

Republic of Iraq

***Ministry of Higher Education and Scientific Research
University of Thi-Qar
College of Computer Sciences and Mathematics***

***Sequencing the Jobs to Minimize Single Machine
Scheduling Problem***

***A thesis
Submitted to the Council of College of Computer Sciences and
Mathematics /Thi-Qar University
in Partial Fulfillment of The Requirements for The Degree
of Master of Science in Mathematics***

***By
Sarah Kamil Hanoon
B.Sc. in Mathematics/University of Thi-Qar 2011***

***Supervised by
Assist. Prof. Dr. Mohammed K. Al-Zuwaini***

2018 A.D.

1439 A.H.



بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

...

وَأَنْزَلَ اللَّهُ عَلَيْكَ الْكِتَابَ وَالْحِكْمَةَ
وَعَلَّمَكَ مَا لَمْ تَكُنْ تَعْلَمُ وَكَانَ فَضْلُ
اللَّهِ عَلَيْكَ عَظِيمًا

صَلَّى
الْعَظِيمِ

سورة النساء من الآية ﴿١١٣﴾



Dedication

I dedicate my Humble effect

*To the victorious, glorious, supported Moulati, Al Zahraa
Fatima [peace be upon her]*

*To those who worked on my rest and the candles that light my
way [My father and My mother]*

*To those who have patience and bear and support me in the path
of knowledge... ..[My husband and my children]*

To my supporters in this life [my brothers and sisters]

*To my supervisor[Dr. Mohammed K, Al-Zuwaini] with
all respect and appreciation .*

To everyone who helped me in the mathematics department

SARAH

Acknowledgment

First, Thanks to **Allah** for all his blessing during my work.

Second, I would like to express my deep gratitude and appreciations to my supervisor: **Dr. Mohammed K. Al-Zuwaini** for his appreciable advice, important comments, support and encouragement during the research.

Third, I am deeply indebted to my family for their help and patience and support for me, "especially" my husband and my father and my mother until the end of my work.

Fourthly, Thanks are also due to Dr. Hussein Jameel and my friends Azhar Mahdi and Tahani Jabbar.

Certification

I certify that this thesis entitled (**Sequencing the Jobs to Minimize Single Machine Scheduling Problem**), by Sarah Kamil Hanoon, has been prepared under my supervision at the Department of Mathematics, College of Computer Sciences and Mathematics, University of Thi-Qar, in partial fulfillment of the requirements for the degree of Master of Science in Mathematics.

Signature:

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

Date: 24\ 6 \ 2018

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

Signature:

Name: Assist. Prof. Dr. Salam Jasim Majeed

Head of the Mathematics Department .

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

Date: 24\6\2018

Committee Certification

We certify that we have read this thesis, entitled (**Sequencing the Jobs to Minimize Single Machine Scheduling Problem**) and examining committee examined the student in its context 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: Akram B. Attar

Title: Professor Dr.

Date: 30\ 9 \2018

(Chairman)

Signature:

Name: Hussam Abed Ali Mohammed

Title: Assistant Professor Dr.

Date: 23\ 9 \2018

(Member)

Signature:

Name: Rashid Basheer Reheima

Title: Assistant Professor Dr.

Date: 23\ 9 \2018

(Member)

Signature:

Name: Mohammed K. Al-Zuwaini

Title: Assistant Professor Dr.

Date: 30 \ 9 \2018

(Member and Supervisor)

Approved by University committee of Graduate Studies .

Signature:

Name: Prof. Dr. Akram B. Attar

Dean of the College of Computer Sciences and Mathematics.

Date: 30 \9 \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). The objective is to find optimal and approximate solutions (Near optimal solutions) for scheduling n independent jobs and minimizing the objective function that consists of a sum of total late work and maximum lateness. This problem is strongly NP-hard. We derive the Lower Bound (LB), some special cases and dominance rule are proposed and proved. The computational results of the first special case solved with up to $n=50000$ jobs.

We use local search methods to find approximate solutions for the problem as: The Ant Colony Algorithm, Cats Swarm Algorithm and Simulated Annealing. Practically and through computational experiments, we found in the local search algorithms can solve the matter to the 100000 jobs at a reasonable time, as well, we found that Cats swarm is the best of the matter when the size $n \leq 10$ jobs. Either to the issues of the size $n > 10$ jobs, the Simulated Annealing algorithm is the best. We also use hybridization techniques HCSA, HASA, HSCS, HSAC and HCASA as approximate methods trying to find better approximations to solve the problem in a reasonable time period. We conclude that the HCASA algorithm is better when $n \leq 1000$ while the HCSA algorithm is better when $1000 < n$.

All programming in this theiss in language MATLAB.

Table of Abbreviations

Abbreviations	Meaning
ACO	Ants Colony Optimization
BAB	Branch and Bound
CE	Complete Enumeration
CSO	Cats Swarm Optimization
C_j	The completion time of job j
C_{max}	The maximum completion time (makespan)
DP	Dynamic programming
d_j	Due date of job j
$\overline{d_j}$	Deadline of job j
EDD	Earliest Due date
E_j	The earliness time of job j
E_{max}	The maximum earliness
F_j	The flow time of job j
F_m	Flow Shop Scheduling
HSAC	Hybridization the Simulated Annealing with Ants Colony

HSCS	Hybridization the Simulated Annealing with Cats Swarm
HCSA	Hybridization the Cats Swarm with Simulated Annealing
HASA	Hybridization the Ants Colony with Simulated Annealing
HCASA	Hybridization the Ants Colony and Cats Swarm in Simulated Annealing
JIT	Just in time
J_m	Job Shop Scheduling
LB	Lower bound
L_j	The lateness time of job j
L_{max}	The maximum lateness
MOF	Multiple Objective Function
MST	Minimum slack time
m	Number of machines
N	Set of jobs
NP	Non-deterministic polynomial
n	Number of jobs
O_m	Open Shop Scheduling

p_j	Processing time of job j
P_m	Parallel Machine Scheduling
$Pmtn$	Preemption
$Prec$	Precedence constraint
r_j	Release date of job j
SPT	Shortest processing time
S_j	Slack time of job j
SA	Simulated Annealing
T_j	The tardiness time of job j
T_{max}	maximum tardiness
UB	Upper bound
U_j	The unit penalty for late job
V_{max}	maximum late work
WSPT	Weighted Shortest Processing Time
w_j	The weight of job j for completion time
$<$	The precedence relation
$\alpha/ \beta / \gamma$	Classification of the problem

Contents

Introduction.....	1
Chapter One: Description of Machine Scheduling Problem	
1.1 Introduction.....	3
1.2 Scheduling Models.....	3
1.3 Basic Definitions.....	6
1.4 Classification Machine Scheduling Problem.....	7
1.5 The Problem Classification ($\alpha/\beta/\gamma$).....	9
1.6 Restrictions	11
1.7 Important Rules	13
1.8 Problem Classes and Computational Complexity.....	13
1.9 Solution Methods	15
Chapter Two: Single Machine Scheduling To Minimize a Function of Two Criteria	
2.1 Introduction	19
2.2 Formulation of the problem	19
2.3 Problem Decomposition	20
2.4 Special Cases	21
2.5 Derivation of Lower Bound (LB)	23
2.6 Dominance Rules	24
2.7 Computational Results of the First Special Case	26
Chapter Three: CSO/ACO/SA Algorithms to Give Approximate Solution on Single Machine Scheduling	
3.1 Introduction	30
3.2 Cat Swarm Optimization (CSO) Algorithm.....	31
3.3 Ants Colony Optimization (ACO)	34

3.4 Simulated Annealing (SA) Algorithm	38
3.5 Computational Results of Local Search Algorithms and Comparison.....	41
Chapter Four: Minimizing Bi-criteria with usage of Hybridization as a Tool	
4.1 Introduction.....	48
4.2 Computational Results of Local Search Algorithms and Comparison.....	50
Chapter Five: Conclusion and Future Works	
5.1 Conclusion.....	59
5.2 Future work.....	60
References.....	61

list of figures

Figure 3.1 CSO nucleus	33
Figure 3.2 Find the shortest path by ants	36
Figure 3.3 Flowchart of ACO algorithm.....	37
Figure 3.4 The results with n larger than 50 jobs for CSO, SA and ACO.....	47
Figure 4.1 The results with n larger than 50 jobs for HCSA, HSCS, HASA, HSAC and HCASA.....	56
Figure 4.2 The results with n larger than 50 jobs for CSO, SA, HCSA and HSCS	57
Figure 4.3 The results with n larger than 50 jobs for SA, ACO, HASA and HSAC	57
Figure 4.4 The results with n larger than 50 jobs for CSO, SA, ACO and HCASA	58

Introduction

This study focused on one of the most important subject in Operation Research which is called (Machine Scheduling Problem). The first theoretical development in scheduling problem was made by Johnson [1]. Two other essential results were shortly followed by Jackson [2], and Smith [3] was solved machine scheduling problems one tends and 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 algorithm are NP-hard. In practice this means that solving large instances of such optimality problems requires impracticable running times. The NP-hard of a problem suggests that it is impossible to find an optimal solution without use essentially implicit enumeration algorithms (branch and bound or dynamic programming algorithms). But this enumeration algorithm may be unable to solve problems with more than a handful of jobs, and the solution generated by simple heuristics may be far from the optimum [4].

In this thesis, we consider the problem of scheduling jobs on a single machine to minimize the total late work and the maximum lateness i.e. to minimize the multiple objective function denoted by $\sum_{j=1}^n V_j + L_{max}$. The first subproblem $1 // \sum_{j=1}^n V_j$ which is NP-hard see[5, 6, 7, 8, 9, 10]; Manal (2012)[11] solved single machine scheduling problem to minimize the sum of total completion times and total late works, Asmaa (2015) [12] solved also scheduling jobs on a single machine to minimize the sum of tardy jobs and total late work. While the problem maximum lateness ($1/ / L_{max}$) which is p-type and studied by Jackson (1955) [2], he proved

that L_{max} was solved by EDD rule. Many researchers have studied this problem see [13, 14, 15, 16, 17, 18, 19].

This study consists of five chapters:

Chapter one: gives a description of machine scheduling problems including a number of assumptions for machine, jobs and optimality criteria. Classification and representation of scheduling problem are mentioned. Furthermore, some types of machine scheduling problems, problem classes and computational complexity are represented.

Chapter two: presents some basic concepts of multi-criteria scheduling optimization with some definitions, special cases, lower bound we also included. As well as the dominance rules.

Chapter three: presents three local search methods to find optimal or near optimal solution.

Chapter four: presents hybridization methods Hybridization the Cats Swarm with Simulated Annealing (HCSA), Hybridization the Ants Colony with Simulated Annealing (HASA), Hybridization the Simulated Annealing with Cats Swarm (HSCS), Hybridization the Simulated Annealing with Ants Colony (HSAC) and Hybridization the Ants Colony and Cats Swarm in Simulated Annealing (HCASA) to solve the problem $1 / \sum_{j=1}^n V_j + L_{max}$.

Chapter five: is devoted to conclusions and outlook for future work.

Chapter One

Description of Machine

Scheduling Problem

1.1 Introduction

Scheduling plays a very important role in most manufacturing and production systems as well as in most information processing environments. The need of scheduling is also crucial in agriculture, hospitals,...etc. The main focus is on the efficient allocation of one or more resources to activities over time. Adopting manufacturing terminology is which job consists of one or more activities and is the machine resource and there are many definitions for machine scheduling, but the simplest one for understanding meant to be the allocation of resources over time to perform a collection of tasks (Baker [20]). Resources and tasks are called machines and jobs respectively, and both of them can take many forms. For example, one can consider a computer (or computers) as a machine (or machines) and the programs that are to be run on that computer (or computers) as jobs. Another example is when we consider the hospital equipment's as machines and the patients in that hospital as jobs. Scheduling generally speaking, means to assign machines to jobs in order and complete all jobs under the imposed constraints. The problem is to find an optimal processing order of these jobs on each machine to minimize the given objective function. Each job is to be processed at most by, one machine at a time such that each machine is capable of processing at most one job at a time.

1-2 Scheduling Models

The machine scheduling problem can be described as follows: There are n jobs (numbered $1, 2, \dots, n$), each of which consists of a given sequence of operations. Each operation corresponds to the processing of a job on one of m machines $m_i (i = 1, 2, \dots, m)$ for a given period of time. A schedule is feasible if there is no overlapping of time intervals

corresponding to the same job (so that a job can't be processed by two machines at once) or of time intervals corresponding to the same machine (so that a machine can't process two jobs at the same time). A scheduling model is defined by three separate elements: the machine environment, the optimality criterion, and a set of side constraint and characteristics [21]. Throughout this thesis, we only state the notations that is used for the single machine, jobs j_i ($i=1,2,\dots,n$) which has the following concepts:

- **Job (j) [22]:** 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.
- **Machine (m) [22]:** A machine is available to execute jobs and tasks. Different machine environments exist, such as single machine and parallel machines.
- **Preemption (pmtn)[22]:** Preemption (or job-splitting) is allowed during the processing of a job if the processing can be interrupted at any time (preempt) and resumed at a later time (even if on a different machine). The amount of processing already done on the preempted job is not lost.
- **Processing time (p_{ij}) [23]:** It is the processing time of job j on machine i . The subscript i is omitted if the processing time of job j does not depend on the machine or if job j is only to be processed on one given machine.
- **Due date (d_j) [24]:** 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 is referred to as deadline ($\bar{d}_j$) and when due date is constant for all jobs, then called common due date.

- **Release date (r_j) [24]:** Also known as ready time is the point of time that a job arrives at a machine and thus the processing starts.
- **Weight (w_j)[25]:** Denoting the importance of a job j relative to another job.

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

1- The completion time $C_j = \sum_{j=1}^j p_j$.

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 $V_j = \min\{T_j, p_j\}$.

The following performance criteria appear frequently in the literature [26].

For a given scheduling δ , we compute:

1. $C_{max}(\delta) = \max_{j \in \delta} \{C_j\}$ (maximum completion time).
2. $E_{max}(\delta) = \max_{j \in \delta} \{E_j\}$ (maximum earliness).
3. $L_{max}(\delta) = \max_{j \in \delta} \{L_j\}$ (maximum lateness).
4. $T_{max}(\delta) = \max_{j \in \delta} \{T_j\}$ (maximum tardiness).
5. $V_{max}(\delta) = \max_{j \in \delta} \{V_j\}$ (maximum late work).
6. $\sum_{j=1}^n w_j C_j(\delta)$ (total weighted completion time).

7. $\sum_{j=1}^n w_j L_j(\delta)$ (total lateness).
8. $\sum_{j=1}^n w_j T_j(\delta)$ (total weighted tardiness).
9. $\sum_{j=1}^n w_j V_j(\delta)$ (total weighted late work).

1.3 Basic Definitions

- **Definition [27]:** Suppose we have a problem P , any schedule (where is the set of all schedules) is said to be feasible if it satisfies the constraints of the problem P .
- **Definition [27]:** 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 [28]:** 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 [29]:** 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[29]:** 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 [28]:** The function $F(f, g)$ is said to be non-decreasing in both arguments, if for any pair of outcome value (x, y) of the

functions f and g , we have $F(x, y) \leq F(x+A, y+B)$ for each pair of nonnegative value A and B .

1.4 Classification Machine Scheduling Problem

Due to the large variety of machine scheduling problem, several classification schemes have been proposed based on different dimensions [30]. An example of such a dimension is the number of the available machines and the way how these machines are organized. 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. There are four classes of job shop scheduling problems.

1.4.1 Single Machine Scheduling [25]

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 give 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 complicity at an exponential rate as the number of jobs to be scheduled increases.

1.4.2 Parallel Machine Scheduling (P_m) [31]

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

1.4.3 Flow Shop Scheduling (F_m) [32]

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

1.4.4 Job Shop Scheduling (J_m) [32]

A job shop consists of two or more machines that perform specific operations and it consists of 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 operation is not fixed. This type of layout is typically used when the product variety is high and the product volume is low.

1.4.5 Open Shop Scheduling (O_m) [33]

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

1.5 The Problem Classification ($\alpha/\beta/\gamma$) [21]

Scheduling problems are often classified according to Graham's classification. This is three field notation $\alpha/\beta/\gamma$, where α describes the machine environment, (i.e, the basic notations of machine environments are only machine or multiple machines, identical or different machines). β provides details of the processing characteristics and the constraints. γ the field contains the objective to be minimize.

1.5.1 Machine Environment (α)[34]

We shall now describe the first field $\alpha = \alpha_1\alpha_2$ specifying the machine environment. $\alpha_1 \in \{\Phi, Q, P, F, R, J\}$, where:

$\alpha_1 = \Phi$: For one machine.

$\alpha_1 = P$: For identical parallel machine.

$\alpha_1 = Q$: For uniform parallel machine.

$\alpha_1 = R$: For unrelated parallel machine.

$\alpha_1 = F$: For flow shop machines.

$\alpha_1 = J$: For job shop machines.

If α_2 is a positive integer, then m (the number of machines) is a constant and equal to α_2 (i.e, $\alpha_2 = m; m \geq 2$)

If $\alpha_2 = 0$, then m assumed to be variable.

1.5.2 Job Characteristics (β)[35]

The second field $\beta \subseteq \{\text{prec}, \text{pmtn}, r_j, \overline{d_j}\}$ indicates a number of job characteristics, which defined as follows:

- **Prec:** Precedence relation $<$ between the jobs are specified when one job depends on completion of other jobs. If $J_k < J_j$ and required that J_k is completed before J_j can start. Otherwise, no precedence relation is specified.
- **Pmtn:** Preemption is allowed. The processing of any operation may be interrupted and resumed at later time other- wise no preemption allowed.
- r_j : Release date is present, and then each job may have different release dates. Otherwise, all jobs arrive at time 0.
- $\overline{d_j}$: Dead line is present, then each job may have deadline. Otherwise, no dead line.

1.5.3 Optimality Criteria (γ)

The third field γ refers to the optimality criterion or objective chosen. Where each f_j denotes an arbitrary cost function. The completion time of each job j (C_j) associated cost by $f_j(C_j)$. There are essentially two types of cost functions

$$f_{\max}(C) = \max\{f_j(C_j), j=1 \dots n\}, \text{ and}$$

$$\sum f_j(C) = \sum_{j=1}^n f_j(C_j)$$

The optimality criteria which are most commonly chosen involve the minimization of

$$f_{\max} \in \{L_{\max}, T_{\max}, V_{\max}, C_{\max}, E_{\max}, \dots, etc\}$$

$$\Sigma f_j \in \{ \Sigma V_j, \Sigma T_j, \Sigma L_j, \Sigma w_j V_j, \Sigma w_j C_j, \dots, etc \}$$

Hence, we set $\gamma \in \{f_{max}, \Sigma f_j\}$

To illustrate the two –field described above.

- i. $1/r_j/C_{max} + L_{max}$ is the problem of scheduling jobs with release dates on a single machine to minimize the sum the maximum of completion time and maximum lateness.
- ii. $F_2 / \text{prce} / \Sigma_{j=1}^n (V_j + F_j)$ is the problem of scheduling jobs with precedence constraints, on two machines flow shop to minimize the total late work and minimize the flow shop.

1.6 Restrictions [36]:

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

1.6.1 Restrictions on the Job

J_1 : The number of jobs are known and fixed.

J_2 : All jobs are available at the same time and are independent. Sometimes, each job j , ($j=1,2,\dots,n$) is characterized by a release date r_j and a due-date d_j (all data being integer).

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

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 : Each job can be processed by only one machine at the same time.

1.6.2 Restrictions on the Machines

M_1 : The number of machines is known and fixed.

M_2 : All machines are available at the same 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 : Each machine can process not more than one job at a time.

1.6.3 Other Restrictions [37]

- The technological constraints known in advance and are immutable.
- There is no randomness. In particular:
 - i. The processing times are known and fixed.
 - ii. The ready times are known and fixed.
 - iii. All other quantities needed to define a particular problem are known and fixed.

1.7 Important Rules:-

- 1- Shortest processing time (**SPT**) rule, that is sequencing the jobs in non-decreasing order of their processing times p_j . This rule solves the problem $1 / \sum_{j=1}^n C_j$ [3].
- 2- The Earliest Due Date (**EDD**) rule, that is sequencing the jobs in non-decreasing order of due dates d_j which solves the problem $1//L_{max}$. This rule also minimizes T_{max} for the $1//T_{max}$ problem [2].
- 3- The Weighted Shortest Processing Time (**WSPT**) rule, that is sequencing the jobs in non-decreasing order of p_j/w_j which solves $1//\sum_{j=1}^n w_j C_j$ problem [3].

1.8 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 [38]. 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 two 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 [39] 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 $1/d_j = d / \sum_{j=1}^n (w_j T_j + w'_j E_j)$ 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 [41].

1.9 Solution Methods

A scheduling problem is a special type of combinatorial optimization problem (A combinatorial optimization problem is either a minimization problem or maximization problem with an associated set of instances). The best known traditional methods of solutions for machine scheduling problem are as follows:-

1.9.1 Complete Enumeration Method (CEM) [42]

Complete enumeration methods generate one by one, all feasible schedules and then pick the best one. For a single machine problem of n jobs there are $n!$ different sequence. Hence for the corresponding m machines problem, there are $(n!)^m$ different sequence. This method may take considerable time as the time as the number $(n!)^m$ is very large even for relatively small values of n and m .

1.9.2 Branch and Bound Methods (BAB) [43]

Branch and bound methods (BAB) can be used for solving many combinatorial minimization problems. These methods are example of implicit enumeration approach, which find an optimal solution by examining subsets of a feasible solution. These procedures can be conveniently represented as a search tree. Each node of the search tree corresponds to subset of feasible solutions to a problem. A branching rule specifies how the feasible solutions at a node are partitioned into subsets, each corresponding to descendant node of the search tree. The scheduling problems that we consider require an objective function, to be minimized. A lower bound scheme associates a lower bound (LB) with each node of the search tree. The idea is to eliminate any node for which the lower bound is greater than or equal to the value of the best known feasible

solution. The branching rule describes how feasible solution at a node is partitioned into subsets. There are several types of branching rules for scheduling problems, the most common of which are forward branching and backward branching. In forward branching rule jobs are sequenced one by one from the beginning, while in a backward branching rule the jobs are sequenced one by one from the end. The bounding rule calculates a lower bound (LB) on an optimal solution for each sub problem generated by the branching rule. The well known methods of obtaining lower bounds for machine scheduling problems are:-

- 1- Relaxation of constraints.
- 2- Relaxation of objectives.
- 3- Langrangian relaxation.
- 4- Dynamic programming state – space relaxation.

To minimize an objective function of particular scheduling problem, first an upper bound (UB) of the minimum of this objective function is needed. This upper bound (UB) is the value for trial solution. At the beginning the trial solution may be found using a heuristic procedure. Finally, the branch and bound method can be improved by applying dominance rules that discard nodes before computing their lower bounds. These dominance rules are computationally useful as they reduce storage requirements on the computer as well as reducing computation time.

1.9.3 Dynamic Programming (DP) Method

Dynamic Programming (DP) method is an implicit enumeration technique which was 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 [4]. 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.

1.9.4 Heuristic Methods

Heuristic methods are applied to replace the enumeration techniques. Finding the optimal schedule take much time than it is useful Or when we do not want to spend much time in the development of a solution procedure for a problem that will occur for few times only or when an optimal schedule cannot be found. Reeves [44] gives the definition of heuristic as follows:-

A heuristic is a technique which seeks well (i.e., near optimal) solution at 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 is. Heuristics for machine scheduling problems can be broadly classified into construction and improvement methods. Construction heuristics are essentially single pass methods, which build a schedule from scratch by fixing, at each step, the position of a job in the sequence. The second category, the improvement methods (meta- heuristics), which are known as local search heuristics. The reason is that most of these methods can be superimposed on existing procedures to guide these procedures towards searching in more desirable regions of the solution space. In recent years, the improvement in

heuristic methods has become under the name of (Local Search Methods). Local search methods are widely used to obtain approximate solution to a variety of combinatorial optimization problems. For example categories of local search method are Ant Colony Optimization, Bees Algorithm and Simulated Annealing....etc.

Chapter Two

Single Machine Scheduling To Minimize a Function of Two Criteria

2.1 Introduction

In this chapter, we tackle the scheduling n independent jobs on a single machine to minimize the objective function.

This problem is the sum of total late work and maximum lateness which is denoted by $1 // \sum_{j=1}^n V_j + L_{max}$, and it consists of two sub problem: $1 // \sum_{j=1}^n V_j$ and $1 // L_{max}$

2.2 Formulation of the Problem

A set of n independent jobs $\{j_1, j_2, \dots, j_n\}$ be scheduled on a single machine with the conditions only one job j can be processed at a time. All jobs are available for processing at time zero, the precedence relation among the jobs are not supposed and preemption is not allowed. The goal is to find a processing order of the jobs that minimize the sum of the total late work and maximum lateness, i.e solve the $1 // \sum_{i=1}^n V_i + L_{max}$ problem.

Each job j has positive integer processing time (p_j) and due date (d_j). For a given schedule the completion time ($C_j = C_{j-1} + p_j$, and $C_0 = 0$) for $j=1, 2, \dots, n$ the late work $V_j = \min\{T_j, p_j\}$ and maximum lateness $L_{max} = \max\{L_j\} = \max\{C_j - d_j\}$, $j = 1, 2, \dots, n$ were computed. The objective is to find the schedule $\sigma = (\sigma(1), \dots, \sigma(n))$ of the job that minimizes the total cost which is formulated in mathematics forms as:

$$\begin{aligned}
R &= \min_{\sigma \in \delta} \{ \sum_{j=1}^n V_{\sigma(j)} + L_{\max(\sigma)} \} \\
\text{subject to:} \\
C_{\sigma(j)} &\geq p_{\sigma(j)} & j = 1, \dots, n \\
V_{\sigma(j)} &\leq T_{\sigma(j)} & j = 1, \dots, n \\
V_{\sigma(j)} &\leq p_{\sigma(j)} & j = 1, \dots, n \\
L_{\sigma(j)} &= C_{\sigma(j)} - d_{\sigma(j)} & j = 1, \dots, n \\
p_{\sigma(j)} &> 0, \quad d_{\sigma(j)} > 0, \quad V_{\sigma(j)} \geq 0
\end{aligned}
\left. \vphantom{\begin{aligned} R &= \min_{\sigma \in \delta} \{ \sum_{j=1}^n V_{\sigma(j)} + L_{\max(\sigma)} \} \right\}} \dots\dots (P)$$

where δ the set of all feasible solutions and $\sigma(j)$ denotes the position of job j in the ordering σ .

2.3 Problem Decomposition

In this section, the problem (P) is decomposed into two sub problem (SP_1) and (SP_2) which are simple structure of the original problem as follows :

- i. The total late work ($1/\sum V_j$) is NP-hard [6]

$$\begin{aligned}
R_1 &= \min_{\sigma \in \delta} \left\{ \sum_{j=1}^n V_{\sigma(j)} \right\} \\
\text{subject to} \\
C_{\sigma(j)} &\geq p_{\sigma(j)} & j = 1, \dots, n \\
V_{\sigma(j)} &\leq T_{\sigma(j)} & j = 1, \dots, n \\
V_{\sigma(j)} &\leq p_{\sigma(j)} & j = 1, \dots, n \\
p_{\sigma(j)} &> 0, \quad d_{\sigma(j)} > 0, \quad V_{\sigma(j)} \geq 0
\end{aligned}
\left. \vphantom{\begin{aligned} R_1 &= \min_{\sigma \in \delta} \left\{ \sum_{j=1}^n V_{\sigma(j)} \right\} \right\}} \dots\dots\dots (SP_1)$$

- ii. The maximum lateness ($1 / L_{\max}$) which is solved by the EDD rule [2] and the formulation form can be given as:

$$\left. \begin{array}{l}
 R_2 = \min_{\sigma(j)} \{L_{\max}\} \\
 \text{subject to} \\
 C_{\sigma(j)} \geq p_{\sigma(j)} \\
 L_{\sigma(j)} = C_{\sigma(j)} - d_{\sigma(j)} \\
 d_{\sigma(j)} > 0, \quad p_{\sigma(j)} > 0,
 \end{array} \right\} \begin{array}{l}
 j = 1, \dots, n \\
 j = 1, \dots, n \\
 j = 1, \dots, n
 \end{array} \dots\dots (SP_2)$$

2.3.1 Theorem [41]

$R_1 + R_2 \leq R$ where R_1 , R_2 and R is the minimum objective function values of SP_1 , SP_2 and P respectively.

2.4 Special Cases

A machine scheduling problem of type **NP**-hard is not easily solved and it is more difficult when the objective function is multi objective. Using some mathematical programming methods to find the optimal solution for this kind of problems, such as dynamic programming, branch and bound methods. Sometimes special cases for this problem can be solved. A special case for scheduling problem means finding an optimal schedule directly without using mathematical programming techniques. 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 [33]. In this section, some special cases for problem (P) are given and proved.

Case (1)

If EDD rule satisfies $C_j \leq d_j$ for each job j in EDD then EDD rule gives the optimal solution for problem $1 / \sum_{j=1}^n V_j + L_{\max}$.

Proof:-

Since $C_j \leq d_j \quad \forall j$ in EDD hence, no job will be tardy i.e, $T_j=0 \quad \forall j$ and $\sum_{j=1}^n V_j = 0$ then the problem $1/ C_j \leq d_j / \sum_{j=1}^n V_j + L_{\max}$ depend on $1/ C_j \leq d_j / L_{\max}$ which was already solved by EDD [2]. So EDD gives an optimal solution for $1/ \sum_{j=1}^n V_j + L_{\max}$. ■

Case (2)

If there is a schedule π satisfying $p_j=p$ and $d_j = jp \quad \forall j \in \pi$, $\pi=(1, \dots, n)$ then π is an optimal solution for the problem $1/ \sum_{j=1}^n V_j + L_{\max}$.

Proof:-

Since $d_j=jp$ and $p_j=p$, then $C_j = d_j$ and hence π gives just in time for each job $j \in \pi$ and $\sum_{j \in \pi} V_j + L_{\max} = 0$. ■

Case (3)

If $d_j = d \quad \forall j, j=1, \dots, n$ then any schedule gives an optimal solution for the problem $1/ d_j = d / \sum_{j=1}^n V_j + L_{\max}$.

Proof:-

First, from the condition of due date, any order of jobs gives an optimal solution for the problem $1/ d_j = d / L_{\max}$.

Second, since $d_j = d \quad \forall j$, so for any ordering of the jobs, the total late works equal to $\max\{\sum_{j=1}^n p_j - d, 0\}$ [6].

Hence, any schedule gives an optimal solution for the problem $1/ d_j = d / \sum_{j=1}^n V_j + L_{\max}$. ■

Case (4)

If $L_{\max}(EDD) = \sum_{j=1}^n V_j(EDD)$ then EDD rule gives an optimal solution for the problem $1/ \sum_{j=1}^n V_j + L_{\max}$.

proof:-

Since $L_{\max}(EDD) = \sum_{j=1}^n V_j(EDD)$ and a schedule which is an optimal with respect to $L_{\max}$ is also optimal with respect to $T_{\max}$ [45], then $T_{\max}(EDD) = \sum_{j=1}^n V_j$ and $T_{\max}(EDD)$ is a lower bound for total late work $\sum_{j=1}^n V_j$ [6] (i.e, $T_{\max}(EDD) \leq \sum_{j=1}^n V_j$). Then (EDD) rule is an optimal for $1/ \sum_{j=1}^n V_j$. Also, it is well known that $L_{\max}(EDD)$ is minimum for $L_{\max}$ [2]. Thus (EDD) rule is an optimal solution for the $1/ \sum_{j=1}^n V_j + L_{\max}$. ■

2.5 Derivation of Lower Bound (LB)

The lower bound for the problem (P) is based on decomposing (P) into two sub problems (SP_1) and (SP_2) as shown in Section (2.3). R_1 and R_2 was calculated to be the lower bound for SP_1 and SP_2 respectively, and thus applying theorem (2.3.1) gives lower bound (LB) for the problem (P).

For subproblem (SP_1) , $R_1 = T_{\max}(EDD)$ is a lower bound [6] while for subproblem (SP_2) , $R_2 = L_{\max}(EDD)$ is an optimal solution [2].

and $LB = (R_1 + R_2) = T_{\max}(EDD) + L_{\max}(EDD)$

2.1 Example: consider the five jobs in the table below.

J	1	2	3	4	5
p_j	3	5	6	7	3
d_j	3	8	8	13	11

Solution:

$$LB = T_{max} + L_{max}$$

$$= 11 + 11 = 22$$

The optimal solution=24 by using the CEM.

2.6 Dominance Rule

Because of branching planner, the size of the search tree is immediately related to the length of the current sequence (which represents the number of nodes). Hence, a preprocessing step is performed in order to remove as many situations as possible. Reducing the current sequence is done by using sundry dominance rules. 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 which has a lower bound that is less than the optimum solution [33]. Some of dominance rules are valid for minimization of the sum of discounted total late work and maximum lateness. As in the preprocessing step, similar dominance rules are also used within the branch and bound procedure to cut nodes that are dominated by others. These improvements lead to very large

decrease in the number of nodes to obtain the optimal solution. Below are the three dominance rules that are stated in order to decrease the number of nodes in search tree as well as decreasing the time.

2.6.1 Theorem

If i and j unscheduled jobs with $d_i \leq d_j$ and $p_i = p_j$, then job $i < j$ is an optimal solution for problem(P).

Proof:-

Let S_k partial sequence which its schedule, $k \subset N$ for $i, j \in \bar{k} = N - k$, and (S_k, i, j) be a schedule which is obtained by interchanging job i and j in (S_k, j, i) . Then all jobs other than i and j have the same completion time in (S_k, i, j) as in (S_k, j, i) . Since Potts and Van wessanhove [6] proved that (EDD) rule gives as optimal for $1/ \sum_{j=1}^n V_j$ proved that $p_i = p_j \quad \forall i$ then $\sum_{j \in S_k, i, j} V_j \leq \sum_{j \in S_k, j, i} V_j$ since $d_i \leq d_j$.

Also by (EDD) rule $L_{max}(S_k, i, j) < L_{max}(S_k, j, i)$, hence $i < j$ is an optimal solution for the problem (P). ■

2.6.2 Theorem

If $d_j = \max\{d_i\}, i=1, \dots, n$ and $d_j \geq t$ where $t = C_{max}$, then there exists an optimal sequence such that job j is ordered the last job.

Proof:

Since $d_j \geq C_{max}$ so job j is early job, and by condition $d_j = \max\{d_i\}$, and (EDD) rule gives an optimal for $1/ \sum_{j=1}^n L_{max}$. Hence there exists an optimal sequence such that job j is the ordered last job. ■

2.6.3 Theorem

If δ_K be a partial sequence which its jobs are scheduled $K \subset N$ for $i, j \in N - K$ and let C be a completion time of the last job δ_K . If $d_i \leq d_j$ and $C + p_i + p_j \leq d_i$ then $i < j$ in an optimal solution for the problem (P).

Proof:

Since $d_i \leq d_j$ and $C + p_i + p_j \leq d_i$ then jobs i and j are be early and $V_i = V_j = 0$, since EDD rule gives an optimal for L_{max} , then $i < j$ in an optimal solution for the problem (P). ■

2.7 Computational Results of the First Special Case

2.7.1 Test Problems

The data were generated in a thesis chapter in the same way as in [46] which can be generated as follows:

- ❖ The processing time p_j is uniformly distributed in the interval $[1,10]$.
- ❖ The due date d_i is uniformly distributed in the interval

$[P(1-TF-RDD/2), P(1-TF+RDD/2)]$; where $P = \sum_{j=1}^n p_j$

depends on the relative range of due date (RDD) and on the average tardiness factor (TF).

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.

2.7.2 Computational Results

The results in this chapter for a first special case are coded in MATLAB 8.3.0 (R2014a) and implemented on Intel (R) Core(TM) i3-6100U CPU @ 2.30 GHZ, with RAM 4.00 GB personal computer. In our computation, we use the condition that if the solution of an example with " n " jobs for any algorithm does not appear after 120 seconds, from its run; then this example is unsolved and this algorithm is active until the problem of size " n ".

2.7.2.1 Computational the First Special Case

Table 2.1 shows the results of the first special case and the value of the objective function with the times it can catch the optimal value for each value of " n " (problem size).

Table 2.2 gives the comparative of computational results and its times (in second which required algorithm to running) for the first special case, $n=30$ for problem (P). where:

Opt.= the optimal value.

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

Av. of opt= the average of cost for (10) examples for each size.

Note that: the first special case depends on the maximum lateness, then the positive value means the delay time and the negative value means the time of early in table 2.1.

Table (2.1): The results of the first special case when $n \in \{10, 20, 30, 40, 50, 100, 200, 400, 500, 1000, 2000, 10000, 20000, 30000, 50000\}$

n	Av. of opt	Av.of time
10	0	0.00004
20	0	0.000051
30	-0.1	0.000067
40	0	0.000088
50	0	0.00011
100	-1.3	0.00031
200	0	0.0011
400	0	0.0041
500	0	0.0064
1000	0	0.0248
2000	0	0.0985
10000	0	2.6020
20000	0	10.2505
30000	0	23.1195
50000	0	66.7267

Table(2.2): The results of the first special case when $n = 30$

n	Ex	Opt	Time
30	1	0	0.000272
	2	0	0.000052
	3	0	0.000047
	4	0	0.000047
	5	0	0.000046
	6	0	0.000044
	7	0	0.000042
	8	-1	0.000043
	9	0	0.000043
	10	0	0.000045
Av. of opt		-0.1	
Av. of time		0.000067	

Chapter Three

***CSO/ACO/SA Algorithms to Give
Approximate Solutions on Single
Machine Scheduling***

3.1 Introduction

We consider the scheduling n independent jobs on a single machine to minimize the multiple objective problem. Our aim is to find a schedule that minimizes the sum of total late work and maximum lateness, by using local search method. This problem is denoted by

$$1/ \sum_{i=1}^n V_i + L_{max}.$$

Local search heuristics in combinatorial dates back to the 1950s, when the first edge-exchange algorithms for the travelling salesman problem were introduced. We refer to Arts and Lenstra (1997) [47], as an overview of local search in combinatorial optimization.

It is clear to solve scheduling problems one tends to use (BAB) or (DP) to find optimal solutions. However, these approaches have two main disadvantages:

- It is mathematically complicated and thus a lot of time needs to be invested [48].
- When it concerns NP-hard problem, the computational time requirements are enormous for large sized problem.

To avoid these drawbacks we can appeal to heuristic methods. In the recent years, the improvement in heuristic methods has become under the name “local search methods”.

Local search is a meta heuristic method for solving computationally hard optimization problems. Local search can be used on problems that can be formulated as finding a solution maximizing a criterion among a number of candidate solutions. Local search algorithms move from solution to solution in the space of candidate solutions (the search space)

by applying local changes until a solution deemed optimal is found or a time bound is elapsed.

Local search algorithms are widely applied to numerous hard computational problems, including problems of computer science (particularly artificial intelligence), mathematics, operations research, engineering, and bioinformatics. In this chapter, we will study Cat Swarm, Simulated Annealing and Ant Colony Algorithms.

3.2 Cat Swarm Optimization (CSO) Algorithm

Cat swarm optimization (CSO) algorithm was first introduced by Chu and Tsai [49] in 2006, and improved in 2011 by M. Orouskhani and et al. [50]. It was applied in combinatorial problem in 2014 [51]. It has been known with the Cat Swarm or CSO algorithm. As its name clearly shows, this algorithm is a based algorithm, and it is one of new induction optimization algorithms based on the swarm intelligence.

Based on cat surveillance, her behaviour was generally divided into two modes. These two behaviors are:

The first, the Seeking Mode (SM) when the cat is resting, Cats in this phase are always alert and slow to move and remain awake and spend most of their time in watching the surrounding environment. If the cats decide to move, the movement will be slow and warned [52].

The second behaviour the Tracing "active" Mode (TM) when the cat is hunting, in this phase cats chases very quickly. If they feel the presence of prey.

These two phases have been formulated mathematically to solve optimization issues. Also, these two modes are combined by mixture ratio (MR) which identifies the SM with the TM [53].

3.2.1 Description Nucleus of CSO

To combine these two methods in algorithms, the MR must be defined. MR that dictates the orders to combine search style with style tracking effect. MR decides how many cats will be moved the process of research method.

The CSO process can be summarized as follows: [54]

First:- create a n-cat and determine the initial values of the site, and the fastest indicator for each cat.

Second:- estimate the fitness value of each cat according to fitness function and keep the best cat in memory.

Thirdly:- the cat is applied in the field of research or the TM according to the flag of the cat.

Fourth:- once the process is completed, the number of cats is captured and placed in the research or tracking mode according to MR value.

Fifth:- verify the status of completion. If satisfactory, the program ends unless; return the second step.

The total CSO process is illustrated in Figure 3.1 as it shows the CSO nucleus: [55]

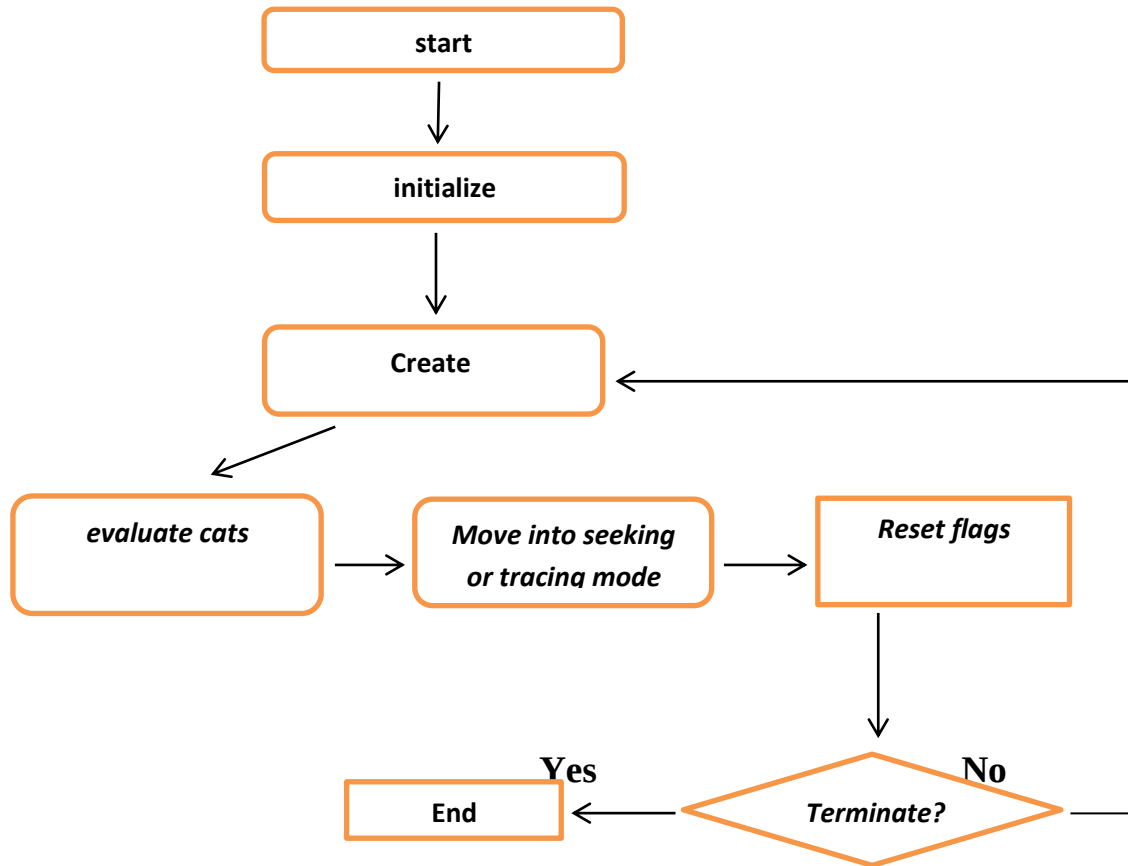


Figure (3.1): CSO nucleus.

3.2.2 The Complete CSO Algorithm [53]

The full mode is composed of the SM and TM combined by MR. The flag is used to determinate the mode of each cat in swarm. The description of the process can be given as follows:

Begin:

- (1) Generate n cats
- (2) Initialize g best with the lowest fitness cat in swarm.
- (3) for each cat in swarm

IF the flag of the selected cat is TM

Apply selected cat into TM process

Else

Apply selected cat into SM process

End if

Update g best

End for

(4) Re-pick number of cats and set them into TM according to MR, and set other cats in SM. If the condition is to terminate yes then complete the program Else repeat (3) and (4).

End.

3.3 Ants Colony Optimization (ACO)

In the early 1990s, ACO [56] was introduced by Dorigo and his colleagues as a novel nature-inspired meta heuristic for the solution of hard Combinatorial Optimization (CO) problems. ACO is a part of the larger field of a swarm intelligence technique through which scientists studied the behavior patterns of termites, ants and other social insects [57] and inspired by the behavior of ants in their search for food. Also. The ACO is an experimental and error-based search algorithm that gives an acceptable solution (may or may not be the best solution). So it is used to solve long-time problems using a computer such as NP-Complete or questions that need to be tried all the possibilities even reach the desired solution if found. Also the ant colony has been studied by Ali and Farouk in 2013 [58], Araibi 2017 [59] and Hussam et al 2015 [60].

3.3.1 ACO Algorithm

The idea of algorithm came from simulating the process of searching of food in the ants are as follows:

1. A group of ants start from the cell in several random directions (this process is done only in the first time in subsequent times, each track is tested and a certain path chosen as we will see later).
2. During the passage of the ant, it secretes a material called pheromone in a certain percentage.
3. When it finds a source of food, it takes a quantity of it and returns to the cell by choosing a certain path (the path that contains the largest amount of pheromone). During its return it secrete the same amount of the pheromone.
4. When the ant starts from the cell again, it will test the amount of pheromone in each path and choose the path that contains the largest amount of pheromone. (during the return trip, the quantity of pheromone that an ant leaves on the ground may depend on the quantity and quality of the food, the pheromone trails will guide other ants to the food source [61]).
5. The amount of pheromone is updated every time period (the concentration of the pheromone fades over time. The ants' life is millions of years .

Note that the shortest path will always contain the largest amount of pheromone and therefore all the ants will pass it. The advantage of this algorithm is that it is dynamic in the sense that if there is an obstacle in the shortest path, the ants will choose a new path in the same way.

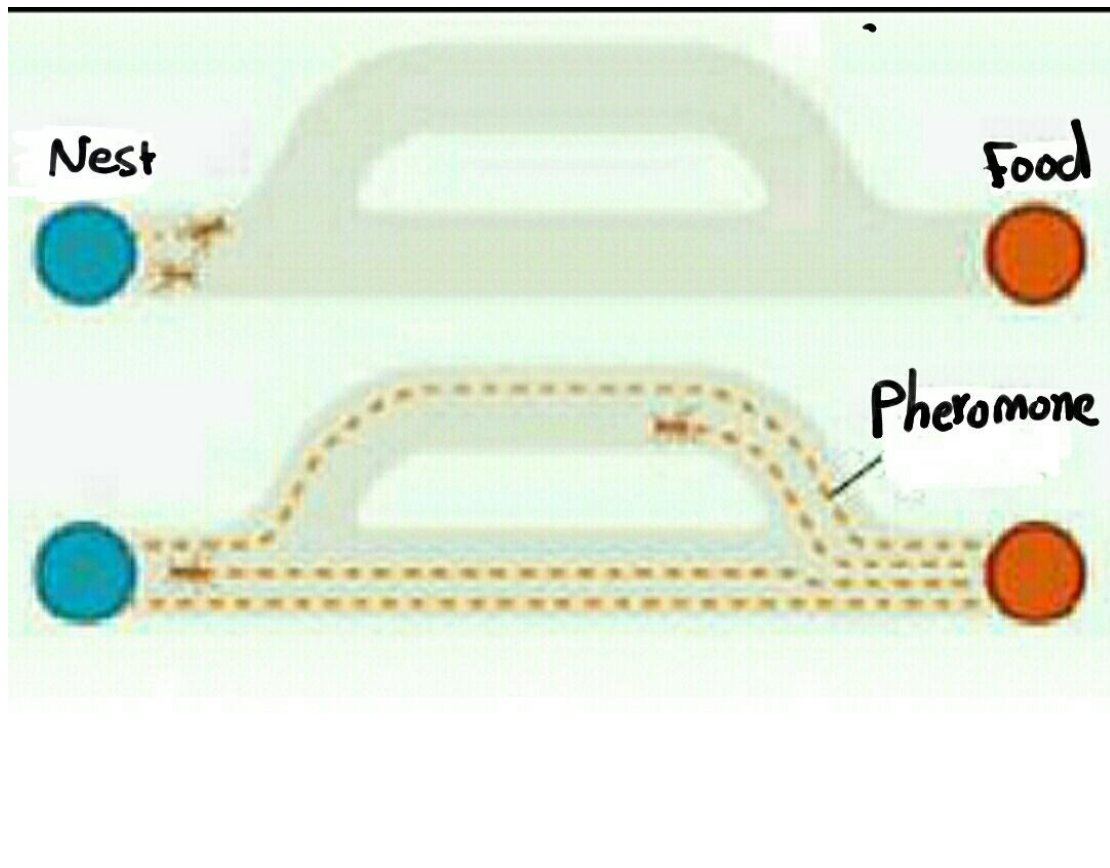


Figure (3.2): Find the shortest path by ants.

Steps of the ants algorithm [62]:

Step1. Set pheromone trails to be small constant.

Step2. While (termination condition not met).

Step3. Place each ant on initial node (its index usually).

Step4. Repeat.

Step5. For each ant do.

Step6. Choose next node with Apply State Transition Rule.

Step7. End for.

Step8. Until "each ant builds one solution".

Step9. Choose the best solution.

Step10. Apply Local Update pheromone.

Step11. Apply Global Update.

Step12. End While.

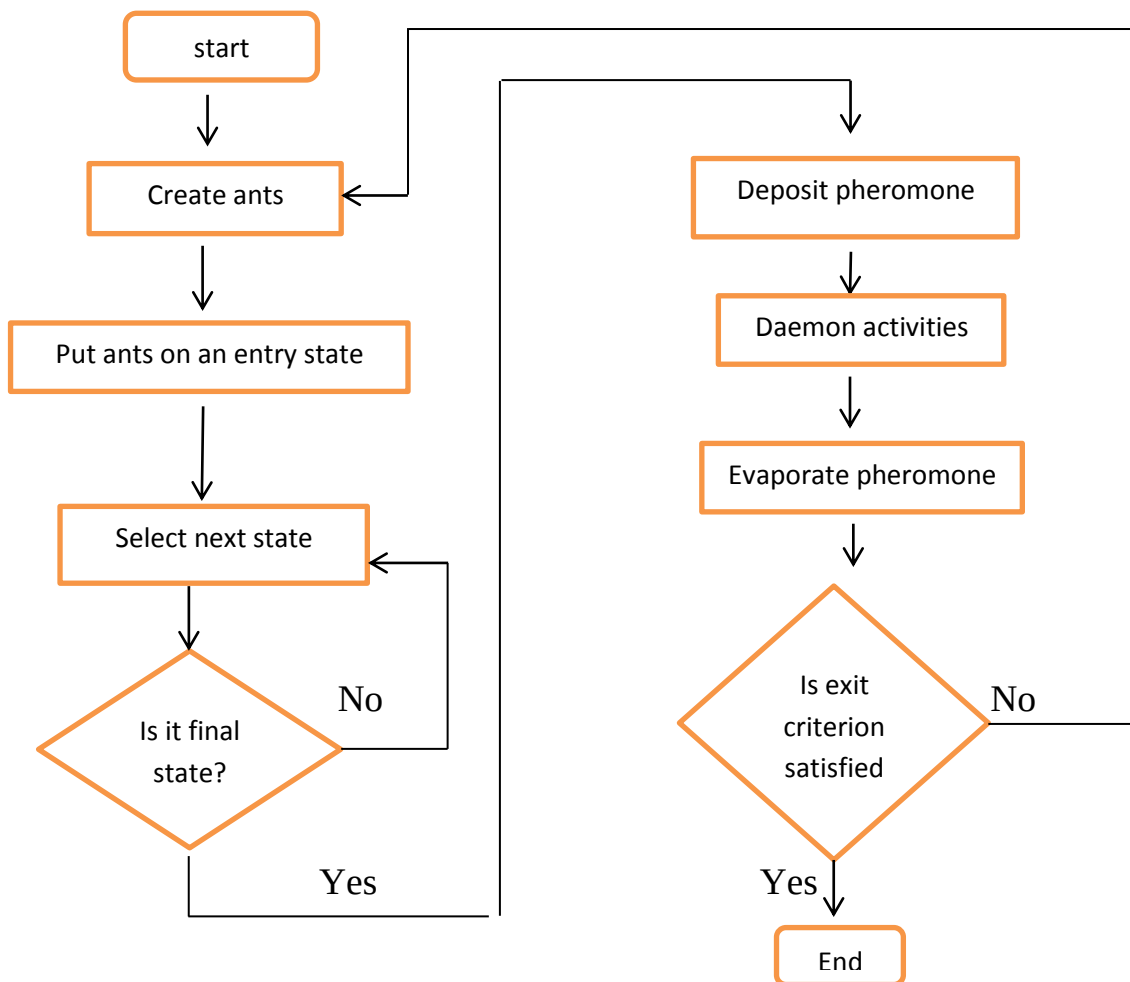


Figure (3.3): Flowchart of ACO algorithm.

3.3.2 Actual Ants

In the experiment "Linepithema humile", Deneubourg et al. (1990) on the argentine ant, which provides a clear demonstration of how to organize the ants by the effects of pheromones and called this experiment,

the double bridge is two paths, one of which twice the length of the other and both used for crossing between the ant site and food source in minutes, the ant Choose in the most cases of the shorter branch. This has been achieved by the ants that leave marks of pheromone and are followed by other ants in their search for food. Obviously the faster ants that go back to the nest are those followed the shorter road back and forth. Although ant species are almost blind, they can still communicate with the environment and with each other by means of substances they release.

3.4 Simulated Annealing (SA) Algorithm

Simulated annealing algorithm is a well-known neighborhood search approach, deriving its acceptance mechanism from annealing process in order to let the current solution escape from the local optimal. It starts with an initial solution and navigates around it by any kind of neighborhood search structure. In order to escape from local optima, it leads ultimately to the optimum solution. 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, steps are not strictly required to produce improved solutions, but each step has a certain probability of leading to improvement; at the start all steps all equally likely, but as the algorithm progresses, the tolerance for solutions are 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 optimal.

This tells us that we can have some confidence in the answer generated by a well formulated SA implementation, but it 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).

Hence one finds a few references in the literature that addresses SA algorithm. In(2009), Seyed, et al [63]used the SA method to solve an problem on more than one machine is a matter of type NP-hard in a reasonable computation time, In(2013), Sergio, et al [64] used it solve an problem of the NP-hard type with release dates, In(2014), Nakandhrakumar and Balachandar[65] found approximate solutions to solve the Job Shop Scheduling Problem by SA method.

The following structure gives the outline of SA:

SA Algorithm [66]:

Let 's term

S_c : Candidate schedule

S_o : Best schedule found so far

S_k : schedule constructed at K iteration (K= iteration counter)

$G(S_o)$: Value of best schedule

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

$G(S_c)$: Value of candidate schedule

$$rand(x) < \exp\left(\frac{G(S_c) - G(S_k)}{\beta_k}\right)$$

Where, x is a random variable having uniform distribution U[0; 1].

β_k is called cooling parameter in annealing terminology usually $\beta_k = a^k$, where $a \in [0,1]$.

Step (1): Initialize

Set $K=1$

Let S_k be a randomly create sequence

Let $S_k = S_o$

Then $G(S_k) = G(S_o)$

Step (2): Generate a perturbed sequence S_c with one of the neighborhoods operators and set $k = k + 1$.

Step (3): Evaluate the $G(S_c)$ values.

Step (4): If $G(S_c) < G(S_o)$, then let $S_k = S_c$, $G(S_o) = G(S_c)$. Set $k = k + 1$ and go to step 7.

Step (5): Generate a random number x .

Step (6): If $rand(x) < \exp(\frac{G(S_c)-G(S_k)}{\beta_k})$, then let $S_k = S_c$, $K = N$.

Step (7): If $k < N$, then go to step 2.

Else

Let $\beta_{k+1} = a\beta_k$ and $k = 0$.

Step(8): Stop best value of S_o stored in best.

3.5 Computational Results of Local Search Algorithms and Comparison

We use the test problems introduced in subsection (2.7.1) for the local search algorithms.

3.5.1 Computational Results

In our computation, we use the condition that if the solution of an example with " n " jobs for any algorithm does not appear after 600 seconds, from its run; then this example is unsolved and this algorithm is active until the problem of size " n ".

3.5.2 Comparative Effective of Local Search Algorithms

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

Table 3.2 shows the values of each local search algorithms and how many times that each of them catch the best value. Table 3.3 gives the comparative of computational results and its times (in second which required each algorithm to running) for 3 approaches, $n=1000$ for problem (P). where:

Opt.= the optimal value by CEM.

CSO = the value found by Cats Swarm Algorithm.

ACO= the value found by Ants Colony Algorithm.

SA= the value found by Simulated Annealing.

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

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

Av. of Cost = the average of cost for (10) examples for each size.

/ = refer to the unsolved example.

Table(3.1):The performance of local search methods and the optimal solution for $n \in \{5, 10\}$

n	Ex	Opt	CSO	SA	ACO
5	1	30	30	31	34
	2	41	41	41	41
	3	22	22	22	22
	4	32	32	32	32
	5	25	25	25	25
	6	31	31	31	33
	7	29	29	30	30
	8	57	57	57	59
	9	40	40	40	47
	10	15	15	15	24
No. of opt.			10	8	4
Av. of time			1.6809	0.1153	0.0012
Av. of cost			32.2	32.4	34.6
10	1	121	121	121	122
	2	67	67	67	79
	3	33	33	33	70
	4	76	76	76	76
	5	22	22	22	43
	6	70	70	70	76

	7	27	27	27	36
	8	122	122	122	125
	9	57	57	58	68
	10	54	54	54	65
No. of opt.		10	9	1	
Av. of Time		1.7733	0.1285	0.0018	
Av. of cost		64.9	65	76.6	

Table (3.2): The performance of local search methods for problem (P) with $n \in \{50, 100, 200, 500, 1000, 2000, 5000, 10000, 15000, 20000, 30000, 50000, 100000\}$

n	CSO		SA		ACO	
	Av. of cost	Av. of time	Av. of cost	Av. of time	Av. of cost	Av. of time
50	278.2	2.6432	263.4	0.1797	391.3	0.0221
100	533.5	3.7746	470.9	0.2513	782.9	0.1084
200	1197.3	5.9832	1051.3	0.4042	1686.8	0.6430
400	1989.7	10.6150	1763	0.7304	3331.4	4.8082
500	2887	12.7189	2534.6	0.8836	4175.3	8.9643
1000	6557.7	24.8451	5781.8	1.7305	8782.1	72.3901
2000	12489	50.3983	11013	3.5265	/	/
5000	26821	128.7733	25899	9.0975	/	/
10000	68932	262.7634	67369	18.4783	/	/
15000	94260	377.0454	93634	23.0555	/	/

20000	/	/	117930	30.9836	/	/
30000	/	/	142000	49.2750	/	/
50000	/	/	374290	86.9954	/	/
100000	/	/	660240	213.4520	/	/

Table (3.3): A Comparison of algorithms for problem (P), n=1000

n	Ex	Best	CSO	SA	ACO
1000	1	5554	6707	5554	8572
	2	7036	7894	7036	9420
	3	7018	7875	7018	9370
	4	7745	8414	7745	9580
	5	5484	5902	5484	8147
	6	3321	4159	3321	7541
	7	5445	6670	5445	8495
	8	1815	2003	1815	7701
	9	7932	8567	7932	9717
	10	6468	7386	6468	9278
No. of Best			0	10	0
Av. of Time			24.8451	1.7305	72.3901
Av. of cost			6557.7	5781.8	8782.1

3.5.3 Summary of Experimental Evaluation of Local Search Methods

In the table 3.4, we summarize the results of table 3.1 by viewing how many times that the algorithm catch the optimal value only and their sum, for each number of jobs and for the two local search methods.

Table (3.4): summary of results of table (3.1)

N	CSO	SA	ACO
5	10	8	4
10	10	9	1
Sum	20/20	17/20	5/20

In the table 3.5, we summarize the results of table 3.2 by viewing how many times that the algorithm catch the best value only, and their sum. For each number of jobs for the two local search algorithms.

Table (3.5): summary of results of table (3.2)

N	CSO	SA	ACO
50	5	9	0
100	0	10	0
200	0	10	0
400	1	10	0
500	0	10	0

1000	0	10	0
2000	1	10	/
5000	3	10	/
10000	4	6	/
15000	3	8	/
20000	/	10	/
30000	/	10	/
50000	/	10	/
100000	/	10	/
Sum	17/140	133/140	0/140

In the table 3.6, we give the activity of local search algorithms, (i.e, give the maximum number of jobs " n ") that the local search algorithms can solve the $(1/ \sum_{i=1}^n V_i + L_{max})$ problem with reasonable time, (i.e, according to the condition that had been given in subsection (3.5.2)).

Table (3.6): shows the activity of the local search methods

Algorithm	Active until (maximum no. of jobs)
CSO	15000
SA	100000
ACO	1000

From comparative study of the introduced local search methods and precise vision of all the above tables, we found that for the $(1/ \sum_{i=1}^n V_i + L_{max})$ problem the CSO Algorithm has obtained the best results for the problem of size less than or equal (10) jobs. For the problem of size larger than (10) jobs, we found that the SA Algorithm has the best results.

In Figure 3.4 it is clear that the SA algorithm has good results as compared to other algorithms CSO and ACO.

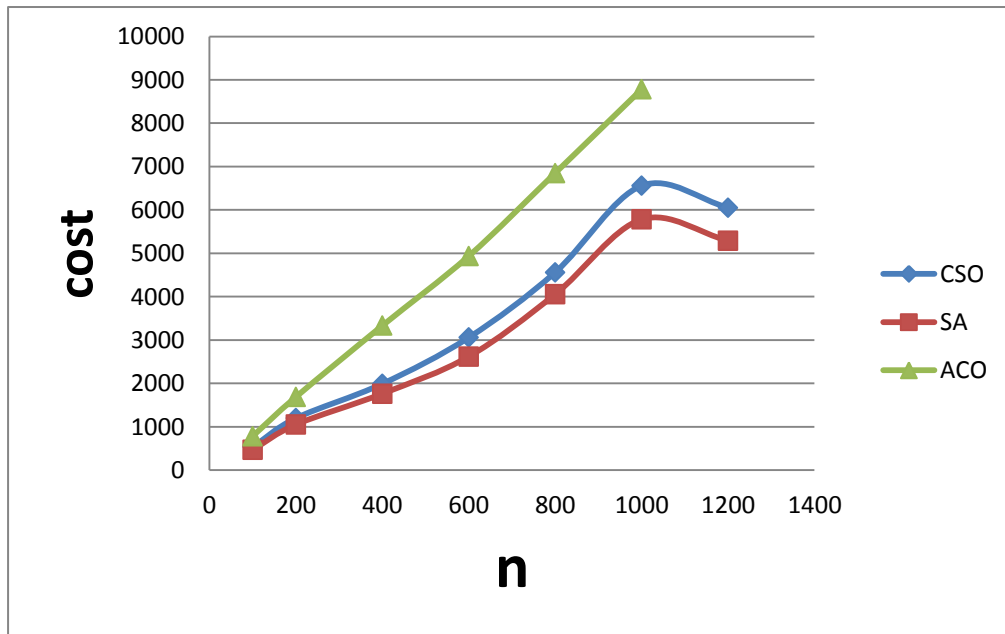


Figure (3.4): The results with n larger than 50 jobs for CSO, SA and ACO

Chapter Four

***Minimizing Bi-criteria with usage
of Hybridization as a Tool***

4.1 Introduction

Meta-heuristics are used to solve the computationally hard optimization problems. Meta-heuristics consist of a high level algorithm that guides the search using other particular methods. Meta-heuristics can be used as a standalone approach for solving hard combinatorial optimization problems. But now the standalone approach is drastically changed and attention of researchers has been shifted to consider another type of high level algorithms, namely hybrid algorithms. There are at least two issues that have been considered while combining more than one meta-heuristics:

- (a) how to choose the meta-heuristic methods.
- (b) how to combine the chosen heuristic methods into new hybrid approaches.

Unfortunately, there are no theoretical foundations for these issues. For the former, different classes of search algorithms can be considered for the purposes of hybridization, such as exact methods, simple heuristic methods and meta-heuristics. Moreover, meta-heuristics themselves are classified into local search based methods, population based methods and other classes of nature inspired meta-heuristics. Therefore, in principle, one could combine any methods from the same class or methods from different classes [67]. Hence, one finds a few references in the literature that addresses Hybridization algorithms, (2006), AL-Zuwaini[35] studied the hybridization of genetic with Simulated to solve the problem, in (2010) A. Jamili, et al [68] hybridized Simulated with electromagnetism-like mechanism to solve a periodic job shop scheduling problem, in (2016) Kashif Akram et al [69], used SA hybridized with quenching for solving job shop scheduling problem.

In this chapter, we study hybridization for the three approximate methods (Cats Swarm, Ants Colony, Simulated Annealing) as follows:

- Hybridization the Cats Swarm with Simulated Annealing (HCSA).
- Hybridization the Ants Colony with Simulated Annealing (HASA).
- Hybridization the Simulated Annealing with Cats Swarm (HSCS).
- Hybridization the Simulated Annealing with Ants Colony (HSAC).
- Hybridization the Ants Colony and Cats Swarm in Simulated Annealing (HCASA).

4.1.1 The HCSA

The best solution is chosen by the Cats Swarm algorithm and then is improved by the Simulated Annealing algorithm. The aim is to obtain a solution in less time and less cost, depending on the improvement of the parents resulting from the (CSO) algorithm to be introduced in the (SA) algorithm to obtain more efficient sons than the parents.

4.1.2 The HASA

The use of the ants algorithm is to find a preliminary solution of the initial problem and then improved by introducing it in the simulated algorithm, i.e (Hybridization of ants algorithm with simulated algorithm). An efficient scheduler is to be obtained at a reasonable time.

4.1.3 The HSCS

In this hybridization, an initial solution is initiated by using the simulated algorithm. This solution is then modified by the cats algorithm to reach the final solution.

4.1.4 The HSAC

Hybridization of the Ants algorithm by sequential taking as an initial solution of the simulated algorithm and then inserted with the Ants algorithm to improve the first solution.

4.1.5 The HCASA

Hybridization of the Simulated algorithm in Ants and Cats i.e, the first solution is determined by choosing the best solution resulting from the Ants or Cats algorithm and then is improved by the simulated algorithm.

4.2 Computational Results of Local Search Algorithms and Comparison

We use the test problems introduced in subsection 2.7.1 for the local

4.2.1 Computational Results

The local search algorithms in this chapter (CSO, SA, ACO, HCASA, HSCS, HASA, HSAC and HCASA),

4.2.2 Comparative Effective of Local Search Algorithms

Table 4.1, table 4.2 and table 4.3 gives difference of the average of computational results and how many average of times that each of them catch the best value of the 4 approaches with $n=50, 100, 200, 500, 1000, 5000$ and 10000 jobs for problem (P).

Table (4.1): Averages deviations about the best mean for problem (P).

n	CSO		SA		HCSA		HSCS	
	Av. of cost	Av. of time	Av. of cost	Av. of time	Av. of cost	Av. of time	Av. of cost	Av. of time
50	278.2	2.6432	263.4	0.1797	265.6	5.4714	264.4	2.9741
100	533.5	3.7746	470.9	0.2513	476.8	7.8043	472.6	4.2695
200	1197.3	5.9832	1051.3	0.4042	1033.8	12.2987	1054.5	6.7204
400	1989.7	5.9832	1763	0.4042	1744.7	12.2987	1755.7	6.7204
500	2887	12.7189	2534.6	0.8836	2483.9	26.3207	2529.8	14.4530
1000	6557.7	24.8451	5781.8	1.7305	5716.5	51.3133	5760.8	28.2348
2000	12489	50.3983	11013	3.5265	10922	104.8888	10986	87.2605
5000	26821	128.7733	25899	9.0975	25462	266.2223	25748	146.1254
10000	68932	262.7634	67369	18.4783	65742	548.5345	/	/

Table (4.2): Averages deviations about the best mean for problem (p).

n	SA		ACO		HASA		HSAC	
	Av. of cost	Av. of time	Av. of cost	Av. of time	Av. of cost	Av. of time	Av. of cost	Av. of time
50	263.4	0.1797	391.3	0.0221	263.9	0.2181	265.4	0.3915
100	470.9	0.2513	782.9	0.1084	471	0.4747	473.2	0.6099
200	1051.3	0.4042	1686.8	0.6430	1056.4	1.6799	1058.3	1.4534
400	1763	0.7304	3331.4	4.8082	1764.4	10.3751	1755.4	6.2230
500	2534.6	0.8836	4175.3	8.9643	2534.8	18.8087	2529	10.6705
1000	5781.8	1.7305	8782.1	72.3901	5796.4	146.3028	5765.8	75.7601

Table (4.3): Averages deviations about the best mean for problem (p).

n	CSO		SA		ACO		HCASA	
	Av. of cost	Av. of time	Av. of Cost	Av. of time	Av. of cost	Av. of time	Av. of cost	Av. of time
50	278.2	2.6432	263.4	0.1797	391.3	0.0221	264.5	2.8613
100	533.5	3.7746	470.9	0.2513	782.9	0.1084	475.9	4.2398
200	1197.3	5.9832	1051.3	0.4042	1686.8	0.6430	1047.8	7.6353
400	1989.7	10.6150	1763	0.7304	3331.4	4.8082	1758.7	21.0052
500	2887	12.7189	2534.6	0.8836	4175.3	8.9643	2475	31.5534

1000	6557.7	24.8451	5781.8	1.7305	8782.1	72.3901	5674.9	171.5062
-------------	--------	---------	--------	--------	--------	---------	--------	----------

HCSA = the value found by HCSA algorithm.

HASA = the value found by HASA algorithm.

HSCS = the value found by HSCS algorithm.

HSAC = the value found by HSAC algorithm.

HCASA = the value found by HCASA algorithm.

4.2.3 Summary of Experimental Evaluation of Local Search Methods

In the table 4.4, table 4.5 and table 4.6, we summarize the results of table 4.1, table 4.2 and table 4.3 respectively. by viewing how many times that the algorithm catch the best value only, and their sum for each number of jobs for the four local search algorithms.

Table (4.4): summary of results of table (4.1).

n	CSO	SA	HCSA	HSCS
50	5	9	6	7
100	0	4	3	4
200	0	2	6	4
400	1	2	7	3
500	0	2	6	2
1000	0	2	4	4

2000	1	4	8	1
5000	3	3	9	4
10000	0	1	7	/
Sum	10/90	29/90	56/90	29/90

Table (4.5): summary of results of table (4.2).

n	SA	ACO	HASA	HSAC
50	9	0	7	7
100	4	0	4	3
200	5	0	3	2
400	3	0	5	5
500	5	0	1	4
1000	6	0	0	5
Sum	32/60	0	20/60	26/60

Table (4.6): summary of results of table (4.3).

n	CSO	SA	ACO	HCASA
50	5	7	0	8
100	0	7	0	3
200	0	3	0	7

400	1	4	0	7
500	1	3	0	7
1000	0	5	0	5
Sum	7/60	29/60	0	37/60

In the table 4.5, we give the activity of local search algorithms, (i.e, give the maximum number of jobs " n ") that the local search algorithms can solve the $(1/ \sum_{i=1}^n V_i + L_{max})$ problem with reasonable time, (i.e, according to the condition that had been given in sub section (4.3.2)).

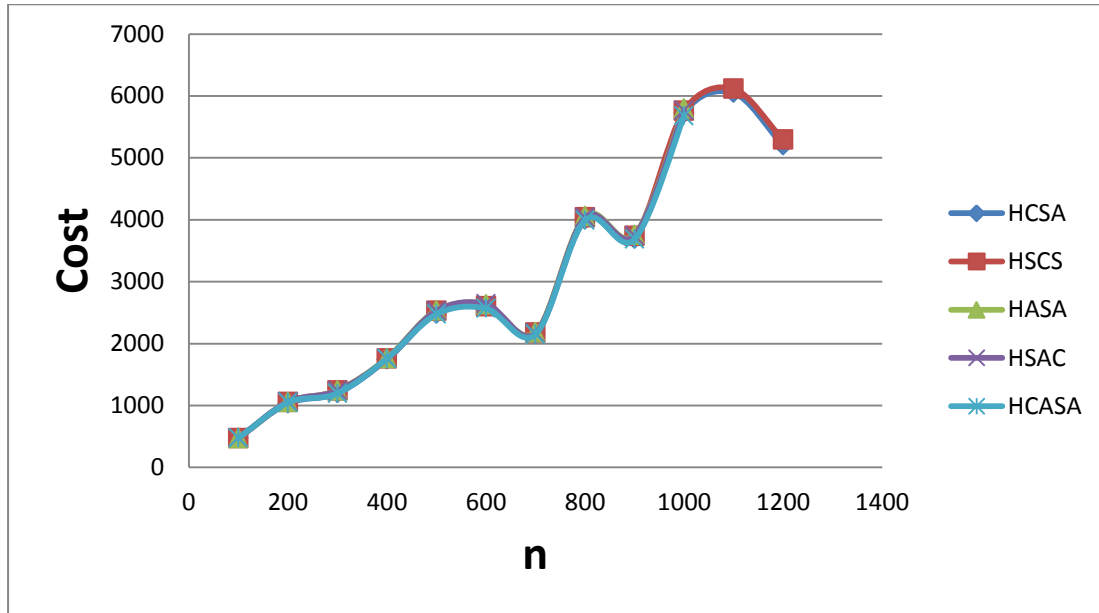
Table (4.5): shows activity of the hybrid local search methods.

Algorithm	Active until (maximum no. of jobs)
HCSA	10000
HSCS	5000
HSAC	1000
HASA	1000
HCASA	1000

From a comparative study of the introduced local search methods and precise vision of all the above tables, we found that for the $(1/ \sum_{i=1}^n V_i + L_{max})$ problem the SA algorithm has obtained the best results for the problem of any local search method. We also concluded

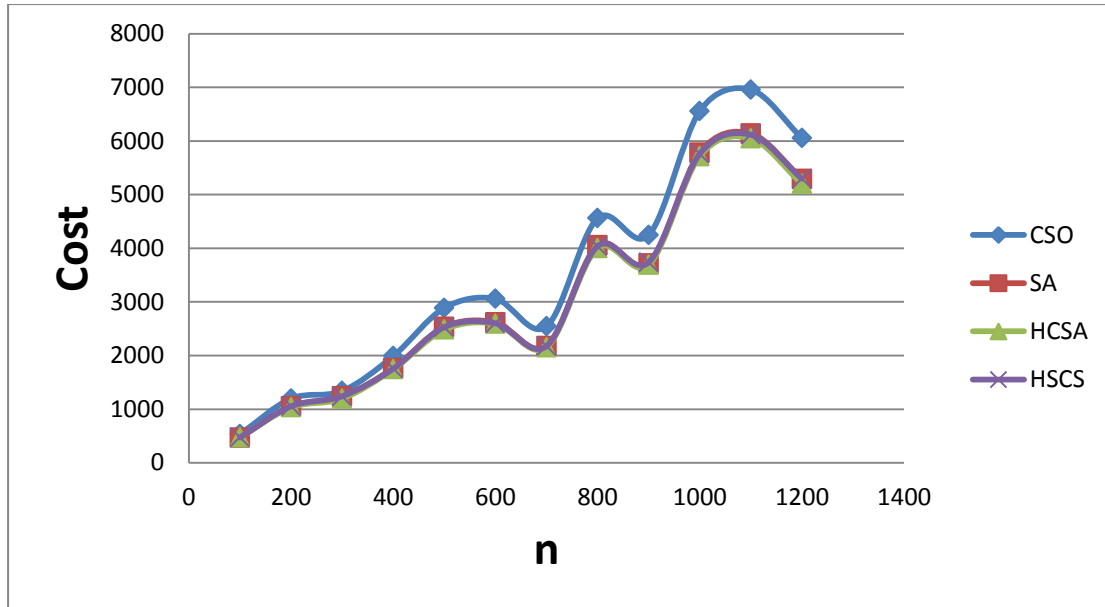
that the HCASA algorithm is better when ($n \leq 1000$) while the HCSA algorithm is better when ($n > 1000$).

In figure 4.1 we note that the results of solving the problem with a simulated hybridization algorithm in cats and ants are good compared to other hybridization algorithms when $n \leq 1000$.



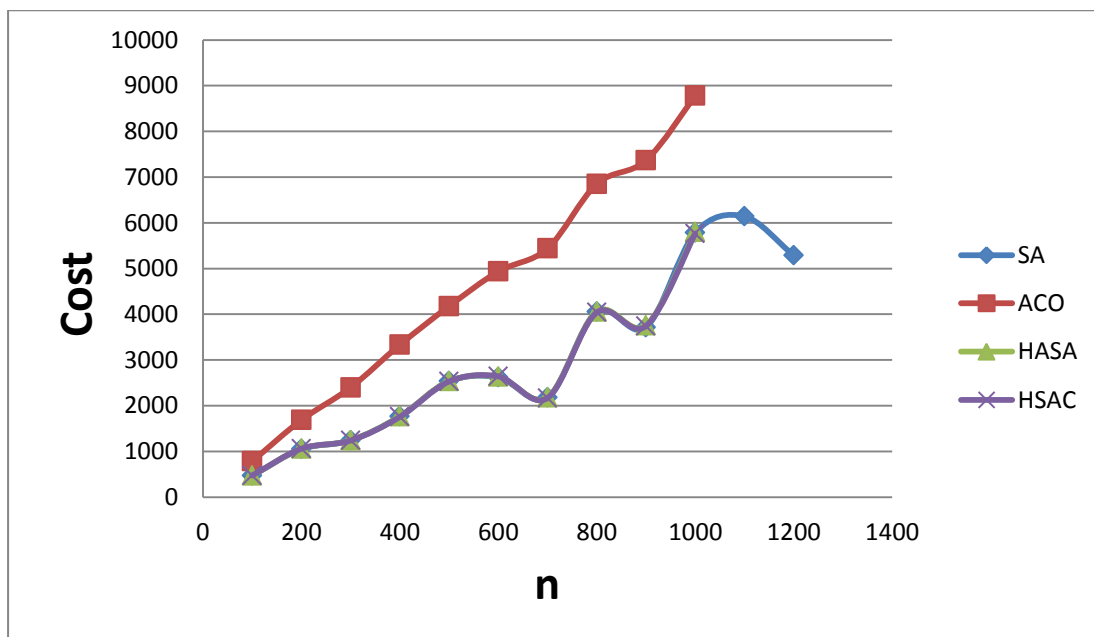
Figure(4.1): The results with n larger than 50 jobs for HCSA, HSCS, HASA, HSAC and HCASA.

In Figure 4.2 we note that the hybridization algorithms (HCSA, HSCS) have improved the results of the CSO algorithm, whereas the hybridization algorithm (HCASA) has improved the results of the SA algorithm when $n \geq 200$ and the hybridization algorithm (HSCS) has improved the SA algorithm results when $n \geq 400$.



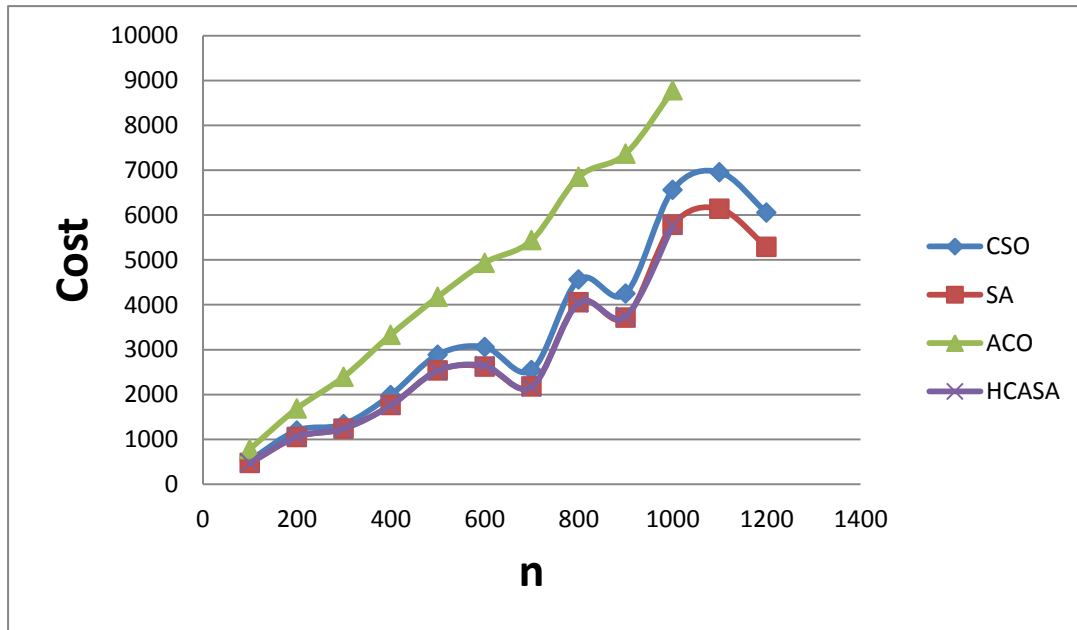
Figure(4.2): The results with n larger than 50 jobs for CSO, SA, HCSA and HSCS.

In Figure 4.3 we note that the hybridization algorithms (HASA, HSAC) have improved the results of the ACO algorithm, whereas the hybridization algorithm (HSAC) has improved the results of the SA algorithm when $n \geq 400$.



Figure(4.3): The results with n larger than 50 jobs for SA, ACO, HASA and HSAC.

In Figure 4.4 we note that the hybridization algorithm (HCASA) have improved the results of the (CSO, ACO) algorithms, whereas the hybridization algorithm (HCASA) has improved the results of the SA algorithm when $n \geq 200$.



Figure(4.4): The results with n larger than 50 jobs for CSO, SA, ACO and HCASA.

Chapter Five

Conclusion and Future Works

5.1 Conclusion

In this thesis, we have developed approximate solutions for the problem of scheduling n jobs on a single machine to minimize the sum of total late work and maximum lateness $1 / \sum_{j=1}^n V_j + L_{max}$. The proposed algorithms are effective in finding an optimal or near optimal schedule. We also concluded a lower bound for the problem P. The proposed CEM can be used to find the optimal solution for the problems solved with up to 10. Several results concerning optimality of five solvable special cases are presented. We prove three dominance rules which can help in reducing the number of branches in the search tree and show the computational results of the first special case. Also, we report on the results of extensive computations tests of the methods: ACO, CSO and SA. Our experimental results indicated that: some of the local search heuristic algorithms can solve problems of size (100000) jobs in reasonable time. Computational results shown in the tables for the problem $1 / \sum_{j=1}^n V_j + L_{max}$. Demonstrated that the CSO algorithm gives the best solution when $(n \leq 10)$ jobs. On the other hand, when the problems is of a larger size, i.e. $(n > 10)$ jobs, the simulation annealing gives the best solution. We have used hybridization methods to try to improve solutions and also concluded that the HCASA algorithm is better when $(n \leq 1000)$ while the HCSA algorithm is better when $(n > 1000)$. We note that the hybridization algorithms (HCSA, HSCS) have improved the results of the CSO algorithm, whereas the hybridization algorithm (HCSA) has improved the results of the SA algorithm when $n \geq 200$ and the hybridization algorithm (HSCS) has improved the SA algorithm results when $n \geq 400$. We note that the hybridization algorithms (HASA, HSAC) have improved the results of the ACO algorithm, whereas the hybridization algorithm (HSAC) has improved the results of

the SA algorithm when $n \geq 400$. We note that the hybridization algorithm (HCASA) have improved the results of the (CSO, ACO) algorithms, whereas the hybridization algorithm (HCASA) has improved the results of the SA algorithm when $n \geq 200$.

5.2 Future work

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

1. $1 / \sum_{j=1}^n V_j + \sum |L_j|$.
2. $F_2 / \sum_{j=1}^n V_j + L_{max}$
3. $1 / r_j / V_{max} + L_{max}$

References

- [1] Johnson, S.M. "Optimal two and three stage production schedules with setup times included", Naval Research Logistic Quart, 1, 61-68, (1954).
- [2] Jackson, J.R., "Scheduling a production line to minimize maximum tardiness", Research Report 43, Management Sciences Research Project, UCLA (1955).
- [3] Smith, W.E., "Various optimizers for single stage production", Naval Research Logistics Quarterly 3/1, 59-66 (1956).
- [4] Blazewics J. Ecker K.H., P Esch E. Schmidt G. and Weglarz J., "Scheduling Computer and Manufacture Processes" Springer Verlag Berlin. Heidelberg (1996).
- [5] Held, M., and Karp ,R.M., "A dynamic programming approach to sequencing problems", Journal of the Society for Industrial and Applied Mathematics 10, 196-210 (1962).
- [6] Potts C. N. and Van Wassenhove L. N., "Single Machine To Minimize Total Late Work "Operations Research 40-3(1991)586-595.
- [7] Hariri, A. M. A., Potts, C. N., & Van Wassenhove, L. N. Single machine scheduling to minimize total weighted late work. ORSA Journal on Computing, 7, 232–242. doi:10.1287/ijoc.7.2.232.
- [8] Kethley, R. B., & Alidaee, B. Single machine scheduling to minimize total weighted late work: A comparison of scheduling rules and search algorithms. Computers & Industrial Engineering, 43, 509–528. doi:10.1016/S0360-8352(02)00123-7(2002).
- [9] Błażewicz, J., Pesch, E., Sterna, M., & Werner, F. "The two machine flow shop problem with weighted late work criterion and

- common due date". *European Journal of Operational Research*, 165,408–415. doi:10.1016/j.ejor.2004.04.011(2005a).
- [10]** Lin, B. M. T., Lin, F. C., & Lee, R. C. T. "Two-machine flow-shop scheduling to minimize total late work". *Engineering Optimization*, 38, 501–509. doi:10.1080/03052150500420439(2006).
 - [11]** Manal G. Ahmed Al-Ayoubi "A Single Machine Scheduling Problem to Minimize the Sum of Total Completion Times and Total Late Works" *Al- Mustansiriyah J. Sci.* Vol. 23, No 7, 2012.
 - [12]** Asmaa Ali Zeyad " Scheduling Jobs on a Single Machine to Minimize Bi-criteria" M.Sc thesis, Dept. of mathematics, college of Education for Pure Sciences, Thi-Qar University (2016).
 - [13]** Horn, W.A., "Some simple scheduling algorithms", *Naval Research Logistics Quarterly*, 21(1):177-185 (1974).
 - [14]** Hariri, A.M.A., and Potts, C.N., "Single machine scheduling with batch set-up times to minimize maximum lateness", *Annals of Ops. Res. O*, 75-92 (1997).
 - [15]** Christos, K., and George J.K., "Single-machine scheduling problems with past-sequence-dependent delivery times", *Int. J. Production Economics* 126. 264–266 (2010).
 - [16]** Habibeh, N., and Lai, S.L., "Solving Single Machine Scheduling Problem with Maximum Lateness Using a Genetic Algorithm", *Journal of Mathematics Research* Vol. 2, No. 3; August (2010).
 - [17]** Radostaw, R., " Minimizing maximum lateness in a single machine scheduling problem with processing time based aging effects", *European J. Industrial Engineering* Vol. x, No. x,xxxx Accepted 12 September (2011).
 - [18]** Suh-Jenq, Y., Jia-Yuarn, G., Hsin-Tao, L., and Dar-Li, Y., "Single machine scheduling problem past sequence dependent delivery

times and deterioration and learning effects simultaneously", ICIC International c ISSN 1349-4198 (2013).

- [19]** Morteza, S., and Mehde , S., "Minimizing Maximum Lateness in a Single Machine Scheduling Problem with a Fixed Availability Constraint", Scientific Research Volume: 4, Issue: 1, Pages: 155-165 (January 2015).
- [20]** Baker, K.R "Introduction to Sequencing and Scheduling ", John Wily, New York, (1974).
- [21]** Fibich, P., Matyska, L., Rudov´a, H ., "Model of Grid Scheduling Problem", In Exploring Planning and Scheduling for Web Service,17-24,(2005),American Association for artificial Intelligence (www.aaai.org).
- [22]** Yeleswarapu, Radhika M., "Scheduling of 2-operation jobs on a single machine to minimize the number of tardy jobs", M.S.c. thesis University of South Florida, College of Engineering, Dep. Of Industrial and Management Systems Engineering (2003).
- [23]** Pinedo, M.L., "Scheduling theory, algorithms, and systems", Springer Science +Business Media, LLC., New York (2012).
- [24]** 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).
- [25]** AL-Assaf, S ., "Solving Multiple Objective Scheduling Problem" M.Sc. thesis, Dept. of mathematics, College of Sciences, AL Mustansiriayah University (2007).
- [26]** 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).

- [27]** Jouni, L., "Multi-objective Nonlinear Parto-Optimization", Lappeenranta University of Technology (2000).
- [28]** Hoogeveen, J.A., "Invited review Multicriteria Scheduling", European Journal of operational Research 167,592-623, (2005).
- [29]** 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).
- [30]** Lenstra, J.K., Rinnooy Kan A.H.G., and Brucker, P., " Complexity of machine scheduling problems", Ann. Discrete Math. 1, 343-362. (1977).
- [31]** David, K., Cliff, S., and Joel W., "Scheduling Algorithms", Massachusetts Institute of Technology, (1998).
- [32]** Delphi, A.M.," Algorithms to Solve Multicriteria Scheduling Problems on Single Machine", M.S.c. thesis University of Al mustansiriyah, College of sinence, Dep. of Mathematics (2011).
- [33]** 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).
- [34]** Graham, R.L., Lawler E.L., Lenstra J.K., and Rinnooy K.A",.Optimization and approximation in deterministic sequencing and scheduling: A survey", Annals of Discrete Mathematics 5, 287-326(1979).
- [35]** AL Zuwaini M.K," A comparative Study of Local Search Methods for Some Machine Scheduling Problems with the Usage of Hybridization As A Tool"., Ph. D. thesis, University of Baghdad, College of Education (Ibn Al-Haitham), Dep. of Mathematics (2006).

- [36]** Ramadhan, A.M., "Single Machine Scheduling using Branch and Bound Techniques", M.Sc. thesis, Dept. of mathematics, College of Science, AL-Mustansiriyah University (1998).
- [37]** Abdul-Razaq, T.S., "machine scheduling problems a branch and bound approach", Ph.D. thesis , University of Keele (1987).
- [38]** Blazewics J. Ecker K.H., P Esch E. Schmidt G. and Weglarz J., "Scheduling Computer and Manufacture Processes" Springer Verlag Berlin. Heidelberg (1996).
- [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]** Araibi, S.M., "Machine Scheduling Problem to Minimize Two and Three Objectives Function", M.Sc thesis, Dept. of mathematics, college of Education for Pur Sciences, Thi-Qar University (2012).
- [42]** Mahmood, A. A., "Multicriteria Scheduling: Mathematical Models, Exact and Approximation Algorithms", Ph.D. Thesis, University of Al-Mustansiriyah, College of Science, Dept. of Mathematics, (2014).
- [43]** Chachan, H.A," Exact and Approximation Algorithms for Scheduling with and without Setup times Problems"., Ph. D. thesis, AL Mustansiriayah University, College of Science, Dep. of Mathematics (2007).
- [44]** Reeves, C.R., "Modern Heuristic Techniques for Combinatorial Problem ", John Wiley and Sons Inc., New York (1993).

- [45]** S. French, Sequencing and scheduling: An introduction to the mathematics of the job-Shop, John Wiley & Sons, New York (1982).
- [46]** Abdul-Razaq, T.S., Potts C.N. and Van Wassenhove, "A survey of algorithms for the single machine total weighted tardiness scheduling problem", Discrete App. Math. 26235-253(1990).
- [47]** Aarts E.H.L and J.K. Lenstra. " Local search in combinatorial optimization" . John. Wiley & Sons, 1997.
- [48]** Crauwels, H. "A comparative study of local search methods for one machine sequence problem". Ph.D. thesis Katholieke University, Heverlee. Belgium (1998).
- [49]** Chu, S. C., Tsai, P. W., & Pan, J. S. Cat swarm optimization. In PRICAI 2006: Trends in Artificial Intelligence (pp. 854-858). Springer Berlin Heidelberg (2006).
- [50]** Orouskhani, M., Mansouri, M., & Teshnehlab, M. Average-inertia weighted cat swarm optimization. In Advances in Swarm Intelligence (pp. 321-328). Springer Berlin Heidelberg (2011).
- [51]** Mohammed ESSAID RIFFI and Abdelhamid BOUZIDI, "Discrete Cat Swarm Optimization for Solving the Quadratic Assignment Problem", International Journal of Soft Computing and Software Engineering [JSCSE], Vol. 4, No.6, pp. 85-92(2014).
- [52]** Jong-Ching Hwang, Jung-Chin Chen, J.S. Pan, Yi-Chao Huang, "CSO and PSO to Solve Optimal Contract Capacity for High Tension Customers", Electrical Engineering Department, National Kaohsiung University of Applied Sciences, Kaohsiung, Taiwan (2009).
- [53]** Abdelhamid Bouzidi, Mohammed Essaid Riffi "Cat Swarm Optimization to Solve Flow Shop Scheduling Problem" Journal of

Theoretical and Applied Information Technology 20th February 2015. Vol.72 No.2.

- [54]** Maysam Orouskhani, Mohammad Mansouri, Mohammad Teshnehlab, " Average-Inertia Weighted Cat Swarm Optimization", Springer-Verlag Berlin Heidelberg (2011).
- [55]** Noel D.Bacarisas, John Paul T. Yusiong, "The Effect of Varying the Fitness Function on the Efficiency of the Cat Swarm Optimization algorithm in Solving the Graph Coloring Problem", Annals, Computer Science Series (2011).
- [56]** Dorigo, M., Maniezzo V., and Colorni ,A., "A positive feedback as a search strategy", Milan, Italy. tech. Rep. 91-016. (1991).
- [57]** John E.B., and Patrick R. M., " Ant colony optimization techniques for the vehicle routing problem", Advanced Engineering Informatics 18 41–48, (2004).
- [58]** Ali Berrichi, Farouk Yalaoui " Efficient bi-objective ant colony approach to minimize total tardiness and system unavailability for a parallel machine scheduling problem " Int J Adv Manuf Technol. Accepted: 15 February 2013.
- [59]** Sami. M Araibi "Ant Colony Method to Minimize Single Machine Scheduling problem "Journal of Thi-Qar Science Vol. 6(3), Dec./2017.
- [60]** Hussam A.A. Mohammed, Fathalh A. Cheachan, Hanan A. Cheachan and Faria A. Cheachan "On Fuzzy Distances and Their Applications on Cost Function $\tilde{L}(\tilde{C}_j, \tilde{D}_j)$ " Transactions on Engineering and Sciences. Vol.3, Issue 2, February 2015.
- [61]** Christian Blum., " Ant colony optimization: Introduction and recent trends", Physics of Life Reviews 2353–373.(2005).

- [62]** Hussein J.M., " Flow shop scheduling problems Using Exact and Local search methods", M.Sc. thesis University of Al-mustansiriyah, College of science, Dep. Of Mathematics (2008).
- [63]** S.A. Seyed-Alagheband, H. Davoudpour, S.H. Hashemi Doulabi, M. Khatibi" Using a Modified Simulated Annealing Algorithm to Minimize Makespan in a Permutation Flow-shop Scheduling Problem with Job Deterioration" Proceedings of the World Congress on Engineering and Computer Science 2009 Vol II WCECS 2009, October 20-22, 2009, San Francisco, USA. ISBN:978-988-18210-2-7.
- [64]** Sergio Fichera, Fulvio Cappadonna, Antonio Costa, Alberto Fichera "A Simulated Annealing Algorithm for Single Machine Scheduling Problem with Release Dates, Learning and Deteriorating Effects" Proceedings of the World Congress on Engineering 2013 Vol I, WCE 2013, July 3 - 5, 2013, London, U.K.
- [65]** Nakandhrakumar R.S and Balachandar M"Implementation of Simulated Annealing Technique for Optimizing Job Shop Scheduling Problem" International Journal of Advanced Mechanical Engineering. ISSN 2250-3234 Volume 4, Number 2 (2014), pp. 169-174.
- [66]** Sergio, F. Fulvio, C. Antonio, C. Alberto, F. " A Simulated Annealing Algorithm for Single Machine Scheduling Problem with Release Dates, Learning and Deteriorating Effects" Proceedings of the World Congress on Engineering Vol I, WCE 2013, July 3 - 5, London, U.K.
- [67]** Thamilselvan Rakkiannan, Balasubramanie Palanisamy "Hybridization of Genetic Algorithm with Parallel Implementation

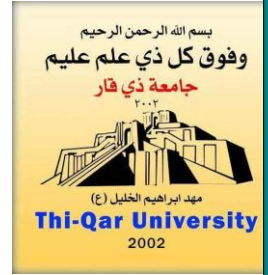
of Simulated Annealing for Job Shop Scheduling" American Journal of Applied Sciences 9 (10): 1694-1705, 2012.

- [68]** A. Jamili , M.A. Shafia, R. Tavakkoli-Moghaddam "A hybridization of simulated annealing and electromagnetism-like mechanism for a periodic job shop scheduling problem" journal homepage: www.elsevier.com/locate/eswa (2010).
- [69]** Kashif Akram , Khurram Kamal and Alam Zeb "Fast simulated annealing hybridized with quenching for solving job shop scheduling problem" Applied Soft Computing 49 (2016) 510–523, journal homepage: www.elsevier.com/locate/asoc.

المستخلص

تناولنا في هذه الرسالة دراسة مسألة جدولة n من المنتجات (jobs) على ماكينة واحدة لتصغير دالة متعددة الأهداف (MOF). والهدف من هذه الرسالة هو إيجاد الحل الأمثل Optimal solution والحلول التقريبية Near optimal solutions لجدولة n من المنتجات لدالة الهدف وهي الكلفة الكلية لمجموع كمية الاعمال المتأخرة مع اكبر زمن تأخير، هذه المسألة هي معقدة، لحل هذه المسألة اقترحنا قيد أدنى LB. وقد أثبتنا بعض الحالات الخاصة واقترحنا بعض قواعد الهيمنة وبرهنت. والنتائج الحسائية للحالة الخاصة الاولى الى $n=50000$ من النتائج. قمنا باستخدام طرائق البحث المحلية لإيجاد حلول تقريبية وهي Ant Colony Algorithm, Cats Swarm Algorithm and Simulated annealing عمليا ومن خلال التجارب الحسائية وجدنا أن خوارزميات البحث المحلي يمكن أن تحل المسألة إلى 100000 نتاج في فترة زمنية معقولة، وكذلك، وجدنا أن طريقة Cats Swarm هي الأفضل في هذه المسألة عندما يكون فيها $n \leq 10$ نتاج، إما في المسائل التي تكون فيها $n > 10$ نتاج فإن خوارزمية Simulated Annealing هي الأفضل. كما استخدمنا طرق التهجين HCSA, HASA, HSCS, HSAC and HCASA كطرق تقريبية في محاولة إيجاد حلول تقريبية افضل لحل المسألة في فترة زمنية معقولة وكذلك وجدنا ان طريقة HCASA algorithm هي الافضل عندما يكون عدد النتائج $n \leq 1000$ ، اما في المسائل التي تكون فيها $n > 1000$ نتاج فان خوارزمية HCSA algorithm هي الافضل.

جميع الأساليب المعتمدة في هذا البحث برمجت باعتماد لغة البرمجة (MATLAB).



جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة ذي قار

كلية علوم الحاسوب والرياضيات

(ترتيب النتائج لتصغير مسألة جدولة الماكنة)

رسالة مقدمة الى مجلس كلية علوم الحاسوب والرياضيات جامعة ذي قار

كجزء من متطلبات نيل درجة الماجستير

في علوم الرياضيات

من قبل

سارة كامل حنون

بكالوريوس رياضيات

جامعة ذي قار ٢٠١١

بإشراف

أ.م.د محمد كاظم زغير