

Republic of Iraq

**Ministry of Higher Education and
Scientific Research**

University of Thi-Qar



College of Computer Science and Mathematics

Fuzzy Machine Scheduling Problem

A thesis

Submitted to the council of College of Computer Science and Mathematics /
University of Thi-Qar

in Partial Fulfillment of The Requirements for The Degree
of Master of Science in Mathematics

By

Marwa Mohammed Kassim

Supervised by

Assist. Prof. Dr. Mohammed K. Al-Zuwaini

Assist. Prof. Dr. Shaker K. Ali

2018 A.D.

1439 A.H.



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

{يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنْكُمْ
وَالَّذِينَ أُوتُوا الْعِلْمَ دَرَجَاتٍ}

صدق الله العلي العظيم

سورة المجادلة ﴿11﴾



Dedication

I dedicate my Humble work.

To my father, who did not spare the days for nothing .

To my mother, who offered me with tenderness and love .

And to all my family.

I tell them: you gave me hope and nurture the passion for knowledge.

Finally, each one who taught me to become a sparkle that lights up the road in front of me.

Certification

I certify that this thesis entitled (**Fuzzy Machine Scheduling Problem**), by Marwa Mohammed Kassim, has been prepared under my supervision at the Department of Mathematics, College of Computer Science and Mathematics, University of Thi-Qar, in partial fulfillment of the requirements for the degree of Master of Science in Mathematics.

Signature:

Supervisor: Asist. Prof. Dr. Mohammed Kadhim Z. Al-Zuwaini

Date: \ \ 2018

Signature:

Supervisor: Asist. Prof. Dr. Shaker K. Ali

Date: \ \ 2018

In view of the available recommendation, I forward this thesis for debate by the examining committee.

Signature:

Name: Asist. Prof. Dr. Salam Jasim Majeed

Head of the Mathematics Department.

College of Computer Science and Mathematics , University of Thi-Qar.

Date: \ \ 2018

Committee Certification

We certify that we have read this thesis, entitled (**Fuzzy Machine Scheduling Problem**) and examining committee examined the student in its contents and that in our opinion it is adequate for the partial fulfillment of the requirements for the degree of Master of Science in Mathematics .

Signature:

Name: Dr. Kadhém Mahdi Hashem

Title: Professor

Date: 18 \ 10 \ 2018

(Chairman)

Signature:

Name: Dr. Faez Hasan Ali

Title: Assistant Professor

Date: 18 \ 10 \ 2018

(Member)

Signature:

Name: Dr. Hanan Ali Chachan

Title: Assistant Professor

Date: 18 \ 10 \ 2018

(Member)

Signature:

Name: Dr. Mohammed K. Al Zuwaini

Title: Assistant Professor

Date: 18 \ 10 \ 2018

(Member and Supervisor)

Signature:

Name: Dr. Shaker Kadhim Ali

Title: Assistant Professor

Date: 18 \ 10 \ 2018

(Member and Supervisor)

Approved by University committee of Graduate Studies .

Signature:

Name: Prof. Dr. Akram B. Attar

Dean of College of Computer Science and Mathematics

Date: 18 \ 10 \ 2018

Acknowledgments

I would like to praise **Allah** for all his blessings during my research period.

I will commence by expressing my deepest gratitude and appreciations to my supervisors, namely, **Assist. Prof. Dr. Mohammed K. Al-Zuwaini** and **Assist. Prof. Dr. Shaker K. Ali** for their suggestions of the work, substantial advice, valuable comments, assistance, and encouragement throughout the research.

In addition, I would like to thank all the **staff of College of Computer Science and Mathematics**, especially the members of the Department of Mathematics for their support, and encouragement.

Finally, special appreciations are likewise entitled to **my friends and family** for their guidance and patience during the research.

Marwa

2018

Abstract

In this thesis, the problem of sequencing a set of n jobs on single machine was considered to minimize multiple objectives function (MOF) with fuzzy processing time. The objective is to find the best sequence which give us the approximate solutions (Near-optimal solutions) for scheduling n independent jobs to minimize the objective function consists from a sum of weighted number of early jobs and weighted number of tardy jobs under fuzzy processing time. This problem is denoted by: $(1 / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$.

To resolve it, we proposed (The Average High Ranking) AHR method to obtain a processing time generated from fuzzy processing time, calculate the costs and reach to penalty cost.

Since the complex type of our problem in the normal case (without fuzzy) is Strongly NP-hard, local search methods were used to find approximate solutions: Simulated Annealing, Tabu Search. In practice, we found that the method Simulated Annealing can solve the problem with up to 12000 jobs in 30 seconds and Tabu Search method can solve the problem with up to 12100 jobs within 30 seconds, from the comparison the results we found that the Tabu Search method was best than the Simulated Annealing method.

All of the methods used in this research are: The Average High Ranking algorithm, the local search algorithms Simulated Annealing and Tabu Search has been programmed using MATLAB 7.10.0 (R2014a) software. The Personal computer was of 4 GB RAM with a CPU of 2.5 GHz speed.

List of Contents

No.	Title	Page
	Abstract	I
	List of Contents	II
	List of Symbols	V
	Abbreviations	VIII
	List of Tables	X
	List of Figures	XI
Introduction		
	Introduction	1
Chapter One Machine Scheduling Problems		
1.1	Definition of the Scheduling Problem	5
1.2	Simple Definitions	6
1.3	Types of Machine Scheduling Problem	7
1.3.1	Single Machine Scheduling	8
1.3.2	Parallel Machine Scheduling (P_m)	8
1.3.3	Scheduling of Flow Shop (F_m)	8
1.3.4	Job Shop Scheduling (J_m)	9
1.3.5	Open Shop Scheduling (O_m)	9
1.4	Basic Concepts of Scheduling	10
1.5	The Machine Scheduling Problem Classification	12
1.5.1	Machine Environment (α)	12
1.5.2	Job Characteristics (β)	13
1.5.3	Objective Function (γ)	14
1.5.4	Examples of Scheduling Problem	14
1.6	Restrictions	14
1.6.1	Restrictions on the jobs	15
1.6.2	Restrictions on the Machines	15
1.6.3	Other Restrictions	16
1.7	Important Rules	16
1.8	Important Algorithms	17
1.9	Problem Classes and Computational Complexity	18
1.10	Solution Approaches for Machine Scheduling Problems	20
1.10.1	Complete Enumeration Method	20

1.10.2	Branch and Bound Methods	20
1.10.3	Dynamic Programming	22
1.10.4	Heuristic Methods	22
Chapter Two Fuzzy System		
2.1	The Fuzzy Set	24
2.1.1	The Membership Function (MF)	25
2.1.1.1	Triangular Membership Function	25
2.1.1.2	Trapezoidal Membership Function	26
2.1.1.3	Gaussian Membership Function	27
2.1.2	Fuzzy Logic Operators	28
2.1.3	Fuzzy Rule	29
2.2	Fuzzy Logic (FL)	30
2.2.1	Fuzzification	32
2.2.2	Fuzzy Inference Systems (FIS)	32
2.2.3	Defuzzification	32
2.2.3.1	Centroid of Area (CoA)	33
2.2.3.2	Bisector of Area (BoA)	33
2.2.3.3	Mean of Maximum (MoM)	34
2.2.3.4	Largest of Maxima (LoM) and Smallest of Maxima (SoM) Methods	34
2.3	Mamdani Fuzzy Inference	35
Chapter Three Solving MSP Under Fuzzy Processing Time Using SA Method		
3.1	Introduction	39
3.2	Basic Definitions	42
3.3	The Concept of Fuzzy Processing Time	43
3.4	Formulation of the Problem	44
3.5	Assumption and Notation	45
3.6	Algorithm	46
3.7	Simulated Annealing Heuristics	50
3.8	Computational Experiments of applying SA Algorithm	55
3.8.1	The Problem Test	55
3.8.2	Computational Results	56
Chapter Four Solving MSP Under Fuzzy Processing Time Using TS Method		
4.1	Introduction	59

4.2	Tabu Search	59
4.3	TS Algorithm	61
4.4	Computational Results of LS Algorithms and Comparison	66
4.4.1	The Problem Test	66
4.4.2	Computational Results	66
4.4.3	Comparative Result of Local Search Algorithms	68
4.4.4	Summary of Experimental Evaluation of LS Methods	73
Chapter Five Conclusions and Future Works		
5.1	Conclusions	75
5.2	Future Works	75
References		
References		77

List of Symbols

Symbol	Meaning
α	Machine environment
$\alpha/ \beta / \gamma$	Classification of the problem
β	Job characteristics
β_k	Cooling parameter in annealing terminology
γ	Objective function
–	Precedence constraints
A_i	Average high ranking of the processing time a, b, c
C_i	The completion time of job i
C_{max}	The maximum completion time (makespan)
d_i	Due date of job i
D_j	Distinct due date
E_j	The earliness time of job j
E_{max}	The maximum earliness
$f_G(\mathbf{x})$	Characteristic function of set G
F_j	The flow time of job j

F_m	Flow Shop Scheduling
$G(S_C)$	Value of candidate schedule
$G(S_K)$	Value of schedule constructed at K iteration
$G(S_0)$	Aspiration criterion (Value of best schedule)
h_j	Weighted of early job j
J_m	Job Shop Scheduling for m machines
L_j	The lateness time of job j
L_{max}	The maximum lateness
MF	Membership function
$\mu_G(x)$	Membership function of set G
$N_{h_j E_j}$	Weighted number of early job j
$N_{w_j T_j}$	Weighted number of tardy job j
O_m	Open Shop Scheduling
P_j	Fuzzy Processing time of job i (a_j, b_j, c_j)
P_j	Processing time of job j
P_m	Parallel Machine Scheduling
$P(S_K, S_C)$	Probability of moving from S_k schedule to S_C schedule at K Iteration
r_j	Release date of job j

S_0	Best schedule found so far
S_C	Candidate schedule
S_K	Schedule constructed at K iteration (K= iteration counter)
Sl_j	Slack time of job j
T_j	The tardiness time of job j
T_{max}	Maximum tardiness
V_{max}	Maximum Late Work
w_j	Weighted of tardy job j

Abbreviations

a, b, c	Processing Time of j Job on the Machine in Fuzzy Environment
AHR	Average High Ranking
MSP	Machine Scheduling Prpblem
AI	Artificial Intelligence
BAB	Branch and Bound
BoA	Bisector of Area
CBS	Current Best Sequence
CoA	Centroid of Area
CPU	Central Processing Unit
DP	Dynamic Programming
EDD	Earliest Due date
FIS	Fuzzy Inference System
FL	Fuzzy Logic
IG	Iterated Greedy
JCT	Job Completion Time
JIT	Just In Time
LB	Lower Bound
LoM	Largest of Maxima
LPT	Longest Processing Time

LS	Local Search
M	Number of Machines
MOF	Multiple Objective Function
MoM	Mean of Maximum
MST	Minimum Slack Time
N	Set of Jobs
n	Number of Jobs
NP	Non-deterministic Polynomial
OBS	Overall Best Sequence
OR	Operations Research
PC	Personal Computer
Pmtn	Preemption
Prec	Precedence Constraint
RAM	Random Access Memory
SA	Simulated Annealing
SoM	Smallest of Maxima
SPT	Shortest Processing Time
TS	Tabu Search
UB	Upper Bound
WSPT	Weighted Shortest Processing Time

List of Tables

No.	Title	Page
Chapter Three		
3.1	The initial data for example 3.1	47
3.2	The calculated AHR and Slack time for example 3.1	49
3.3	The calculated penalty cost for example 3.1	50
3.4	The Performance of SA Algorithm for n= 500 jobs	57
3.5	The performance to average of cost and average of time in SA algorithm for $n \in \{10,50,100,500,1000,5000,10000,12000\}$	58
Chapter Four		
4.1	The Performance of TS Algorithm for n = 500 jobs	67
4.2	The Performance to average of cost and average of time in TS algorithm for $n \in \{10,50,100,500,1000,5000,10000,12000,12100\}$	68
4.3	Performance of both LS methods with the best solution for $n \in \{10,50,100,500,1000,5000,10000,12000\}$	69-72
4.4	The summary of Table (4.3) results	73
4.5	Activity of the LS methods	74

List of Figures

No.	Title	Page
Chapter One		
1.1	Classification of machine scheduling problems	7
Chapter Two		
2.1	Triangular membership function	26
2.2	Trapezoidal membership function	27
2.3	Gaussian membership function	28
2.4	Fuzzy logical operations	29
2.5	Range of logical values in Boolean and fuzzy logic: (a) Boolean logic; (b) multi-valued logic	30
2.6	The classical set theory is a subset of the theory of fuzzy sets	31
2.7	Fuzzy Inference System	31
2.8	Defuzzification methods CoA, BoA, MoM, LoM, and SoM.	34
2.9	A two input, two rule Mamdani FIS with fuzzy inputs	36
2.10	Fuzzy rule	37
Chapter Three		
3.1	Membership function	48
3.2	Output membership function	49
Chapter Four		
4.1	Illustrative approach for finding the sequences of S_1 in example 4.1 using adjacent pairwise interchange.	63
4.2	Illustrative approach for finding the sequences of S_2 in example 4.1 using adjacent pairwise interchange.	64
4.3	Illustrative approach for finding the sequences of S_3 in example 4.1 using adjacent pairwise interchange.	65
4.4	Illustrative approach for finding the sequences of S_4 in example 4.1 using adjacent pairwise interchange.	66

Introduction

Operations Research (OR) is one of the wide spreading scientific methods that are currently used for solving practical problems, that has witnessed major successes in diverse areas of research and development. OR has arisen during the Second World War when the military administration in Britain gave a team of scientists and researchers the task of studying the technological and strategic problems, especially in terrestrial and voyage defense. The aim of this team was to find the best use of resources in addition to study the action of the new kind of bombs. The creation of this group was regarded as the birth to what is now called OR in 1941 [1]. Scheduling problems have been under study for a long time, but the first theoretical development in scheduling was made by Johnson [2]. Two other important results followed shortly by Jackson [3] and Smith [4]. Methods which guarantee optimal solutions to machine scheduling problems are: Dynamic Programming (DP) methods, which were firstly used by Held and Karp [5], Branch and Bound (BAB) method which was used for the first time in scheduling problems by Lomnicki [6], and Ignall and Schrage [7]. Recently, many problems in production scheduling often solved by applying Branch and Bound (BAB) to obtain optimal solutions. As a result of the difficulties in finding the problems optimal schedule within a reasonable time; heuristic methods (methods that do not guarantee optimality) were used to find approximate solution. Graham [8] was a pioneer in investigating the theoretical analysis of heuristic methods. Near optimal solutions are commonly resulting from local search algorithm (LSM) in machine scheduling problem (MSP), which includes a genetic algorithm and a particle swarm optimization that were very successful in generating high-quality solutions.

Numerous present researchs on the scheduling problem take into account single objective optimization. In reality, nonetheless, certain circumstances wherein a decision maker can assess the schedules regarding many measures are occurring.

A number of trending multi-criteria scheduling researchs are coping with the single machine [9]. Generally, methodologies to deal with the problem of multi-criteria are of two distinct methods; in the first one there is a hierarchical tactic, while in the second one there is a simultaneous tactic.

To explain more about these tactics, a hierarchical methodology takes into perspective a primary and a secondary criterion, the problem is then deals with the minimization of the primary criterion despite the fact that breaking ties that supports the schedule which has minimum value of secondary criterion. On the other hand, a simultaneous methodology which consists of two approaches, the former usually creates all effective schedules and chooses the finest value of the composite objective function for these criteria, whereas, the latter is concerned mainly with finding the sum for these objectives [10].

The problem $1/ \sum_{j=1}^n N_{h_j} E_j + \sum_{j=1}^n N_{w_j} T_j$ will be studied in this thesis, to outline the significance of this problem is by reducing weighted number for early jobs and weighted number for tardy jobs, this has been addressed to storage problems if we take into account some of the jobs needed to store a complex and expensive operation which is a serious problem when jobs are perishable, where the force short validity can be outlined such as foodstuffs, chemicals, medicines and other validity. The problem also aims to fix damaged products resulting from the time of completion before the due date is agreed by both the consumer and the manufacturer of the products and reduce the number of late jobs date delivered to the consumer and the lateness of the return on the bad

reputation of the plant may result in fines the plant that are because of lateness in the completion of jobs. However, this problem consists of two sub problems. In this thesis, we will consider the sum of problems $1/ \sum_{j=1}^n N_{h_j E_j}$ and $1/ \sum_{j=1}^n N_{w_j T_j}$ under fuzzy processing time.

The utmost of the research concerning bicriteria scheduling in literature, regards a single machine with minimum couples of criteria. For the problem $1/ \sum N_{h_j E_j}$ minimizing summation of weighted number of early jobs which is NP-hard [11].

At this point, a review of the researchers who studied the problem $\sum_{j=1}^n N_{h_j E_j}$, Soroush, (2007) [11] found an optimal sequence that reduces the predictable summation of both tardy and early weighted numbers of jobs; he approached was NP-hard. Kacem, et al., (2015) [12] considered maximization for weighted numbers of early job on a single machine using constraints of non-availability. Al- Zuwaini, and Taher, (2018) [13] studied the function sum of weighted number of early jobs and total weighted of completion time.

For the problem $1/ \sum_{j=1}^n N_{w_j T_j}$ minimizing summation of weighted number of tardy jobs which is NP-hard, this criteria studied with other criteria by the former researchers which are mentioned above in [11]. Also, Foster and Boxma, (1986) [14] found a sequence of jobs which can reduce the weighted number of tardy jobs. Peha, (1995) [15] studied the minimization of weighted completion and weighted number of tardy jobs. It is fair to mention the fact that this problem was intensively studied by many researchers worldwide [16-24].

This thesis consists of five chapters:

Chapter one describes the problem of machine scheduling including traditional scheduling problem containing machine environment, job

characteristics and the objective function. The well-known methods for finding the exact solution such as BAB method and DP are discussed.

Chapter two presents the fuzzy system including fuzzy set, types of membership function, fuzzy logic operators, fuzzy rule, fuzzy logic, defuzzification, and Mamdani fuzzy inference.

Chapter three shows the Average High Ranking (AHR) and Simulated Annealing (SA) to find near-optimal solution.

Chapter four shows fuzzy processing time and Tabu search (TS) to calculate the approximate solution.

Chapter five provides conclusions and future works.

Chapter One

Machine Scheduling Problem

Chapter One

Machine Scheduling Problems

1.1 Definition of the Scheduling Problem

Although there is a plenty of definitions relating to machine-scheduling, Baker has provided a modest definition to easily grasp the term, "the allocation of machines over time to perform a collection of jobs" [25]. Machines and jobs are mentioned in the definition by resources and tasks correspondingly and they are provided with different customs.

A personal computer (PC) can work as a machine and the software that run by the particular PC can be set as jobs. One may also regard machines as hospital apparatus or devices in the healthcare sector and jobs can be set as the number of patients being cured using the specialized equipment [26].

To allocate machines for jobs so that all jobs are accomplished with the obligatory restrictions is scheduling. Finding the optimal order of processing for jobs on each machine is a problem and the solution relying on minimizing the known objective function [27]. In classical scheduling theory mainly there are two restrictions: restriction on jobs, restriction on machine [28]. Each machine is able to process a single job specified within a given timescale adding to the necessity that every job ought to get handled using a single machine. The schedule is considered to be practical if it fulfills the numerous necessities linking to the definite problem category. The problem category is identified using the environment of the machine and the features of a job also with standard optimal conditions.

1.2 Simple Definitions

Definition (1) [29]: The term "optimize" in a multi-objective decision making problem refers to a solution around which there is no way of improving any objective without worsening at least one other objective.

Definition (2) [29]: suppose we have a problem P , any schedule $\pi \in \Pi$ (where Π is the set of all schedules) is said to be feasible if it satisfies the constraints of the problem P .

Definition (3) [30]: A schedule S is said to be efficient if there does not exist another schedule S' satisfying $f_i(S') \leq f_i(S), i = 1, 2, \dots, n$, with at least one of the above holding as a strict inequality. Otherwise S is said to be dominated by S' .

Definition (4) [31]: A measure of performance is said to be regular if it is a non-decreasing function of job completion times and the scheduling objective is to minimize the performance measure.

Example of regular measures are job mean flow time ($\bar{F}$), schedule make span (C_{max}) and tardiness based performance measures.

Definition (5) [31]: A non-regular performance measure is usually not monotone function of the job completion times.

An example of such a measure is job earliness.

Definition (6) [30]: The function $F(f, g)$ is said to be non-decreasing in both argument, if for any pair of outcome value (x, y) of the function

f and g , we have $F(x, y) \leq F(x + A, Y + B)$ for each pair of non-negative value A and B .

1.3 Types of Machine Scheduling Problem

Due to the large variety of machine scheduling problem, several classification schemes have been proposed based on different dimensions [32]. An example of such a dimension is the number of the available machines and the way how these machines are organized [33]. Then one machine sequencing problem is the simplest one: all jobs have to be processed on single machine and no two jobs can be processed simultaneously. A more detailed description of each problem class is given in the following subsections [34].

By classifying scheduling problem according to the number of machines and jobs, ordering becomes as in Figure (1.1).

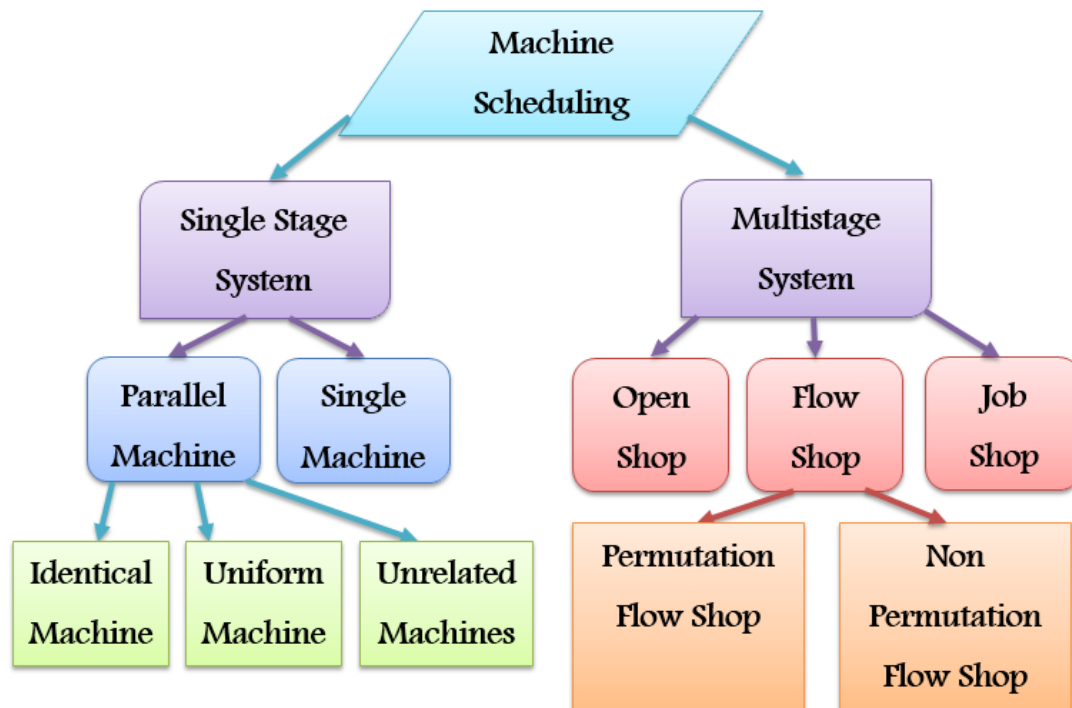


Figure (1.1): Classification of machine scheduling problems

1.3.1 Single Machine Scheduling

The single machine scheduling problem involves scheduling a set of jobs to a single resource [35]. This is accomplished by determining a sequence that includes each job and it assigns the jobs to its source, Each job can give a priority, ready time, processing time, and due date [36]. The value of the performance measure can be computed on the base of this information and the sequence of jobs [37]. This problem grows in complicity at an exponential rate as the number of jobs to be scheduled increases [38].

1.3.2 Parallel Machine Scheduling (P_m)

Parallel Machine includes scheduling a group of two or even more machines that are operating in parallel with each other [39]. The machines implement the same processes and could operate at identical place. There are three cases [40]:

1. **Identical Parallel Machine**, in which every time of processing doesn't depend on the machine that performs the particular job.
2. **Uniform Parallel Machine**, speeds are different for the operating machines.
3. **Unrelated Parallel Machine**, in which the machine assignment determines the processing time of any job.

1.3.3 Scheduling of Flow Shop (F_m)

A flow shop scheduling comprises of at least two and maybe more machines with a job set that should be handled on every single machine. This organization is commonly entitled as flow shop since the products flow end to end with a definite unidirectional route [41]. Every product should be handled by every machine keeping orderly fashion, such as the

first-machine with number one, second-machine with number two and m-machine with number m. The processing time relating to any job is contrasting from a machine to another and also on each machine is contrasting from a job to another. Furthermore, this form can be categorized with two categories using n job sequence with every machine, the categories are [42]:

1. **Permutation flow shop problem**, n job sequencing with every machine are identical. ((Passing is not allowed)).
2. **Non-Permutation flow shop problem**, n job sequencing with every machine are not the same. ((Passing is allowed)).

1.3.4 Job Shop Scheduling (J_m) [43]

A job shop can be made up of at least two or more machine that can perform precise processes and it comprises of a set of jobs that should be handled by some or the entire machines. Job shop differs from flow shop in that fixed path in which products should follow through the system doesn't exist. Consequently, operation order is varied. This type of arrangement is normally employed when the product diversity is huge and the product volume is small.

1.3.5 Open Shop Scheduling (O_m)

This is a further wide-ranging example of J_m problem of scheduling. A job of number n comprising of steps of number n' are to be handled by m' machines. Since a step sequences within every job should be defined as well as the jobs processing schedule. Every job sequence inside of a machine will be dissimilar this results for an optimum sequence in every job is likewise a share of problem [44].

1.4 Basic Concepts of Scheduling

The performance criteria are concentrated on without elaborating on the machine environments. Several significant symbolizations are presented ahead. Jobs of number n are supposed to get scheduled on machine sets that are constantly presenting from time equal to zero ahead and is able to manage a single job timewise only which are given as $j_1, j_2, \dots, j_n$.

We will focus on state the symbolization that is employed to a single machine jobs j_i ($i = 1, \dots, n$) has:

Preemption (pmtn) [45]: Preemption (or job-splitting) is allowed during the processing of a job, if the processing of the job can be interrupted at any time (preempt) and resumed at a later time, even on a different machine. The amount of processing already done on the preempted job is not lost.

Machine (m) [45]: A machine is available to execute jobs and tasks. Different machine environments exist, such as single machine and parallel machines.

Job (j) [45]: A job can be made up of any number of tasks. It is easy to think of a job as making a product, and each task as an activity that contributes to making that product, such as a painting task, assembling task and so on.

Release date (r_j) [46]: Also known as ready time, the point of time that job arrives at the machine and thus processing starts.

Processing time (p_{ij}) [47]: The processing time of job j on machine i . The subscript i is omitted if job j is only to be processed on one given machine.

Weight (w_j) [35]: Denoting the importance of a job j relative to another job.

Due date (d_j) [46]: The date when the jobs should ideally be completed, the completion of job after its due date is allowed, but a penalty is incurred. When the due date absolutely must be met, it's referred to as deadline ($\overline{d_j}$) and when due date is constant for all jobs, then called common due date.

Now for given sequence of jobs the following can be computed for job j :

1. $C_j = \sum_{k=1}^j p_k$ (Completion time).

2. $F_j = C_j - r_j$ (Flow time).

3. $T_j = \max\{C_j - d_j, 0\}$ (Tardiness).

4. $E_j = \max\{d_j - C_j, 0\}$ (Earliness).

5. $L_j = C_j - d_j$ (Lateness).

6. Unit penalty

- i. $N_{E_j} = \begin{cases} 1 & \text{if } d_j > C_j \\ 0 & \text{if } d_j \leq C_j \end{cases}$

- ii. $N_{T_j} = \begin{cases} 1 & \text{if } C_j > d_j \\ 0 & \text{if } C_j \leq d_j \end{cases}$

7. $V_{max}(\pi) = \max_{j \in \pi} \{V_j\}$ (Max. of late work).

8. $C_{max}(\pi) = \max_{j \in \pi} \{C_j\}$ (Max. of completion time).

9. $L_{max}(\pi) = \max_{j \in \pi} \{L_j\}$ (Max. of lateness).

10. $E_{max}(\pi) = \max_{j \in \pi} \{E_j\}$ (Max. of earliness).

11. $T_{max}(\pi) = \max_{j \in \pi} \{T_j\}$ (Max. of tardiness).

12. N_{E_j} number of early job.

13. N_{T_j} number of tardy job.

14. $\sum_{j=1}^n w_j C_j (\pi)$ total weighted completion time.

15. $\sum_{j=1}^n h_j E_j (\pi)$ total weighted earliness.

16. $\sum_{j=1}^n w_j T_j (\pi)$ total weighted tardiness.

17. $\sum_{j=1}^n N_{h_j E_j}(\pi)$ total weighted number of early jobs.

18. $\sum_{j=1}^n N_{w_j T_j}(\pi)$ total weighted number of tardy jobs.

All these criteria, are regular except for E_{max} , N_{E_j} and $\sum_{j=1}^n N_{h_j E_j}$ are irregular, i.e. objective function values can't be reduced even when idle time is inserted in scheduling.

Note that the scheduling problems considered in this thesis are characterized by three sets, set of n jobs, (m) machines also an additional constraints [7].

1.5 The Machine Scheduling Problem Classification [48]

The Machine Scheduling problems (MSP) are frequently categorized as stated by Graham's classification (Pinedo in 1995; Brucker in 1998; Lueng in 2004; Baptiste in 2001). Graham's classification consists of three field notations ($\alpha/\beta/\gamma$) the α labels environment of a machine, i.e., the elementary symbolizations of machine environments are a single-machine or multiple machines, identical or different machines. β Delivers specifics for restrictions and processing characteristics. γ Covers the objective function that is required to get reduced.

1.5.1 Machine Environment (α) [40]

First field $\alpha = \alpha_1 . \alpha_2$ describes the machine environment parameters where α_1 and α_2 are interpreted as follows:

$$\alpha_1 \in \{\emptyset, F, J, O, P, Q, R\}$$

$$\alpha_1 = \emptyset \quad : \text{for single machine.}$$

$$\alpha_1 = F \quad : \text{in flow shop.}$$

$$\alpha_1 = J \quad : \text{in job shop.}$$

$$\alpha_1 = O \quad : \text{in open shop.}$$

$$\alpha_1 = P \quad : \text{for identical parallel machine problem.}$$

$\alpha_1 = Q$: for uniform parallel machine problem.

$\alpha_1 = R$: for unrelated parallel machine problem.

$\alpha_2 \in \{\emptyset, m\}$

$\alpha_2 = \emptyset$: the machine numbers that are supposed to vary.

$\alpha_2 = m$: the machine numbers is like m ($m \geq 2$).

1.5.2 Job Characteristics (β) [48]

The second field β is a list of the variables and the restrictions indicating job characteristics and jobs j ($j=1, \dots, n$). The following parameters are regarded:

r_j : release dates for job j .

d_j : due dates for job j .

S_f : If jobs possess a specific setup dependent sequence times it is used.

prmp: By repeating transport of input data, the preemption for jobs may convey a great cost. Thus this parameter won't be considered for our analysis and computations since it will probably be very sophisticated in our modeling.

prec: If users are able to segment an enormous job into tiny fractions and identifies the necessary order for job accomplishment, then precedence restrictions could be involved.

$d_j = d$: refers to matching due dates (common due dates).

$p_j = p$: refers to matching time of processing.

1.5.3 Objective Function (γ) [48]

The third field γ determines the objective function or the optimality criterion; that is the value we wish to optimize. The parameter $\gamma \in \{f_{max}, \sum_{j=1}^n f_j\}$ is defined as follows:

$$f_{max} \in \{C_{max}, L_{max}, T_{max}, E_{max}, V_{max}, V_{max}^w, T_{max}^w, L_{max}^h\}$$

or

$$\sum_{j=1}^n f_j \in \left\{ \sum_{j=1}^n C_j, \sum_{j=1}^n w_j C_j, \sum_{j=1}^n E_j, \sum_{j=1}^n N_{E_j}, \sum_{j=1}^n N_{T_j}, \sum_{j=1}^n h_j E_j, \sum_{j=1}^n w_j T_j \right\}$$

Or f_j is equal to sum or weighted sum for two or more of objective function.

1.5.4 Examples of Scheduling Problem

1. $1/r_j / \sum_{j=1}^n N_{T_j} + \sum_{j=1}^n T_j$ it includes release dates with a single-machine for the purpose of minimizing sum number of tardy jobs and total tardiness.
2. $1/r_j, p_j = 1 / \sum_{j=1}^n N_{h_j E_j}$ refers to reducing weighted number of early tasks by single-machines subject to non-trivial release dates and unit processing times.
3. $1/S_f / \sum_{j=1}^n w_j T_j$ is the problem of reducing total weighted tardiness with single-machine where the jobs are divided into $\mathbf{f}$ families.

1.6 Restrictions [49]

In this section, a number of restrictions on the jobs, on the machines, and other restrictions are listed.

1.6.1 Restrictions on the jobs [49]

J_1 : The set of jobs is fixed and known.

J_2 : All jobs are available at the same time and are independent. Sometimes, every job (j) is regarded using due dates d_j with release-date r_j (all data being integer) where $j = 1, 2, \dots, n$.

J_3 : Each job has the same degree of importance. This assumption is sometimes, relaxed (i.e. job j has a w_j weight representing (j) job significance).

J_4 : Any operation started is not interrupted by other operations and continues its completion. This assumption is sometimes relaxed (i.e. job splitting is allowed).

J_5 : Each job can be in one of three states: waiting for the next machine; being operated on by a machine; or having passed its last machine.

J_6 : Jobs are handled using a single-machine simultaneously.

1.6.2 Restrictions on the Machines [50]

M_1 : The machine numbers are identified and stays constant.

M_2 : All machines should be available at similar instance and are independent of each other.

M_3 : At each instant, each machine can wait for the next job, or process a job or has finished its last job.

M_4 : All machines are equally important.

M_5 : Each machine has to process all jobs assigned to it.

M_6 : Machines are not able to handle more than a single job simultaneously.

1.6.3 Other Restrictions [51]

- a) The technical restrictions should be earlier identified and unchallengeable.
- b) Randomness doesn't exist.
- c) The following parameters should be identified and remain constant.
 - i. Times of readiness.
 - ii. Fuzzy processing time.
 - iii. Any particular quantities necessary for problem definition.

1.7 Important Rules

1. Shortest processing time (**SPT**) rule, in which sequencing jobs in an increasing order for related processing times p_j . This rule solves the problem $1/ \sum_{j=1}^n C_j$ [4].
2. Longest Processing Time (**LPT**) rule, as of Graham's specifications the LPT algorithm makes two ordered functions, it categories information making sure of $p_{n-1} \geq p_n$ where $n = (2, 3, 4, \dots)$ at first after this function is done properly a second function will allocate jobs in sequence with the machine that is marginally loaded [8].
3. Earliest Due Date (**EDD**) rule, means job sequencing with (d_j) due dates with an increasing order which, in turn, solves the problem $1/ L_{max}$. This rule also minimizes T_{max} for the $1/ T_{max}$ problem [3].
4. Minimum Slack Time (**MST**) method, which represents job sequencing with an increasing manner relating to their slack time $s_j = d_j - p_j$. In single machine environment which solves $1/ E_{max}$ problem [49].

5. The Weighted Shortest Processing Time (**WSPT**) rule, refers to job sequencing with an increasing manner of p_j/w_j which solves $1 / \sum_{j=1}^n w_j C_j$ problem [4].

1.8 Important Algorithms:

- i. Huang and Yang Algorithm (H.Y.A) [52]: Solved the $1 / \sum_{j=1}^n N_{E_j}$ problem (the number of early jobs).

A practical schedule mainly consists of two sets Q and E . set Q follows Set E . Therefore, Set E jobs should take the urgency with respect to Set Q jobs. This is commonly done in four steps are follows:

Step 1

All jobs are to be placed within Set Q , assuming Set $E = \phi$. Then they are sequenced by employing the method of (**MST**).

Step 2

In Set Q , locate the last early job, be it as $(J_{[k]})$, in the present schedule containing both Sets Q and E . When Set Q jobs are late, goto Step 4.

Step 3

$(J_{[k]}, J_{[k+1]}, \dots, J_{[n]})$ Is called a partial sequence, in this step, locating jobs by this sequence using (**LPT**) is mainly done, in addition, this sequence is to be moved to the (E) set than could be sequenced with arbitrary order. Return to Step 2.

Step 4

Create an Optimal Schedule with employing Set Q following Set E .

- ii. Moor Algorithm (M.A) [54]: Solves the $1 / \sum_{j=1}^n N_{T_j}$ problem (the number of tardy jobs) in $O(n \log n)$ time. This algorithm repeatedly adds jobs in (**EDD**) order to the end of partial schedule of on time jobs. If the addition of job j results in this job being complete after

time d_j , then a job in the partial schedule with the largest processing time is removed and declared late. This is commonly done in four steps are follows:

Step 1

Sequence the jobs according to the **EDD** rule to find the current sequence $(J_{i(1)}, J_{i(2)}, \dots, J_{i(n)})$ such that $d_{i(k)} \leq d_{i(k+1)}$ for $k = 1, 2, \dots, n-1$.

Step 2

Find the first tardy job, say $J_{i(1)}$, in the current sequence. If no such job is found, go to Step 4.

Step 3

Find the job in the sequence $(J_{i(1)}, J_{i(2)}, \dots, J_{i(n)})$ with the largest processing time and reject this from the current sequence. Return to Step 2 with a current sequence one shorter than before.

Step 4

From an optimal schedule by taking the current sequence and appending to it the rejected jobs, which may be sequenced in any order.

1.9 Problem Classes and Computational Complexity

Some problems are polynomials- solvable. That means there exists an algorithm which requires a computing effort that could be shown to grow as a polynomial in the size of the problem. Other problems can only be solved by an enumerative algorithm which, in essence, has an exponential time complexity [28]. The problems that have known polynomials algorithms are said to be well solved. The complexity theory is concerned with the decision problem instead of optimization problem. A decision problem is a question to which the

answer is either "yes" or "no". The question is: Does a solution exist a solution with an objective function value less than or equal to fixed value k ? (we transform the optimization problem into a finite series of decision problems by varying the value k).

Decision problem can be classified into three classes:

1- The class **P** (Polynomial) which contains all decision problems that are polynomial solvable.

2- The class **NP** (Non-deterministic polynomial) which contains decision problems which can be easily verified whether a given solution is "yes" or "no" answer. It is clear that the class **P** is a subclass from the class **NP**.

The scheduling problem that is to be consider with in this thesis can be solved by non-deterministic algorithm and thus is a member of **NP**. Cook [55] proved that there are hardest problems in **NP**; such problems are called **NP-Complete**.

3- The class open which contains all decision problems that did not prove to be neither **P** nor **NP**, for instance the problem

$1/d_j = d / \sum_{j=1}^n (w_j E_j + w'_j T_j)$ is open [40].

A problem **P** is **NP**-complete if the existence of polynomial algorithm.

for **P** implies the existence of a polynomial algorithm for any problem in **NP** (i.e. **P** = **NP**). The location of the border separating the easy problem (in **P**) and the hard ones (in **NP-Complete**) has been under wide investigation of many researches, and it turns out that a minor change in the value of an easy problem parameters often transforms this problem into a hard one. An optimization problem is called **NP**-hard if the association decision problem is **NP**-Complete. Since the computation time needed to solve a scheduling problem is very important, recent development in the theory of computational complexity applied to

machine scheduling problems has aroused the interest of many researchers [56].

1.10 Solution Approaches for Machine Scheduling Problems

In this subsection, a discussion is carried out for the most well-known methods that have been used to solve machine scheduling problems. Enumerative approaches (Branch and Bound (**BAB**) and Dynamic programming (**DP**)) and heuristic approaches are considered for solving the machine scheduling problems [1].

1.10.1 Complete Enumeration Method

Enumeration method generate schedules one by one, to find an optimal solution and list all possible schedules and then eliminate the non-optimal schedule from the list, leaving those which are optimal. Clearly, searching for an optimal schedule among all possible schedules using complete enumeration is not appropriate even for problems of small size. Thus the complete enumeration method (CEM) may be rejected immediately [1].

1.10.2 Branch and Bound Methods

Branch and Bound (BAB) methods are enumeration techniques which can find an optimal solution by systemically examining subsets of feasible solutions. These methods are usually described by means of search tree with nodes that are corresponding to these subsets. BAB methods were applied to scheduling problems by Lgnall and Schrage [7] and Lomnickiz [6]. BAB finds a schedule π^* by implicit enumeration all a schedule $\pi \in \Pi$ through examination of smaller subsets of the set of feasible solutions Π . These subsets can be treated as sets of solutions of

corresponding sub problems of the original problem. Abdul-Razaq (1987) applied the BAB method by the following procedures [49]:

i. The branching procedure

This describes the method to partition a subset of possible solutions. These subsets can be treated as a set of solution of corresponding sub-problems of the original problem.

ii. The bounding procedure

This indicates how to calculate a lower bound (LB) on the optimal solution value for each sub-problem generated in the branching process.

iii. The search strategy

This strategy describes the method of choosing a node of the search tree to branch from it; usually branch from the node with the smallest LB among the recently created nodes.

A general description of the BAB methods will now be given [51].

The set of all possible schedules is divided up into disjoint subsets, each of which contains more than one possible schedule. For each subset LB is calculated which is the cost of the sequenced jobs (depending on the objective function) and the cost of the unsequenced jobs (depending on the derived LB). If the LB of this subset is greater than or equal to the upper bound (UB), then this subset is ignored since any subset with value less than UB can only exist in the remaining subsets. These remaining subsets have to be considered one at a time. One of these subsets is chosen, according to some search strategy, from which to branch. These subsets are then divided (as above) into smaller disjoint subsets. As soon as one of these subsets contains one element only, a complete sequence of the jobs should exist, this sequence is evaluated and if its value is less than the current UB, this UB is then adjusted accordingly. The procedure is then repeated until all subsets (nodes) have been considered. The UB at the end of this BAB procedure is the optimal

solution for the particular problem, depending on the particular problem a node of the search tree could also be eliminated by dominance properties or feasibility conditions developed for the given problem.

1.10.3 Dynamic Programming

Dynamic Programming (DP) method is an implicit enumeration technique which used for machine scheduling problems for the first time by Held and Karp [5]. It is clear that the fundamentals of DP have been elaborated by Bellman in the 1950's. The notion (dynamic programming) is slightly misleading, but generally accepted. A better description would be (recursive) or (multistage) optimization, since it interprets an optimization problem as a multistage decision processes. The problem is divided into a number of stages, and at each stage the solution is derived from recurrence relation from the solutions of the preceding stages [28]. This means that the problem is divided into a number of stages, and at each stage a decision is required which impacts on the decisions to be made in later stages.

1.10.4 Heuristic Methods

It is clear that, to solve a scheduling problem one tends to use implicit enumerative approaches to find an optimal solution. However, these approaches have two disadvantages. Firstly, they are mathematically complex and thus a lot of time to be invested. Secondly, when it concerns an NP-hard problem, the computational requirements are enormous for large sized problem, to avoid these drawbacks can appeal to heuristics. Reeves [57] defined the heuristic method as follows: A heuristic is a technique which seeks good (i.e. near optimal) solution at a reasonable computational cost without being able to guarantee either

feasibility or optimality, or even in many cases to state how close to optimality a particular feasible solution.

In recent years, the improvement in heuristic methods has become under the name of Local Search Methods (LSM) or Meta heuristic methods.

Chapter Two

Fuzzy System

Chapter two

Fuzzy System

2.1 The Fuzzy Set

A fuzzy set can be defined as a set with fuzzy boundaries. Let X be the universe of discourse and its elements be represented as x . In classical set theory, crisp set G of X is identify as function $f_G(x)$ called the Characteristic function of G [58], s.t.

$$f_G(x): X \rightarrow \{0, 1\}$$

Where:

$$f_G(x) = \begin{cases} 1, & \text{if } x \in G \\ 0, & \text{if } x \notin G \end{cases} \quad (2.1)$$

This set defines universe X to a set of two elements. For any element x of universe X , characteristic function $f_G(x)$ is equal to 0 if x is not an element of G and is equal to 1 if x is an element of set G .

In the fuzzy theory, fuzzy set G of universe X is defined by function $f_G(x)$ called the membership function of set G [59], s.t.

$$f_G(x): X \rightarrow [0, 1] \quad (2.2)$$

Where

$$f_G(x) = 1 \text{ if } x \text{ is exactly in } G$$

$$f_G(x) = 0 \text{ if } x \text{ is not in } G$$

$$0 < f_G(x) < 1 \text{ if } x \text{ is partly in } G$$

which set makes a range of possible choices. For any element x of universe X , membership function $\mu_G(x)$ equals the degree to that x is an element of set G .

A value in $[0, 1]$ represents the degree of membership which called membership value of element x in set G [60].

2.1.1 The Membership Function (MF)

Membership functions have different forms such as Gaussian, trapezoidal, triangular, piecewise linear and singleton. The most common types of membership functions are trapezoidal, Gaussian, and triangular shapes. The type of the MF can be chosen according to the user experience [61].

2.1.1.1 Triangular Membership function

The first kind of membership functions is triangular. These functions of $f_G(x)$ that relies mainly upon the scalar constraint "a" defined by lower limit, and "b" defined by upper limit, also $a < m < b$ [62].

$$\mu_G(x) = \begin{cases} 0 & \text{if } x \leq a \text{ or } x \geq b \\ \frac{x-a}{m-a} & \text{if } a < x \leq m \\ \frac{b-x}{b-m} & \text{if } m < x < b \end{cases} \quad (2.3)$$

Figure (2.1) show the diagram of Triangular MF, as in relation (2.3).

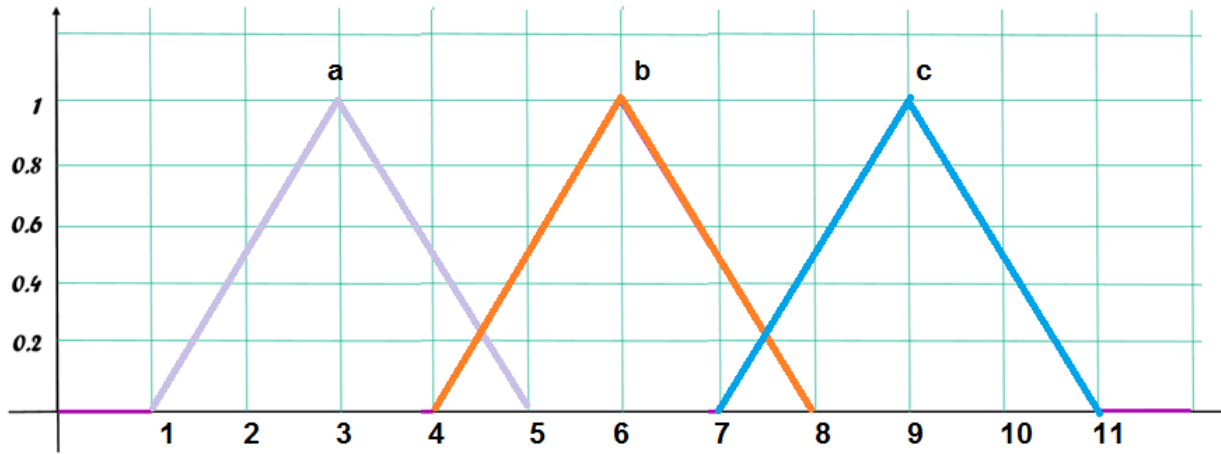


Figure (2.1): Triangular membership function [63].

2.1.1.2 Trapezoidal Membership function [62]

The third kind of membership functions is the trapezoidal function which represents some function of $\mu_G(x)$ that depends on a lower limit a , an upper limit d , a lower support limit b , and an upper support limit c , where $a < b < c < d$.

$$\mu_G(x) = \begin{cases} 0 & \text{if } (x < a) \text{ or } (x > d) \\ \frac{x-a}{b-a} & \text{if } a \leq x \leq b \\ 1 & \text{if } b \leq x \leq c \\ \frac{d-x}{d-c} & \text{if } c \leq x \leq d \end{cases} \quad (2.4)$$

Figure (2.2) shows the diagram of Trapezoidal MF, as in relation (2.4).

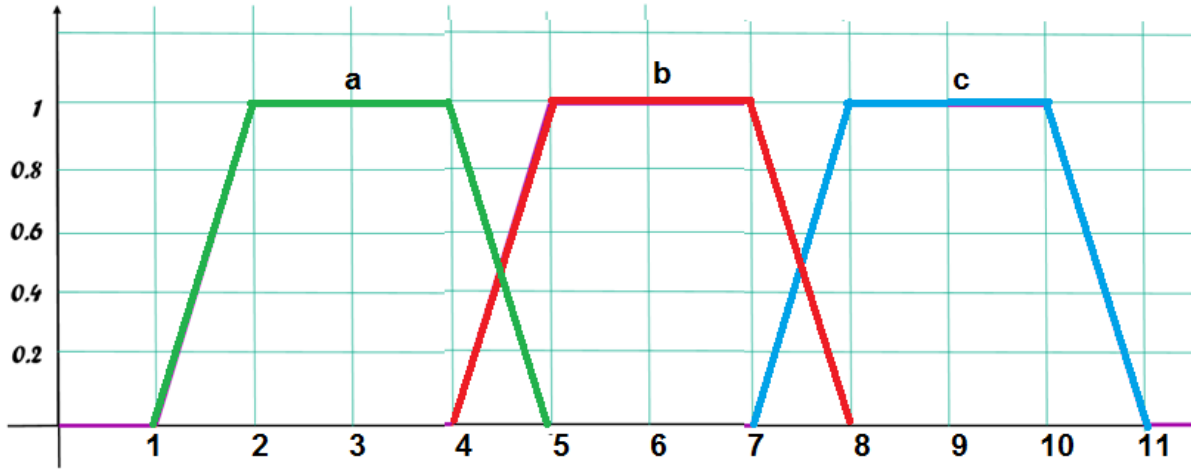


Figure (2.2): Trapezoidal membership function [59].

2.1.1.3 Gaussian Membership Function

The second kind of fuzzy membership functions that is frequently used to represent ambiguous, linguistic terms is the Gaussian which is given by:

$$\mu_{G_j}(x) = \exp\left(-\frac{(C_j - x)^2}{2\sigma_j^2}\right) \quad (2.5)$$

Where σ_j and C_j are the width and the center of the j^{th} fuzzy set G_j , respectively.

Figure (2.3) show the diagram of Triangular MF [64], as in relation (2.5).

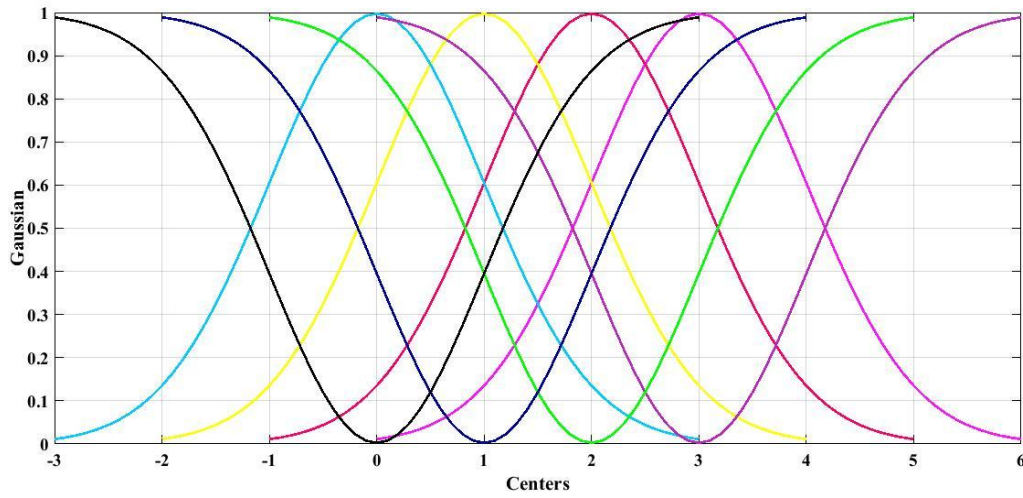


Figure (2.3): Gaussian membership function [63].

2.1.2 Fuzzy Logic Operators [65]

There are three major standard fuzzy set operations: OR (disjunction or union), AND (conjunction or intersection) and NOT (negation or complementation).

$$\begin{aligned}
 \mu_{G \cup H}(x) &= \max(\mu_G(x), \mu_H(x)) \\
 \mu_{G \cap H}(x) &= \min(\mu_G(x), \mu_H(x)) \\
 \mu_{\bar{G}}(x) &= 1 - \mu_G(x)
 \end{aligned}
 \tag{2.6}$$

Figure (2.4) shows the diagram for each logical operation, as in relation (2.6).

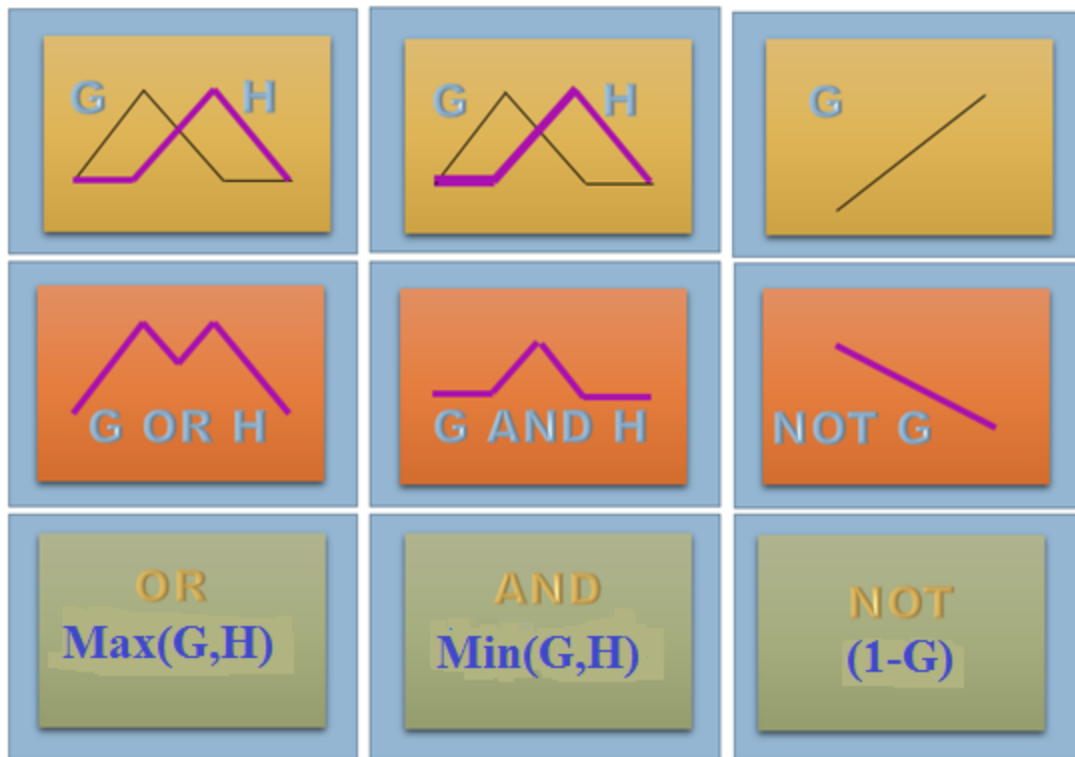


Figure (2.4): Fuzzy logical operations [58].

2.1.3 Fuzzy Rule [66]

This rule of fuzzy logic can be described in terms of a certain conditional statement that can have the following form:

IF x is G
 THEN y is H

Where G and H are linguistic values, and x and y are linguistic variables defined by fuzzy sets on the universes of discourses x and y, respectively.

We can describe fuzzy sets and fuzzy operators the subjects and verbs of fuzzy logic. But to say anything useful we must make complete sentences. Conditional statements IF...THEN rules are the things that make fuzzy logic useful.

IF...THEN rule connections a condition explained using linguistic variables and fuzzy sets to an output. The IF part is usually used to take knowledge by using the elastic conditions, and the THEN part can be used to give the output in linguistic variable form. IF-THEN rule is commonly used by the fuzzy inference system to calculate the degree to which the input data matches the condition of a rule.

2.2 Fuzzy Logic (FL) [58]

Fuzzy logic which introduced by Zadeh in 1965 is defined as a set of mathematical ideas for knowledge representation based on degrees of membership rather than on crisp membership of the classical theory of logic.

Unlike binary Boolean logic, Fuzzy logic is multi-valued. It deals with degrees of membership and degrees of truth. It uses the continuum of logical values between 0 (completely false) and 1 (completely true). Instead of black and white only, it employs the spectrum of colors, see that in figure (2.5), fuzzy logic adds a range of logical values to Boolean logic. Now the classical theory of logic can be considered as a special case of multi-valued fuzzy logic as can be seen in Figure (2.6).

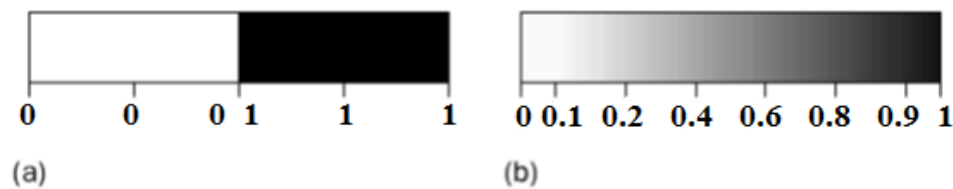


Figure (2.5) Range of logical values in Boolean and fuzzy logic: (a) Boolean logic; (b) multi-valued logic [67]

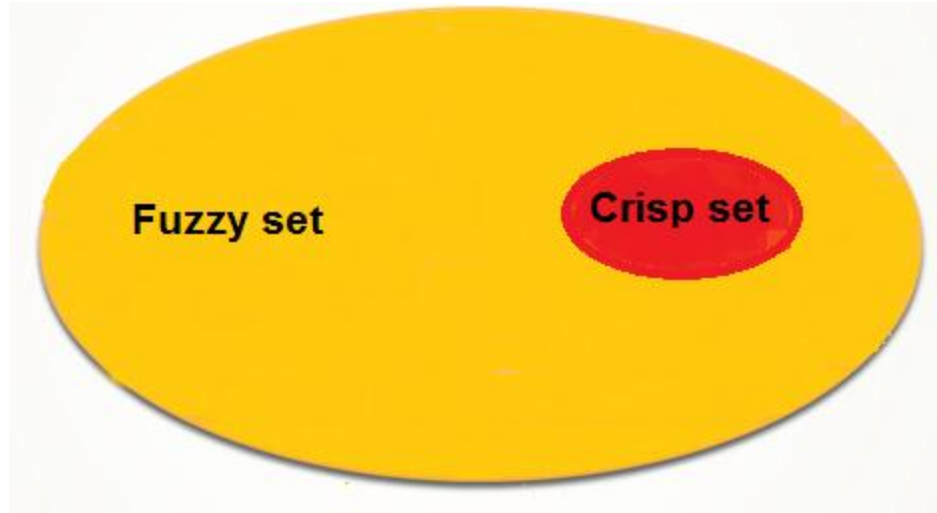


Figure (2.6): The classical set theory is a subset of the theory of fuzzy sets [65]

In any case, the technique of fuzzy logic can be executed for the real applications requirements in the following three steps:

- 1- Fuzzification.
- 2- Fuzzy Inference systems (FIS).
- 3- Defuzzification.

Each step is explained in details as shown in Figure (2.7).

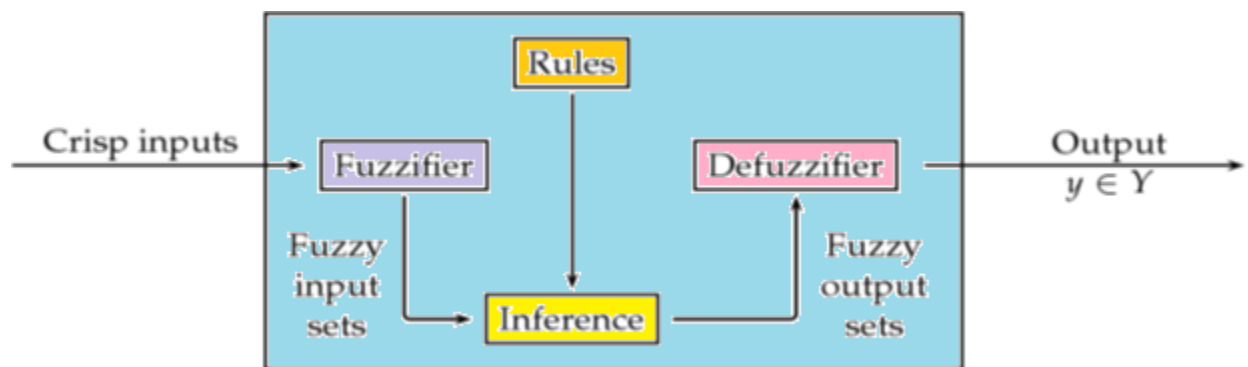


Figure (2.7): Fuzzy Inference System [58]

2.2.1 Fuzzification

It is the first step to apply a fuzzy inference system. Most variables existing in the real world are crisp variables. If we want to change those crisp variables (both input and output) to fuzzy variables, we shall apply fuzzy inference to process those data to find the wanted output [58].

At last those fuzzy outputs must to be altered back to crisp variables to complete the wanted control objectives [61].

Usually, fuzzification includes two processes: derive the membership functions for input variables and its expected output and represent them with linguistic variables. This process is the same to converting classical set to fuzzy set to varying degrees [66].

2.2.2 Fuzzy Inference Systems (FIS)

The output will be computed from the combination of input, output membership functions and fuzzy rules is still a vague element, and this process is called fuzzy inference.

In general, there are three types of FIS the first type is the Mamdani fuzzy inference, the second type is the Tsukamoto fuzzy inference, and finally the third type is the Sugeno fuzzy inference. In this thesis, we will use Mamdani fuzzy inference [64].

2.2.3 Defuzzification

Defuzzification process is applied for the purpose of making the fuzzy output available to real applications. The main target of this process is to convert the fuzzy output back to the crisp output [65]. There are numerous defuzzification approaches that can be used for that purpose such as, the bisector of area, the

centroid of area, the largest of maxima, the smallest of maxima and the mean of maximum methods [66], a detailed analysis of these methods are shown below.

2.2.3.1 Centroid of Area (CoA) [64]

This method involves the computation of the area under the scaled membership functions by the fuzzy logic controller within the range of output variable, after computational analysis, it then manages to make use of the following formula for the purpose of calculation of the geometric center of this area.

$$CoA = \frac{\int_{x_{min}}^{x_{max}} f(x) \cdot x dx}{\int_{x_{min}}^{x_{max}} f(x) \cdot dx} \quad (2.7)$$

Where: CoA stand for the center of the area, x represents the value of the linguistic variable, and x_{max} and x_{min} characterizes the maximum and minimum range of the linguistic variable as shown in Figure (2.7) where the illustration of CoA is out-labeled.

2.2.3.2 Bisector of Area (BoA) [59]

This method returns the vertical line which divides the region into two sub-regions of equal area, Figure (2.7) gives the necessary indications for the BoA method.

$$\int_{\alpha}^{x^*} \mu_G(x) dx = \int_{x^*}^{\beta} \mu_G(x) dx \quad (2.8)$$

Where: $\alpha = \min\{x|x \in X\}$ is the minimum bound, and $\beta = \max\{x|x \in X\}$ is the maximum bound.

2.2.3.3 Mean of Maximum (MoM) [60]

This method takes the average of the base-variable values at that their membership values reach the maximum. The function is given by:

$$X^* = \sum_{j=1}^k \frac{x_j}{k} \quad (2.9)$$

Where k is the number of discrete elements of the output fuzzy set that reach the maximum memberships. Figure (2.7) shows the diagram of MoM.

2.2.3.4 Largest of Maxima (LoM) and Smallest of Maxima (SoM) Methods [61]:

The LoM method takes the maximum value (in terms of magnitude) at that the membership values reach the maximum, the SoM method, takes the minimum value (in terms of magnitude) at that the membership values reach the maximum. Figure (2.7) shows the diagram for both LoM and SoM.

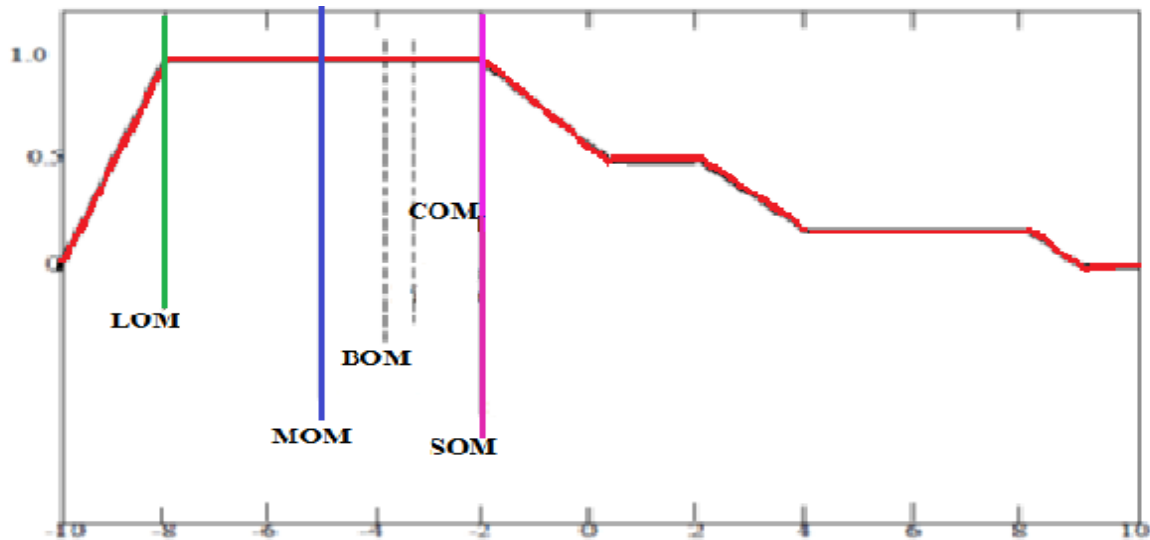


Figure (2.8): Defuzzification methods CoA, BoA, MoM, LoM, and SoM [59].

2.3 Mamdani Fuzzy Inference

Mamdani-type of FIS is a system in which the fuzzy sets from the consequent of each rule are combined through the aggregation operator and the resulting fuzzy set is defuzzified to yield the output of the system [68].

Mamdani type fuzzy inference process is done in the following steps: [64]

Step 1: Defining a set of fuzzy rules

Fuzzy rules are a set of linguistic statements which refer to how the FIS should make a decision about classifying an input or controlling an output. Usually, Fuzzy rules are written in the following form.

IF (input₁ is (MF)₁) or/and (input₂ is (MF)₂) or/and ...input_n is(MF)_n then (output_m is output (MF)_m) [61].

Step 2: We shall fuzzify the inputs by using the input MF

In this step, the crisp numerical values of input variables will be transformed into the equivalent membership values of the suitable fuzzy sets by MF. The output is usually degree of membership in the related fuzzy linguistic sets within the interval between 0 and 1 [68].

Step 3: According to the fuzzy rules, the fuzzified inputs are combined to establish rule strength

The fuzzy operator (OR or AND) is used to get a single number that denotes the result of the antecedent evaluation. If a given fuzzy rule has multiple antecedents. This number (the truth value) is then applied to the resulting MF [64].

Union (OR operation)

$$G \cup H = \mu_G(x) \cup \mu_H(x) = \max(\mu_G(x), \mu_H(x)) \quad (2.10)$$

Intersection (AND operation)

$$G \cap H = \mu_G(x) \cap \mu_H(x) = \min(\mu_G(x), \mu_H(x)) \quad (2.11)$$

Where X is the discourse universe that contain the element x and G and H are two fuzzy sets [65].

Step 4: By combining the rule strength and the output MF, the result of the rule is found [68]

The result of a fuzzy rule is calculated using two steps:

1. Calculating the rule strength by combining the fuzzified inputs using the fuzzy combination process discussed in step 3. This is shown in Figure (2.8).
2. Clipping or scaling the output MF at the rule strength. Figure (2.8) explains how this is done for a two inputs, two rules Mamdani FIS.

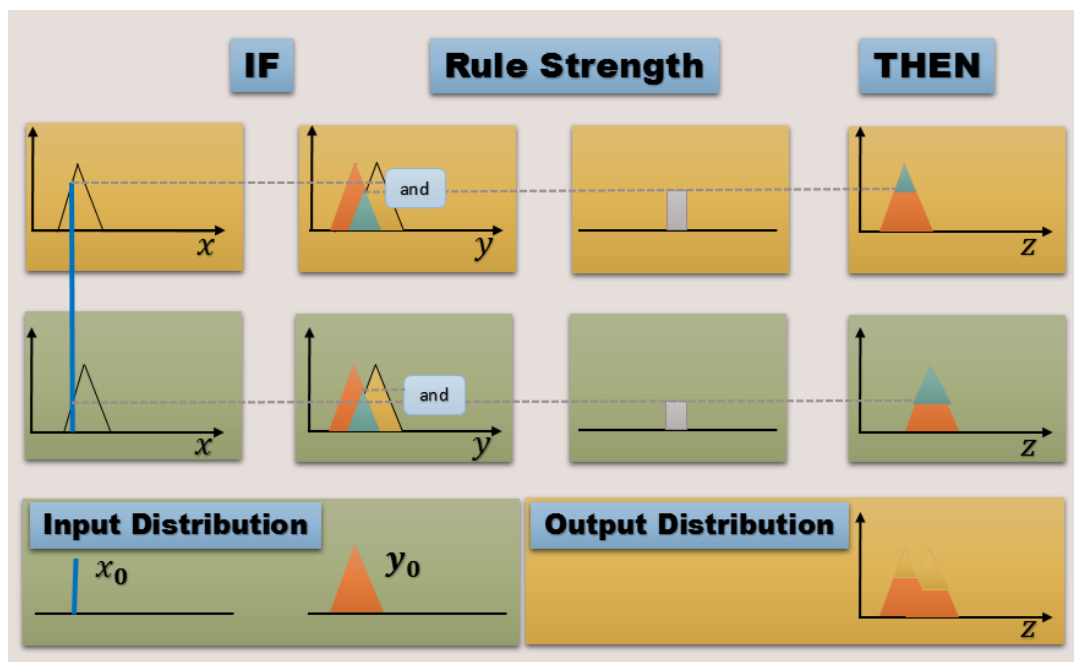


Figure (2.9) A two inputs, two rules Mamdani FIS with fuzzy inputs [58].

We built the fuzzy Rule to organize the fuzzy processing time with the following Rule:

$$R1 = \text{if } (a = \text{Low} \ \& \ b = \text{Mid} \ \& \ c = \text{High}) \text{ then output} = \text{AHR}$$

Where:

(a, b, c) is the fuzzy Processing time.

AHR is the Average High Ranking (Processing time) the fuzzy Rule as shown in Figure (2.10).

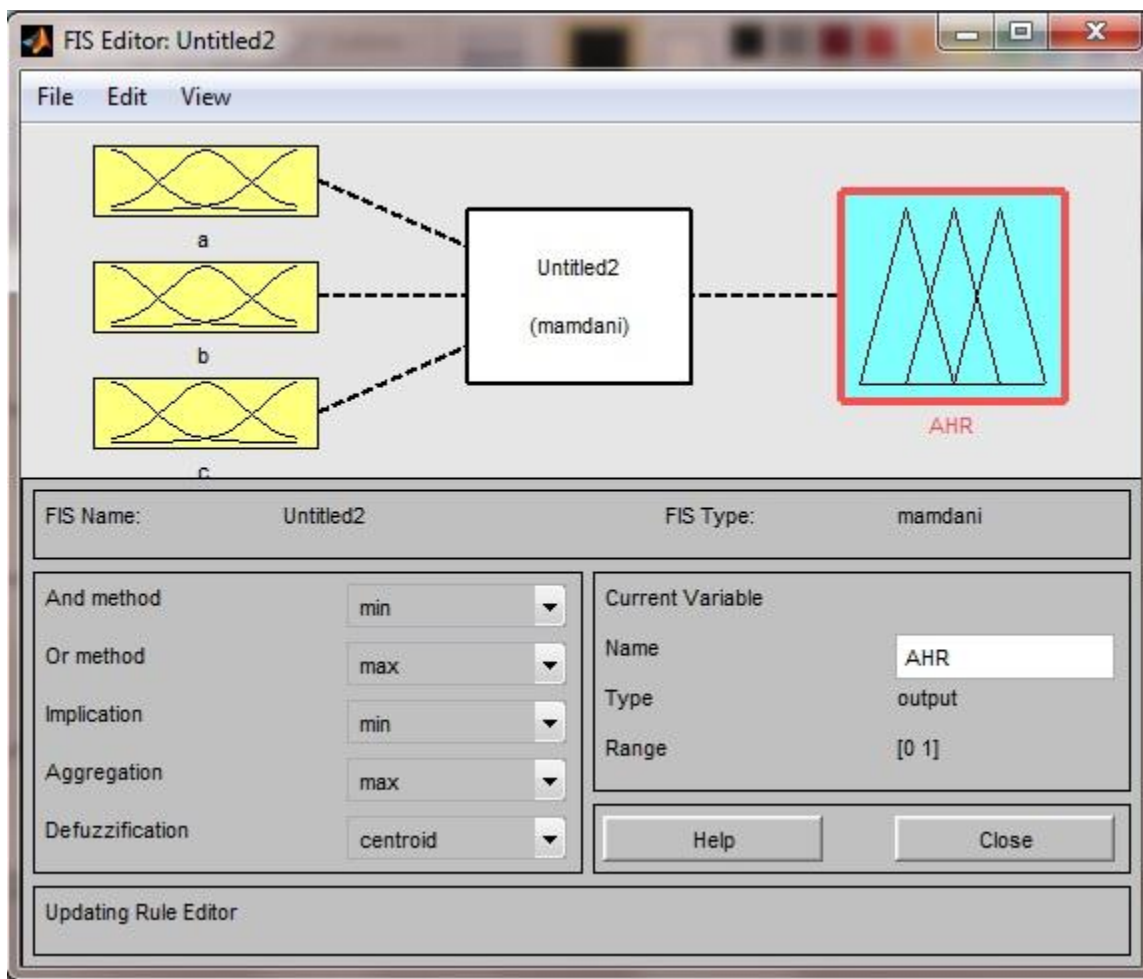


Figure (2.10): Fuzzy rule [68]

Step 5: To get an output distribution, the results combined [58]

Combination is the process of unification of the outputs of all rules. In other words, we take the MF of all rule result earlier clipped or scaled and combine them into a single fuzzy set. So, the input of the combination process is the list of clipped or scaled resulting membership functions, and the output is one fuzzy set for each output variable.

Step 6: Defuzzification [66]

The last step is defuzzification, through which the combined fuzzy set from combination process will output a single scalar quantity. As the name denotes, defuzzification is the reverse operation of fuzzification. Then in the first procedure, the crisp values of input variables are fuzzified into degree of membership with respect to fuzzy sets, the last procedure extracts a precise quantity out of the range of fuzzy set to the output variable.

Chapter Three

*Solving Machine Scheduling
Problem Under Fuzzy
Processing Time Using
Simulated Annealing Method*

Chapter Three

Solving MSP under Fuzzy Processing Time Using SA Method

3.1 Introduction

The optimal sequence with time wherein jobs can be handled in addition to displaying timetables for tasks, individuals, facilities, equipment, and every necessary resource for backing the plan of production are demonstrated by short-term schedules. An efficient usage of resources ought to be done by the schedules in order to provide both higher utilization and lower costs. An additional purposes of scheduling can be, minimizing client delay time for certain product, promising guaranteed delivery dates, maintaining low levels of stock, providing favored working arrangement, reducing delays for patients in a hospital for diverse sorts of checkups, and the list goes.

Sequencing or general scheduling problem can be labelled as if the assumption provides x tasks (jobs) that are executing individually by every y processor (machine) [69]. The research on penalties of tardiness as well as earliness in prototypes of scheduling can be regarded as a moderately hot zone of investigations. In previous studies the investigation of scheduling mainly dedicated to measures for a single performance. Most of the works manage ordered measures with deterministic time, for instance; mean tardiness, mean flow, mean earliness, and percentage of tardy jobs, and so on; but in contemporary societies, the environments are not probabilistic and never fixed. Therefore, it's obligatory for fuzzy environments to be considered this means that the time required for processing any job lies with indeterministic environment.

In this thesis, fuzzy processing time is regarded in three conditions that are ranging from best to worse conditions as follows; the high or favorable one, medium or normal one, and finally bad or worst one. Meaning that the number of tardy criterion is considered to be a normal method for assessing performance with due dates, even though this criterion disregards penalties with jobs that are early completed. Baker, and Scudder (1990) [70] has investigated the sequences that possesses penalties of both tardiness and earliness in an environment with JIT scheduling, the jobs which finish earlier are saved with completed decent record while waiting for their due date to end.

On the other hand, jobs which finish late with the assigned due dates can initiate shutdown operation with the client. For that reason, optimal scheduling refers to any completed job with its allocated due date. This may be interpreted to a scheduling objective by numerous means.

The best apparent objective is to decrease deviations in job completion time (JCT) nearby these due dates with indeterministic time. In penalizing notion tardiness and earliness has both to be produced with novel and fast evolving stream of studies in the scheduling area of research. Since usage of penalties in tardiness and earliness with fuzzified environment promises an irregular measure of performance, it paved the way towards novel operational topics in the enterprise of solution procedures.

Scheduling of independent n jobs in a single machine will be regarded here to minimize bi-criteria. The aim lies in finding a schedule which is the summation of weighted number of early jobs in addition to weighted number of tardy jobs under fuzzy processing time in employing local search algorithm Simulated Annealing (SA). This problem is denoted by $1/ \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j}$

Where h_j , w_j are weighted of early job j , weighted of tardy job j respectively.

$$N_{E_j} = \begin{cases} 1 & \text{if } c_j < d_j \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

$$N_{T_j} = \begin{cases} 1 & \text{if } c_j > d_j \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

The problem in normal case (that is without fuzzy processing time) was considered by H. Soroush (2007) [11] and he show that it is NP-hard. SA method is used to find approximate solutions, for our problem solved approximately with up to 12000 jobs.

Local search (LS) can frequently be regarded as the chosen technique especially if one faces problems in combinatorial optimizations for instance, multi-criteria scheduling, absence of perception with problem structures or problem size can prevent the use of accurate algorithm of optimization. An algorithm in LS begins by a primary solution after that it repeatedly attempts to add superior solutions by neighborhoods searching. **LS** heuristics in combinatorial originated at the moment of firstly introducing algorithms of edge-exchange for the famous problem traveling salesman in the 1950s. Aarts, and Lenstra (1997) [71] showed a summary for which our adopted ways for briefing LS in combinatorial optimizations are shown.

The algorithm describes the space of arrangements in the following: all viewed solutions are neighbors for previously viewed ones. If all solutions neighbors have identical or bad value for the objective function, then they are known as local minima regarding a neighborhood function. It is evident that when algorithms continually chose the best neighbor or slightly improved costly, they eventually turn out to be in local minima [72].

LS algorithms are extensively employed with plentiful of solid computations problems; incorporating artificial intelligence (AI) problems from computer science, also other scientific and engineering disciplines such as operations research, bioinformatics, mechanical engineering, and mathematics. In this chapter, the heuristic method namely, SA was developed using fuzzy processing time.

3.2 Basic Definitions [73]

Definition (3.1)

For an instance of a combinatorial optimization problem (COP) with feasible solutions set S , a neighborhood function is a mapping $N: S \rightarrow P(S)$, which defines for each feasible solution $s \in S$, a set $N(s) \subseteq S$, of feasible solutions that are in some sense close to s . The set $N(S)$ is the neighborhood of the feasible solutions, and each $s' \in N(s)$ is a neighbor of s .

Definition (3.2)

Let S be the solutions set of an instance for a combinatorial optimization problem with cost function f and let N be a neighborhood function. A feasible solution $s' \in S$ is locally optimal with respect to N if $f(s') \leq f(s)$ for all $s \in N(s')$ in case of a minimization problem and if $f(s') \geq f(s)$ for all $s \in N(s')$ in case of a maximization problem.

Definition (3.3)

Let S be the set of feasible solutions and f is the cost function defined by $f: S \rightarrow R$. s' is said to be a global minimum optimal solution if $f(s') \leq f(s)$ for all $s \in S$.

Definition (3.4)

A local minima is a solution $(s \in S)$, such that $f(s) \leq f(t)$ for all $t \in N(s)$, while a global minima is a solution $(s \in S)$, such that $f(s) \leq f(t)$ for all $t \in S$ (set of all feasible solutions). A neighborhood function N is called exact, if every local minima is global.

3.3 The Concepts of Fuzzy Processing Time [74]

Job processing time may differ in plenty of ways, the reason could be because of an environmental factor or because of the altered workplaces. We find that when a service provider gets the work from a sector, he or she may compute the overall spending during share. However, because of several reasons such as labor lack, unpromising weather, or other nonstandard circumstances, this will change costs. Thus, because of the above-mentioned causes; work will probably be late in completion this would create a problem of due date, meaning; time delivery of an order will become out of control, alternatively, if the work completes before the due time it arises the inventory problem. So to overcome these factors, the processing time of a job considered here is in three situations: favorable, normal and worse conditions.

In this thesis, a new concept for the different processing time of each job is considered which helps the contractor to estimate the cost of the work at the time of allotment. In this thesis, different due dates for each of the job be considered which meets the demand maker with more satisfaction level. So using the algorithm developed here, the contractor can save the penalty cost and can satisfy the demand maker to a great extent.

3.4 Formulation of the Problem

Single machine scheduling models seem to be very important for understanding and modeling multiple machines models. A set $N=\{1,2,\dots,n\}$ of n independent jobs has to be scheduled on a single machine in order to minimize a given criterion. This study concerns the one machine scheduling problem with multiple objectives function which is denoted by $(1/ \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$.

In this problem, preemption is not allowed, no precedence relation among jobs is assumed and all jobs are available at the time zero, (that is $r_j=0 \forall j$). Each job j has a_j, b_j, c_j fuzzy processing time, d_j due date, h_j weighted of early job j and w_j weighted of tardy job j .

The start time of job j is denoted by t_j , where $t_1=0$ and $t_j = \sum_{i=1}^{j-1} p_i$ and its completion time by C_j , ($C_j = t_j + p_j$), if job j completed before its due date ($C_j < d_j$ then $N_{E_j}=1$) and job j is said to be early otherwise ($C_j \geq d_j$, then $N_{E_j}=0$), and job j is said to be tardy job or just in time, for each job j can be calculate the slack time $Sl_j = |A_j - d_j|$. In this chapter, next jobs are scheduled in increasing order of their slack time. Consider the processing time in fuzzy environment (a, b, c) , which is integer numbers. We calculate the Average High Ranking (AHR) Method to generate a processing time from a fuzzy processing time.

$$AHR = [3b + (c - a)] / 3 \quad (3.3)$$

Where a, b, c is fuzzy processing time.

For a given schedule we can calculate weighted number of early jobs $N_{h_j E_j}$, as follow:

$$N_{h_j E_j} = \begin{cases} h_j & \text{if } d_j > C_j \\ 0 & \text{if } d_j \leq C_j \end{cases} \quad (3.4)$$

And weighted number of tardy jobs $N_{w_j T_j}$:

$$N_{w_j T_j} = \begin{cases} w_j & \text{if } C_j > d_j \\ 0 & \text{if } C_j \leq d_j \end{cases} \quad (3.5)$$

The objective is to find the schedule $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ of the jobs that minimize the total penalty cost R which is formulated in mathematical form as:

$$\left. \begin{array}{l} \min R = \min_{\pi \in \delta} \{ \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j} \} \\ \text{subject to} \\ C_{\pi(j)} \geq p_{\pi(j)} \quad j = 1, 2, \dots, n \\ C_{\pi(j)} = C_{\pi(j-1)} + P_{\pi(j)} \quad j = 2, 3, \dots, n \\ N_{E_{\pi(j)}} \in \{0, 1\} \quad j = 1, 2, \dots, n \\ N_{T_{\pi(j)}} \in \{0, 1\} \quad j = 1, 2, \dots, n \\ p_{\pi(j)} > 0, \quad d_{\pi(j)} > 0, \quad w_{\pi(j)} > 0, \quad h_{\pi(j)} > 0 \end{array} \right\} \quad (Q)$$

Where δ the set of all feasible solutions, $\pi(j)$ denoted the position of j in the ordering π .

3.5 Assumption and Notation [69]

The machine scheduling problem is studied in this chapter and requires n independent jobs j_i ($i=1, 2, 3, \dots, n$) to be processed on a single machine with the following assumptions:

i) Only one job can be processed on a given machine at a time.

- ii) All jobs are available for processing at time zero.
- iii) The single machine can process at most one job at a time.
- iv) No pre-emption is allowed.
- v) The processing times are fuzzy and sequence independent.
- vi) The processing order of each job is given and fixed.

Notations:

- S : Schedule for the n jobs.
- a_j, b_j, c_j : Processing time of job j on the machine in fuzzy environment.
- A_j : Average high ranking of the processing time a, b, c
 $A_j = [3b_j + (c_j - a_j)] / 3.$
- d_j : Due date for the job j .
- C_j : Completion time of job j .
- Sl_j : Slack time of job j , $Sl_j = |A_j - d_j|$.
- h_j : The weighted of early job j .
- w_j : The weighted of tardy job j .

3.6 Algorithm

Step 1: Define three input (a, b, c) and output of jobs (initialization).

$$a = [1,10], b = [1,10], c = [1,10]$$

Step 2: Construct the membership functions of three inputs(initialization).

Step 3: Construct the rule base (initialization).

Step 4: Convert crisp input data to fuzzy values using the membership functions (fuzzification).

Step 5: Evaluate the rules in the rule base (inference).

Step 6: Combine the results of each rule (inference).

Step 7: Convert the output data to non-fuzzy values (defuzzification).

Step 8: Determine Average High Ranking (AHR) of the fuzzy processing time (a, b, c) of all the jobs.

$$A_j = [3b_j + (c_j - a_j)] / 3$$

Step 9: Determine the slack time of all the jobs $Sl_j = |A_j - d_j|$.

Step 10: Order the jobs in MST rule. If more than job have the same slack time then breaking ties the sequence by EDD rule.

Step 11: Using the sequence obtained in step 3 find the total penalty cost of all the jobs using weighted of early jobs (h_j) and weighted of tardy jobs (w_j).

Step 12: END.

Example 3.1

Consider four jobs having fuzzy processing time, single machine, due dates, weighted of early jobs (h_j) and weighted of tardy jobs (w_j) are given in Table (3.1)

Table (3.1): The initial data for example 3.1

Job	p_j	d_j	h_j	w_j
1	1 2 4	5	2	3
2	3 8 9	11	4	6
3	5 7 8	25	7	9
4	7 8 10	30	10	12

Solution:

The three inputs will be fuzzy processing time a , b , c and the output will be Average High Ranking AHR.

$$X = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10]$$

$$a = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8]$$

$$b = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ 8]$$

$$c = [4 \ 5 \ 6 \ 7 \ 8 \ 9 \ 10]$$

The membership functions of inputs as showing in Figure (3.1):

$$\mu_{a \cap b \cap c} = \min(\mu_a(x), \mu_b(x), \mu_c(x))$$

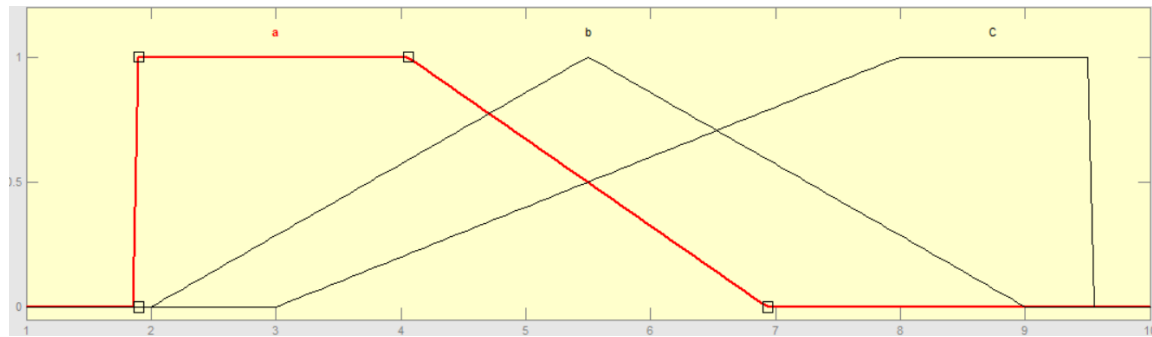


Figure (3.1): Membership function.

Construct the rule base and Convert crisp input data to fuzzy values using the membership functions as following:

R1: if a is low and b is medium and c is high then output is AHR.

Convert the output data to non-fuzzy values as showing in Figure (3.2)

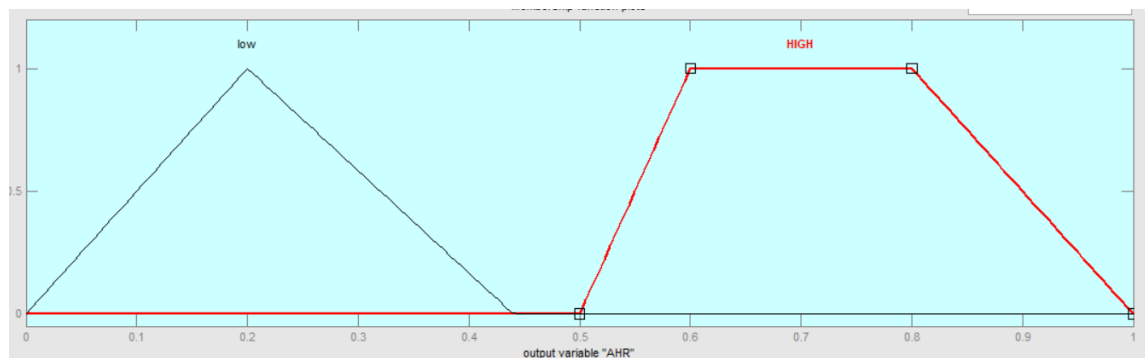


Figure (3.2): Output membership function

The threshold will be ≥ 0.7

Determine Average High Ranking (AHR) of the fuzzy processing time (a, b, c) of all the jobs.

$$A_j = [3b_j + (c_j - a_j)] / 3$$

Table (3.2): The calculated AHR and Slack time for example 3.1

Job	p_j	AHR	d_j	Sl_j	h_j	w_j
1	1 2 4	3	5	2	2	3
2	3 8 9	10	11	1	4	6
3	5 7 8	8	25	17	7	9
4	7 8 10	9	30	21	10	12

Then MST rule gives the sequence:

$$S = 2 - 1 - 3 - 4$$

The table (3.3) shows the flow time of the system and the total penalty cost (R).

Table (3.3): The calculated total penalty cost for example 3.1

Job	P_j	d_j	c_j	h_j	w_j	cost
2	10	11	10	4	6	4
1	3	5	13	2	3	3
3	8	25	21	7	9	7
4	9	30	30	10	12	0

The total Penalty cost (R) is = 14

3.7 Simulated Annealing Heuristics

The simulated annealing (SA) algorithm is a way of finding solutions to problems which have a large set of possible solutions, in an analogous fashion to the physical annealing of solids to attain minimum internal energy states. The basic idea is to create a path through the solution space, from one solution to another nearby solution, leading ultimately to the optimum solution [46]. In generating this path, solutions are chosen from the locality of the preceding solution by a probabilistic function of the improvement gained by this move. So, stages are not strictly required to produce improved solutions, but each stage has a certain probability of leading to improvement; at the start, all stages are equally likely, but as the algorithm progresses, the tolerance for solutions worse than the current one decreases, eventually to the point where only improvements are accepted. In this way, the algorithm can attain the optimum solution without becoming trapped in local optima [75].

This tells us that we can have some confidence in the answer which is created by a well-formulated SA implementation, but says nothing about how long it might

take to find that answer. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities) [75].

SA was independently described by Kirkpatrick, et.al., (1983) [76] and V. Černý (1985) [77], it is an adaptation of the Metropolis-Hastings algorithm, a Monte Carlo method to generate sample states of a thermodynamic system, invented by Rosenbluth and published in a paper by Metropolis, et al., (1953) [78]. The following structure gives the outline of SA:

SA Algorithm: [75]

Let 's term,

S_C : Candidate schedule.

S_0 : Best schedule found so far.

S_K : Schedule constructed at K iteration (K= iteration counter).

$G(S_0)$: Aspiration criterion (Value of best schedule).

$G(S_K)$: Value of schedule constructed at K iteration.

$G(S_C)$: Value of candidate schedule.

$P(S_K, S_C)$: Probability of moving from S_K schedule to S_C schedule at K Iteration.

$$P(S_K, S_C) = \exp\left(\frac{G(S_K) - G(S_C)}{\beta_K}\right)$$

Where,

β_K is called cooling parameter in annealing terminology usually,

$$\beta_K = a^K \text{ where } a \in [0,1].$$

Step 1: Initialization

Set $K=1$, Set β_1 equal to a given value.

From starting sequence by any heuristic; call it S_1

Let $S_0 = S_1$, then $G(S_0) = G(S_1)$

Step 2: Conditioning

Select S_C from S_K

IF $G(S_0) < G(S_C) < G(S_K)$,

THEN $S_{K+1} = S_C$

GOTO Step (3)

END

IF $G(S_C) < G(S_0)$,

THEN $S_0 = S_{K+1} = S_C$

GOTO Step (3)

END

IF $G(S_C) > G(S_K)$,

THEN generate a random number $U_K \sim [0,1]$

IF $U_K \leq P(S_K, S_C)$,

THEN $S_{K+1} = S_C$

ELSE $S_{K+1} = S_K$

END

Step 3: Finalization

Set $\beta_{K+1} \leq \beta_K$, Set $K = K+1$

IF $K \leq N$

THEN GOTO Step (2)

ELSE Stop

Example 3.2

Solve the problem $(1 / \sum_{j=1}^n N_{h_j} E_j + \sum_{j=1}^n N_{w_j} T_j)$ using SA method. Apply the technique with four iteration ($K \leq 4$), and $N = 4$. Use U values = $\{0.8, 0.01, 0.02, 0.43\}$, take initial value of $\beta = 0.9$, using the data of Ex (3.1).

Solution

$\beta_1 = 0.9, K = 1, S_1 = \{j_2, j_1, j_3, j_4\}$

Set $S_0 = S_1$

$G(S_0) = 14, G(S_1) = 14,$

Let $S_C = \{j_1, j_2, j_3, j_4\},$

$G(S_C) = 15,$

$G(S_C) > G(S_1),$

$U_1 = 0.8$

$$P(S_1, S_C) = \exp\left\{\frac{G(S_1) - G(S_C)}{\beta_1}\right\} = 0.2865$$

$$U_1 > P(S_1, S_C)$$

$$\text{set } S_2 = S_1, S_2 = \{j_2, j_1, j_3, j_4\}, G(S_2) = 14$$

$$\beta_2 = (\beta_1)^2 = (0.9)^2 = 0.81$$

$$K = 2, N = 4$$

$$\text{Let } S_C = \{j_2, j_1, j_4, j_3\},$$

$$G(S_C) = 26,$$

$$G(S_C) > G(S_2)$$

$$U_2 = 0.01$$

$$P(S_2, S_C) = \exp\left\{\frac{G(S_2) - G(S_C)}{\beta_2}\right\} = 0.0000003681$$

$$U_2 > P(S_2, S_C)$$

$$\text{set } S_3 = S_2, S_3 = \{j_2, j_1, j_3, j_4\}, G(S_3) = 14$$

$$\beta_3 = (\beta_1)^3 = 0.729$$

$$K = 3, N = 4,$$

$$S_C = \{j_3, j_2, j_1, j_4\}$$

$$G(S_C) = 16$$

$$G(S_C) > G(S_3)$$

$$U_3 = 0.02$$

$$P(S_3, S_C) = \exp\left\{\frac{G(S_3) - G(S_C)}{\beta_3}\right\} = 0.0643$$

$$U_3 < P(S_3, S_C)$$

$$\text{set } S_4 = S_C, S_4 = \{j_3, j_2, j_1, j_4\}, G(S_4) = 16$$

$$\beta_4 = (\beta_1)^4 = (0.9)^4 = 0.6561$$

$$K = 4, N = 4$$

$$\text{Let } S_C = \{j_1, j_3, j_2, j_4\}$$

$$G(S_C) = 15$$

$$G(S_0) < G(S_C) < G(S_4)$$

$$\text{set } S_5 = S_C, S_5 = \{j_1, j_3, j_2, j_4\}, G(S_5) = 15$$

$$\beta_5 = (\beta_1)^5 = 0.5905$$

$$K = 5, N = 4,$$

$$S_0 = \{j_2, j_1, j_3, j_4\}, G(S_0) = 14$$

3.8 Computational Experiments of applying SA Algorithm

An intensive effort has been implemented in relation to numerical investigations. We will present in the next subsection how the problem under study is created randomly

3.8.1 The Problem Test [79]

There exists in the literature a classical way to randomly generate test problems of scheduling problems.

- ❖ The processing times a_j , b_j and c_j is uniformly distributed in the interval $[1,10]$.

- ❖ The due date d_j is uniformly distributed in the interval $[P(1-TF-RDD/2), P(1-TF+RDD/2)]$; where $P = \sum_{j=1}^n p_j$ depending on the relative range of due date (RDD) and on the average tardiness factor (TF).
- ❖ The integer h_j and w_j are generated from uniform distribution $[1,10]$.

For both parameters, the values 0.2, 0.4, 0.6, 0.8 and 1.0 are considered. For each selected value of n (where n is the number of jobs), ten problems were generated.

3.8.2 Computational Results

In this section, we provide our results that were obtained from the SA algorithm with our proposed function. The test problems are provided in this chapter. It's worth mentioning that we used the MATLAB 7.10.0 (R2010a) software for coding the algorithms and the processing personal computer was of 4 GB RAM with a CPU of 2.5 GHz speed.

Table (3.4) displays a calculated cost by SA algorithm for 10 problems at $n=500$, the time of each problem and the number of iterations, in addition to the average of time and cost.

Table (3.5) shows both the cost and times averages for 10 problems for each of its values from the number of jobs n where $n \in \{10, 50, 100, 500, 1000, 5000, 10000, 12000\}$ using SA.

From the observation of the two tables (3.4) and (3.5) we find that the method of SA gives an approximate solution or near optimal solution, we concluded from our observation that this method can solve up to $n = 12000$ during 30 seconds, and SA algorithm will cease when a constant time amount is passed, here after 30 seconds.

Table abbreviations:

n	Number of jobs.
No. of iterations	Number of iterations.
No.	Number of problems.
Av. of Cost	Average of cost.
Av. of Time	Average of time for (10) examples in Simulated Annealing algorithm.

At this point, we find the solution of this problem with h_j and w_j to SA for $n = 500$.

Table (3.4): The Performance of SA Algorithm for $n= 500$ jobs

n	No.	Cost	Time	No. of Iterations
500	1	2307	29.0160	122
	2	2306	29.2483	160
	3	2270	29.1916	169
	4	2345	29.1700	170
	5	2186	29.0170	168
	6	2210	29.0610	163
	7	2408	29.2995	167
	8	2327	29.0784	168
	9	2271	29.1836	168
	10	2290	29.1533	166
Av. of Cost		2292		
Av. of Time		29.1419		

Table (3.5): The performance to average of cost and average of time in SA algorithm for $n \in \{10, 50, 100, 500, 1000, 5000, 10000, 12000\}$

n	Av. of Cost	Av. of Time
10	44.6	0.2758
50	225.2	2.8994
100	452.3	9.0964
500	2292	29.1419
1000	4516.7	29.1454
5000	22482	29.4675
10000	45072	29.7156
12000	54291	30.4109

Chapter Four

Solving Machine Scheduling Problem Under Fuzzy Processing Time Using Tabu Search Method

Chapter Four

Solving MSP under Fuzzy Processing Time Using TS Method

4.1 Introduction

In this chapter, we will study a single machine with its' scheduling for n independent jobs (fuzzy processing time, due date, weighted of early jobs and weighted of tardy jobs (i.e. the same data calculated and adopted in chapter 3)). To minimize the sum of two functions, our objective is to get a schedule which is given an approximate solution for the problem sum weighted number of early jobs and weighted number of tardy jobs under fuzzy processing time, this problem is strongly NP- hard and denoted by $1/ / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j}$ (which we described in chapter 3) by using Tabu Search method.

4.2 Tabu Search

At Fred Glover [80], was willing to overcome local optima arising in the methods of LS, so he proposed a novel methodology by developing over a number of his earlier research, and he named it as Tabu Search (TS). Actually, a numerous foundations of this leading TS proposal, and also a particular foundations of advanced TS elaborations, were a novelty presented by Gendreau [81], comprising of a temporary memory to avoid the reversal of contemporary moves and a longer term frequency memory to strengthen attractive components. Hence one finds several references in the literature that addresses algorithm of tabu search, Beausoleil [82], used the TS to solve the problem of weighted tardiness sequence-dependent setups and compare with Re-start method. AL-Anzi, and A. Allahverdi [83], studied the problem of total completion time; the suggested heuristics is to be comparing with the current heuristics tabu search with SA this shown to be more efficient. Haung, and Yu [84], presented a novel algorithm of TS in order to solve

a scheduling problem that possesses a several robust project that is constrained resourcefully. Selt [85], proposed two algorithms to solve the NP-hard problem and compare them with TS.

Local (neighborhood) searches tackled the problem and found a probable solution by checking its instant neighbors (i.e., the solutions are alike, apart from very little minor specifics) with the anticipation of discovering an enhanced solution. Current applications of TS span areas related to telecommunication, resource-planning, scheduling, financial enquiry, space development, molecular engineering, pattern recognition, energy sharing, classification, malleable manufacturing, waste management, logistics, biomedical examination, environmental safeguarding, mineral investigation, and plenty of additional applications.

TS improves the above mentioned techniques performance by making use of the memory structures that are designated to the previous solutions or to the groups of instructions provided by Glover [86].

It's worth mentioning that when a probable solution was formerly visited contained by a specific short duration period or when the solution was violating a rule, then it's labeled as "tabu" (prohibited) in such a way that the algorithm will potentially not take care of that possibility continually.

4.3 TS Algorithm

TS is the LS method used for mathematical optimization, it was invented by Fred Glover in 1986 [80].

The algorithm requires certain number of parameters for being set, as follow,

Algorithm: [75]

Assumptions,

S_C	The schedule of the candidate.
$G(S_C)$	The value of schedule of the candidate.
S_0	The best schedule initiated up until now.
$G(S_0)$	The value of best schedule (aspiration criterion).
K	The number of iterations.
S_K	The schedule created through K iterations.
$G(S_K)$	The value of schedule assembled by K iterations.

Step 1 : Initialization

Set $K = 1$

From starting sequence by any heuristic; call it S_1

Let $S_0 = S_1$,

Then $G(S_0) = G(S_1)$

Step 2 :

Select S_C from neighborhood of S_K

IF move from S_K to S_C is not allowed in the Tabu List.

THEN $S_{K+1} = S_K$, GOTO Step 3

IF $G(S_C) < G(S_0)$,

THEN $S_0 = S_C$

$$G(S_0) = G(S_C)$$

Delete the oldest Tabu move in the Tabu List.

GOTO Step 3

Step 3 :

Set $K = K+1$

IF $K \leq N$

THEN GOTO Step 2

ELSE Stop

Example (4.1): Using the data in Table (3.3) and algorithm in sub sec. 3.6 for getting the starting sequence to solve the problem $(1/ / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$ by employing the Tabu Search method, then apply the procedure for four iterations, in addition making the Tabu list length equal to two.

Solution:

The example can be set to solve as the following sequence

First iteration

Tabu List = $\emptyset$,

$S_1 = \{j_2, j_1, j_3, j_4\}$

Let $S_0 = S_1$.

So, $S_0 = \{j_2, j_1, j_3, j_4\}$

Then, $G(S_0) = 14, G(S_1) = 14$.

Find neighborhoods of S_1 By using adjacent pairwise interchange, as shown in Figure (4.1) below:

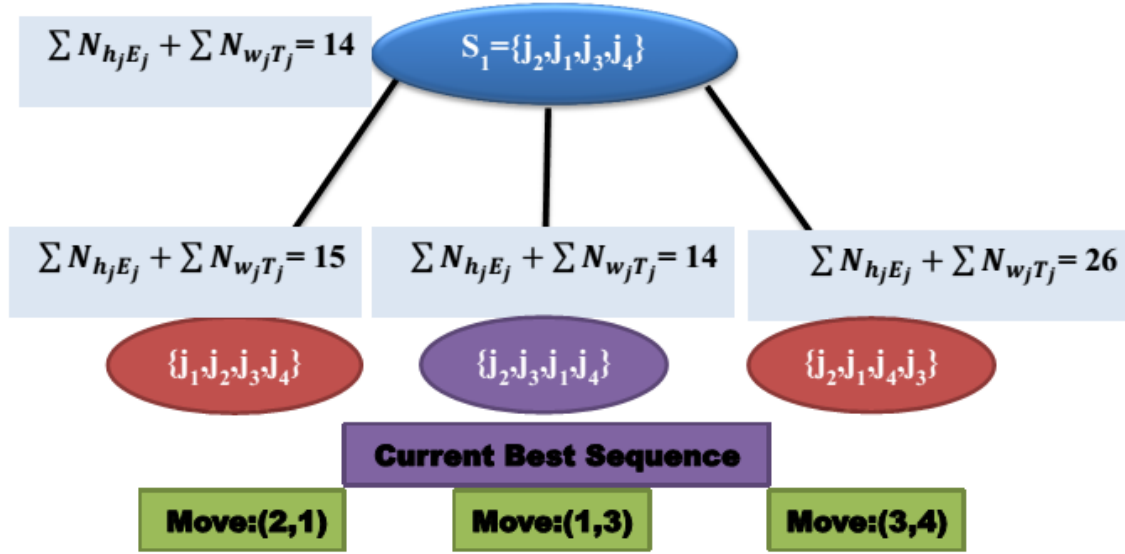


Figure (4.1): Illustrative approach for finding the sequences of S_1 in example 4.1 using adjacent pair wise interchange.

Let S_C = Current Best Sequence (CBS)

$S_C = \{j_2, j_3, j_1, j_4\}$.

$G(S_C) = 14$.

Since, $G(S_C) = G(S_0)$,

$S_0 = S_C$, and, $G(S_0) = 14$.

Then, Tabu Move = (1,3) and Tabu List = {(1,3)}

Let, $S_2 = S_C = \{j_2, j_3, j_1, j_4\}$, $G(S_0) = 14$.

Second iteration

Generating all the sequences of S_2 using interchanging adjacent pairwise as shown in the list below.

Sequence S_2	Pair to be interchanged	New Sequence
$\{j_2, j_3, j_1, j_4\}$	(2,3)	$\{j_3, j_2, j_1, j_4\}$
$\{j_2, j_3, j_1, j_4\}$	(3,1)	$\{j_2, j_1, j_3, j_4\}$
$\{j_2, j_3, j_1, j_4\}$	(1,4)	$\{j_2, j_3, j_4, j_1\}$

Then, The CBS is = $\{j_3, j_2, j_1, j_4\}$

$$G(S_C) = 16$$

Since, $G(S_C) > G(S_0)$,

$$G(S_0) = 14.$$

Then, Tabu Move = (2,3) and Tabu List = $\{(1,3), (2,3)\}$.

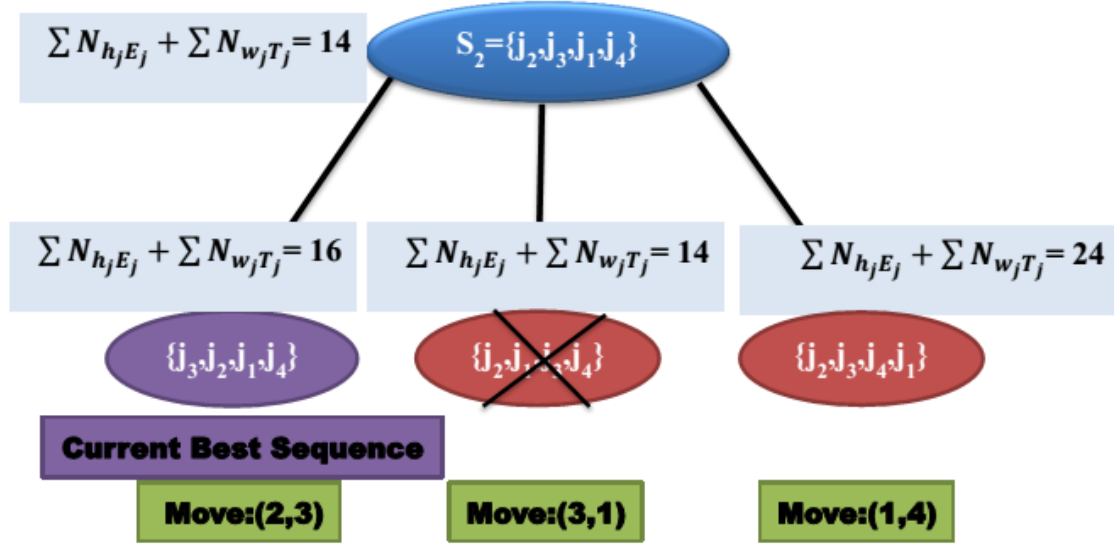


Figure (4.2): Illustrative approach for finding the sequences of S_2 in example 4.1 using adjacent pairwise interchange

Let, $S_3 = S_C = \{j_3, j_2, j_1, j_4\}$, $G(S_0) = 14$.

Third iteration

Generating all the sequences of S_3 using interchanging adjacent pairwise as shown in Figure (4.3)

Only possible move is (2,1)

Then, The CBS is $\{j_3, j_1, j_2, j_4\}$

$$G(S_C) = 16.$$

Since, $G(S_C) > G(S_0)$,

$$G(S_0) = 14.$$

Then, The new Tabu Move = (2,1) and, the new Tabu List = $\{(2,3), (2,1)\}$.

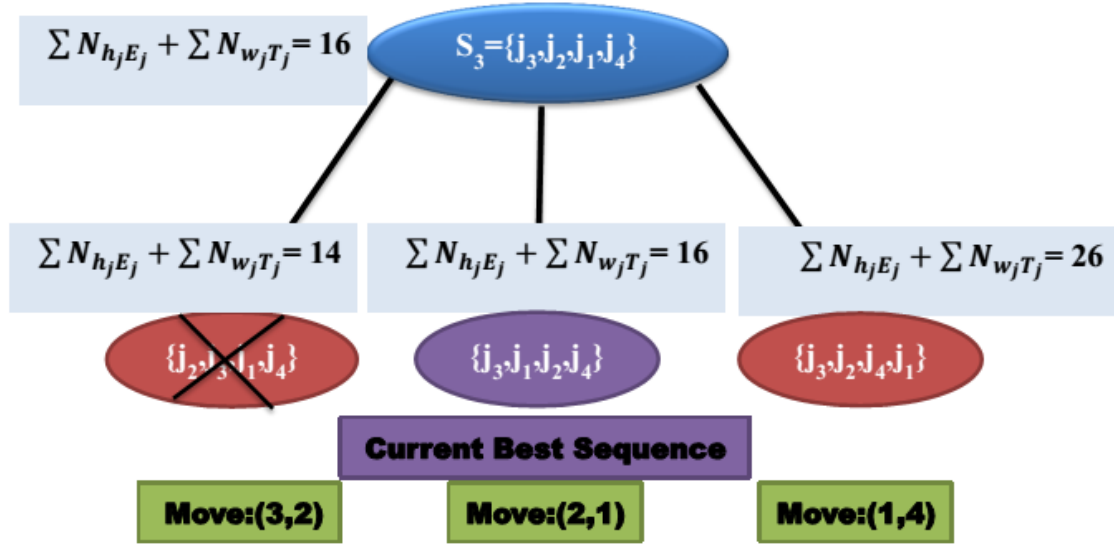


Figure (4.3): Illustrative approach for finding the sequences of S_3 in example 4.1 using adjacent pairwise interchange.

Let, $S_4 = S_C = \{j_3, j_1, j_2, j_4\}$, $G(S_0) = 14$.

Fourth iteration

Generating all the sequences of S_4 using interchanging adjacent pairwise as shown in Figure 4.4

Only possible move is (3,1)

Then, The CBS is $\{j_1, j_3, j_2, j_4\}$

$G(S_C) = 15$,

Since, $G(S_C) > G(S_0)$,

$G(S_0) = 14$,

Then, new Tabu Move = (3,1) and, new Tabu List = $\{(2,1), (3,1)\}$.

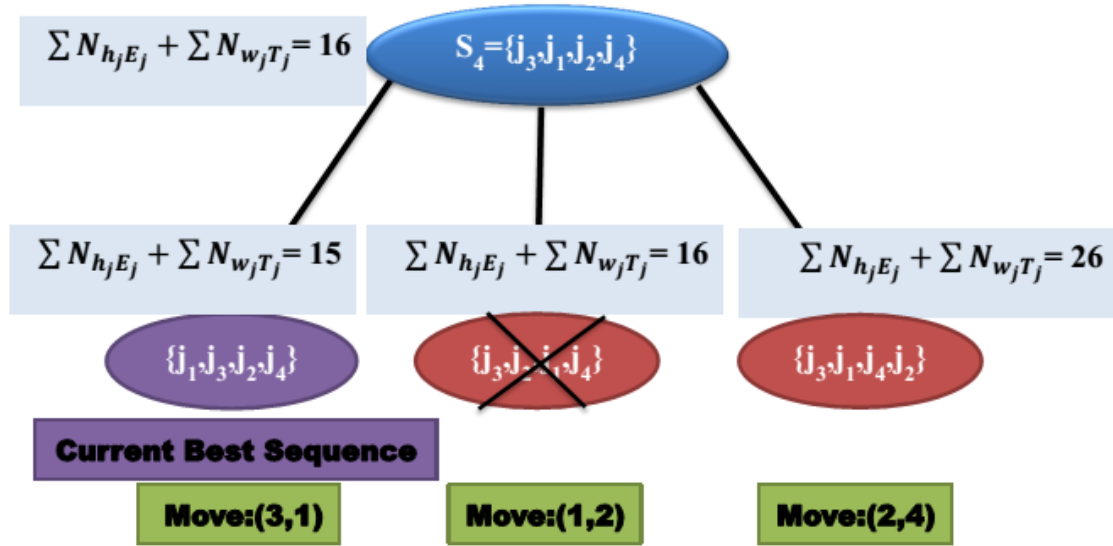


Figure (4.4): Illustrative approach for finding the sequences of S_4 in example 4.1 using adjacent pairwise interchange.

Let, $S_5 = S_C = \{j_1, j_3, j_2, j_4\}$. And $G(S_0) = 14$.

At the end of iteration 4, $G(S_0) = 14$,

Best Overall Sequence BOS = $\{j_2, j_1, j_3, j_4\}$.

4.4 Computational Results of LS Algorithms and Comparison

4.4.1 The Problem Test

We use the test problems that are introduced in subsection (3.9.1) for the local search algorithms.

4.4.2 Computational Results

In this section, we provide our results that were obtained for SA and TS algorithms with our proposed function. The test problems are provided in this chapter and chapter three. It's worth mentioning that we used the MATLAB 7.10.0 (R2010a) software for coding the algorithms and the processing personal computer was of 4 GB RAM with a CPU of 2.5 GHz speed. In our computations we proved that the TS algorithm is active for 12100 jobs to a processing time of 30 seconds.

Now,

we find the solution of this problem with h_j and w_j to TS algorithm for $n = 500$ s.t. (n = number of jobs) as revealed by the Table (4.1) below:

Table (4.1): The Performance of TS Algorithm for $n = 500$ jobs.

n	No.	Cost	Time	No. Of Iteration
500	1	2295	0.8385	10
	2	2220	0.4426	10
	3	2261	0.4424	10
	4	2189	0.4441	10
	5	2220	0.4658	10
	6	2255	0.4407	10
	7	2348	0.4251	10
	8	2365	0.4649	10
	9	2284	0.4440	10
	10	2207	0.4422	10
AV. of Cost		2264.4		
AV. of Time		0.4850		

The solution of this problem with h_j and w_j to TS algorithm for $n \in \{10,50,100,500,1000,5000,10000,12000,12100\}$ is as revealed by the following Table (4.2):

Table (4.2): The Performance to average of cost and average of time in TS algorithm for $n \in \{10, 50, 100, 500, 1000, 5000, 10000, 12000, 12100\}$

n	AV. of Cost	AV. of Time
10	39.1000	0.1540
50	220.7000	0.2121
100	452.3000	0.2484
500	2264.4	0.4850
1000	4455.8	1.0531
5000	22460	10.7370
10000	45015	29.7351
12000	53989	29.4697
12100	54419	29.9722

4.4.3 Comparative Result of Local Search Algorithms

The best solution for examples is provided in Table (4.5) next page which displays the results of between SA and TS algorithms and the amount of time for establishing the best value for a size of problem " n " where:

Best	The calculated best value.
SA	The value found by simulation annealing.
TS	The value found by Tabu search.
No. of Best	The number of examples that performs the best value.
Av. Time	The time average for ten examples in both algorithms in seconds.

Table (4.3): Performance of both LS methods with the best solution for $n \in \{10, 50, 100, 500, 1000, 5000, 10000, 12000\}$.

n	No.	Best	SA	TS
10	1	38	55	38
	2	42	48	42
	3	39	49	39
	4	20	41	20
	5	36	36	49
	6	46	47	46
	7	37	51	37
	8	29	29	45
	9	36	36	47
	10	28	54	28
No. of Best			3	7
Av. Time			0.2758	0.1540
50	1	237	238	237
	2	227	230	227
	3	205	205	225
	4	219	219	253
	5	207	233	207
	6	195	237	195
	7	213	213	231
	8	183	252	183
	9	194	194	220
	10	229	231	229
No. of Best			4	6

Av. Time			2.8994	0.2121
100	1	469	480	469
	2	430	430	460
	3	417	417	430
	4	432	436	432
	5	429	498	429
	6	424	435	424
	7	436	504	436
	8	414	419	414
	9	447	447	504
	10	457	457	525
No. of Best			4	6
Av. Time			9.0964	0.2484
500	1	2295	2307	2295
	2	2220	2306	2220
	3	2261	2270	2261
	4	2189	2345	2189
	5	2186	2186	2220
	6	2210	2210	2255
	7	2348	2408	2348
	8	2327	2327	2365
	9	2271	2271	2284
	10	2207	2290	2207
No. of Best			4	6
Av. Time			29.1419	0.4850
	1	4514	4531	4514

1000	2	4376	4376	4450
	3	4471	4505	4471
	4	4370	4542	4370
	5	4529	4578	4529
	6	4426	4578	4426
	7	4437	4492	4437
	8	4482	4571	4482
	9	4419	4498	4419
	10	4460	4496	4460
No. of Best			1	9
Av. Time			29.1454	1.0531
5000	1	22260	22325	22260
	2	22282	22311	22282
	3	22452	22595	22452
	4	22402	22402	22414
	5	22101	22101	22944
	6	22130	22555	22130
	7	22555	22735	22555
	8	22680	22680	22852
	9	22592	22659	22592
	10	22119	22459	22119
No. of Best			3	7
Av. Time			29.4675	10.7370
	1	44925	45153	44925
	2	44945	45401	44945
	3	44911	45094	44911

10000	4	44638	44986	44638
	5	44852	45228	44852
	6	45253	45253	45602
	7	44938	44938	45415
	8	44926	45313	44926
	9	44407	44407	45056
	10	44881	44946	44881
No. of Best			3	7
Av. Time			29.7156	29.7351
12000	1	54009	54009	54059
	2	54010	54650	54010
	3	54286	54453	54286
	4	54040	54882	54040
	5	53760	53837	53760
	6	54194	54194	54577
	7	53820	54723	53820
	8	53921	54111	53921
	9	53930	53930	53957
	10	53455	54125	53455
No. of Best			3	7
Av. Time			30.4109	29.4697

4.4.4 Summary of Experimental Evaluation of LS Methods

Since the Table (4.3) is very condensed we provide a summary for the table as shown in Table 4.4 below focusing on the number of times that produces the best value with their summation. This is done to all the above mentioned number of jobs, showing both SA and TS side by side for comparison purposes.

Table (4.4): The summary of Table (4.3) results.

n	SA	TS
10	3	7
50	4	6
100	4	6
500	4	6
1000	1	9
5000	3	7
10000	3	7
12000	3	7
Sum	25/80	55/80

In the Table (4.4) we give the activity of LS algorithms, (i.e. give the maximum number of jobs " n ") that the LS algorithms can solve the problem $(1 / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$ with 30 seconds.

Table (4.5): Activity of the LS methods.

Algorithm	Active until (maximum no. of jobs)
SA	12000
TS	12100

By comparing results of both LS methods namely, SA and TS and accurate revision of result tables provided earlier, we confidently deduce that for the functional problem $(1/ \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$ the TS has achieved the optimum results for a problem size comprising of 12100 jobs through 30 seconds.

Chapter Five

Conclusions and Future Works

Chapter Five

Conclusions and Future Works

5.1 Conclusions

In this thesis, we have improved approximate solutions for the problem of scheduling n independent jobs on single machine to minimize the sum of weighted number of early jobs and weighted number of late (tardy) jobs under fuzzy processing time. To resolve it proposed the AHR (Average High Ranking) method to obtain a processing time generated from fuzzy processing time, and then we calculated the costs and reach to total penalty cost as this method gave a faster speed and accuracy compared to previous work. We offer the results of tests of capacitive calculations of the method: simulated annealing (SA) and Tabu search (TS). The main conclusion to be extracted from our computation results is that ((SA), (TS)) of the local search methods can resolve $(1 / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$ problem of size (12000) jobs in 30 seconds. We found that the method (Simulated Annealing) can solve the problem to 12000 jobs in 30 seconds, Tabu Search method can solve the problem to 12100 jobs within 30 seconds and from the tables (4.5) and (4.6) we concluded that the Tabu Search method is the best than Simulated Annealing method. The example 3.1 in chapter three, which we solved by the algorithm in subsection 3.6, we concluded that this algorithm is a good heuristic which is giving us an optimal result (best sequence).

5.2 Future Works

1. We suggest the following problems for future research to be solved:
 - i. $1/r_i / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j}$.

ii. $F_2 / \sum_{j=1}^n N_{h_j E_j}.$

iii. $O_2 / \sum_{j=1}^n N_{w_j T_j}.$

2. We suggest using intelligent algorithms or local search algorithms such as genetic algorithms, ants, bees and swarm, etc... to solve these problems with fuzzy processing time.
3. We suggest using a parallel machine to solve these problems.

- [1] Abdul-Razaq, T.S. and Hummady, L.Z. (2010). Using Genetic Algorithm and Local Search to Solve Flow Shop NP-complete; AL-Mustansiriya Journal of Science, Vol. 21, No. 6, Pp. 501-518.
- [2] Johnson, S.M. (1954). Optimal Two and Three Stage Production Schedules with Setup Times Included; Naval Research Logistics, Vol. 1, No. 1, Pp. 61-68.
- [3] Jackson, J.R. (1955). Scheduling a Production Line to Minimize Maximum Tardiness; Research Report 43, University of California, Management Sciences Research Project, Office of Technical Services, Los Angeles.
- [4] Smith, W.E. (1956). Various Optimizers for Single-Stage Production; Naval Research Logistics Quarterly, Vol. 3, No. 1-2, Pp. 59-66.
- [5] Held, M. and Karp, R.M. (1962). A Dynamic Programming Approach to Sequencing Problems; Journal of the Society for Industrial and Applied Mathematics, Vol. 10, No. 1, Pp. 196-210.
- [6] Lomnicki, Z.A. (1965). A Branch and Bound Algorithm for the Exact Solution of the Three Machine Scheduling Problem; Journal of the Operational Research Society, Vol. 16, No. 1, Pp. 89-100.
- [7] Ignall, E. and Schrage, L. (1965). Applications of the Branch and Bound Technique to Some Flow Shop Scheduling Problems; Operations Research, Vol. 13, No. 3, Pp. 400-412.
- [8] Graham, R.L. (1969). Bounds on Multiprocessing Timing Anomalies; SIAM Journal on Applied Mathematics, Vol. 17, No. 2, Pp. 416-429.
- [9] Erenay, F.S. Sabuncuoglu, L. Toptal, A. and Tiwari, M.K. (2010). New Solution Methods for Single Machine Bicriteria Scheduling Problem: Minimization of Average Flowtime and Number of Tardy Jobs; European Journal of Operational Research, Vol. 201, No. 1, Pp. 89-98.

- [10] Hoogeveen, J.A. (1996). Single-Machine Scheduling to Minimize a Function of Two or Three Maximum Cost Criteria; *Journal of Algorithms*, Vol. 21, No. 2, Pp. 415-433,
- [11] Soroush, H. (2007). Minimizing the Weighted Number of Early and Tardy Jobs in a Stochastic Single Machine Scheduling Problem; *European Journal of Operational Research*, Vol. 181, No. 1, Pp. 266–287.
- [12] Kacem, I. Kellerer, H. and Lanuel, Y. (2015). Approximation Algorithms for Maximizing the Weighted Number of Early Jobs on a Single Machine with Non-Availability Intervals; *Journal of Combinatorial Optimization*, Vol. 30, No. 3, Pp. 403–412.
- [13] Al- zuwaini M.K, and Taher, H.K. (2018). Branch and Bound Method to Solve the Sum of Two Objective Functions; *Journal of Progressive Research in Mathematics (JPRM)*, Vol. 13, No. 1, Pp. 2151–2163.
- [14] Foster, F.G. and Boxma, O.J. (1986). Minimizing the Expected Weighted Number of Tardy Jobs in Stochastic Flow Shops; *Operations Research Letters*, Vol. 5, No. 3, Pp. 119–126.
- [15] Peha, J.M. (1995). Heterogeneous-Criteria Scheduling: Minimizing Weighted Number of Tardy Jobs and Weighted Completion Time; *Journal of Computers and Operations Research*, Vol. 22, No. 10, Pp. 1089–1100.
- [16] Lawler, E. (1983). Scheduling of Single Machine to Minimize the Number of Late Jobs; Technical Report No. UCB/CSD-83-139, University of California, Berkeley, Pp. 1–14.
- [17] Baptiste, P. (1999). Polynomial Time Algorithms for Minimizing the Weighted Number of Late Jobs on a Single Machine with Equal Processing Times; *Journal of Scheduling*, Vol. 2, No. 6, Pp. 245–252.

- [18] Sevaux, M. and Dautère-Pères, S. (2003). Genetic Algorithms to Minimize the Weighted Number of Late Jobs on a Single Machine; *European Journal of Operational Research*, Vol. 151, No. 2, Pp. 296–306.
- [19] Dautère-Pères, S. and Sevaux, M. (2003). Using Lagrangean Relaxation to Minimize the Weighted Number of Late Jobs; *Naval Research Logistics*, Vol. 50, No. 3, Pp. 273–288.
- [20] Dautère-Pères, S. and Sevaux, M. (2004). An Exact Method to Minimize the Number of Tardy Jobs in Single Machine Scheduling; *Journal of Scheduling*, Vol. 7, No. 6, Pp. 405–420.
- [21] Erel, E. and Ghosh, J. (2007). Batch Scheduling to Minimize the Weighted Number of Tardy Jobs; *Computers and Industrial Engineering*, Vol. 53, No. 3, Pp. 394–400.
- [22] Forst, F.G. (2010). Minimizing the Expected Weighted Number of Tardy Jobs with Non-Identically Distributed Processing Times and Due Dates; *AABRI Journals: Research in Business and Economics Journal*, Vol. 2, Pp. 1–7.
- [23] Adamu, M. and Adewumi, A. (2014). A Survey of Single Machine Scheduling to Minimize Weighted Number of Tardy Jobs; *Journal of Industrial and Management Optimization*, Vol. 10, No. 1, Pp. 219–241.
- [24] Zhao, C. Hsu, C.J. Cheng, S.R. Yin, Y. and Wu, C.C. (2014). Due Date Assignment and Single Machine Scheduling with Deteriorating Jobs to Minimize the Weighted Number of Tardy Jobs; *Applied Mathematics and Computation*, Vol. 248, Pp. 503–510.
- [25] Baker, K.R. and Trietsch, D. (2009). *Principles of Sequencing and Scheduling*; 1st Edition, John Wiley and Sons.

- [26] Marynissen, J. and Demeulemeester, E. (2018). Literature Review on Multi-Appointment Scheduling Problems in Hospitals; *European Journal of Operational Research*, (In Press).
- [27] Sterna, M. (2011). A Survey of Scheduling Problems with Late Work Criteria; *Omega*, Vol. 39, No. 2, Pp. 120–129.
- [28] Błażewicz, J. Ecker, K.H. Pesch, E. Schmidt, G. and Weglarz, J. (2001). *Scheduling Computer and Manufacture Processes*, 2nd Edition, Springer-Verlag Berlin Heidelberg.
- [29] Soper, A.J. and Strusevich, V.A. (2018). Schedules with a Single Preemption on Uniform Parallel Machines; *Discrete Applied Mathematics*, (In Press).
- [30] Hoogeveen, H. (2005). Multicriteria Scheduling; *European Journal of Operational Research*, Vol. 167, No. 3, Pp. 592-623.
- [31] Nagar, A. Haddock, J. and Heragu, S. (1995). Multiple and Bicriteria Scheduling: A Literature Survey; *European Journal of Operational Research*, Vol. 81, No. 1, Pp. 88-104.
- [32] Allahverdi, A. (2016). A Survey of Scheduling Problems with No-Wait in Process; *European Journal of Operational Research*, Vol. 255, No. 3, Pp. 665-686.
- [33] Lenstra, J.K. Rinnooy Kan, A.H.G. and Brucker, P. (1977). Complexity of Machine Scheduling Problems," *Annals of Discrete Mathematics*, Vol. 1, Pp. 343-362.
- [34] Shioura, A. Shakhlevich, N.V. and Strusevich, V.A. (2018). Preemptive Models of Scheduling with Controllable Processing Times and of Scheduling with Imprecise Computation: A Review of Solution Approaches; *European Journal of Operational Research*, Vol. 266, No. 3, Pp. 795–818.
- [35] Gu, M. Lu, X. Gu, J. and Zhang, Y. (2016). Single-Machine Scheduling Problems with Machine Aging Effect and an Optional

- Maintenance Activity; *Applied Mathematical Modelling*, Vol. 40, No. 21-22, Pp. 8862-8871.
- [36] Kononov, A.V. Lin, B.M.T. and Fang, K.T. (2015). Single-Machine Scheduling with Supporting Tasks; *Discrete Optimization*, Vol. 17, Pp. 69–79.
- [37] Perez-Gonzalez, P. and Framinan, J.M. (2018). Single Machine Scheduling with Periodic Machine Availability; *Computers and Industrial Engineering*, (In Press).
- [38] Cheng, J. Chu, F. Liu, M. Wu, P. and Xia, W. (2017). Bi-Criteria Single-Machine Batch Scheduling with Machine on/off Switching under Time-of-Use Tariffs; *Computers and Industrial Engineering*, Vol. 112, Pp. 721-734.
- [39] Karger, D. Stein, C. and Wien, J. (1999). Scheduling Algorithms, In: *Algorithms and Theory of Computation Handbook*; M.J. Atallah (ed.), Ch. 35, CRC Press, Pp. 1–20.
- [40] Chen, B. Potts, C.N. and Woeginger, G.J. (1998). A Review of Machine Scheduling: Complexity Algorithms and Approximability; In: *Handbook of Combinatorial Optimization*; D.Z. Du, and P.M. Pardalos (eds.), Kluwer Academic Publishers, Pp. 21-169.
- [41] Rossit, D.A. Tohme, F. and Frutos, M. (2017). The Non-Permutation Flow-Shop Scheduling Problem: A Literature Review; *Omega*, Vol. 77, Pp. 143-153.
- [42] Ruiz, R. and Vázquez-Rodríguez, J.A. (2010). The Hybrid Flow Shop Scheduling Problem; *European Journal of Operational Research*, Vol. 205, No. 1, Pp. 1-18.
- [43] Naderi, B. Fatemi Ghomi, S.M.T. Aminnayeri, M. and Zandieh, M. (2011). Scheduling Open Shops with Parallel Machines to Minimize Total Completion Time; *Journal of Computational and Applied Mathematics*, Vol. 235, No. 5, Pp. 1275-1287.

- [44] Lawler, E.L. Lenstra, J.K. Rinnooy Kan, A.H.G. and Shmoys, D.B. (1993). Sequencing and Scheduling: Algorithms and Complexity, In: Handbooks in Operations Research and Management Science; S.C. Graves et al., (eds.), Elsevier Science Publishers, Ch. 9, Vol. 4, Pp. 445–522.
- [45] Yeleswarapu, R.M. (2003). Scheduling of 2-Operation Jobs on a Single Machine to Minimize the Number of Tardy Jobs; M.S.I.E. Thesis, University of South Florida, College of Engineering, Department of Industrial and Management Systems Engineering.
- [46] Pinedo, M.L. (2016). Scheduling: Theory, Algorithms, and Systems; 5th Edition, Springer, New York.
- [47] Zhang, X. Liao, L. Zhang, W. Cheng, T.C.E. Tan, Y. and Ji, M. (2018). Single-Machine Group Scheduling with New Models of Position-Dependent Processing Times; Computers and Industrial Engineering, Vol. 117, Pp. 1-5.
- [48] Fibich, P. Matyska, L. and Rudova, H. (2005). Model of Grid Scheduling Problem; Technical Report WS-05-03, In: Exploring Planning and Scheduling for Web Service, American Association for Artificial Intelligence Press (AAAI), Menlo Park, California, Pp. 17-24.
- [49] Abdul-Razaq, T.S. (1987). Machine Scheduling Problems: a Branch and Bound Approach," Ph.D. Thesis, University of Keele, Department of Mathematics.
- [50] Schmidt, G. (2000). Scheduling with Limited Machine Availability; European Journal of Operational Research, Vol. 121, No. 1, Pp. 1-15.
- [51] Mostefa, M.S. (2014). The Branch-and-Bound Method, Genetic Algorithm, and Dynamic Programming to Determine a Safe Ship Trajectory in Fuzzy Environment; 18th International Conference on

- Knowledge-Based and Intelligent Information and Engineering Systems (KES), *Procedia Computer Science*, Vol. 35, Pp. 348-357.
- [52] Huang, R.H. and Yang, C.L. (2007). Single-Machine Scheduling to Minimize the Number of Early Jobs; *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, Pp. 955-957.
- [53] Moore, J.M. (1968). An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs; *Management Science*, Vol. 15, No. 1, Pp. 102-109.
- [54] Franch, S. (1982). *Sequencing and Scheduling an Introduction to Mathematics of Job Shop*; John Wiley & Sons, New York.
- [55] Cook, S.A. (1971). The Complexity of the Theorem-Proving Procedure; *STOC '71 Proceedings of the Third Annual ACM Symposium on Theory of Computing*, Pp. 151-158.
- [56] Shen, J. and Zhu, K. (2018). An Uncertain Single Machine Scheduling Problem with Periodic Maintenance; *Knowledge-Based Systems*, Vol. 114, Pp. 32–41.
- [57] Reeves, C.R. (1993). *Modern Heuristic Techniques for Combinatorial Problems*; 1st Edition, John Wiley and Sons.
- [58] Adewunmi, A.A. (2008). Selection of Simulation Variance Reduction Techniques through a Fuzzy Expert System; Ph.D. Thesis, University of Nottingham, School of Computer Science, UK.
- [59] Neogi, A. (2012). Studies on Performance Evaluation of Indian Education Using Fuzzy Systems; Ph.D. Thesis, the University of Burdwan, Department of Computer Science, West Bengal, India.
- [60] Bai, Y. and Wang, D. (1982). Fundamentals of Fuzzy Logic Control — Fuzzy Sets, Fuzzy Rules and Defuzzifications; In: *Advanced Fuzzy Logic Technologies in Industrial Applications*; Y. Bai, et al., (eds.), Springer, Ch. 2, Pp. 17–36.

- [61] Nagori, V. (2014). Design of Expert System Prototype for Analysing and Structuring Motivational Strategies on ICT Human Resources to Reduce Employee Turnover Ratio; Ph.D. Thesis, the University of Kadi Sarva Vishwavidyalaya, Department of Computer Science, Gujarat, India.
- [62] Fuller, R. (2005). Neural Fuzzy Systems; Abo Akademi University Press, Ch. 1, Pp. 11-141.
- [63] Mahalakshmi, P. and Ganesan, K. (2015). Mamdani Fuzzy Rule Based Model to Classify Sites for Aquaculture Development; Indian Journal of Fisheries, Vol. 62, No. 1, Pp. 110–115.
- [64] Iancu, I. (2012). A Mamdani Type Fuzzy Logic Controller, In: Fuzzy Logic- Controls, Concepts, Theories and Applications; E. Dadios, (ed.), InTech, Rijeka, Ch. 16, Pp. 325-350.
- [65] Dernoncourt, F. (2013). Introduction to Fuzzy Logic; Massachusetts Institute of Technology, Pp. 5–18.
- [66] Ghallab, S.A. Badr, N. Hashem, M. Salem, A.B. and Tolba, M.F. (2013). A Fuzzy Expert System for Petroleum Prediction; In: World Scientific and Engineering Academy and Society (WSEAS), Dubrovnik, Croatia, Vol. 2, Pp.77-82.
- [67] Negnevitsky, M. (2011). Artificial Intelligence: A Guide to Intelligent Systems; 3rd Edition, Addison Wesley, Ch. 4, Pp. 87–129.
- [68] Bhatnagar, R. Ghose, M.K. and Bhattacharjee, V. (2011). Predicting the Early Stage Software Development Effort Using Mamdani FIS; International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 2, No. 4, Pp. 1675-1678.
- [69] Jadhav, V.S. and Bajaj, V.H. (2012). Single Machine Scheduling Problem Under Fuzzy Processing Time and Fuzzy Due Dates; International Journal of Computer Engineering Science (IJCES), Vol. 2, No. 5, Pp. 12–19.

- [70] Baker, K.R. and Scudder, G.D. (1990). Sequencing with Earliness and Tardiness Penalties: A Review; *Operations Research*, Vol. 38, No.21, Pp. 22-36.
- [71] Aarts, E.H.L. and Lenstra, J.K. (1997). *Local Search in Combinatorial Optimization*; Wiley-Interscience Series in Discrete Mathematics and Optimization, 1st Edition, John Wiley and Sons.
- [72] Steinhöfel, K. Albrecht, A. and Wong, C.K. (2003). An Experimental Analysis of Local Minima to Improve Neighbourhood Search; *Computers and Operations Research*, Vol. 30, No. 14, Pp. 2157-2173.
- [73] Vredereld, T. (2002). *Combinatorial Approximation Algorithms: Guaranteed Versus Experimental Performance*; Ph.D. Dissertation, the Technical University of Eindhoven, Eindhoven, The Netherlands.
- [74] Gupta, S. and Rambha, M. (2011). Single Machine Scheduling With Distinct Due Dates Under Fuzzy Environment; *International Journal of Enterprise Computing and Business Systems*, vol. 1, no. 2, pp. 1–9.
- [75] Al-Harkan, I.M. (1997). *Algorithms for Sequencing and Scheduling*; University of Riyadh, College of Engineering, Industrial Engineering Department, King Saud, Riyadh, Saudi Arabia, Ch. 8, Pp. 8-1, 8-19.
- [76] Kirkpatrick, S. Gelatt Jr, C.D. and Vecchi, M.P. (1983). Optimization by Simulated Annealing; *Science*, Vol. 220, No. 4598, Pp. 671–680,
- [77] Černý, V. (1985). Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm; *Journal of Optimization Theory and Applications*, Vol. 45, No. 1, Pp. 41–51.
- [78] Metropolis, N. Rosenbluth, A.W. Rosenbluth, M.N. Teller, A.H. and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines; *The Journal of Chemical Physics*, Vol. 21, No. 6, Pp. 1087-1092.

- [79] Abdul-Razaq, T.S. Potts, C.N. and Van, Wassenhove. (1990). A survey of Algorithms for The Single Machine Total Weighted Tardiness Scheduling Problem; *Discrete App. Math.* Pp. 26235-253.
- [80] Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence; *Computers and Operations Research*, Vol. 13, No. 5, Pp. 533-549.
- [81] Gendreau, M. (2003). An Introduction to Tabu Search; In: F. Glover, G.A. Kochenberger (eds) *Handbook of Metaheuristics*, International Series in Operations Research and Management Science, Springer, Boston, Vol. 57, Pp. 37-54.
- [82] Beausoleil, R.P. (2002). A Tabu Search Approach for the Weighted Tardiness with Sequence-Dependent Setups in One-Machine Problem; *Journal of Mathematics: Theory and Applications*, Vol. 9, No. 1, Pp. 35–46.
- [83] Al-Anzi, F.S. and Allahverdi, A. (2006). A Hybrid Tabu Search Heuristic for the Two-Stage Assembly Scheduling Problem; *International Journal of Operations Research*, Vol. 3, No. 2, Pp. 109-119.
- [84] Huang, R.H. and Yu, T.H. (2013). A Novel Tabu Search Algorithm for Solving Robust Multiple Resource-Constrained Project Scheduling Problem; *International Journal of Computer and Communication Engineering*, Vol. 2, No. 2, Pp. 213-215.
- [85] Selt, O. (2016). Numerical Solution for Solving Scheduling Problem; *Global Journal of Pure and Applied Mathematics*, Vol. 12, No. 5, Pp. 4325–4333.
- [86] Glover, F. (1989). Tabu Search–Part I; *Operations Research Society of America Journal on Computing*, Vol. 1, No. 3, Pp. 190-206.

المُستخلص

في هذه الرسالة تناولنا دراسة مسألة جدولة n من المنتجات (jobs) على ماكينة واحدة لتصغير دالة متعددة الأهداف (MOF) مع وقت العمل الضبابي. والهدف من ذلك هو إيجاد أفضل متابعه التي تعطينا الحلول التقريبيه **Approximate solutions** او الحلول الأقرب للحل الأمثل **Near optimal solutions** لجدولة n من المنتجات المستقلة لتصغير لدالة الهدف المتكونه من مجموع العدد الوزني للنتائج المبكرة والعدد الوزني للنتائج المتأخرة تحت وقت العمل الضبابي ، وهذه المسألة تمثل بالصيغة:

$$(1 / \sum_{j=1}^n N_{h_j E_j} + \sum_{j=1}^n N_{w_j T_j})$$

لحلها اقترحنا طريقة المعدل العالي **Average High Ranking (AHR)** للحصول على وقت عمل متولد من وقت العمل الضبابي وحساب الكلف والوصول الى الكلفه الكليه .

بما أن النوع المعقد لمسألتنا في الحالة الطبيعية (بدون الضبابية) هي **Strongly NP-** (**hard**) تم استخدام طرائق البحث المحلية لإيجاد حلول تقريبية وهي :

Simulated Annealing ، **Tabu Search** . عمليا فقد وجد بأن طريقة **Simulated Annealing** يمكن ان تحل المسألة الى 12000 نتاج خلال 30 ثانية و طريقة **Tabu Search** يمكن ان تحل المسألة الى 12100 نتاج خلال 30 ثانية. من مقارنة النتائج وجد بأن طريقة **Tabu Search** كانت أفضل من طريقة **Simulated Annealing** .

جميع الأساليب المعتمدة في هذا البحث:

خوارزمية **Average High Ranking** وخوارزميات البحث المحلية
(**Simulated Annealing** و **Tabu Search**)

قد تمت برمجتها باستخدام برنامج **MATLAB 7.10.0 R2014a** وكان جهاز الكمبيوتر الشخصي المعالج عبارة عن ذاكرة وصول عشوائي سعتها 4 غيغابايت مع وحدة معالجة مركزية تبلغ سرعتها 2.5 غيغاهرتز.



جمهورية العراق
وزارة التعليم العالي والبحث العلمي
جامعة ذي قار
كلية علوم الحاسوب والرياضيات



مسألة جدولة الماكنة الضبابية

رسالة مقدمة الى مجلس كلية علوم الحاسوب والرياضيات جامعة ذي قار
وهي جزء من متطلبات نيل درجة الماجستير
في علوم الرياضيات

من قبل
مروة محمد قاسم

بإشراف
أ.م.د محمد كاظم زغير الزويني
أ.م.د شاكر كاظم علي الشريف