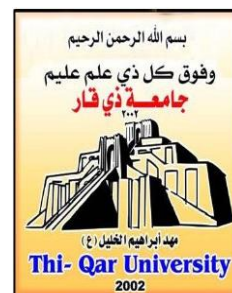




Republic of Iraq
Ministry of Higher Education
and Scientific Research
University Of Thi-Qar
College of Education for Pure Sciences



An n-Jobs One Machine Scheduling for Minimization the Sum of Two Criteria

A Thesis Submitted to
The Department of Mathematics, College of Education
for Pure Sciences, University of Thi-Qar as a Partial
Fulfillment of the Requirements for the Degree
of Master of Sciences in Mathematics

by

AZHAR MAHDI ABADI

(B.Sc. in Mathematics University of Thi-Qar 2014)

Supervised by

Assist. Prof. Mohammed K. Al-Zuwaini

(Ph.D.)



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

رَبِّ أَوْزِعْنِي أَنْ أَشْكُرَ
نِعْمَتَكَ الَّتِي أَنْعَمْتَ عَلَيَّ وَعَلَىٰ وَالِدَيَّ وَأَنْ أَعْمَلَ صَالِحًا
تَرْضَاهُ وَأَدْخِلْنِي بِرَحْمَتِكَ فِي عِبَادِكَ الصَّالِحِينَ ﴿١٩﴾

صَلَّى
الْعَظِيمِ

سورة النمل من الآية ﴿19﴾



DEDICATION

*Every challenging work needs grace of Allah and self efforts
as well as
guidance of all great teachers especially those who
were very close to our hearts*

*I dedicate this work to my family especially
father and mother*

*whose affection ,love ,encouragement, and prays of day
and night make me able to get such success and honour*

along with all hard working and respected teacher

ACKNOWLEDGEMENTS

All praise and thanks be to ALLAH, the most gracious and most merciful, for his grace that enabled me to continue the requirements of my study.

I would like to express my deepest gratitude to my supervisor **Dr. Mohammed K. Al-Zuwaini** for his unwavering support, patience, help and prudent guidance throughout this work. His support and guidance have allowed me to complete one of the goals in my life. Without his help, completion of this goal would have been more difficult.

My thanks and pride to my colleagues in the higher studies. Also, my thanks go to everyone helped me throughout the fulfillment of this research especially Dr. Hussein Jamel Mutashar .

I would like to extend my thanks to my family who supported me in this journey. Thank you.

AZHAR

Supervision Certification

I certify that this thesis entitled (**An n-Jobs One Machine Scheduling for Minimization the Sum of Two Criteria**) was prepared under my supervision at Department of Mathematics/College of Education for Pure Sciences/University of Thi-Qar, as a partial fulfillment of the requirements for the degree of Master of Science (**M.Sc.**) in Mathematics.

Signature:

Supervisor: **Assist. prof. Mohammed Kadhim Z. Al-Zuwaini (Ph.D)**

Date: 21 / 6/2017

Address: **College of Computer Science and Mathematics, Thi-Qar University**

Head recommendation of the mathematics department

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

Signature:

Name: **Assist. Prof. Waleed K. Jaber**

Head of the Department of Mathematics, College of Education for Pure Sciences, Thi-Qar University

Date: 21/ 6 /2017

Abstract

In this thesis, we considered the problem of scheduling n jobs on a single machine. The aim of this study is to find the optimal and near optimal solutions for the sum of cost of total flow time and maximum late work with unequal ready time, this problem denoted by $1/r_j/\sum_{j=1}^n F_j + V_{max}$. The problem is strongly NP-hard, the branch and bound method was using to find optimal solution. Two lower bounds (LB₁, LB₂) are proposed each of them based on decompose the problem into two sub problems. The lower bounds of the problem is the sum of the lower bounds of the two sub problems. A heuristic which gives an upper bound in the root node of BAB algorithm was proposed, its effective in finding an optimal or near optimal schedule. Also, we proved some special cases of the problem which lead to optimal solution, three dominance rules were stated and proved. The results of extensive computational tests show that the proposed BAB algorithm is effective in solving problems up to (35) jobs at a time less than or equal to (30) minutes.

We apply two local search methods to find near optimal solutions: Artificial Fish Swarm Algorithm (AFSA) and Fruit Fly Optimization Algorithm (FOA) .

Computational experience found that these local search methods can solve the problem up to (6000) jobs with reasonable time, also found that: The AFSA has better results for the problem of size less than or equal to (35) jobs, but for the problems of size larger than (35) jobs, The FOA has the best results. All methods used in this research are programmed by using a programming language (MATLAB Language).

List of Symbols and Abbreviations

AFSA	Artificial Fish Swarm Algorithm
BAB	Branch and Bound
DP	Dynamic programming
FOA	Fruit Fly Optimization Algorithm
f_{max}	Maximum of function
$\sum f_j$	The total cost
$\sum F_j$	The total flow time
ILB	Initial Lower bound
JIT	Just in time
LA	Lawler Algorithm
LB	Lower bound
m	Number of machines
MOF	Multiple Objective Function
n	Number of jobs
N	Set of jobs
NP	Non-deterministic polynomial
P_{mtn}	Preemption
$prec$	Precedence constraints
r_j	Release date of job j
$R_i(t)$	The earliest beginning time of job i at time t
SPT	Shortest processing time
SRD	Shortest release date
SRPT	Shortest remaining processing time
V_{max}	The maximum late work
UB	Upper bound

Contents

Introduction.....	1
-------------------	---

Chapter One: Description of Machine Scheduling Problem

1.1 Definition of the Scheduling Problem.....	6
1.2 Basic Definitions	6
1.3 Basic Scheduling Concepts.....	7
1.4 Classification of Scheduling Problem.....	9
1.4.1 Single Machine Scheduling.....	10
1.4.2 Parallel Machine Scheduling.....	10
1.4.3 Flow Shop Scheduling(F_m).....	10
1.4.4 Job Shop Scheduling(J_m).....	10
1.4.5 Open Shop Scheduling(O_m).....	11
1.5 Fundamental Theorem and Algorithms.....	12
1.6. Solution Approaches for Machine Scheduling Problems.....	13
1.6.1 Exact Methods.....	13
1.6.2 The Heuristic Methods.....	13
1.7 Problem Classes and Computational Complexity.....	14

Chapter Two: Branch and Bound Method to Solve Multi Objectives

Function

2.1 Introduction	17
2.2 Problem Formulation.....	17
2.3 Problem Decomposition(P)	18

2.4 Special Cases.....	20
2.5 Branch and Bound Method.....	21
2.5.1 The Upper bound (UB) Procedure.....	22
2.5.2 The Lower Bound (LB) Procedure	23
2.5.2.1 The First Lower Bound (LB_1).....	23
2.5.2.2 The second Lower Bound (LB_2).....	23
2.6 Dominance Rules.....	25
2.7 Computational Experiments	28

Chapter Three: AFSA/ FOA To Solve Multiple Objective Function

3.1 Introduction	33
3.2 Artificial Fish Swarm Algorithm (AFSA).....	33
3.3 The basic functions of AFSA.....	36
3.4 Fruit Fly Optimization Algorithm(FOA).....	39
3.5 Applications Fruit Fly Optimization Algorithm.....	41
3.6 Computational Results of Local Search Algorithms and Comparison.....	41
3.6.1 Test Problem.....	41
3.6.2 Computational Results	41
3.6.3 Summary of Experimental Evaluation of Local Search Methods...	50

Chapter Four: Conclusion and Future Works

4.1 Conclusion.....	54
4.2 Future work.....	54
References.....	56

Introduction

Operational research (OR) encompasses a wide range of problem-solving techniques and methods applied in the pursuit of improved decision-making and efficiency. It is applied in almost all sectors ranging from manufacturing to the service industry. OR started just before World War II in Britain with the establishment of teams of scientists to study the strategic and tactical problems involved in military operations. The objective was to find the most effective utilization of limited military resources by the use of quantitative techniques. Following the war, numerous peacetime applications emerged, leading to the use of OR and management science in many industries and occupations. OR is the study of mathematical models for complex organizational system. Optimization is a branch of OR which uses mathematical techniques such as linear and nonlinear programming to derive values for system variables that will optimize performance. Scheduling theory is subject of interest in several areas of computer science and arises in many other fields. Scheduling theory has been developed to solve problems occurring in for instance production facilities. The first scheduling paper appeared in 1954, the basic scheduling problem has been formulated by differentiating between machine environments, side-constraints, and objective functions. Scheduling is the task of determining when each operation is to start and finish, methods for solving scheduling problems depend on the computational complexity. For machine (processor) scheduling problems a classification scheme has been introduced. After describing it we survey some polynomial solvable and NP-hard scheduling problems. This study focuses on one of the most important subjects in OR which is called (Machine Scheduling Problem) [1]. To solve machine scheduling problems one tends to use optimization algorithms which always find an

optimal solution. However, not for all optimization problems, polynomial time optimization algorithms can be constructed. This is because, some of these problems, which cannot be solved in polynomial time algorithms are NP-hard. A variety of algorithms and formulations have been developed for various classes of problems. Scheduling is one of such classes of problems. Solving a machine schedule problem means finding the decision that makes us to determine which job should be sequenced first and on which machines. An exact method has been used to solve machine scheduling problem. The important approach was BAB method, this method is probably the solution technique most widely used in scheduling. It is an approach of optimal which is applied to scheduling problem. The problem of sequencing n jobs on one machine under different assumptions and multiple criteria . In this thesis, we consider the problem $1/ r_j/ \sum_{j=1}^n F_j + V_{\max}$. The importance of studying this problem is being addressed to reduce the flow of the time of all jobs, in addition to reducing the maximum of the late work for the jobs (that is the amount of processing performed after its due date). Flow time, defined as the time elapsed since a request is submitted until it finishes, is one of the most relevant measures of the quality of service received.

The following are some of Literature Review:

For the problem $1 / r_j / \sum_{j=1}^n F_j$, several researchers have studied and proved some dominance properties and proposed a number of algorithms. Lenstra et al. [1] show that its NP-hard. If all jobs are released at the same time, the problem reduces to the $1 / r / \sum_{j=1}^n F_j$, which can be solved optimally by the well-known Shortest Processing Time (SPT) rule Smith [2]. The preemptive version, $1/r_j, pmtn / \sum_{j=1}^n F_j$ of the problem can be solved optimally in polynomial time by the Shortest Remaining Processing Time (SRPT) rule. Baker [3]. The solution of the preemptive

problem provides a lower bound for the objective value of the non-preemptive problem. Ahmadi and Bagchi [4] shown that this lower bound dominates all other known lower bounds for the $1 / r_j / \sum_{j=1}^n F_j$ problem, Chandra et al. [5] have developed branch-and-bound algorithms. Chu [6] have proposed approximation algorithms for the problem. Al-Zuwaini [7] used efficient branch and bound technique with a suitable lower bound and proved some dominance rules for $1 / r_j / \sum_{j=1}^n F_j$ problem. Several researchers studied this criteria ($\sum F_j$) with other criteria to consider multi criteria for example $(\sum_{j=1}^n F_j + \sum_{j=1}^n U_j)$ [8]. The problem $1 / r_j / \sum_{j=1}^n F_j$ has many applications, for example, in network systems and parallel computing [9].

The problem $1 / r_j / V_{max}$ is NP-hard, Lawler developed an algorithm (LA) and proved that LA solved $1 // f_{max}$ in $O(n^2)$ time [10], in our study $f_{max} = V_{max}$. The parameter V_j was first introduced by Blazewicz [11], who called it "information loss", referring to a possible application of the performance measures based on it. The phrase "late work" was proposed by Potts and Van Wassenhove [12]. Applications of the late work minimization problems arise in control systems [11]. Also several researchers studied this criteria (V_{max}) with other criteria to study multi criteria scheduling, Shahbaa and Saad [13], studied a multi criteria scheduling problem on single machine to minimize $1 // \sum_{j=1}^n C_j + \sum_{j=1}^n T_j + V_{max}$ by using exact and local search methods. Al-Nuaimi [14], proposed a branch and bound algorithm for minimizing the problem $1 // \sum_{j=1}^n V_j + V_{max}$. T.S. Abdul-Razaq and Karar F [15], proposed some algorithms to find exact solution in hierarchical and simultaneously cases for $(1 // E_{max} + V_{max})$. T.S. Abdul-Razaq and Sh

Fandi [16], studied multi criteria problem $1 // (\sum_{j=1}^n C_j, V_{max})$ to find efficient solutions for simultaneous case.

This thesis consists of four chapters , chapter one gives a description of machine scheduling problems, basic definition and scheduling concepts, classification of scheduling problem, fundamental theorems and algorithms. Also, in this chapter some solution approaches for machine scheduling problems, problem classes and computational complexity. In chapter two, discussed the problem of scheduling n jobs on a single machine to minimize the sum of total cost of flow time and maximum late work with unequal release dates and using branch and bound method to find optimal solution to our scheduling problem. Lower bound, heuristic method (upper bound), special cases and dominance rules will also be included. In chapter three, we present two local search methods which include Artificial Fish Swarm Algorithm and Fruit Fly Optimization Algorithm. These methods are well studied to deal with multi criteria problem to find near optimal solution to our scheduling problem; performance of these heuristic methods is evaluated on a large set of test problem. Finally, Chapter four is the conclusion and the lookout for the future works.

Chapter One

Description of Machine Scheduling

Problem

1.1 Definition of The Scheduling Problem

There are many definitions for machine scheduling, but the simplest one for understanding is that, the allocation of resources over time to perform a collection of tasks [3]. Resources and tasks are called machines and jobs respectively and both of them can take many forms. For example: We can consider a computer (or computers) as a machine (or machines) and the programs that run on that computer (or computers) as jobs. Another example: we can consider hospital equipment's as machines and the patients in that hospital as jobs.

Scheduling, general speaking, means to assign machines to jobs in order to complete all jobs under the imposed constraints. The problem is to find the optimal processing order of these jobs on each machine to minimize the given objective function. There are two general constraints in classical scheduling theory [17]. Each job is processed by; at most, one machine at a time and each machine is capable of processing at most one job at a time. A schedule is feasible if it satisfies the various requirements relating to the specific problem type. The problem type is specified by the machine environment, the job characteristics and an optimality criterion.

1.2 Basic Definition :

Definition (1.2.1): 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 [18].

Definition (1.2.2): suppose we have a problem P , any schedule $S \in \delta$ (where δ is the set of all schedules) is said to be feasible if it satisfies the constraints of the problem P [18].

Definition (1.2.3): A schedule S is said to be efficient if there does not

exist another schedule S' satisfying $f_i(S') \leq f_i(S), i = 1, 2, \dots, k$, with at least one of the above holding as a strict inequality. Otherwise, S is said to be dominated by S' [19].

Definition (1.2.4): 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 makespan (C_{max}) and tardiness based performance measures(T_i) [20].

Definition (1.2.5): A non-regular performance measure is usually not monotone function of the job completion times. An example of such a measure is job earliness(E_i) [20].

1.3 Basic Scheduling Concepts:

Some important notations are introduced first, The performance criteria are concentrated on without elaborating on the machine environments. It is assumed that there are n jobs which are denoted by $j_1, j_2, \dots, j_n$, are to be scheduled on a set of machines that are continuously available from time zero onwards and that can handle only one job at a time.

In this thesis, we only state here the notation which is used for the single machine, jobs j_i ($i = 1, 2, \dots, n$) has:

Job (j): 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 [21].

Machine (m): A machine is available to execute jobs and tasks. Different machine environments exist, such as single machine and parallel machines[21].

Preemption ($pmtn$): 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 [21].

Processing time (p_{ij}): The processing time of job j on machine i . The subscript i is omitted if job j is only to be processed on one a given machine [22].

Due date (d_j): The date when the jobs should ideally be completed, the completion of the job after its due date is allowed, but a penalty is incurred. When the due date absolutely must be met, it is referred to as deadline ($\overline{d_j}$), and when the due date is constant for all jobs, then called common due date [23].

Release date (r_j): Also known as ready time, the point of time that job arrives at the machine and thus processing starts [23].

Weight (W_j): Denoting the importance of a job j relative to another job [24].

Now for a given sequence of jobs, we can compute for job j :

- 1- The earliest completion time $C_j = \sum_{i=1}^j p_i$.
- 2- The flow time $F_j = C_j - r_j$.
- 3- The lateness $L_j = C_j - d_j$.
- 4- The tardiness $T_j = \max\{C_j - d_j, 0\}$.
- 5- The earliness $E_j = \max\{d_j - C_j, 0\}$.
- 6- The late work $V_j = \min\{T_j, p_j\}$.
- 7- The idle time

$$I_j = \begin{cases} r_j & \text{if } j = 1 \\ r_j - C_{j-1} & \text{if } r_j > C_{j-1}, j = 2, \dots, n \\ 0 & \text{otherwise} \end{cases}$$

The following criteria of performance appear frequently in the literature [25]. For a given scheduling σ we compute:

- $C_{max}(\sigma) = \max_j(C_j)$ (maximum completion time).
- $E_{max}(\sigma) = \max_j(E_j)$ (maximum earliness).
- $L_{max}(\sigma) = \max_j(L_j)$ (maximum lateness).
- $T_{max}(\sigma) = \max_j(T_j)$ (maximum tardiness).
- $V_{max}(\sigma) = \max_j(V_j)$ (maximum late work).
- $\sum W_j C_j(\sigma)$ (total weighted completion times).
- $\sum W_j E_j(\sigma)$ (total weighted earliness).
- $\sum W_j T_j(\sigma)$ (total weighted tardiness).
- $\sum W_j V_j(\sigma)$ (total weighted late work).

1.4 Classification of Scheduling Problem

The standard three-field notation is used to describe a scheduling problem. Such a notation of problem types would greatly facilitate the presentation and discussion of scheduling problem. A notation proposed by Graham et al. [26]. This notation is composed of three fields $\alpha/\beta/\gamma$. Let n and m be the number of jobs and machines, respectively. Let i and j be a job and machine, respectively let (i, j) be the operation of job i on machine j . The subscript j is omitted if job i is only to be processed on one given machine. With respect to $\alpha/\beta/\gamma$ notation, they have the following meaning:

The first field α describes the machine environment, due to the large variety of machine scheduling problems, several classification schemes have been proposed based on different dimensions. An example of such a dimension is the number of available machines and the way how these machines are organized. The single machine sequencing problem is the simplest one: All jobs have to be processed on a single machine and no

two jobs can be processed simultaneously[23]. There are five classes of machine scheduling problem[27]:

1.4.1 Single Machine Scheduling[28]:

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

1.4.2 Parallel Machine Scheduling [29].

Parallel machine scheduling involves scheduling a set of jobs on two or more machines that work in parallel with each other. The machine perform identical operations and may or may not operate at the same pace.

1.4.3 Flow Shop Scheduling [27].

A flow shop scheduling problem consists of two or more machines and a set of tasks(jobs) that must be processed on each of these machines. This arrangement is called a flow shop because it products flow along a specific unidirectional path. Each product must be processed on each machine in the same order e.g. 1st - machine 1, 2nd – machine 2, ..., mth – machine m . The processing times for each job can vary from machine to machine and the processing times on each machine can vary from job to job.

1.4.4 Job Shop Scheduling[27].

A job shop consists of two or more than two machines that perform specific operations, and a set of jobs that must be processed on some or all of these machines. Unlike the flow shop, there is no fixed path that the

products must follow through the system. Therefore, the order of operations is not fixed.

1.4.5 Open Shop Scheduling [30]

The open shop is a more general case of the job shop scheduling problem. There are n jobs consisting of m operations to be processed in m machines. The sequencing of each job through the machines is different and finding the optimal sequence for each of the n jobs is also part of the problem. Since the sequence of operations within each job has to be determined in addition to the jobs processing schedule.

The second field $\beta \subseteq \{pmtn, r_j, prec\}$ indicates certain job characteristics:

- If $pmtn$ is present, then preemptions are allowed; otherwise, no preemptions are allowed.
- If r_j is present, then each job may have different release dates.

Otherwise, all jobs arrive at time 0.

- If $prec$ is present, then there is precedence relation among the jobs.

The third field $\gamma \in \{f_{max}, \sum f_j\}$ refers to the optimality criterion or the objective, the value which is to be optimized (minimized).

The parameter $\gamma \in \{f_{max}, \sum f_j\}$ which are defined as follows: $f_{max} \in \{C_{max}, L_{max}, T_{max}, E_{max}, V_{max}, V_{max}^w, T_{max}^w\}$.

$\sum f_j \in \{\sum C_j, \sum W_j C_j, \sum E_j, \sum T_j, \sum V_j, \sum W_j T_j, \sum U_j, \sum W_j U_j\}$.

To illustrate the three field descriptors, three examples are presented:

- 1) $1/prec/\sum F_j$: is the problem of scheduling jobs with precedence constraints, on a single machine to minimize the total flow time.
- 2) $F_2/r_j/\sum U_j$: is the problem of scheduling jobs with release dates on a two machine flow shop to minimize the number of tardy jobs.
- 3) $P_2/r_j/\sum W_j C_j$: is the problem of scheduling jobs with release dates on identical parallel two machines to minimize the total weighted completion time.

1.5 Fundamental Theorems and Algorithms:

Theorem (1.1) (Jackson1955) The $1//L_{max}$ problem is minimized by sequencing the jobs according to the earliest due date (EDD) rule, that is, in order of non-decreasing d_i [31].

Theorem (1.2) (Smith 1956) The $1//\sum C_i$ problem is minimized by sequencing the jobs according to the shortest processing time (SPT) rule, that is, in order of non-decreasing p_i [2].

Theorem(1.3)(Lawler 1973). The $1// f_{max}$ problem, f_{max} is minimized as follows: while there are unassigned jobs, assign the job that has minimum cost when scheduled in the last unassigned position in that position [10].

Lawler algorithm which solves the $1/prec/ f_{max}$ problem or $1// f_{max}$ problem where $f_{max} \in \{ L_{max}, T_{max}, V_{max}, V_{max}^w, T_{max}^w \}$ [10].

Theorem(1.4). The $1// E_{max}$ problem is solved by sequencing the jobs according to the minimum slack time (MST) rule, that is, in order non-decreasing $d_i - p_i$ [32].

Lawler algorithm

Step(1):Let $N=\{1,\dots,n\}$, $\sigma= (\varphi)$ and F set of all jobs with no successors.

Step(2):Let j^* such that $f_{j^*}(\sum_{i \in N} p_i) = \min_{j \in F} \{f_j(\sum_{i \in N} p_i)\}$.

Set $N=N-\{j^*\}$ and sequence j^* in σ , i.e $\sigma=(j^*, \sigma)$.

Modify F to represent the new set of schedulable jobs.

Step(3):If $N = \varphi$ stop, otherwise go to step(2).

For a multi criteria scheduling problem, Van Wassenhove and Gelder [33], proposed a pseudo-polynomial algorithm for finding all efficient schedules with respect to $\sum C_i$ and L_{max} . Their algorithm searches all

possible L_{max} values. Since a given L_{max} value imposes job deadline, the backward algorithm of Smith [2] is used to solve the corresponding $1/\bar{d}_i/\sum C_i$ problem. Hoogeveen and Van de Velde [34] provided a generalization to the case that the two criteria are $\sum C_i$ and f_{max} , where f_{max} is regular cost function.

1.6. Solution Approaches for Machine Scheduling Problems

The best known methods of optimal solution or near optimal solution for machine scheduling problems are stated as follow:

1.6.1 Exact Methods:

a. Complete Enumeration Method[35].

Complete Enumeration is the simple method that generate one by one, searching for an optimal solution. For a single machine problem of n jobs, there are $n!$ different schedules. This method may take considerable time and this time is very large even for relatively small values of n .

b. Branch and Bound (BAB) Method [36].

BAB method is an exact method for finding optimal solution to an NP-hard problem. To minimize an objective function for a particular scheduling problem, the BAB method successive partition subsets using a branching procedure and compute bounds using a lower bound procedure and by these procedures the subsets are excluded which are not found to include any optimal solution.

c. Dynamic Programming (DP)[37].

DP method is implicit enumeration technique and it can be applied to any optimization problem which can be divided into a number of stages and at each stage the solution is derived by recurrence relation from the solution at the preceding stages.

1.6.2 The Heuristic Methods

It is clear from above that the computational requirement to solve a

particular problem using BAB or DP algorithm might require much time than it is useful consuming for large problems, in fact, even for relatively small problem there is no guarantee that a solution can be found quickly. Reeves[38] 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 .

1.7 Problem Classes and Computational Complexity

Some problems are polynomials solvable, which means that there exists an algorithm requiring 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[17]. 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 there 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 problems can be classified into three classes:

- (1) The class P (Polynomial), which contains all decision problems that are polynomially 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 subclass from class NP. The scheduling problem that will be consider in this thesis can be solved by non-deterministic algorithm and thus is a

member of NP. Cook [39] proved 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 so far P or NP, for instance the problem $1/d_j = d / \sum_{j=1}^n (W_j E_j + W'_j T_j)$ [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[30].

Chapter Two
Branch and Bound Method to Solve
Multiple Objective Function

2.1 Introduction

In this chapter, we consider the scheduling n independent jobs on a single machine with unequal release dates to minimize the multi objective problem.

Our objective is to find a schedule that minimize sum of total flow time and maximum late work by using BAB method. This problem is denoted by $1/r_j/\sum_{j=1}^n F_j + V_{max}$, and it is complete NP-hard because $1/r_j/\sum_{j=1}^n F_j$ NP-hard [1].

2.2 Problem Formulation

We consider the n -job, one-machine scheduling problem. Each job of the set $N = \{1, 2, \dots, n\}$ is to be processed on a single machine which performs at most one job at a time. No preemption is allowed. Each job j becomes available at its release date r_j , requires a processing time p_j and due date d_j . This study concerns the one machine scheduling problem with multiple objective function denoted by $1/r_j/\sum_{j=1}^n F_j + V_{max}$. For any given schedule $(1, 2, \dots, n)$, the flow time of job j , F_j and maximum late work V_{max} , can be defined respectively as: $F_j = C_j - r_j$, where C_j be a completion time for job j , given by the relationship: $C_1 = p_1 + r_1$, $C_j = \max\{C_{j-1}, r_j\} + p_j$, $j=2, 3, \dots, n$ and $V_{max} = \max\{V_j\}$, $V_j = \min\{T_j, p_j\}$, and $T_j = \max\{C_j - d_j, 0\}$, for $j=1, 2, \dots, n$. The objective is to find a schedule that minimize the sum of total flow time and maximum late work with unequal release dates on single machine (i.e. minimize the objective function $(\sum_{j=1}^n F_j + V_{max})$). The problem is strongly NP-hard because the $1 / r_j / f_{max}$ problem is NP- hard [41].

Our scheduling problem can be state mathematically more precisely as follows:

Given a schedule $\delta = (1, 2, \dots, n)$, then for each job $j \in \delta$ can be calculate the flow time F_j and the late work V_j . The objective is to find a schedule $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ belong to a neighborhood of δ that minimize the total cost $Z(\pi)$, where $Z(\pi) = \sum_{j=1}^n F_{\pi(j)} + V_{max}(\pi)$.

Let S be a set of all schedules, $|S| = n!$ then we can formulate our problem in mathematical form as:

$$\begin{aligned}
 M = \min_{\pi \in S} \{Z(\pi)\} &= \min_{\pi \in S} \left\{ \sum_{j=1}^n F_{\pi(j)} + V_{max}(\pi) \right\} \\
 \text{Subject to:} & \\
 \left. \begin{aligned}
 C_{\pi(j)} &\geq r_{\pi(j)} + p_{\pi(j)} & j = 1, 2, \dots, n \\
 C_{\pi(j)} &\geq C_{\pi(j-1)} + p_{\pi(j)} & j = 2, 3, \dots, n \\
 F_{\pi(j)} &= C_{\pi(j)} - r_{\pi(j)} & j = 1, 2, \dots, n \\
 F_{\pi(j)} &\geq p_{\pi(j)} & j = 1, 2, \dots, n
 \end{aligned} \right\} (A) \\
 \left. \begin{aligned}
 C_{\pi(j)} &> p_{\pi(j)} & j = 1, 2, \dots, n \\
 V_{\pi(j)} &\leq T_{\pi(j)} & j = 1, 2, \dots, n \\
 V_{\pi(j)} &\leq p_{\pi(j)} & j = 1, 2, \dots, n \\
 r_{\pi(j)} \geq 0, d_{\pi(j)} > 0, V_{\pi(j)} \geq 0, p_{\pi(j)} > 0 & j = 1, 2, \dots, n
 \end{aligned} \right\} (P)
 \end{aligned}$$

Where $\pi(j)$, indicates the position of job j the requesting in π and S means the arrangement of all identified schedules.

2.3 Problem Decomposition

Since problem P is NP-hard, then we can decomposed it in to three sub problems which are simple structure as following:

- 1- $1/r_j / \sum_{j=1}^n F_j$.
- 2- $1/r_j / V_{max}$.
- 3- $1 / \sum_{j=1}^n C_j + V_{max}$.

Each of them is NP-hard and was studied by many researchers [1,2,3,6,7,10,11,16].

For sub problem (1) we can formulate mathematical form as:

$$\left. \begin{array}{l} M_1 = \min_{\pi \in S} \left\{ \sum_{j=1}^n F_{\pi(j)} \right\} \\ \text{Subject to :} \\ \quad (A) \\ \quad r_{\pi(j)} \geq 0, p_{\pi(j)} > 0, d_{\pi(j)} > 0 \quad j = 1, 2, \dots, n \end{array} \right\} (P_1)$$

For sub problem (2) we can formulate mathematical form as:

$$\left. \begin{array}{l} M_2 = \min_{\pi \in S} \left\{ \max\{V_{\pi(j)}\} \right\} \\ \text{Subject to:} \\ \quad C_{\pi(j)} \geq r_{\pi(j)} + p_{\pi(j)} \quad j = 1, 2, \dots, n \\ \quad C_{\pi(j)} \geq C_{\pi(j-1)} + p_{\pi(j)} \quad j = 2, \dots, n \\ \quad V_{\pi(j)} \leq T_{\pi(j)} \quad j = 1, 2, \dots, n \\ \quad V_{\pi(j)} \leq p_{\pi(j)} \quad j = 1, 2, \dots, n \\ \quad r_{\pi(j)} \geq 0, V_{\pi(j)} \geq 0, p_{\pi(j)} > 0, d_{\pi(j)} > 0 \quad j = 1, 2, \dots, n \end{array} \right\} (P_2)$$

Theorem 2.1: [24] If N_1, N_2 , and N_3 are the minimum objective function values of (P_1) , (P_2) and (P) respectively then $N_1 + N_2 \leq N_3$.

Theorem 2.2:[2] If $r_j = r$ for all $j \in N$, then SPT schedule gives the optimal solution for the problem $1/r_j / \sum_{j=1}^n F_j$.

Theorem 2.3:[10] If $r_j = r$ for all $j \in N$, then LA gives the optimal for $1/r_j = r/V_{max}$.

2.4 Special Cases

Finding a special case for scheduling problem means finding an optimal schedule directly without using enumeration methods. A special case if it exists depends on satisfying some conditions in order to make the problem easily solvable. These conditions depend on the objective function as well as the jobs. We give some special cases for problem P.

Case (1): If $r_i = r, p_i = p, d_i = d$, then $1/r_i / \sum_{i=1}^n F_i + V_{max}$ problem is independent on sequence.

Proof :

$$C_i = r + ip, F_i = r + ip - r = ip, \sum_{i=1}^n F_i = \sum_{i=1}^n ip$$

$$V_i = \min\{T_i, p\} = \min\{r + ip - d, p\}$$

- if $C_i < d$, then $V_i = 0$ and $V_{max} = 0$
- if $d < C_i < d + p$, then $V_i = C_i - d$ and $V_{max} = r + np - d$
- if $C_i > d + p$, then $V_i = p$ and $V_{max} = p$

Then any schedule gives optimal solution for the problem (P) ■

Case (2) : If $p_i = p, r_i = r$ for all $i \in N$, then Lawler – algorithm is optimal solution for the problem (P).

Proof : sine $p_i = p, r_i = r \quad \forall i \in N$, then $\sum F_i$ is independent on a sequence [42], then the problem depend on V_{max} so LA gives optimal solution for the problem(P). ■

Case (3) : If $r_i = r$ and $V_{max}(SPT) = V_{max}(LA)$, then SPT rule gives optimal solution for the problem (P).

Proof : From the condition $r_i = r$ the SPT rule is optimal solution for $1/r_i = r / \sum F_i$ and sine $V_{max}(SPT) = V_{max}(LA)$, so SPT gives optimal solution for the problem (P). ■

Case(4): If $r_i = r$ and there is a schedule σ ordered its jobs according to SPT rule and for all $i \in \sigma, i$ is early job then σ is optimal for problem (P).

Proof : Since j is early job for all $j \in \sigma$ then $V_{max} = 0$, and the problem depend on $\sum F_i$, but $r_i = r$, implies that SPT rule gives an optimal solution for problem $1/r_i = r / \sum F_i$. ■

Case(5): If $r_1 \leq r_2 \leq \dots \leq r_n, p_1 \leq p_2 \leq \dots \leq p_n$ and SPT(SRT) schedule satisfying $C_i \leq d_i$ then SPT(SRD) schedule rule gives an – optimal solution for the problem (P).

Proof : The condition $C_i \leq d_i$ means that each job $i \in SPT(SRT)$ is early job then $V_i = 0 \forall i$, then the problem reduced to $1/r_i / \sum F_i$ but Al-Zuwaini [7] proved that SPT is optimal solution for $1/r_i / \sum F_i$ problem provided that $r_i \leq r_{i+1}$ for each job i , then SPT(SRT) is optimal solution for the problem (P). ■

2.5 Branch and Bound Method:

The problem (P), described in previous sections classified as NP-hard type, to find optimal solution to that problem, we used the BAB method forward sequencing branching rule for which nodes at level (L) of the search tree correspond to initial partial sequenced in the first (L) positions to find optimal solution for problem (P).

2.5.1 The Upper bound (UB) Procedure :

To find an (UB) for the objective function, we propose the following heuristic method which is applied once at the root node of search tree in (BAB).

Heuristic (UB)

The following algorithm (heuristic) has been proposed to obtain (UB) for the problem (P).

Step (1): Set $C'_i = r_i + p_i$, $\forall i \in N$.

Step (2): Order the jobs according to a non-decreasing of C'_i to obtain a sequence $\pi = (\pi_1, \pi_2, \dots, \pi_n)$.

Step (3): If there exists jobs with same C'_i , break ties by using (LA).

Step (4): Compute $UB = \sum_{i=1}^n F_i + V_{max}$.

Step (5): Stop.

Example (2.1): Data for the ready times, processing times and due dates.

J	1	2	3	4	5
r_j	6	0	6	7	3
p_j	3	4	10	9	7
d_j	9	10	18	17	12

Solution:

The schedule π is (2,1,5,3,4). Applying step (3) on the jobs (3,4) by (LA) to get (2,1,5,4,3).

Compute $UB = \sum_{i=1}^n F_{\pi(i)} + V_{max} = 77$

2.5.2 The Lower Bound (LB) Procedure

We used the decomposition way, to find the lower bound for the problem (P), because of difficulty finding minimum cost for the multi objective function. In this sub section we derived two lower bounds LB_1 and LB_2 for problem(P).

2.5.2.1 The First Lower Bound (LB_1)

Let N_1 was calculated to be the lower bound for (P_1) by (SRPT) rule (sequencing the jobs in non-decreasing order of Shortest Remaining Processing Time) to solve the $1/r_i, pmtn/\sum F_i$ [3]. N_2 was calculated to be the lower bound for (P_2) by using (LA) with $(1/r_i, pmtn/V_{max})$ [41]. to obtain a lower bound for P_2 , and then applying Theorem (2.1) to get the first lower bound for problem (P).

2.5.2.2 The second Lower Bound (LB_2)

The lower bound for the sub problem P_1 is obtained by using the lower bound which is proposed by Chu [6]. First it must state some useful definitions:

Definition 2.1: Given a partial sequence S_k , the earliest start time of a job $j \in \bar{k}$, where $\bar{k} = N - K$ is the set of jobs not sequenced, can be expressed as[43]:

$$(1) \quad t_j(S_k) = \max(r_j, C_k), \text{ with } t_j = r_j, \forall j \in N, \text{ if } k = 0.$$

The completion time of a job $j \in \bar{k}$, $C_j(S_k)$, can be expressed as

$$(2) \quad C_j(S_k) = t_j(S_k) + p_j.$$

Definition 2.2: The earliest completion time (ECT) rule :select job j with $\min_{j \in \bar{k}} C_j$. Break ties by choosing j with $\min t_j$ and further ties with $\min j$. Update t_j and C_j , $j \in \bar{k}$ after each addition using (1) and (2) [43].

The lower bound which is computed in the following way: we construct a schedule by ECT rule and modifying the release dates of jobs in the following way (where $[i]$ is the job occupying the i^{th} position in a schedule)

$$C'_{[0]} = 0 ;$$

For i from 1 to n do

begin

$$m_i = \min_{j \geq i} (r_j);$$

$$r'_i = \min(r_i, \max(C'_{i-1}, m_i));$$

$$C'_i = \max(r'_i, C'_{i-1}) + p_i ;$$

The quantity $\sum_{i=1}^n F_i = \sum_{i=1}^n (C'_i - r_i)$ is the lower bound of the problem $1/r_i / \sum_{i=1}^n F_i$ [6].

The lower bound for sub problem P_2 can be calculated by using the relaxation of constraints of objective function as follows:

We assume that $r^* = \min_{1 \leq i \leq n} \{r_i\}$ and $r_i = r^*$; $i = 1, 2, \dots, n$, to get the problem $1/r_i = r^* / V_{max}$, which is solved by Lawler-algorithm, and then applying Theorem (2.1) to get the lower bound for problem (P).

Hence the lower bound is $LB = \max\{LB_1; LB_2\}$.

By use the same data for previous example:

Optimal value of the problem $1/r_j / \sum_{j=1}^n F_{\pi(j)} + V_{max}$ equal to 73 with sequence (2,5,1,4,3) by using complete enumeration method.

$$LB_1(P_1) = 61, LB_2(P_1) = 59, LB_1(P_2) = 7, LB_2(P_2) = 9,$$

$$LB_1 = 61 + 7 = 68, LB_2 = 59 + 9 = 68, \text{ then } LB = \max\{LB_1, LB_2\} = 68 .$$

2.6 Dominance Rules:

The proposed dominance rule provides a sufficient condition for local optimality for the $1/r_j / \sum_{j=1}^n F_{\sigma(j)} + V_{max}$ problem. If it can be shown that an optimal solution can always be generated without branching from a particular node of the search tree, then that node is dominated and can be eliminated. Dominance rules usually specify whether a node can be eliminated before its lower bound is calculated. Clearly, dominance rules are particularly useful when a node can be eliminated whose lower bound that is less than the optimum solution. Some of dominance rules are valid for minimization of the sum of total flow time and maximum late work. As in the preprocessing step, similar dominance rules are also used within the branch and bound procedure to cut nodes that is dominated by others. These improvements lead to very large decrease in the number of nodes to obtain the optimal solution.

Notations:

A partial schedule G is a permutation of jobs in a subset of N . For a partial schedule G , we define the following notations.

$J(G)$: The set of jobs in G

t :The earliest completion time of last job of G

$R_i = \max(t, r_i)$, $R_j = \max(t, r_j)$, where i, j two unsequence jobs

Theorem (2.4): Consider a set N of n jobs; and a partial sequence H_G of G jobs, $G \subset N$ and let $i \in \bar{G}$ have $r_j \geq C(H_G, i) \forall j \in \bar{G}$ and i, j are early, then a partial sequence (H_G, i) dominates (H_G, j) .

Proof:

Dessouky and Deogun [43] show that (H_G, i) dominates (H_G, j) if $r_j \geq C(H_G, i) \forall j \in \bar{G}$ for $1/r_j / \sum_{j=1}^n F_j$ problem

Since i, j early jobs and $V_i=0$ for each $i \in N$.

Then (H_G, i) dominates (H_G, j) in optimal solution for the problem (P). ■

Theorem(2.5): Given H_G a partial sequence and two jobs $i, j \in \bar{G}$ if $C(H_G, j) \geq C(H_G, i)$, $C(H_G, j) - C(H_G, i) \geq (P_j - P_i)(n - |J(G)| - 1)$ and $V_j (\sum_{i=k+1}^n p_i \leq \min_{t \in \bar{G}} \{V_t (\sum_{i=|G|+1}^n p_i)\})$, then (H_G, i) dominates (H_G, j) .

Proof:

Yanai and Fujie [44] show that (H_G, i) dominates (H_G, j) , provided $C(H_G, j) \geq C(H_G, i)$, $C(H_G, j) - C(H_G, i) \geq (P_j - P_i)(n - |J(G)| - 1)$ for the problem $1/r_i / \sum_{i=1}^n F_i$.

Since $V_j (\sum_{i=k+1}^n p_i \leq \min_{t \in \bar{G}} \{V_t (\sum_{i=|G|+1}^n p_i)\})$ then sequenced job j in last position of schedule .

Then (H_G, i) dominates (H_G, j) . ■

Theorem (2.6): If H_G be a partial sequence which it's jobs are scheduled, $G \subset N$. For $i, j \in \bar{G} = N - G$, if $r_i < r_j$, $p_i < p_j$, $d_i = d_j$, $C_i < r_j$ and $C_j < p_i + d_i$, then (H_G, i) dominates (H_G, j) in optimal solution.

Proof:

Let (H_G, j, i) be the partial schedule which is obtained by interchanging jobs i and j in (H_G, i, j) . Then all jobs other than i and j have the same completion time in (H_G, i, j) as in (H_G, j, i) , the difference in completion

time between (H_G, i, j) and (H_G, j, i) depends only on the completion time of jobs i and j

From the condition $r_i \leq r_j$ and $p_i \leq p_j$, then $\sum_{i \in H_{G,i,j}} F_i \leq \sum_{i \in H_{G,j,i}} F'_i$ [7].

The maximum late work in (H_G, i, j) is:

$$V_{max} = \max\{V, V_i, V_j\}, V = \max_{b \in H_G}\{V_b\}$$

$$V_i = \min\{T_i, p_i\}, \text{ where } T_i = \max\{R_i + p_i - d_i, 0\}$$

$$V_j = \min\{T_j, p_j\}, \text{ where } T_j = \max\{r_j + p_j - d_j, 0\}$$

The maximum late work in (H_G, j, i) is: $V'_{max} = \max\{V, V'_j, V'_i\}$,

$$V'_j = \min\{T'_j, p_j\}, T'_j = \max\{R_j + p_j - d_j, 0\}$$

$$V'_i = \min\{T'_i, p_i\}, T'_i = \max\{R_j + p_j + p_i - d_i, 0\}.$$

In (H_G, i, j) :

Since $C_i \leq C_j$ and $p_i \leq p_j$, then $V_i \leq V_j$ and $V_{max} = \max\{V, V_j\}$.

In (H_G, j, i) :

Since $C'_i \geq C'_j$ and $d_i = d$, then $T'_j \leq T'_i$, $T_j \leq T'_j$ and $V_j \leq V'_j$

Since $C_j < p_i + d_i$ Then $V_j = T_j$ and $V_j \leq V'_j$

If $V'_i = T'_i$ Then $V_j \leq V'_i$

If $V'_i = p_i$, and $C_j < p_i + d_i$ Then $T_j \leq p_i$ and $V_j \leq V'_i$

Then $V_{max} \leq V'_{max}$

we get $\sum_{i \in H_{G,i,j}} F_i + V_{max} \leq \sum_{i \in H_{G,j,i}} F'_i + V'_{max}$. ■

2.7 Computational experiments:

The BAB algorithm was tested by coding it in MATLAB (R2013b) and implemented on Intel (R) Core (TM) i3-3217U CPU @ 1.80 GHZ, with RAM 4.00 GB personal computer. Test problems are generated as follows: for each job j , the job processing times p_j were generated uniformly in the interval $P=[1, P_{max}]$ where the values P_{max} considered were: (25, 50, 100). The due date d_j were generated uniformly in the interval $d=[1, \alpha * P_{max}]$, where α is the tightness factor. Three values of α were considered: (1, 2, 3). The release date r_j were generated uniformly in the interval $r = (0, 0.5 P_{max} + n * \gamma)$, where the values of γ = (0.3, 0.6, 1).

Table (2.1) shows the results for problem (P) obtained by BAB algorithm. We list 10 problems for each value of n , where $n \in \{5, 10, 15, 20, 25, 30, 35\}$, and the branch and bound value, upper bound (UB), initial lower bound (ILB), the number of open nodes (NOON), the computational time in second (Time), and for solved and unsolved problems (Status).

Table (2.2) illustrates the number of nodes which are requires to solve the problems in the search tree of BAB method. The stopping condition for the BAB algorithm was determined and we consider that the problem is unsolved (state is 1), that the BAB algorithm is stopped after a fixed period of time, here after 1800 seconds (i.e. after 30 minutes).

We observed from Table (2.1), the heuristics of upper bound is good algorithm. It gives the value for objective function equal to optimal or near optimal value for some values of n , and we notice that lower bound gives the values near the optimal.

Table (2.1): The performance of initial lower bound, upper bound, number of nodes and computational time in second of BAB algorithm.

n	EX	OPT	UB	ILB	NOON	Time	Status
5	1	122	122*	112	14	3.1929	0
	2	122	122*	108	14	1.7675	0
	3	254	254*	180	16	0.001248	0
	4	147	150	105	14	1.7105	0
	5	70	70*	46	23	0.009003	0
	6	57	57*	41	16	0.001087	0
	7	137	155	93	21	0.005060	0
	8	170	178	161	14	1.7105	0
	9	124	129	114	14	1.8245	0
	10	113	121	71	14	0.00018	0
10	1	727	792	573	56	0.00138	0
	2	593	599	590	63	0.01015	0
	3	574	618	565	108	0.10204	0
	4	711	731	702	54	6.44283	0
	5	532	532*	529	89	0.06769	0
	6	414	440	409	76	0.04603	0
	7	577	581	446	54	6.55686	0
	8	381	422	289	54	6.38581	0
	9	478	540	375	81	0.05598	0
	10	576	576*	570	54	6.44283	0
15	1	1231	1231*	1231**	0	0	0
	2	918	929	895	731	1.5147	0
	3	966	966*	965	221	0.2195	0
	4	1043	1063	1038	294	0.6753	0
	5	1128	1186	1109	286	0.4419	0
	6	1138	1139	1111	240	0.3221	0
	7	851	956	831	191	0.1733	0
	8	1205	1254	1201	551	1.1451	0
	9	953	978	929	339	0.7182	0

	10	641	703	633	181	0.1056	0
20	1	1598	1627	1579	2411	4.0394	0
	2	2252	2383	2241	389	0.9859	0
	3	1723	1829	1489	346	0.3876	0
	4	2247	2259	2240	474	0.7728	0
	5	2026	2066	2006	353	0.3409	0
	6	2143	2217	2130	2387	6.4559	0
	7	1475	1516	1467	1083	2.3365	0
	8	1681	1727	1476	9869	20.8898	0
	9	2031	2109	2022	4452	15.1250	0
	10	1476	1665	1474	476	0.4860	0
25	1	2740	2914	2441	1158	2.4712	0
	2	2914	3070	2887	26818	113.0406	0
	3	2638	2638*	2638**	0	0	0
	4	2915	2955	2877	1019	2.7591	0
	5	3148	3189	3145	10092	36.3353	0
	6	2905	3075	2902	5839	18.3209	0
	7	2180	2379	2175	49684	147.025	0
	8	2706	2706*	2705	10996	38.9284	0
	9	2463	2497	2461	8937	14.9000	0
	10	2685	2685	2684	2398	6.4419	0
30	1	3380	3415	3355	240474	506.721	0
	2	4630	4907	4599	39041	114.260	0
	3	4543	4621	4543**	464	0	0
	4	4552	4576	4497	668510	1404.2469	0
	5	3919	4149	3919**	464	0	0
	6	3234	3286	3214	17987	68.242024	0
	7	4116	4164	4113	820488	1359.5698	0
	8	3379	3403	3377	8099	15.514328	0
	9	3999	4093	3983	42231	145.3450	0
	10	3512	3715	3511	23756	96.6089	0
	1	3849	4060	3843	65472	253.9585	0

35	2	4277	4277*	4277**	0	0	0
	3	6517	6640	6473	394614	15302.57	1
	4	3698	3748	3690	118195	414.6198	0
	5	5468	5670	5440	223458	445.2409	0
	6	6257	6283	6249	789743	1800.001	1
	7	5674	6032	5660	792696	1800.155	1
	8	3990	4020	3985	284580	614.8058	0
	9	4418	4512	4395	143404	378.4872	0
	10	5055	5059	5011	78119	147.7937	0

Where:

OPT= The optimal value obtained by BAB method.

UB = Upper bound.

ILB = Initial lower bound.

NOON = Number of open nodes.

Time = Computational time in seconds.

* = The upper bound gives the optimal value.

** = The initial lower bound gives the optimal value.

$$\text{Status} = \begin{cases} 0 & \text{if the problem is solve} \\ 1 & \text{if the problem isn't solved} \end{cases}$$

Table(2.2): Comparison between the number of nodes and problems are solved.

No. of nodes	No. of problems
0-100	22
101-500	15
501-1000	1
1001-5000	7
5001-10000	4
10001-500000	16
500001-900000	4

Chapter Three

AFSA/FOA to Solve

Multiple Objective Function

3.1 Introduction

We consider the scheduling n independent jobs on a single machine with unequal release dates to minimize the multiple objective problem. Our objective is to find a schedule that minimizes sum of total flow time and maximum late work, by using Local search methods. This problem is denoted by $1/ r_i / \sum_{i=1}^n F_i + V_{max}$. This problem is strongly NP-hard (see Sec. 2.2).

Local Search Methods or local improvement starts with some feasible solutions of the problem and tries to progressively improve it. Each step of the procedure carries out a movement from one solution to another one with a better value. The method terminates when, for a solution, there is no another accessible solution that improves it. The techniques depicted in this chapter are called heuristic calculations or essentially heuristics. This term originates from the Greek word *heuriskein* which intends to discover or find. It is used in the field of optimization to characterize a certain kind of problem-solving methods. These techniques are called heuristics or approximate calculations. As opposed to exact methods, which guarantee to give an optimum solution of the problem, heuristic methods only attempt to yield a good, but not necessarily an optimum solution.

3.2 Artificial Fish Swarm Algorithm (AFSA):

In underwater world, fish can find areas with more food based on their individual or swarm search. AFSA is one of the best methods of optimization among the swarm intelligence algorithms. This algorithm is inspired by the collective movement of the fish and their various social behaviors. Based on a series of instinctive behaviors, the fish always try to maintain their colonies and accordingly demonstrate intelligent behaviors. Searching for food, immigration and dealing with dangers all

happen in a social form and interactions between all fish in a group will result in an intelligent social behavior. This algorithm has many advantages including high convergence speed, flexibility and high accuracy. Objective function of AFSA is to find maximum food density. Solving the NP-complete problems is one of the most challenging issues that computer scientists always faced with. Swarm intelligence algorithms have been significantly proved their capabilities in solving these problems. The behavior of fishes has a great impact on the performance of the algorithm, such as global search and convergence speed. The fish can discover the more nutritious area by individual search or following after other fish, the area with much more fish is generally most nutritious. The basic idea of the AFSA is to imitate the fish behaviors such as praying, swarming, and following with local search of fish individual for reaching the global optimum Li (2003)[45]. AFSA proposed by Li et al. (2002) [46], is a stochastic population-based algorithm motivated by intelligent collective behavior of fish groups in nature . Artificial fish (AF) is a fictitious entity of true fish, which is used to carry on the analysis and explanation of problem.

The basic idea of AFSA [47-48] is simulated fish behaviors such as swarming, praying, following with local search of fish individual for reaching the global optimum; it is random and parallel search algorithm.

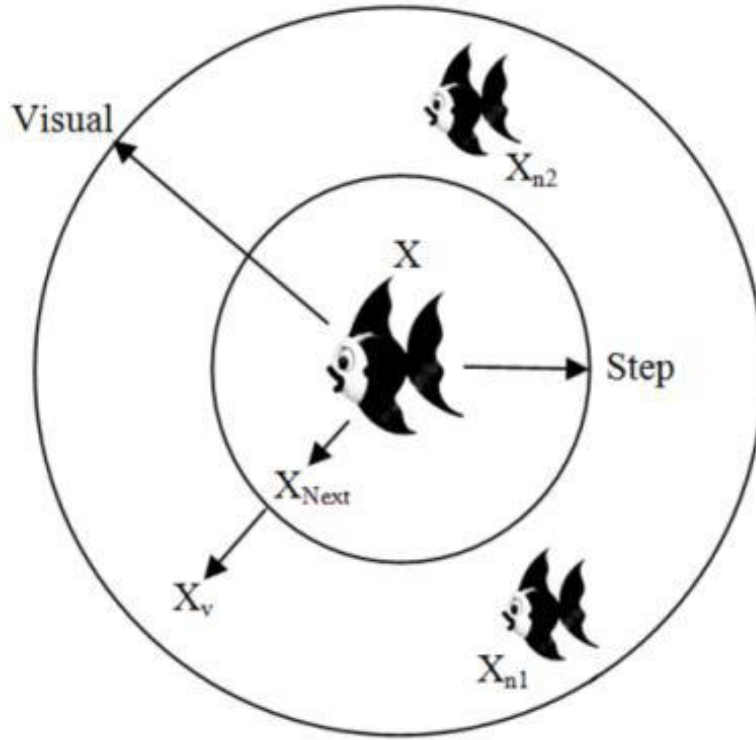


Figure.(3.1) Vision concept of the artificial fish

As Figure (3.1) shows, AF observes external concepts with its visual perception. Current position of AF is shown by vector $X = (X_1, X_2, \dots, X_n)$. The visual is equal to visibility domain of AF and X_v is an intended position in visual where the AF selects to move towards. If X_v has better food density than current position, then AF moves one step towards X_v , which results in displacement of AF from X to X_{next} . Otherwise, if the current position X of AF is better than X_v , it selects another position in its visual. Food density in position X is the fitness value of the position and is shown with Y . The step is equal to maximum length of the movement. The distance between two AFs which are placed in X_i and X_j is shown by (Euclidean distance) $\text{Dist}(ij) = \|X_i - X_j\|$.

Let $X = (x_1, x_2, \dots, x_n)$ and $X_v = (x_{v1}, x_{v2}, \dots, x_{vn})$ then this process can be expressed as follows:

$$X_{Vi} = x_i + Visual.rand(0,1), i \in (0, n] \quad \dots (3-1)$$

$$X_{next} = X + \frac{X_v - X}{\|X_v - X\|} \cdot Step.rand(0,1). \quad \dots (3-2)$$

where *Step* is the step length, and x_i is the optimizing variable, n is the number of variables. The AF model includes two parts (variables and functions). The variables include: X is the current position of the AF, *Step* is the moving step length, *Visual* represents the visual distance, and δ is the crowd factor ($0 < \delta < 1$). The functions include the behaviors of the AF: AF_Prayer, AF_Swarm, AF_Follow [49].

3.3 The basic functions of AFSA:

Fish usually stay in the place with a lot of food, so we simulate the behaviors of fish based on this characteristic to find the global optimum, which is the basic idea of the AFSA. The basic behaviors of AF are defined as follow for maximum objective function value (food) :

1-Prayer behavior:

This is a basic biological behavior that tends to the food. Supposed the state of artificial fish is X_i , select a state X_j within its sensing range randomly. If X_j superior to X_i , then move to X_j ; on the contrary, selected randomly state X_j and determine whether to meet the forward conditions, repeated several time, if still not satisfied forward conditions, then move one step randomly, where Y is the food concentration (objective function value),

$$X_j = X_i + \text{Visual.rand}(0,1) \quad \dots (3-3)$$

If $Y_i < Y_j$ in the maximum problem, it goes forward a step in this direction;

$$X_i^{t+1} = X_i^t + \frac{X_j - X_i^t}{\|X_j - X_i^t\|} \cdot \text{Step.rand}(0,1) \quad \dots (3-4)$$

2- AF_Swarm:

The fish will assemble in groups naturally in the moving process, which is a kind of living habits to guarantee the existence of the colony and avoid dangers. Behavior description: Let X_i be the AF current state, X_c be the center position and nf be the number of its companions in the current neighborhood ($d_{ij} < \text{Visual}$), n is total fish number. If $Y_c > Y_i$ and $\frac{n_f}{n} < \delta$, which means that the companion center has more food (higher fitness function value) and is not very crowded, it goes forward a step to the companion center; otherwise pray behavior.

$$X_i^{t+1} = X_i^t + \frac{X_c - X_i^t}{\|X_c - X_i^t\|} \cdot \text{Step.rand}(0,1) \quad \dots (3-5)$$

3- AF_Follow:

In the moving process of the fish swarm, when a single fish or several ones find food, the neighborhood partners will trail and reach the food quickly. Behavior description: Let X_i be the AF current state, and it explores the companion X_j in the neighborhood ($d_{ij} < \text{Visual}$), which has the greatest Y_j . If $Y_j > Y_i$ and $\frac{n_f}{n} < \delta$, which means that the companion X_j state has higher food concentration (higher fitness function value)

and the surrounding is not very crowded, it goes forward a step to the companion X_j , Otherwise, executes the preying behavior. As in equation (3-4).

The following algorithm illustrates the Artificial fish swarm algorithm.

- 1). Initialize the parameters of artificial fish, such as *Step* ,*Visual* , the number of exploratory try number, maximum number of iterations ,and randomly generated n fishes;
- 2). Set bulletin board to record the current status of each fish, and select the optimal value recorded;
- 3). Implementation of pray behavior, swarm behavior and follow behavior;
- 4). Optimal value in bulletin board is updated;
- 5). If the termination condition is satisfied, output the result; otherwise return to step 2.

3.4 Fruit Fly Optimization Algorithm(FOA)

In this section one of the recent swarm optimization algorithms namely Fruit Fly Optimization Algorithm (FOA) and some of its variants are investigated. FOA is suggested by Tsao Pan a scholar of Taiwan in 2011 [50-51]. It is one of the newest optimization algorithms. It is a fast, easy to code and easy to understand. FOA is developed by inspiring the foraging behavior of fruit flies. Fruit flies are superior to other species in terms of especially smell and vision. They can smell the food even if it is 40 km away. They fly toward to the food according to smell. Then, they can reach the food by using their vision capability. FOA is consist of two phases. The first phase is smell phase. In this phase, the flies fly through the food by using their smell capability. After getting closer to the food source the second phase starts. This phase is the vision phase. In this phase they use their vision capability to get nearer to the food. This phase is repeated until the fruit fly reaches the food. The steps of FOA is demonstrated in Figure (3.2) and described as follows:

- 1- Initialization of the parameters (iteration number, fruit fly number)
- 2- Initialization of the initial positions of the fruit fly swarm (Init X axis, Init Y axis)
- 3- Update the locations of the fruit flies by using smell sensing

$$X_i = X_{axis} + \text{Random Value}$$

$$Y_i = Y_{axis} + \text{Random Value}$$

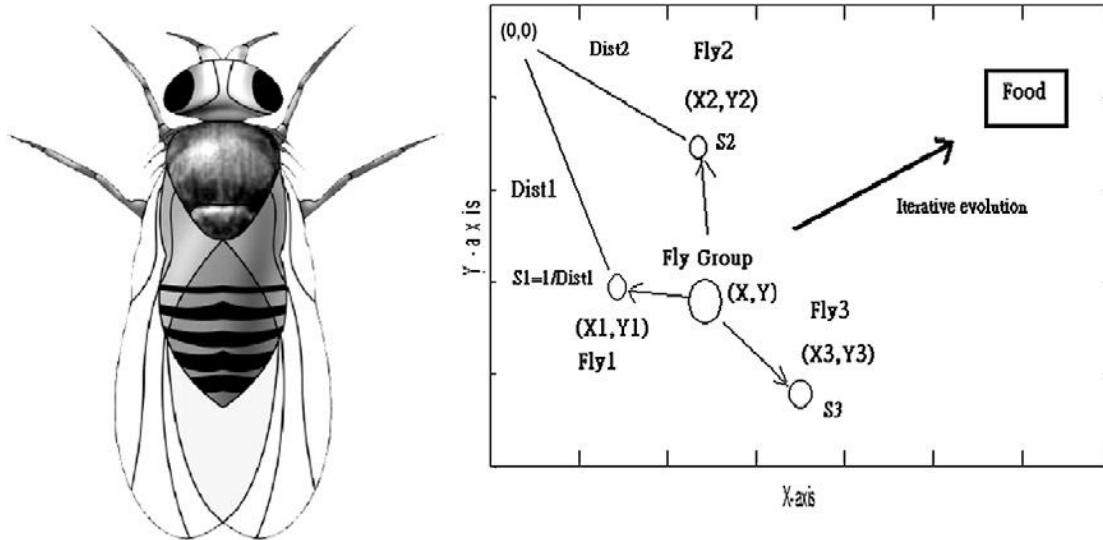


Fig.(3. 2): Illustration of the body look of the fruit fly and group iterative food searching of fruit fly.

4- Calculation of the distance of the fly to the origin (Dist) and smell concentration judge value (S)

$$(Dist)_i = \sqrt{X_i^2 + Y_i^2}$$

$$S_i = 1/(Dist)_i \quad \dots (3-6)$$

5- Calculation of smell concentration value

$$(Smell)_i = Function(S_i) \quad \dots (3-7)$$

6- Detection of the best fly

$$[best\ Smell\ best\ Index] = \max (Smell) \quad \dots (3-8)$$

7- Saving the best concentration value and location of the best

Smell best best Smell

$$X_{axis} = X(bestIndex)$$

$$Y_{axis} = Y(bestIndex) \quad \dots (3-9)$$

8- Repeat 3-6 until the maximum iteration number is reached.

3.5 Applications Fruit Fly Optimization Algorithm

Fruit Fly Optimization Algorithm (FOA) belongs to the category of evolutionary computing, also belongs to the field of artificial intelligence. In the application is part of a research method, without any restrictions on the field, covered the various fields of military, engineering, medicine, management, finance and so on. In addition, we can be combined with other data mining techniques are used together, such as decision trees, fuzzy math, and neural network, and so on. Therefore, readers may need to be applied in accordance with their different areas, can be mixed with different algorithms [50-52].

3.6 Computational Results of Local Search Algorithms and Comparison

3.6.1 Test Problem

We use the test problems that are introduced in subsection (2.7) for the local search algorithms.

3.6.2 Computational Results

In this chapter, discuss the local search that content (AFSA , FOA) are coded in MATLAB (R2013b) and implemented on Intel (R) i3-3217U CPU@ 1.80 GHZ, with RAM 4.00 GB personal computer.

Table (3.1) shows for each algorithm the value of objective function and how many it can catch the optimal value for each value of " n " (problem size). In addition, it describes the deviation of local search methods from the optimal solution. The optimal solution for examples in Table (3.1) was found by using BAB algorithm in chapter two.

Table (3.2) shows the values of each local search algorithms and how many times that each of them catch the best value, where:

Optimal= the optimal value.

AFSA = the value found by Artificial Fish Swarm Algorithm.

FOA = the value found by Fruit Fly Optimization Algorithm.

No of opt.= number of examples that catch the optimal value.

Av. Time = the average of time for (10) examples for each algorithm.

Best = the best value.

No. of best = number of examples that catch the best value.

Table (3.1) : The performance of local search methods and the optimal solution for

$$n \in \{5, 10, 15, 20, 25, 30, 35\}$$

n	EX	OPT	AFSA	FOA
5	1	122	122	122
	2	122	122	122
	3	254	254	254
	4	147	147	147
	5	70	70	70
	6	57	57	57
	7	137	137	146
	8	170	174	174
	9	124	124	124
	10	113	113	113

No of opt.			9	8
Av. Time			4.473	4.678
10	1	727	784	776
	2	593	593	593
	3	574	595	599
	4	711	712	712
	5	532	532	532
	6	414	414	420
	7	577	581	581
	8	381	396	396
	9	478	526	533
	10	576	576	576
No of opt.			4	3
Av. Time			5.384	5.552
15	1	1231	1231	1231
	2	918	918	924
	3	966	966	966
	4	1043	1059	1053
	5	1128	1166	1180
	6	1138	1138	1138
	7	851	910	941
	8	1205	1243	1247

	9	953	957	957
	10	641	675	691
No of opt.			4	3
Av. Time			6.178	6.378

n	EX	OPT	AFSA	FOA
20	1	1598	1623	1626
	2	2252	2378	2378
	3	1723	1782	1814
	4	2247	2256	2256
	5	2026	2036	2036
	6	2143	2211	2211
	7	1475	1511	1475
	8	1681	1699	1721
	9	2031	2087	2077
	10	1476	1577	1577
No of opt.			0	1
Av. Time			7.058	7.246
25	1	2740	2900	2898
	2	2914	2986	3038
	3	2638	2638	2638
	4	2915	2940	2927
	5	3148	3148	3182

	6	2905	3045	3054
	7	2180	2302	2332
	8	2706	2706	2706
	9	2463	2482	2479
	10	2685	2685	2685
No of opt.			4	3
Av. Time			8.220	8.425
30	1	3380	3411	3386
	2	4630	4807	4882
	3	4543	4619	4543
	4	4552	4573	4573
	5	3919	4143	4098
	6	3234	3234	3265
	7	4116	4146	4161
	8	3379	3379	3402
	9	3999	4073	4062
	10	3512	3711	3708
No of opt.			2	1
Av. Time			9.499	9.595
	1	3849	3849	4054
	2	4277	4277	4277
	3	6517	6575	6635

35	4	3698	3747	3747
	5	5468	5593	5656
	6	6257	6281	6281
	7	5674	5897	5892
	8	3990	3992	3990
	9	4418	4498	4510
	10	5055	5056	5057
No of opt.			2	2
Av. Time			10.342	10.634

Table (3.2): The performance of local search methods and the best solution for $n \in \{50, 100, 500, 1000, 5000, 6000\}$

n	EX	Best	AFSA	FOA
50	1	10858	10863	10858
	2	10820	10820	10861
	3	9722	9730	9722
	4	9170	9170	9178
	5	8870	8870	8901
	6	12353	12398	12353
	7	11837	11837	11900
	8	11711	11713	11711
	9	10813	10813	10824
	10	11177	11186	11177

No of best.			5	5
Av. Time			13.349	13.484
100	1	37775	37779	37775
	2	41101	41101	41147
	3	43159	43192	43159
	4	41066	41102	41066
	5	46231	46434	46231
	6	46847	46943	46847
	7	45680	45680	45680
	8	35564	35564	35624
	9	47421	47426	47421
	10	46283	46388	46283
No of best.			3	8
Av. Time			26.552	26.645
500	1	1084646	1084880	1084646
	2	1071767	1071767	1071770
	3	1149952	1149952	1150057
	4	1112501	1112600	1112501
	5	1080767	1080767	1080867
	6	1080256	1080258	1080256
	7	1052706	1052706	1052874
	8	1052807	1052807	1052807

	9	1098132	1098168	1098132
	10	1147315	1147315	1147387
No of best.			6	6
Av. Time			10.9277	10.9543
1000	1	4234575	4234575	4235394
	2	4124110	4124110	4124111
	3	4107156	4107628	4107156
	4	4316244	4316304	4316244
	5	4225595	4225595	4226228
	6	4246634	4246635	4246634
	7	4253528	4253530	4253528
	8	4474877	4524633	4474877
	9	4195598	4195609	4195598
	10	4308980	4308981	4308980
No of best.			3	7
Av. Time			27.4293	27.4869
5000	1	107296598	107304466	107296598
	2	111148264	111171932	111148264
	3	106012745	106012746	106012745
	4	107332114	107332114	107332114
	5	107793828	107796496	107793828
	6	108639639	108640003	108639639

	7	107760159	107760159	107763173
	8	105404776	105404776	105404776
	9	107192082	107192082	107192313
	10	104098158	104098158	104101438
No of best.			5	8
Av. Time			36.5077	36.4867
6000	1	156999500	156999500	156999500
	2	154406199	154406199	154407802
	3	156001033	156001182	156001033
	4	152784564	152784564	152784564
	5	154194250	154194250	154194250
	6	156591932	156591932	156591933
	7	152803313	152803313	152811471
	8	155471459	155471459	155471460
	9	152919144	152919144	152923242
	10	154333821	154333821	155243651
No of best.			9	4
Av. Time			154.1748	612.8386

3.6.3 Summary of Experimental Evaluation of Local Search Methods

In the table (3.3) and (3.4), we summarize the results of table (3.1) and (3.2) by viewing how many times that the algorithm catch the optimal value only for $5 \leq n \leq 35$ and the best value for $50 \leq n \leq 6000$, and their sum, for each number of jobs and for the two local search methods.

Table (3.3): Summary of results of table (3.1)

n	AFSA	FOA
5	9	8
10	4	3
15	4	3
20	0	1
25	4	3
30	2	1
35	2	2
Sum	25/70	21/70

Table (3.4): Summary of results of table (3.2)

n	AFSA	FOA
50	5	5
100	3	8
500	6	6
1000	3	7

5000	5	8
6000	9	4
Sum	31/60	38/60

From comparative study of the introduced local search methods and precise vision of all the above tables, we found that for the $(1 / r_j / \sum_{j=1}^n F_j + V_{\max})$ problem the Artificial Fish Swarm Algorithm has the best results for the problem of size less than or equal (35) jobs. For the problem of size larger than (35) jobs, we found that the Fruit Fly Optimization Algorithm has the best results.

Fig. (3.3) Illustrates the comparison between the values given by both AFSA and FOA with the method BAB for the number of jobs $5 \leq n \leq 35$

It is also apparent from the diagram that two AFSA and FOA give very close results to the value of the optimal solution.

Fig. (3.4) Shows the results given by both AFSA and FOA to the jobs $50 \leq n \leq 6000$, and the two methods show almost identical results, especially when the number of jobs between $50 \leq n \leq 1000$, in $1000 < n \leq 6000$ note the FOA give better results than the results of AFSA. Note that the AFSA solved the same number of jobs until $n = 6000$, and at a time less than the time when the FOA by $n = 6000$.

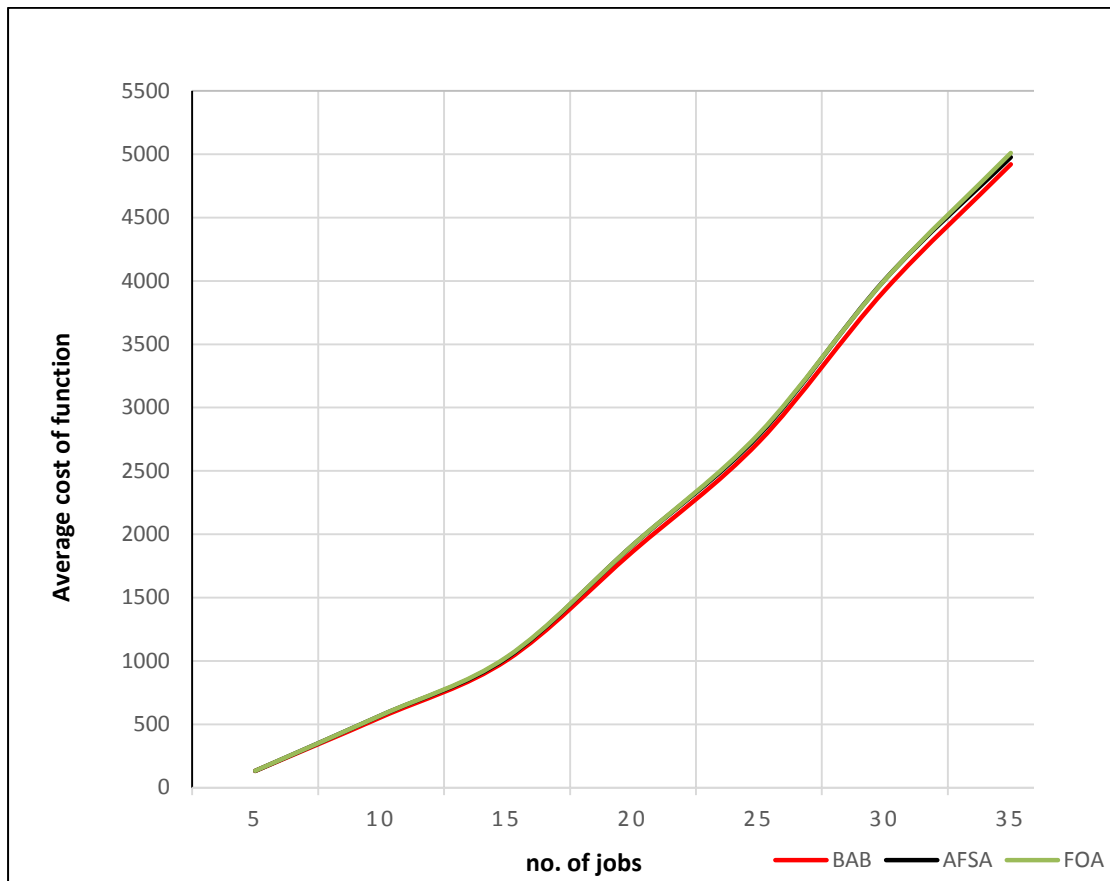


Fig.(3.3) Compares the results with n less than 35 for the problem (P).

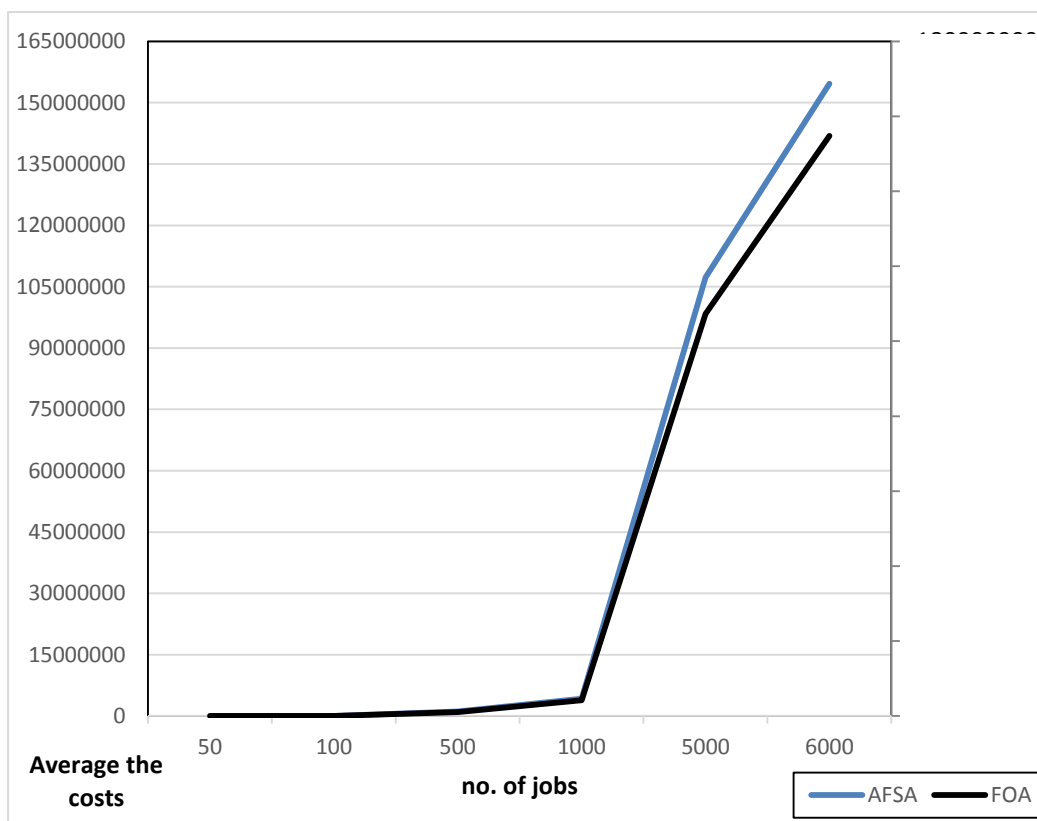


Fig.(3.4) Compares the results with n more than 35

Chapter Four

Conclusion and Future

works

4.1 Conclusion

In this thesis, we have developed exact and approximate solutions for the problem of scheduling n jobs on a single machine to minimize the sum of total flow time and maximum late work with unequal release dates. The proposed heuristic algorithm, which is used as an upper bound in the branch and bound algorithm, is effective in finding an optimal or near optimal schedule. The lower bounds which are used in branch and bound algorithm based on decomposing the problem into two sub problems. The proposed branch and bound algorithm can be used to find optimal solution for the problem up to 35 jobs. Several results concerning optimality of five solvable special cases are presented. We prove three dominance rules which help in reducing the number of branches in the search tree. Also, we report on the results of extensive computations tests of the methods: AFSA and FOA. The main conclusion to be drawn from our computation results is that AFSA and FOA of the local search heuristic algorithms can solve $(1/r_i / \sum F_i + V_{max})$ problem of size (6000) jobs in reasonable time. It is also apparent from that two AFSA and FOA give very close results to the value of the optimal solution.

The AFSA has the best results for the problem of size less than or equal to (35) jobs. For the problem of size larger than (35) jobs, we found that the FOA has the best results. And the AFSA solved the same number of jobs until $n = 6000$, and at a time less than the time when the FOA by $n = 6000$.

4.2 Future work

An interesting future research topic would involve experimentation with the following problems:

$$1- F_2/r_j/\sum F_j + V_{max}.$$

$$2- 1/ F(\sum_{j=1}^n C_j, V_{\max}^W).$$

$$3- P_3/ r_i / V_{\max} + \sum F_j.$$

References

- [1] Lenstra J.K., Rinnooy Kan A.H.G., Bruck B., "Complexity of machine scheduling problems", *Annals of discrete math.* 1, 343-36, (1977).
- [2] Smith W.E., "Various Optimization for Single Stages Production", *Naval Research Logistics Quarterly* 3/1, 59-66, (1956).
- [3] Baker, K.R "Introduction to Sequencing and Scheduling " Wile New York, (1974).
- [4] Ahmad R. H. and Bagchi U., "Lower bound for single machine scheduling problem" ,*Naval Research Logistics*, 37, 967-79 (1990).
- [5] Chandra ,R. On $n/1/\bar{F}$ dynamic deterministic problems. *Naval Research Logistics*, 26, 537–544, (1979).
- [6] Chu, C. Efficient heuristics to minimize total flow time with release dates. *Operations Research Letters*, 12, 321–330, (1992).
- [7] Al-Zuwaini M. K., " Single machine scheduling to minimize total cost of flow time with unequal ready times", *Journal of Basra Researches*, V.25, Par.3, 94-108 (2000).
- [8] Al-Zuwaini M.K., " One machine scheduling problem with release dates and two criteria", *Journal of Thi-Qar University*, No.2, Vol.6, 1-14, (2011).
- [9] Woeginger, G. J." Approximability and nonapproximability results for minimizing total flow time on a single machine", *Proceedings of the 28th Annual ACM Symposium on Theory of Computing*, pp. 418–426, (1996).
- [10] Lawler, E.L., "Optimal sequencing of a single machine subject to precedence constraints", *Management Science* 19/5, 544-546 (1973).
- [11] Blazewicz J. Scheduling preemptible tasks on parallel processors with information loss. *Technique et Science Informatiques* 1984; 3(6):415-20.
- [12] Potts CN. Van Wassenhove LN. "Single machine scheduling to minimize total late work". *Operations Research* 1991; 40 (3) : 586-95.

- [13] Shahbaa Y. and Saad H., "Minimizing Multiple Objective Function for Scheduling Machine Problems", *Mathematical Theory and Modeling*, ISSN 2224-5804 (Paper) ISSN 2225-0522 (Online), Vol.5, No.6, 2015.
- [14] Al_Nuaimi. A.A. "Muiti criteria scheduling: Mathematical, Models, Exact and Approximation algorithms". M.Sc. thesis, Dept. of mathematics, college of science university of AL-Mustansiriya (2014).
- [15] Abdul-Razzaq T.S. and Karrar F., "Solving Multi-criteria Scheduling Problems", M.Sc. Thesis, University of Al-Mustansiriyah, College of Science, Dep. Of Mathematics (2014).
- [16] Abdul-Razaq T.S. and F.Sh. Fandi ., "Minimizing total Completion Time and Maximum late Work Simultaneously". *Ibn Al-Haitham Journal for Pure and Applied Science*, No. 3 Vol. 25,(2012).
- [17] Blazewics J. Ecker K.H., P Esch E. Schmidt G. and Weglarz J., "Scheduling Computer and Manufacture Processes" Springer velary Berlin. Heidelberg (1996).
- [18] Jouni L., "Multi-objective Nonlinear Parto-Optimization", Lappeenranta University of Technology (2000).
- [19] Hoogeveen J.A., "Invited review Multi criteria Scheduling", *European Journal of Operational Research* 167,592-623, (2005).
- [20] Nagar A., Jorge H., and Sunderesh H., "Multiple and Bi-criteria Scheduling: A Literature Survey", *European Journal of Operation Research* North-Holland, 81,88-104 (1995).
- [21] Yeleswarapu Radhika M., "Scheduling of 2-Operation Jobs on A single Machine to Minimize The Number of Tardy Jobs", M.Sc. Thesis University of South Florida, College of Engineering, Dep. of Industrial and Management Systems Engineering (2003).

- [22] Pinedo M.L., "Scheduling Theory, Algorithms, and Systems", Springer Science +Business Media, LLC., New York (2012).
- [23] Mohammed H.A.A., "Using Genetic and Local Search Algorithms as Tools for Providing Optimality for Jobs Scheduling", M.Sc. Thesis, Univ. of Al-Mustansiriyah, College of Science, Dept. of Mathematics (2005).
- [24] AL-Assaf S ., "Solving Multiple Objective Scheduling Problem" M.Sc. Thesis, Dept. of Mathematics, College of Sciences, AL-Mustansiriayah University (2007).
- [25] Ignall E., and Schrage L.E., "Application of the Branch and Bound Technique to Some Flow Shop Scheduling Problem", Oper. Res. 13,400-412 (1965).
- [26] Graham R.L., Lawler E.L., Lenstra J.K., and Rinnooy Kan A.H.G. "Optimization and Approximation in Deterministic Sequence and Schedule: a Survey Annals of Discrete Mathematics, volum5,287- 326, (1979).
- [27] Delphi A. M., " Algorithms to solve multi criteria scheduling problems on single machine ", M.Sc. thesis Univ. of Al-Mustansiriyah, College of Science, Dep of Mathematics (2011).
- [28] Abbas, I.T., "The performance of multi criteria scheduling in one machine", M.Sc. thesis Univ. of Al-Mustansiriyah, College of Science, Dep. of Mathematics (2009).
- [29] David K., Cli_ S., and Joel W., "Scheduling Algorithms", Massachusetts Institute of Technology, (1998).
- [30] Hussein Jameel, " Flow shop scheduling problems Using Exact and Local search methods", M.Sc. thesis University of Al-mustansiriyah, College of sinence, Dep. Of Mathematics (2008).
- [31] Jackson, J.R., "Scheduling a production line to minimize maximum tardiness", Res. Report 43, Management Science Res. Project, Univ. of

California, Los Angeles (1955).

[32] Hoogeveen J.A., "Minimizing maximum earliness and maximum lateness on a single machine", CWI, BS-R9001 (1990).

[33] Van Wassenhove L.N., and Gelders F., "Solving a bicriterion scheduling problem", European Journal of Operational Research 4/1, 42-48(1980).

[34] Hoogeveen, J.A., van de Velde, S.L., "Minimizing total completion time and maximum cost simultaneously is solvable in polynomial time", Operations Research Letters 17, 205–208(1995).

[35] Hummady L. Z., "Using Genetic algorithm to solve (NP-Compete) problem", M.Sc. thesis Univ. of Al- Mustansiriyah, College of Science, Dep. of Mathematics (2005).

[36] Lominickiz, A., "A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem", oper.Res.Quart.16,89-100, (1965).

[37] Bellman R. and Dreyfus, "Applied Dynamic Programming Princeton" University press, Princeton, N.J., (1962).

[38] Reeves CR. "Modern Heuristic Techniques for Combinatorial Problems". John Wiley and Sons, Inc, New York, (1993).

[39] Cook S.A. "The Complexity of The Theorem-Proving procedure", proc. 3rd annual ACM Symp. Theory compute,151-158 (1971).

[40] Chen B., Potts C. N., and Woeginger G.J., "A Review of Machine Scheduling: Complexity Algorithms and Approximability", Hand book of Combinatorial Optimization D. Z. Du and P. M. Pardalos (Eds) (1998).

[41] K. R. Baker, E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, "Preemptive Scheduling of a Single Machine to Minimize Maximum Cost Subject to Release Dates and Precedence Constraints", Operations Research 31(2):381-386.(1983).

- [42] Husein N.A., "Machine Scheduling Problem to Minimize Multiple Objective Function", M.Sc. thesis, Dept. of mathematics, college of Education (Ibn AL-Haitham), Baghdad University (2012).
- [43] Dessouky, M. L. and Deogun, J. S. "Sequencing jobs with unequal ready times to minimize mean flow time ", SIAM, J. Comput. Vol. 10, No. 1 ,pp. 192-202. Feb. (1981)
- [44] Yanai S. and Fujie T., " On a dominance test for the single machine scheduling problem with release dates to minimize total flow time ", Journal of the Operation Research Society of Japan 47, No.2,96-111, (2004).
- [45] Li XL ."A new intelligent optimization-artificial fish swarm algorithm". PhD thesis, Zhejiang University, China, June(2003).
- [46] Li LX, Shao ZJ, Qian JX "An optimizing method based on autonomous animals: fish-swarm algorithm". Syst Eng Theory Practice 22(11):32–38(2002).
- [47] Jiang, M.Y., Yuan, D.F., "Artificial Fish Swarm Algorithm and Its Applications", *International Conference on Sensing, Computing and Automation*, pp.1782-1787, (2006).
- [48] Xiao, J.M., Zheng , X.M., Wang , X.H., "A Modified Artificial Fish-Swarm Algorithm", *IEEE the 6th World Congress on Intelligent Control and Automation*, Dalian China, pp.3456-3460,(2006).
- [49] Neshat, M., Sepidnam, G., Sargolzaei, M., Toosi, A. N., "Artificial Fish Swarm Algorithm: A Survey of the State-of-the-art, Hybridization, Combinatorial and Indicative Applications". *Artificial Intelligence Review*, 1-33, DOI 10.1007/s10462-012-9342-2.(2012).
- [50] W.T. Pan ,"A new evolutionary computation approach: Fruit Fly Optimization Algorithm", *Conference of Digital Technology and Innovation Management*(2011).
- [51] W.T. Pan, "A new fruit fly optimization algorithm: Taking the financial distress model as an example," *Knowledge-Based*

Systems, 26, pp.69-74.(2012).

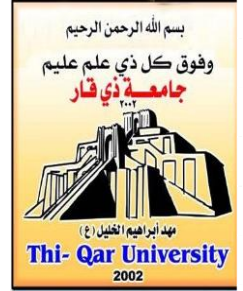
[52] H. Iscan., M. Gunduz," A Survey on Fruit Fly Optimization Algorithm", International Conference on Signal-Image Technology & Internet-Based Systems. IEEE 978-1-4673-9721-6/15 (2015).

المستخلص

تناولنا في هذه الرسالة دراسة مسألة جدولة n من المنتجات (jobs) على ماكينة واحدة لتصغير دالة هدف تتكون من مجموع دالتين. والهدف من هذه الرسالة هو إيجاد الحل الأمثل (Optimal solution) والحلول التقريبية (Near optimal solutions) (جدولة المنتجات) لتصغير دالة الهدف وهي مجموع الكلفة الكلية لزمن انسياب المنتجات وأكبر عمل متأخر عندما يكون للنتائج أزمنة تحضير غير متساوية، وتمثل المسألة بالصيغة: $(1/ r_i / \sum_{i=1}^n F_i + V_{\max})$ وهي مسألة معقدة من نوع NP-hard. لحل هذه المسألة استخدمنا طريقة التفرع والتقيد واقتراحنا قيدين أدنيين (LB2، LB1) كل من القيد الأدنى يعتمد على تجزئة المسألة الى مسألتين جزئيتين وإيجاد القيد الأدنى لكل مسألة جزئية ويكون القيد الأدنى للمسألة من مجموع القيد الأدنى. وطريقة تنقيبه للحصول على القيد الأعلى اللذان يستخدمان في عقدة الجذر في شجرة التفرع والتقيد. برهنا بعض الحالات الخاصة للحصول على الحل الأمثل دون استخدام طريقة التفرع والتقيد واقتراحنا وبرهنا بعض قواعد الهيمنة. نتائج الاختبارات الحسابية أثبتت بأن خوارزمية التفرع والتقيد المقترحة فعالة لغاية (35) نتاج في وقت أقل من أو يساوي (30) دقيقة.

استخدمنا طريقتين لإيجاد حلول تقريبية وهي: خوارزمية سرب السمك وخوارزمية أمثلية ذباب الفاكهة. التجارب الحسابية والنتائج العملية وجدت أن خوارزميات البحث يمكن أن تحل المسألة لغاية (6000) نتاج في فترة زمنية معقولة، وكذلك وجدت أن خوارزمية سرب السمك (Artificial Fish Swarm Algorithm) هي الأفضل في هذه المسألة من خوارزمية أمثلية ذباب الفاكهة عندما يكون $(n \leq 35)$ نتاج، إما في المسائل التي تكون فيها $(n > 35)$ نتاج فإن خوارزمية أمثلية ذباب الفاكهة (Fruit Fly Optimization Algorithm) هي الأفضل.

جميع الأساليب المعتمدة والخوارزميات والبرامج في هذا الرسالة برمجت باعتماد لغة البرمجة (MATLAB).



جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة ذي قار

كلية التربية للعلوم الصرفة

جدولة n من النتائج على ماكينة واحدة لتصغير مجموع دالتين

رسالة مقدمة الى

قسم الرياضيات/ كلية التربية للعلوم الصرفة / جامعة ذي قار وهي جزء من متطلبات نيل

درجة الماجستير في علوم الرياضيات

من قبل

أزهار مهدي عبادي

بكالوريوس رياضيات جامعة ذي قار 2014

بإشراف

الأستاذ المساعد الدكتور

محمد كاظم زغير الزويني