reliability and availability most. If a machine requires more time to repair then it's priority is low & vice versa.

In order to overcome the uncertainty and imprecision, fuzzy logic theory is used to restore the integration of reliability and availability. In the following FIS, triangular membership function is chosen for all input and output variables and five membership functions for inputs and five membership functions for output is used. All these are determined by the expert knowledge of scheduling. The following figure shows the membership functions used in this FIS for machines priority determination.



FIS Name: stage1
FIS Type: mamdani
And method: min
Or method: max
Implication: min
Aggregation: max
Defuzzification: centroid
Current Variable
Name: mttf
Type: input
Range: [0 1000]

Figure 5.7: FIS for machine priority calculation

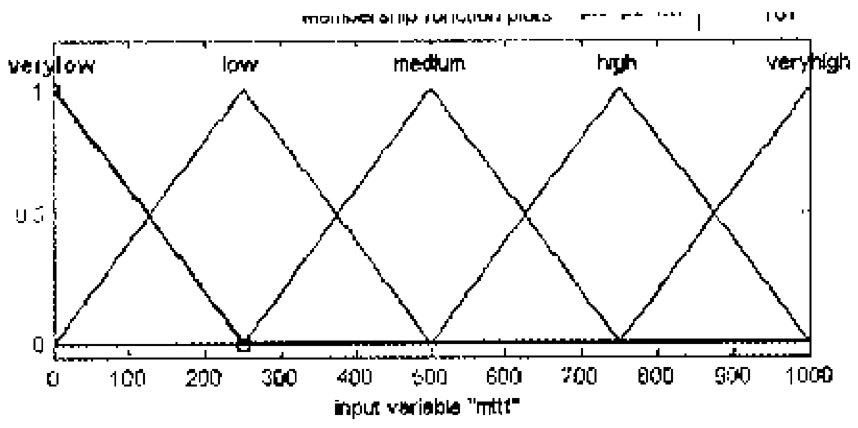


Figure 5.8: Membership function for MTTF

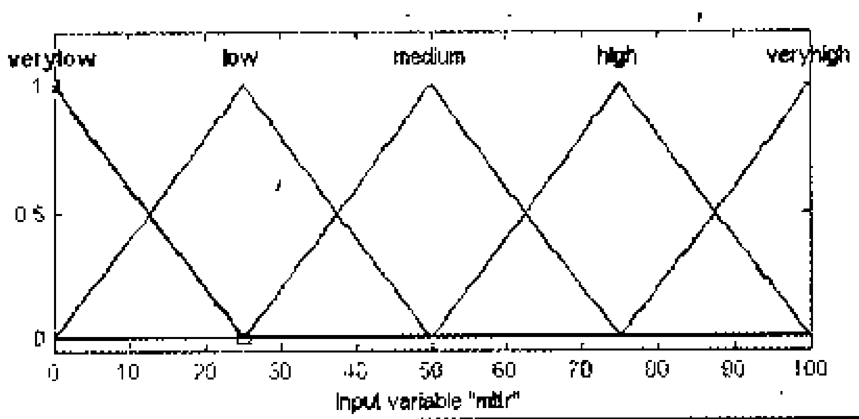


Figure 5.9: Membership function for MTTR

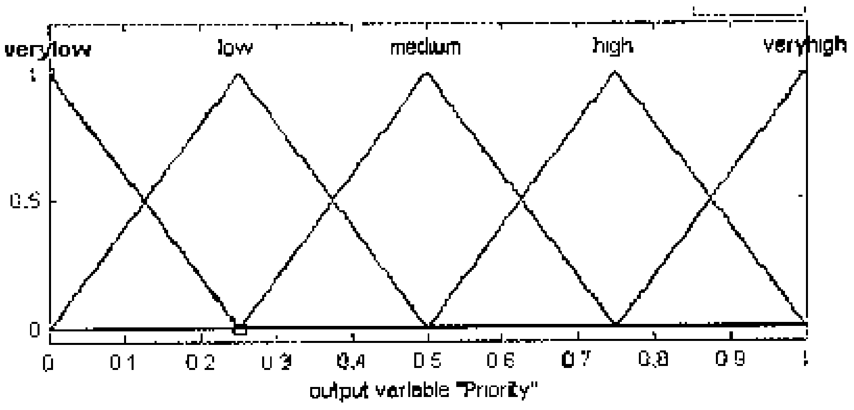


Figure 5.10: Membership function for Priority of Machine

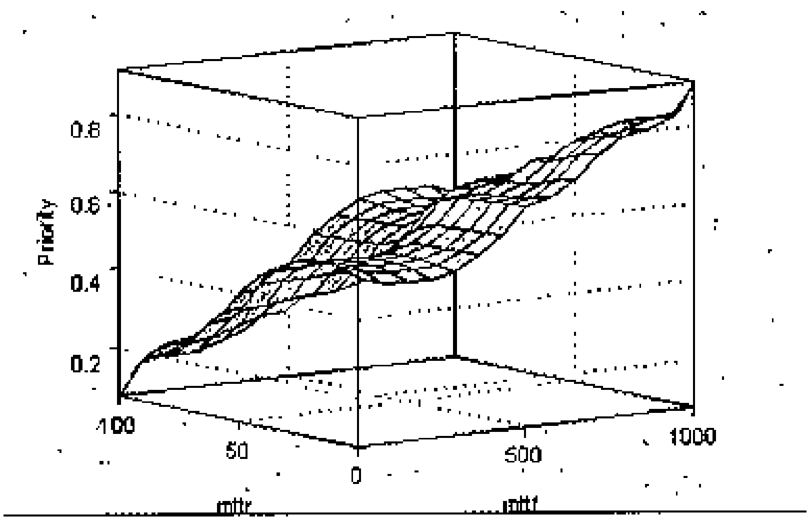


Figure 5.11: Relationship Diagram of Machine's Priority, MTTR, MTTF

Following 10 rules are used in the fuzzy inference system to determine the priority of each machine in every stage (each of them has equal rule weight 1):

If (MTTF is very low) then (machine priority is very low)
If (MTTF is low) then (machine priority is low)
If (MTTF is medium) then (machine priority is medium)
If (MTTF is high) then (machine priority is high)
If (MTTF is very high) then (machine priority is very high)
If (MTTR is very low) then (machine priority is very high)
If (MTTR is low) then (machine priority is high)
If (MTTR is medium) then (machine priority is medium)
If (MTTR is high) then (machine priority is low)
If (MTTR is very high) then (machine priority is very low)

5.3 Grouping and Sequencing Algorithm

After getting the priority of each job & machine for every stage using the two different FIS the following measures are taken:

Job priority is determined based on their priority value. The job which has highest priority value has ranked top, second highest priority has ranked second and so on. Similarly machine priority is determined based on the reliability and availability value and using this priority value and processing time of each job the target utilization is calculated with the following equation:

Suppose the machines have Priority $R_1, R_2, R_3, \ldots\ldots\ldots R_n$

Normalized Priority (for machine i) $NR_i = (R_i) / \left(\sum_{i=1}^{n} R_i \right)$

Target Utilization (for machine i) $T_i = NR_i \times TPT$

Where TPT=Total Processing Time

5.3.1 Grouping

Main principle of this grouping algorithm is to perform the top priority job in the top priority machine. So first assign the top priority jobs to the highest priority machine until it satisfy the target utilization.

When it does not satisfy the target utilization go to the second highest priority machine & assign the rest of the job until it satisfy the target utilization of the second highest priority machine.

If it does not satisfy in first assignment calculate percentage of over utilization for that assignment (U2%) & also calculate the percentage of over utilization (U1%) if it is assigned to the highest priority machine with previous assignment. Assign the Job to the machine in which percentage of over utilization is minimized & assign the rest of the job until it satisfy the target utilization of the second highest priority machine.

If it does not satisfy other than first assignment go to the third highest priority machine & assign the rest of the job until it satisfy the target utilization of the third highest priority machine.

If it does not satisfy in first assignment calculate percentage of over utilization for that assignment & also calculate the percentage of over utilization if it is assigned to the second highest priority machine with previous assignment. Assign the Job to the machine in which percentage of over utilization is minimum & assign the rest of the jobs until it satisfy the target utilization of the third highest priority machine.

If it does not satisfy other than first assignment go to the fourth highest priority machine & assign the rest of the job until it satisfy the target utilization of the fourth highest priority machine.

Similarly grouping is performed in other stages.

After that a re-grouping operation is performed starting from the lowest priority machine which has over utilized allocation. Among the group the job that has highest priority, send to the next higher priority machine and the over utilization is calculated. For which machine over utilization is minimum, keep the job to that machine. Similarly check the all machine having over utilization from bottom to top.

5.3.2 Sequencing

Sequencing is determined based on the priority of the job. Highest priority job on the group goes first then second third & so on.

But except the first stage sequencing other sequencing may need to modify in order to minimize the make span without hampering the main principle of groping & sequencing i.e., the higher priority job does not need to wait for the job which has less priority. It is modified in such a way that if the arrival time of the higher priority job is greater than the completion time of the less priority job in that stage of the same group then do the less priority job first.

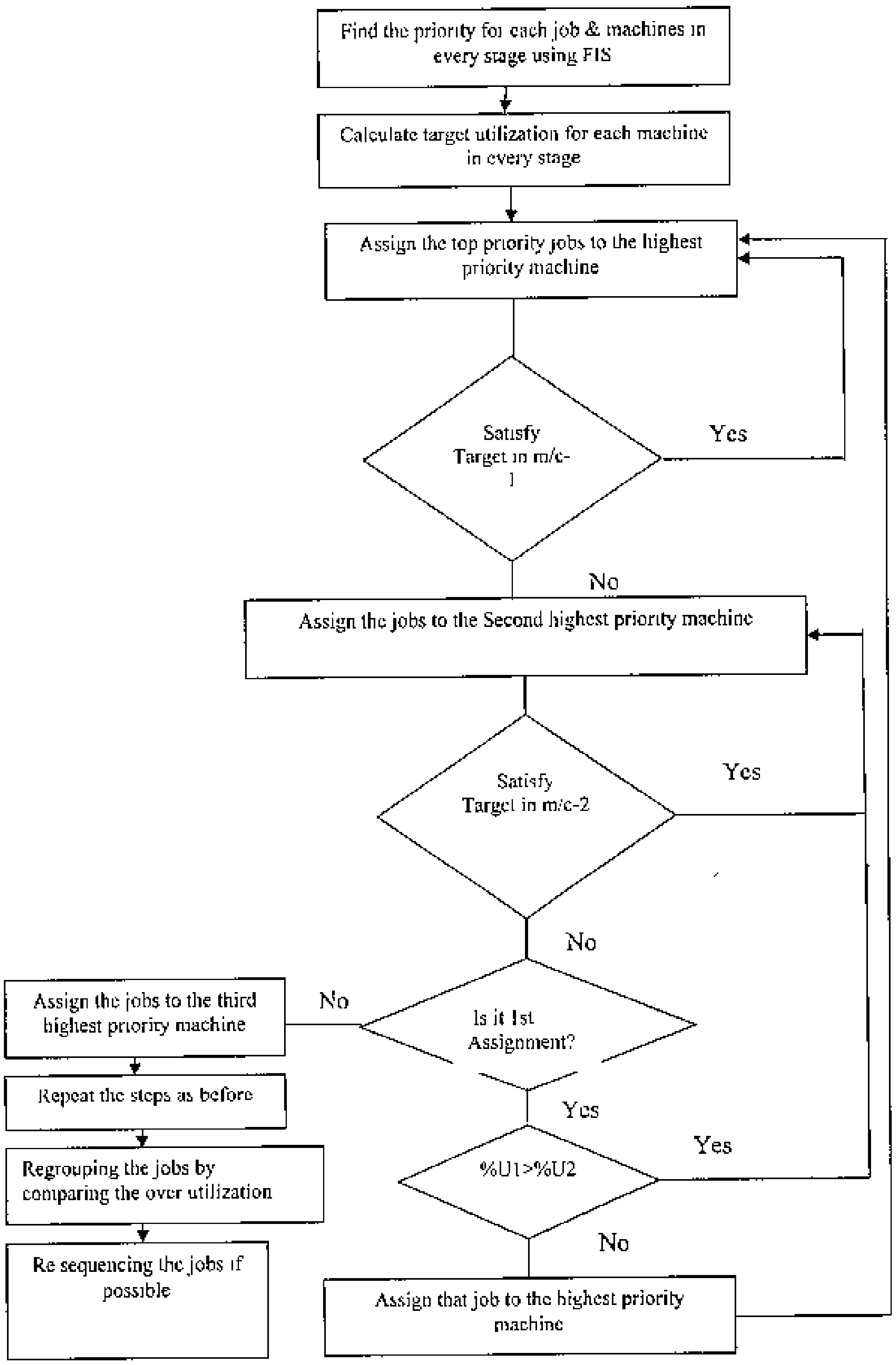

Find the priority for each job & machines in every stage using FIS
Calculate target utilization for each machine in every stage
Assign the top priority jobs to the highest priority machine
Satisfy Target in m/c-1
Yes
No
Assign the jobs to the Second highest priority machine
Satisfy Target in m/c-2
Yes
No
Is it 1st Assignment?
No
Yes
Assign the jobs to the third highest priority machine
Repeat the steps as before
Regrouping the jobs by comparing the over utilization
Re sequencing the jobs if possible
%U1>%U2
Yes
No
Assign that job to the highest priority machine

Figure 5.12: Grouping, Sequencing Algorithm

CHAPTER 6

RESULT ANALYSIS

6.1 Introduction

The developed algorithm for fuzzy multi objective machine reliability based multi processor flow shop scheduling has been coded in C programming languages with MATLAB fuzzy logic tool box to put the System into practice. MATLAB fuzzy logic tool box is used to prioritize the jobs and different machines in an individual stage to full fill the multi objective criteria. Using these priorities of jobs and machines the developed scheduling algorithm then provides us the schedule. For analyzing the performance of the developed algorithm a case study is presented here.

6.2 Case Study

A case study is presented here using the hypothetical data to clarify the proposed process. A three stage multi processor flow shop is presented here as like the following figure:

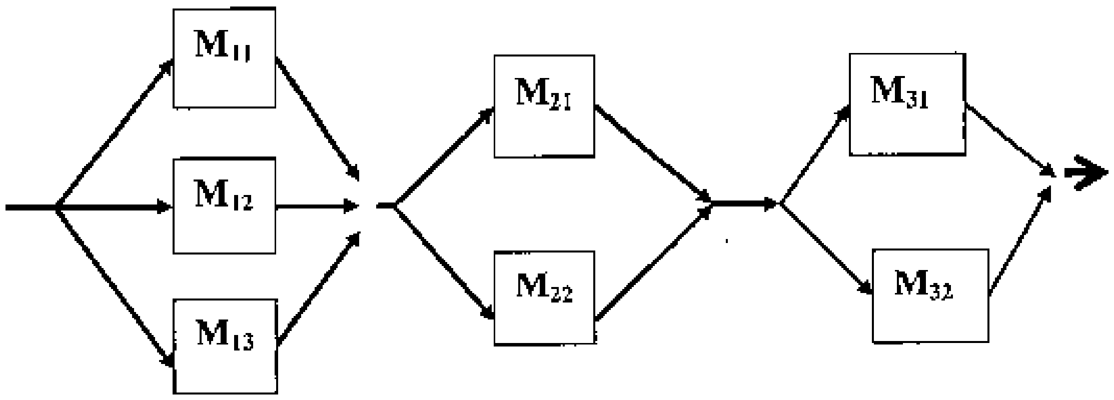


Figure 6.1: 3 stage hybrid flow shop

In first stage there are 3 machines and in stage 2 ; stage 3 there are 2 machine taken into consideration. All the machines in every stage have identical processing time but different reliability which is calculated based on their failure and repair characteristics. Five jobs are considered here having the following 3 different processing time in 3 stage, due date for each jobs and cost over time (COVERT) for each job which are determined by the customer requirements.

Table 6.1: Information of jobs in stage 1, stage2, stage3

Job name	**Processing Time (Stage 1)**	**Processing Time (Stage 2)**	**Processing Time (Stage3)**	**Due Date**	**Covert**
A	15	12	10	25	95
B	20	10	15	20	80
C	12	9	20	21	25
D	10	15	9	15	70
E	3	4	3	5	100
Total Processing Time	60	50	57		

Priority of the job is calculated using the fuzzy inference system. Based on this priority the Top priority job in stage 1 is E, then D, then B, then A and lowest priority Job is C. Similarly priority of the machines in other stage is determined.

Priorities of the machines are also determined based on the reliability of the machines found from the second fuzzy inference system. The machine which has the highest reliability has ranked top priority machine, the machine which has second highest reliability has ranked second top priority machine in a stage and so on. Then based on their normalized priority proportion the target utilization for each machine is determined.

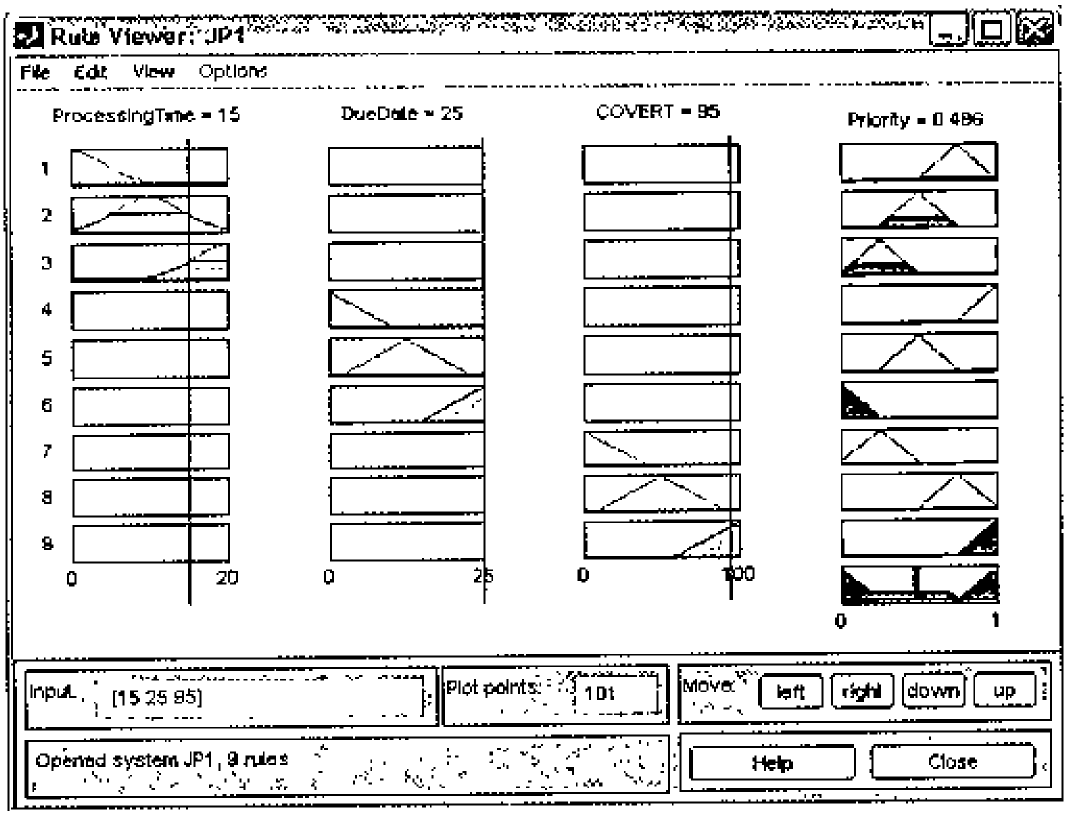
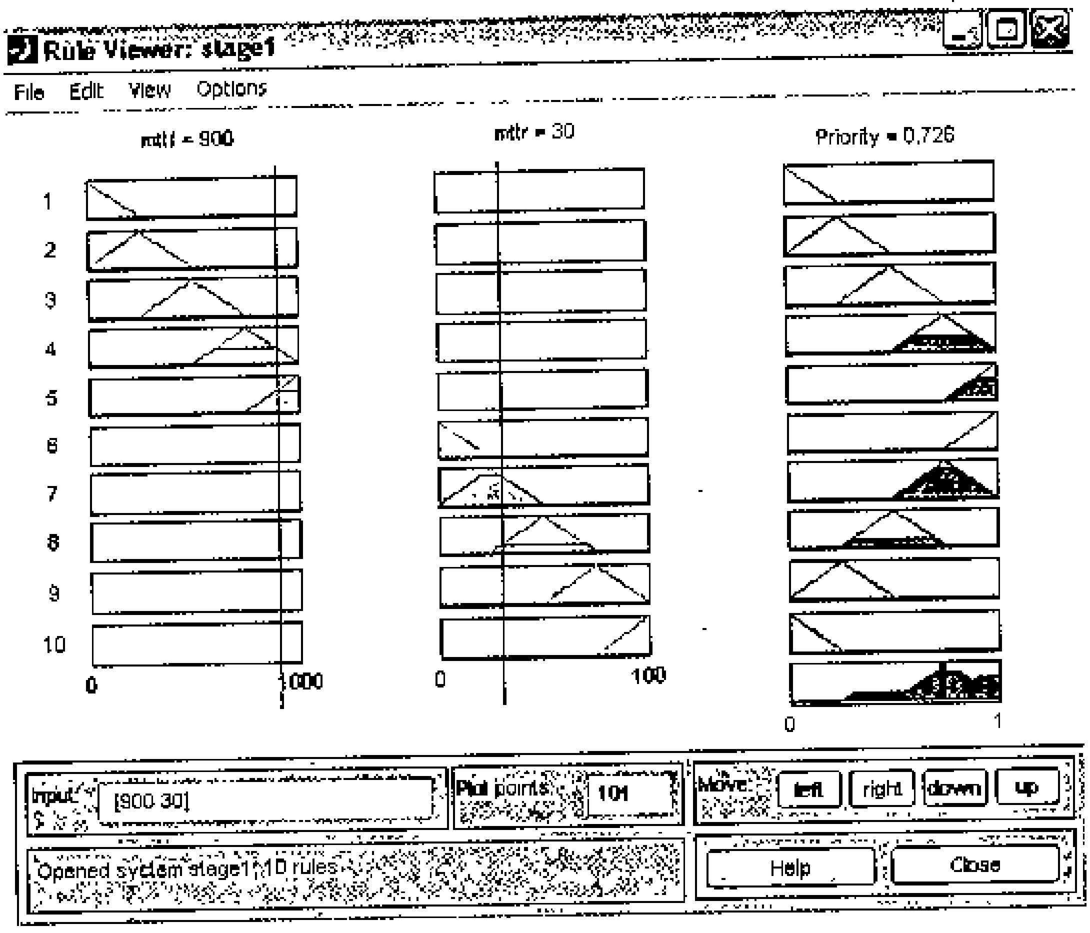


Figure 6.2: Job's Priority Calculation using FIS

Table 6.2: Priority of Job in stage 1, stage2, stage3

Job name	**Priority (Stage 1)**	**Priority (Stage 2)**	**Priority (Stage 3)**
A	0.486	0.474	0.497
B	0.465	0.507	0.5
C	0.446	0.446	0.424
D	0.608	0.509	0.612
E	0.726	0.721	0.726



Opened system stage1, 10 rules

Figure 6.3: Machine Priority Calculation using FIS

Table 6.3: Information & priority of machines in stage 1

M/c no.	MTTF (minute)	MTTR (minute)	Priority	Normalized Priority	Target Utilization
1	900	30	0.726	0.51599147	30.959
2	500	60	0.407	0.28926795	17.356
3	300	90	0.274	0.19474058	11.684
				1	60

Table 6.4: Information & priority of machines in stage 2

M/c no.	MTTF (minute)	MTTR (minute)	Priority	Normalized Priority	Target Utilization
1	695	20	0.691	0.6098853	30.494
2	500	55	0.442	0.3901147	19.506
				1	50

Table 6.5: Information & priority of machines in stage 3

M/c no.	MTTF (minute)	MTTR (minute)	Priority	Normalized Priority	Target Utilization
1	605	30	0.636	0.5606732	31.958
2	500	50	0.50	0.4393268	25.042
				1	57

According to the developed algorithm following table provide the ultimate grouping of the jobs to the machines and accordingly the jobs should be allocated to restore the priority of the jobs and machines.

Table 6.6: Final Schedule

Machine No. (sequenced as priority)	Stage1	Stage2		Stage3	
		Without Re-Sequencing	With Re-Sequencing	Without Re-Grouping	With Re-Grouping
1	E,D,A	E,D,B	E,D,B	E,D,B	E,D,B,A
2	B	A,C	**C,A**	A, C	C
3	C			U%=20%	U%=16%
Total Make Span		69	63		

Sequencing is done easily here based on the priority of each job in a group. Suppose in stage 1, machine 1 the job E should be process first, then D and then

A. But re-sequencing algorithm may alter the sequencing based on the arrival time of the highest priority job in a group. Here for stage 2, machine 2 according to re sequencing algorithm the job C should be processed first instead of job A having higher priority because the arrival time of job A (28 minute) in Stage 2 is greater than the completion time of job C (21 minute) in stage 2.This re sequencing algorithm is used to minimize the make span without hampering the priority restoration. Simultaneously a re grouping algorithm is used in stage 3 to minimize the over utilization.

6.3 Computational Result

The developed program has been run on a TOSHIBA Laptop MACH010AQT07 Intel (R) Celeron (R) M having 1.60 GHz of processor speed, 1.60 GHz 224 MB Ram and Microsoft Windows XP operating system for the processing of 50 jobs in 50 stages each of having 50 parallel machines hybrid flow shop. The computational time was 3.51 seconds, which shows a good performance of the algorithm in view of computational speed.

CHAPTER 7

CONCLUSION AND RECOMMENDATIONS

7.1 Conclusion

Instead of choosing job randomly, this research proposed the use of fuzzy inference system to determine the job priority based on their processing time, due date, cost over time (COVERT) which are fuzzy in nature most of the time. This is an alternative way to incorporate the influence of processing time, cost over time, earliest due date during multi processor flow shop scheduling to fulfill multi objective because many real-world scheduling problems are multi objective. That is, there exist several criteria that must be taken into consideration when evaluating the quality of the proposed schedule.

On the other hand, inherent complexity and uncertainty makes the scheduling process complicated and here fuzzy logic is used in order to build aggregated rules allowing obtaining a compromise between the satisfactions of several criteria. The inability of the normal scheduling methods to address the imprecision and uncertainty paved the path for the fuzzy scheduling techniques. By incorporating the expert knowledge of experienced scheduler the fuzzy inference system can be developed in such a way that it provides a realistic result.

To realize the integration between maintenance and production planning machines are prioritized using another fuzzy inference system based on their reliability and availability because if machine reliability and availability is not considered during the scheduling process then the schedule may not be a realistic schedule due to the uncertain shutdown of the machine.

To balance loads on each machine an algorithm is developed for grouping, sequencing & allocating the jobs to the machines at every stage in such a way that total percentage of over utilization is minimum and sends the top priority jobs to the top priority machine. A re sequencing mechanism is also incorporated to minimize the make span without hampering the priority restoration.

7.2 Recommendations

Future aim of this research is to perform a sensitivity analysis by changing the values of meantime to failure (MTTF) & mean time to repair (MTTR) of each machine & see the significant differences each of change makes. An algorithm can be developed for automatic selection of the more sensitive machine, for which if the reliability and availability is increased by deploying more maintenance work then the make span will be decreased significantly and according to this, a realistic production and maintenance plan then can be possible to implement. To facilitate the proposed scheduling method the fuzzy logic tool box can be interfaced with the developed tool so that it can automatically read the out put of the fuzzy inference system.

REFERENCE

Allet, S.,"Handling flexibility in a "generalized job shop" with a fuzzy approach ",European Journal of Operational Research, Vol-147, pp. 312-333, 2003.

Andreas, A.C. and Nearchou, C., "A novel meta heuristic approach for the flow shop scheduling problem -Engineering Applications of Artificial Intelligence", Vol-17, pp. 289-300, 2004.

Anilkumar, K. G. and Tanprasert, T. "Neural network based priority assignment for job scheduler"Australian Journal of Technology, Vol-9, no.3: pp.181-186, 2006

Ben-Daya, M. and Al-Fawzan, M.A., "tabu search approach for the flow shop scheduling problem" ,European Journal of Operational Research, Vol-109,pp.88-95,1998.

Chang, P., Chen, S. and Liu, C., "Sub-population genetic algorithm with mining gene structures for multi objective flow shop scheduling problems.", Expert Systems with Applications, In Press.

Chan,W. and Hu, H., "An application of genetic algorithms to precast production scheduling" Computers & Structures, Vol- 79, , pp.1605-1616, 2001.

Cheng, J., Steiner G., and Stephenson, P., "A computational study with a new algorithm for the three-machine permutation flow-shop problem with release times ", European Journal of Operational Research, Vol-130, pp. 559-575, 2001.

Cho, H., Paik, C., Yoon, H. and Kim, H., "A Robust Design of Simulated Annealing Approach for Mixed-Model Sequencing" Computers and Industrial Engineering, Vol- 48, pp., 2005.

Dagli, C. and Sittisathanchai, S.,"Genetic neuro-scheduler for job shop scheduling", Computers & Industrial Engineering, Vol- 25, pp. 267-270 ,1993.

Felix, T. S. and Abhary, C. K., "A fuzzy approach to operation selection", Engineering Applications of Artificial Intelligence, Vol-10,pp. 345-356,1997

Fink, A. and Vob, S., "Solving the continuous flow-shop scheduling problem by metaheuristics", European Journal of Operational Research, Vol-151, pp 400-414, 2003.

Garcia, J. M. and Lozano, S., "Production and delivery scheduling problem with time windows" Computers and Industrial Engineering, Vol-48: pp. 733-742, 2005.

Grabot, B. and Geneste, L., "Dispatching rules in scheduling: a fuzzy approach" International journal of production research, Vol-32,No. 4: pp.903-915, 1994

Grabowski, J. and Pempera, J , "Some local search algorithms for no-wait flow-shop problem with makespan", Computers & Operations Research, Vol-32, pp. 2197-2212, 2005.

Grabowski, J. and Wodecki, M.A., "very fast tabu search algorithm for the permutation flow shop problem with makespan criterion", Computers Operations Research,Vol-31,pp.1891-1909,2004.

Gupta J. N. D., Hennig, K. and Werner, F, "Local search heuristics for two-stage flow shop problems with secondary criterion", Computers & Operations Research, Vol-29, pp.123-149,2002

Hoogeveen, J. A., Lenstra, J. K., and Veltman, B.,"Preemptive scheduling in a two-stage multiprocessor flow shop is NP-hard" European Journal of Operational Research, Vol- 89,pp. 172-175, 1996.

Hong, T., and Wang, T., "Fuzzy flexible flow shops at two machine centers for continuous fuzzy domains", Information Sciences, Vol-129, pp. 227-237, 2000.

Ishibuchi, H., Yamamoto, N., Misaki, S. and Tanaka, H., "Local search algorithms for flow shop scheduling with fuzzy due-dates", International Journal of Production Economics, Vol- 33,pp.53-66, 1994.

Janiak, A., Kozan, E., Lichtenstein, M. and Oğuz, C., "Metaheuristic approaches to the hybrid flow shop scheduling problem with a cost-related criterion", International Journal of Production Economics, In Press

Kacem ,I., Hammadi,S., and Borne,P., "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic", Mathematics and Computers in Simulation, Vol-60, pp.245-276, 2002.

Keung, K. W., Ip, W. H. and Yuen, D., "An intelligent hierarchical workstation control model for FMS", Journal of Materials Processing Technology, Vol-139,pp.134-139,2003.

Lee, I and Shaw, M. J., "A neural-net approach to real time flow-shop sequencing .",Computers & Industrial Engineering, Vol-38,pp.125-147,2000.

Liu, S.Q., Ong, H.L. and Ng, K M., "A fast tabu search algorithm for the group shop scheduling problem" Advances in Engineering Software, Vol-36, pp.533-539,2005.

Loukil, T., Teghem, J. and Tuyttens, D., "Solving multi-objective production scheduling problems using metaheuristics", European Journal of Operational Research,Vol-161,pp.42-61,2005.

Mansouri, S. A. "A simulated annealing approach to a bi-criteria sequencing problem in a two-stage supply chain" Computers and Industrial Engineering, vol-50: pp. 105-119, 2006.

McCahonE, C. S. and Lee, S., "Fuzzy job sequencing for a flow shop", European Journal of Operational Research, Vol-62, pp.294-301,1992.

Nowicki, E. and Smutnicki, C., "Some aspects of scatter search in the flow-shop problem, European Journal of Operational Research, Vol- 169,pp 654-666 2006.

Petroni,A and Rizzi, A., "A fuzzy logic based methodology to rank shop floor dispatching rules ", International Journal of Production Economics, Vol- 76, pp.99-108, 2002.

Pinedo M., "Scheduling Theory, Algorithms, and Systems", Second Edition, Prentice Hall, 2002.

Sabuncuoglu, I. and Gurgun,B., "A neural network model for scheduling problems " European Journal of Operational Research, Vol-93,pp.288-299, 1996.

Solimanpur ,M., Vrat , P. and. Shankar, R.,"A neuro-tabu search heuristic for the flow shop scheduling problem"Computers & Operations Research, Vol- 31, pp. 2151-2164,2004.

Tian, P., Ma, J. and Zhang, D.-M., "Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation property: An investigation of generation mechanism", European Journal of Operational Research, Vol- 118 ,pp 81-94, 1999.

Tsujimura , Y., Park, S H., Chang ,I. S. and Gen, M., "An effective method for solving flow shop scheduling problems with fuzzy processing times " Computers & Industrial Engineering,Vol-25,pp.239-242,1993.

Wang, L., Zhang, L. and Zheng, D., "An effective hybrid genetic algorithm for flow shop scheduling with limited buffers", Computers & Operations Research, Vol-33,pp.2960-2971,2006.

Watanabe, T., Tokumaru , H. and Hashimoto, Y., "Job-shop scheduling using neural networks", Control Engineering Practice, Vol- 1, pp. 957-96, 1993.

Ying, K.-C. and Liao, C.-J., "An ant colony system for permutation flow-shop sequencing" , Computers & Operations Research, Vol- 31, 2004,pp 791-801, 2004.

Yun, Y. S., "Genetic algorithm with fuzzy logic controller for preemptive and non preemptive job-shop scheduling problems", Computers & Industrial Engineering,Vol-43,pp.623-644,2002.

Zandieh, M., Ghomi S.M.T. F. and Husseini, S.M. M., "An immune algorithm approach to hybrid flow shops scheduling with sequence-dependent setup times" Applied Mathematics and Computation, In Press.

Zadeh, L.A. "Fuzzy Sets", Information and Control, Vol- 8,pp.338-353, 1965.

Appendices

A INPUT DATA

Total stage: 3

Total job: 5

Total machine in stage 1: 3

Jobs Details:

Jobs a
Priority: .486
Time: 15

Jobs b
Priority: .465
Time: 20

Jobs c
Priority: .446
Time: 12

Jobs d
Priority. .608
Time: 10

Jobs e
Priority: .726
Time: 3

Machine details:
Machine 1
Reliability: .726

Machine 2
Reliability: .407

Machine 3
Reliability: .274

Total machine in stage 2: 2

Jobs Details:

```
        Jobs a
          Priority:  .474
          Time:  12

        Jobs b
          Priority:  .507
          Time:  10

        Jobs c
          Priority:  .446
          Time:  9

        Jobs d
          Priority:  .509
          Time:  15

        Jobs e
          Priority:  .721
          Time:  4

Machine details:
        Machine 1
          Reliability:  .714

        Machine 2
          Reliability:  .5


Total machine in stage 3:  2


Jobs Details:

        Jobs a
          Priority:  .497
          Time:  10

        Jobs b
          Priority:  .5
          Time:  15

        Jobs c
          Priority:  .424
          Time:  20

        Jobs d
          Priority:  .612
          Time:  9
```

Jobs e
  Priority: .726
  Time: 3

Machine details:
  Machine 1
    Reliability: .633

  Machine 2
    Reliability: .496

B Result

```
-------------------------------Stage: 1----------------------------------

Jobs given according to machine priority:

 e d a

 b

 c

---------------------------

Machine11:  e d a

Machine12:  b

Machine13:  c

---------------------------

 job a:   machine:  1  Time consume:  28

 job b:   machine:  2  Time consume:  20

 job c:   machine:  3  Time consume:  12

 job d:   machine:  1  Time consume:  13

 job e:   machine:  1  Time consume:  3


-------------------------------Stage: 2----------------------------------

Jobs given according to machine priority:

 e d b

 c a


---------------------------

Machine21:  e d b
Machine22:  c a

---------------------------
```

```
job a:   machine:  2  Time consume:  40

job b:   machine:  1  Time consume:  38

job c:   machine:  2  Time consume:  21

job d:   machine:  1  Time consume:  28

job e:   machine:  1  Time consume:  7

------------------------------Stage:  3----------------------------------

Jobs given according to machine priority:

 e d b a

 c



---------------------------

Machine31:   e d b a

Machine32:   c



---------------------------

job a:   machine:  1  Time consume:  63

job b:   machine:  1  Time consume:  53

job c:   machine:  2  Time consume:  41

job d:   machine:  1  Time consume:  37

job e:   machine:  1  Time consume:  10

Make span of this process 63 for the job a
```

C COMPUTER CODE

```
#include<iostream.h>
#define SIZE 10;

int total_stage;
int stage_per[SIZE];
int reliabity[SIZE][SIZE];
float time_machine[SIZE][SIZE];
int machine[SIZE][SIZE][SIZE];

struct rec
{
        int time;
        int pirority;
        int index;
        float time_consume;
};

int total_job;

void grouping(int number);
 void sorting();

void main()
{
        struct rec job[SIZE];
        int sum=0;
        float total_time=0;

        cin>>total_stage;

        for(int i=0;i<total_stage;i++)
                cin>>stage_per[i];

        for(i=0;i<total_stage;i++)
                for(int j=0;j<stage_per[i];j++)
                        cin>>reliabity[i][j];

        cin>>total_job;

        for(i=0<total_job;i++)
        {
                job[i].index=i;
        }

          for(i=0;<total_machine;i++)
          {
```

```
                    total_time=0;
                    sum=0;

                    for(j=0<total_job;j++)
                    {
                            cin>>job[j].pirority;
                            cin>>job[j].time;
                            total_time+=job[i].time;
                    }

                    for(j=0,j<per_stage[i];j++)
                             sum+=reliabity[i][j];

                    for(int k=0;k<per_stage[i];k++)
                            time_machine[k]=(reliabity[i][k]/sum)*total_time;

                    sorting();
                    grouping(i);
            }

    }

    void sorting()
    {
            for(int i=0;i<total_job;i++)
                    for(int j=i;j<total_job;j++)
                            {
                                    if(job[j].pirority>job[i].pirority)
                                            {
                                            rec m;
                                            m=job[i];
                                            job[i]=job[j];
                                            job[j]=m;
                                            }
                            }
    }

    void grouping(int number)
    {
            float time,k=0,l=0;
            float t[SIZE];

            time=job[0].time;

                            for(int j=0;j<total_job;j++)
                            {
                                    if(time>time_machine[number][k])
                                            {
                                                    k++;
```

```
                                                  time=0;
                                                  l=0;
                                            }

                                    machine[number][k][l++]=job[j].index;
                                    time+=job[j].time;

                                    if(number==0);
                                            job[j].time_consume+=time;

                    }

                    if(number>0)
                            for(int i=0;i<=l;i++)

if(job[i].time_consume>(job[i+1].time_consume+job[i+1].time))
                                            {
                                             machine[number][k][l]=job[i+1].index;
                                             machine[number][k][l+1]=job[i].index;


job[i].time_consume+=job[i].time+job[i+1].time;
              job[i+1].time_consume+=job[i+1].time;
                                            }

 }


#include<iostream.h>
#include<stdio.h>
#define SIZE 50

int total_stage;
int total_job;
float total_time;
float total_reliability;
int machine[SIZE][SIZE];
int machine_allocation[SIZE];
float machine_cost[SIZE];
int machine_record[SIZE][SIZE][SIZE];
int machine_priority[SIZE][SIZE][SIZE];
int total_m[SIZE];

struct jobs_compute
{
        float consume_time;
        int machine[SIZE];
        float time_process[SIZE];
};
```

```
struct job_details
{
        int job_index;
        char a;
        float job_time;
        float job_priority;
};

struct machine_details
{
        float reliability;
        int stage;
        int index;
        float time_given;
};

jobs_compute compute[SIZE];
job_details job_main[SIZE];
job_details jobs[SIZE];
machine_details machine_stage[SIZE];

void Assign(int stage_count,int total_machine);
void Reassign(int stage_count,int total_machine);
void Reallocate(int stage_count,int total_machine);
void output(int stage_count,int total_machine);
void computation(int stage,int total_machine);

void main()
{
        int total_machine,stage_count,j,k;
        job_details temp_job;
        machine_details temp_machine;
        float p;
        int flag;


        cout<<"Total stage: ";
        cin>>total_stage;

        cout<<"\n\nTotal job:  ";
        cin>>total_job;

        for(int i=0;i<total_job;i++)
                compute[i].consume_time=0;

        for(i=0;i<total_stage;i++)
                for(j=0;j<SIZE;j++)
                        for(k=0;k<SIZE;k++)
                {
                        machine_record[i][j][k]=-1;
                        machine_priority[i][j][k]=-1;
                }

        for(stage_count=0;stage_count<total_stage;stage_count++)
        {
                total_time=0;
                total_reliability=0;


                cout<<"\n\nTotal machine in stage "<<stage_count+1<<": ";
                cin>>total_machine;

                for(j=0;j<SIZE;j++)
                        machine_allocation[j]=0;

                for(j=0;j<SIZE;j++)
                        machine_cost[j]=0;

                cout<<"\n\nJobs Details: \n";

                for(j=0;j<total_job;j++)
                {
                        jobs[j].job_index=j;

                        jobs[j].a='a'+j;

                        cout<<"\n\tJobs "<<jobs[j].a;


                        flag=1;
                        while(flag)
                        {
                                cout<<"\n\t   Priority: ";
                                cin>>p;

                                if(p<=1 && p>0)
                                {
                                        jobs[j].job_priority=p;
                                        flag=0;
                                }
                                else
                                        cout<<"\n\t   Priority range
0<priority<=1,input again\n";
                        }

                        flag=1;
                while(flag)
                {
                cout<<"\t   Time:  ";
                cin>>p;

                if(p>=0)
                        {
                                jobs[j].job_time=p;
                                flag=0;
                        }
                else
                        cout<<"\n\t   Time is positive\n";
                }

                total_time+=jobs[j].job_time;
        }

        for(j=0;j<total_job;j++)
                job_main[j]=jobs[j];
        cout<<"\n\nMachine details:  ";
        for(j=0;j<total_machine;j++)
        {
                machine_stage[j].stage=stage_count;
                machine_stage[j].index=j;

                cout<<"\n\tMachine "<<j+1;

                flag=1;

                while(flag)
                {
                cout<<"\n\t   Reliability:  ";
                cin>>p;

                if(p<=1 && p>0)
                        {
                                machine_stage[j].reliability=p;
                                flag=0;
                        }
                else
                        cout<<"\n\t   Reliability range
0<reliability<=1,input again\n";
                }

                total_reliability+=machine_stage[j].reliability;
        }

        for(j=0;j<total_machine;j++)
                for(k=j;k<total_machine;k++)
```

```
	if(machine_stage[j].reliability<machine_stage[k].reliability)
					{
						temp_machine=machine_stage[j];
						machine_stage[j]=machine_stage[k];
						machine_stage[k]=temp_machine;
					}
		for(j=0;j<total_machine;j++)

	machine_stage[j].time_given=(machine_stage[j].reliability/total_reliabilit
y)*total_time;

		for(j=0;j<total_job,j++)
			for(k=j;k<total_job,k++)
				if(jobs[j].job_priority<jobs[k].job_priority)
				{
					temp_job=jobs[j];
					jobs[j]=jobs[k];
					jobs[k]=temp_job;
				}

		Assign(stage_count,total_machine);
		Reassign(stage_count,total_machine);

		if(stage_count>0)
			Reallocate(stage_count,total_machine);

		computation(stage_count,total_machine);

		for(int i=0;i<total_machine;i++)
			for(j=0;j<machine_allocation[i];j++)

	machine_record[stage_count][machine_stage[i].index][j]=machine[i][j];

		for(i=0;i<total_machine;i++)
			for(j=0;j<machine_allocation[i];j++)
				machine_priority[stage_count][i][j]=machine[i][j];

		total_m[stage_count]=total_machine;

		}

		for(i=0;i<total_stage;i++)
			output(i,total_m[i]);

		float max=0;

		for(i=0;i<total_job;i++)
			if(max<compute[i].consume_time)
```

```
                        max=compute[i].consume_time;

            cout<<"\n\n";
            for(i=0;i<total_job;i++)
                    if(compute[i].consume_time==max)
                            cout<<"Make span of the process "<<max<<" for
the job "<<job_main[i].a<<"\n";

            cin>>j;
}
void computation(int stage,int total_machine)
{
        int k=0;
        for(int i=0;i<total_machine;i++)
                for(int j=0,j<machine_allocation[i];j++)
                {
                        int index;
                        index=machine[i][j];

                        compute[index].machine[stage]=machine_stage[i].index;

                        if(stage==0)
                                for(k=j;k>=0;k--)

        compute[index].consume_time+=job_main[machine[i][k]].job_time;


        compute[index].time_process[stage]=compute[index].consume_time;
                }
}

void Assign(int stage,int total_machine)
{
        int k=0,m=0;
        float cost=0;

        for(int i=0;i<total_job;i++)
        {
                cost+=jobs[i] job_time;

                if(cost<machine_stage[k].time_given)
                {
                        machine[k][m++]=jobs[i].job_index;
                        ++machine_allocation[k];
                        machine_cost[k]=cost;
                }
                else
                if(cost==machine_stage[k].time_given ||
machine_allocation[k]==0)
                {
                        ++machine_allocation[k];
                        machine_cost[k]=cost,
                        machine[k++][m]=jobs[i].job_index;
                        m=0;
                        cost=0;
                }
                else
                if(cost>machine_stage[k].time_given)
                {
                        if(k<total_machine-1)
                        {
                        m=0;
                        machine[++k][m++]=jobs[i].job_index;
                        ++machine_allocation[k];
                        cost=jobs[i].job_time;
                        machine_cost[k]=cost;
                        }
                        else
                        {
                        machine[k][m++]=jobs[i].job_index;
                        ++machine_allocation[k];
                        machine_cost[k]=cost;
                        }
                }
        }
}

void Reassign(int stage,int total_machine)
{
        float ratio1=0,
        float ratio2=0;

        for(int i=total_machine-1;i>0;i--)
        {
                if((machine_cost[i]>machine_stage[i].time_given) &&
machine_allocation[i]>1)
                {
                        ratio1=(machine_cost[i]-
machine_stage[i].time_given)/machine_stage[i].time_given;

                        float cost;
                        int index;
                        index=machine[i][0];
                        cost=job_main[index].job_time;
```

```
                    ratio2=(cost+machine_cost[i-1]-machine_stage[i-
1].time_given)/machine_stage[i-1].time_given;

                    if(ratio1>ratio2)
                    {
                            machine_cost[i]-=cost;
                            for(int k=1;k<machine_allocation[i];k++)
                                    machine[i][k-1]=machine[i][k];

                            --machine_allocation[i];
                            machine_cost[i-1]+=cost;
                            machine[i-1][machine_allocation[i-1]]=index;
                            ++machine_allocation[i-1];

                    }
            }
        }
}

void Reallocate(int stage,int total_machine)
{
        int finish[SIZE];
        int a,b,c,temp;

        for(int i=0;i<total_job;i++)
                finish[i]=0;

        for(i=0;i<total_machine;i++)
                for(int j=0;j<machine_allocation[i];j++)
                {
                        a=machine[i][j];
                        c=j;

                        for(int k=j+1;k<machine_allocation[i];k++)
                        {

                                b=machine[i][k];


        if(compute[a].consume_time>(compute[b].consume_time+job_main[b].jo
b_time))
                                {
                                                        a=b;
                                                        c=k;
                                }
                        }

                        temp=machine[i][c];
                        for(int m=c-1;m>=j;m--)
                                machine[i][m+1]=machine[i][m];
                        machine[i][j]=temp;


        compute[machine[i][j]].consume_time+=job_main[machine[i][j]].job_tim
e;
                        float time;
                        time=compute[machine[i][j]].consume_time;
                        for(m=j+1;m<machine_allocation[i];m++)
                                if(time>compute[machine[i][m]].consume_time)

        compute[machine[i][m]].consume_time=time;

                }
}

void output(int stage,int total_machine)
{

        cout<<"\n\n-------------------------------"<<"Stage:  "<<stage+1<<"----------
-------------------------\n\n";

        cout<<"Jobs given according to machine priority:\n\n";

        for(int i=0;i<total_machine;i++)
                {
                for(int k=0;k<total_job;k++)
                        if(machine_priority[stage][i][k]!=-1)
                                cout<<"
"<<job_main[machine_priority[stage][i][k]].a;

                cout<<"\n\n";
                }

        cout<<"\n\n--------------------------\n\n";

        for(i=0;i<total_machine;i++)
                {
                cout<<"Machine"<<stage+1<<i+1<<":  ";

                for(int k=0;k<total_job;k++)
                        if(machine_record[stage][i][k]!=-1)
                                cout<<"
"<<job_main[machine_record[stage][i][k]].a;

                cout<<"\n\n";
                }
                cout<<"\n\n--------------------------\n\n";
        for(i=0;i<total_job;i++)
                {
                        cout<<" job "<<job_main[i].a<<":  "<<"  machine:
"<<compute[i].machine[stage]+1<<"  Time consume:
"<<compute[i].time_processs[stage]<<"\n\n";
                }
}
```