Republic of Iraq
Ministry of Higher Education
and Scientific Research
University of Babylon
College of Education for Pure Sciences

Exact, Approximation and Meta-Heuristic Methods for Solving Combinatorial Optimization Problems

A Dissertation

Submitted to the Council of the College of Education for Pure Sciences in University of Babylon as a Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Education / Mathematics

By

Adel Hashem Nouri Jassim

Supervised by

Prof.Dr. Kareema Abed Al-Kadim

Asst. Prof.Dr. Hussam Abid Ali Mohammed

2023 A.D. 1445 A.H

بسم الله الرحمن الرحيم

يَرْفَعِ اللَّهُ الَّذِينَ آمَنُوا مِنكُمْ وَالَّذِينَ أُوتُوا الْعِلْمَ
دَرَجَاتٍ وَاللَّهُ بِمَا تَعْمَلُونَ خَبِيرٌ

سورة المجادلة – الاية ١١

Supervisor's Certification

I certify that the dissertation entitled **"Exact,Approximation and Meta-Heuristic Methods for Solving Combinatorial Optimization Problems "** by **" Adel Hashem Nouri Jassim"** has been prepared under my supervision in University of Babylon / College of Education for Pure Sciences as a partial requirement for the degree of Doctor of Philosophy in Education / Mathematics.

Signature :

Name : Dr. Kareema Abed Al-Kadim

Title: Professor

Date: / / 2023

Signature :

Name : Dr. Hussam Abid Ali Mohammed

Title: Assistant Professor

Date: / / 2023

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

Signature:

Name: Dr. Azal Jaafar Musa

Head of Mathematics Department

Title: Assistant Professor

Certification of linguistic Expert

I certify that I have read this dissertation entitled **"Exact, Approximation and Meta-Heuristic Methods for Solving Combinatorial Optimization Problems"** and corrected its grammatical mistakes; therefore, it has qualified for debate.

Signature:

Name: Dr. Hussain Hameed Mayuuf

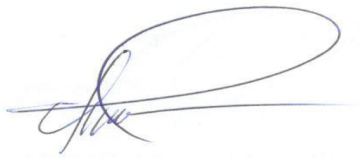
Title: Assistant Professor

Address: University of Babylon / College of Education for Human Science

Date: / / 2023

Certification of Scientific Expert

I certify that I have read the scientific content of this dissertation **"Exact, Approximation and Meta-Heuristic Methods for Solving Combinatorial Optimization Problems"** and I have approved this dissertation is qualified for debate.



Signature:

Name: Dr. Ahmed Abass AL-Adilee

Title: Professor

Address: University of Kufa / College of Education

Date: / /2023

Certification of Scientific Expert

I certify that I have read the scientific content of this dissertation **"Exact,Approximation and Meta-Heuristic Methods for Solving Combinatorial Optimization Problems"** and I have approved this dissertation is qualified for debate

Signature:

Name: Dr. Basim Karim Kadhir Albuohimad

Title: Assistant Professor

Address: University of Karbala / College of Education for Pure Science

Date: / /2023

Committee Certification

We, the examining committee, after reading this dissertation, **Exact,Approximation and Meta-Heuristic Methods for Solving Combinatorial Optimization Problems** "and examining the student **" Adel Hashem Nouri Jassim "**, in its content ,find that it is adequate as a dissertation for the Degree of Doctor of Philosophy in Education / Mathematics.

Signature:
Name: Dr. Mushtak A. K. Shiker Al- Jenabi
Title: Professor
Date: / /2023
(Chairman)

Signature:
Name: Dr. Zahir Abdul Haddi Hassan
Title: Professor
Date: / /2023
(Member)

Signature:
Name: Dr. Ahmed Sabah Al-Jilawi
Title: Assist.Professor
Date: / /2023
(Member)

Signature:
Name: Dr. Ruma Kareem K. Ajeena
Title: Assist.Professor
Date: / /2023
(Member)

Signature:
Name: Dr. Iraq Tareq Abbas
Title: Assist. Professor
Date: / /2023
(Member)

Signature:
Name: Dr. Kareema Abed Al-Kadim
Title: Professor
Date: / /2023
(Member/Supervisor)

Signature:
Name: Dr. Hussam Abid Ali Mohammed
Title: Assist. Professor
Date: / /2023
(Member/Supervisor)

Approved by the Dean of College Committee for Post Graduate Studies

Signature:
Name: Prof. Dr. Bahaa Hussein Salih Rabee
Date: / / 2023
(Dean of the College of Education for Pure Sciences)

Aknowledgements

First and foremost, I thank God who enabled me to handle all scientific and other difficulties associated with this work.

I would like to express my gratitude and appreciation to my supervisors, Prof. Dr. Kareema Abed Al-Kadim and Asst. Prof.Dr. Hussam Abid Ali Mohammed for their continuous guidance and valuable support during my research.

I would like to express my thanks to my family for their ongoing support during my study.

Finally, I would like to thank the Departments of Mathematics and all my colleagues for their valuable and discussions.

Dedication

To

My Parents

and

My friends

Abstract

Combinatorial Optimization Problems (COPs), which have a finite number of Feasible Solutions (FS), include scheduling machines and landing aircraft. Checking all of the FSs in the search area is the best technique to solve a COP. It is not always practicable to check every FS, especially if the search region is extensive. To address these issues, local search algorithms, or meta-heuristics, have been developed and improved. The Machine Scheduling Problem (MSP) and the Aircraft Landing Problem (ALP) are two optimization problems that are studied in this dissertation, along with the use of both exact and heuristic optimization strategies to solve them.

The Proportional Flow Shop (PFS) and Identical Parallel Machines (IPM) in Machine Scheduling Problem (MSP) and multi-runaway in Aircraft Landing Problem (ALP) are studied in this work.

All of these calculations were carried out using exact methods such as Complete Enumeration Method (CEM) and Branch and Bound (BAB). Meta-heuristic methods have also been used, such as Genetic Algorithm (GA) , Memetic Algorithm and other.

In Aircraft Landing Problem (ALP) ,we describe several multi-objective algorithms for the BOALP that calculate both the overall scheduling landing times (SLTs) for all aircraft and the total penalty cost of divergence between each aircraft's SLT and its Target Landing Time (TLT).

Contents

List of Tables

List of Figures

List of symbols

Symbol	Meaning
$\alpha\|\beta\|\gamma$	Classification of the Machine Scheduling Problem
α	Machine Environment
β	Job Characteristics
γ	The Objective Function
$\sum C_j$	The Sum of Completion Times
ALP	Aircraft Landing Problem
BAB	Branch and Bound
CEM	Complete Enumeration Method
C_j	The Completion Tme of Job j
COP	Combinatorial Optimization Problems
d_j	Due Date of Job j
DP	Dynamic Programming
EDD	Earliest Due Date
ELT_j	The Earliest Landing Time for Plane $j \in P$
ELT	Earliest Landing Time
EES_s	Number of Efficient Solutions
FCFS	First Come, First Served
FP	Flow Shop Problem

FSs	**Feasible Solutions**
GA	**Genetic Algorithm**
gj	**The Penalty Cost (≥0) Per Unit of Time for Landing Before Thetarget Time for Plane j∈P**
hj	**The Penalty Cost (≥0) Per Unit of Time for Landing After the Targettime for Plane j∈P**
IA	**Intensification Algorithm**
ISA	**Improvement Scheduling Aircraft**
JR	**Johnson's Rule**
GA	**Genetic Algorithm**
MPIA	**Modified Parallel Improving Algorithm**
MSP	**Machine Scheduling Problem**
n	**Number of Jobs**
N	**The Number of Planes**
$NODE$	**Number of Nodes**
NF	**Number of Feasible Solutions**
Nnd	**Number of Non-Dominated Jobs**
NP	**Non-Deterministic Polynomial Time**
O	**Set of Aircraft Which Have Previously Been Assigned a Landing Time**
OS	**Optimal Solution**

P	**Set of Planes, P={1,2,…,N}**
PIA	**Parallel Improving Algorithm**
pj	**Processing Time of Job j**
R_L	**Range on Lateness**
RTA	**Runway Throughput Algorithm**
rj	**A Release Date of Job j**
S	**The Set of Feasible Solutions of a COP**
SP	**Scheduling Problem**
Sij	**The Required Separation Time (≥0) between Plane i Landing and Plane j Landing (where Plane i Lands before Plane j), i,j∈P, i≠j**
Sli	**Slack Time for Job i**
SOF	**Single Objective Functions**
SPT	**Shortest Processing Time**
TLTj	**Target (preferred) landing time for plane j∈P**
Tmax	**The maximum tardiness**

Introduction:

Machine Scheduling Problem (MSP):

The Machine Scheduling Problem (MSP) entails scheduling a defined number of tasks on a set number of machines during a specified time period and toward a set objective. Each job may include many activities.

Nagar et al. (1995) [1] and Hoogeveen (2005) [2] provide the state-of-the-art in multi-criteria scheduling, which includes Multi-Objective Single Machine Scheduling (MOSMS). Only a few of researchers are interested in Minmax models (Sidney [3], Li and Cheng [4], Mosheiov and Oron [5]). Maximum early arrival (E_{max}) and maximum late arrival (T_{max}) may be minimized to achieve this goal. The ideal method for determining the idle insert value in a given sequence was proposed by Tavakkoli-Moghaddam et al. (2005) [6]. To reduce the sum of ET_{max} when an idle insert is present in single-machine sequencing, Tavakkoli-Moghaddam et al. (2006) [7] presented a method.

Branch and Bound (BAB) method was a solution proposed by Tapan et al. (1988) [8] to address the problem of $1||\sum Ci + R_L$, The method for reducing R_L on a single computer was provided by Liao and Huang (1991) [9]. For n =15, the problem was solved by considering a linear combination of total completion times (C_i) and the range of lateness (R_L).

For this reason, multi-objective optimization is a rare use to the Variable Neighborhood Search (VNS) meta-heuristic. In order to solve the Bi-Objective Flow Shop Scheduling (BOFSS) by minimizing a number of different criteria, the Multi-Objective Variable Neighborhood Search

(MOVNS) was proposed by Geiger (2004) [10]. Another multi-objective problem was resolved with the help of Geiger's MOVNS [11].

Researchers have provided a plethora of methods for coping with such predicaments (Baker (1974) [12]; French (1982) [13]). Flow shop scheduling (FSS) issues have been studied from the vantage point of a single goal. The two-machine flow shop scheduling issue with a C_{max} minimization objective was studied by Johnson (1954) [14]. He proposed a rational strategy for finding the Optimal Solution(OS) . Maximum early arrival (E_{max}) and maximum late arrival (T_{max}) were among the objective functions discovered in the literature review of the flow shop scheduling problem [15].

Due to the complexity of the objective functions, various optimum strategies were proposed. Bertrand (2001) [16] tackled the two-machine flow shop issue with the objective function T_{max} and presented several lemmas for it in simplified circumstances. Tapan and Dilpen (1991) [17] attempted to apply Johnson's rule (1954) to the objective function L_{max} in a two-machine flow shop.

Using the BAB method, they were able to solve as many as 10 problems.

In order to switch from the L_{max} objective function to the C_{max} objective function, two simple methods were described by Ladhari and Hauari (2000) [18]. The academic community is shifting its focus from single-objective solutions to scheduling and sequencing problems to multi-objective approaches. One worthwhile goal is to reduce the total percentage difference between the earliest and later arrival times at work.

When resources need to be distributed effectively to perform a set of tasks, machine scheduling is a significant challenge in manufacturing,

logistics, and other related industries. The scheduling process may be made more effective when numerous computers can work on the same job at once, as is the case with parallelism.

Scheduling issues for parallel machines may be tackled in a number of ways: using heuristic algorithms, mathematical programming, or metaheuristics. The genetic algorithm is often used because it employs the rules of natural selection to determine the best plan of action.

Azimi et al. (2020) introduced a new scheduling strategy for parallel computers that is inspired by the ant colony optimization technique. Results from applying the technique to many instances of the scheduling issue revealed substantial gains in makespan (the time it takes to finish all jobs). [19]

Aircraft Landing Problem (ALP):

We do studies on both SOALP (Single-Objective ALP) and BOALP (bi-objective ALP).In this preliminary SOALP analysis, we focus on two primary goals,

We do studies on both SOALP (single-objective ALP) and BOALP (Bi-Objective ALP).In this preliminary SOALP analysis, we focus on two primary goals:: $Z = \min \sum_i^p (g_i\alpha_i + h_i\beta_i)$ to reduce the costs of penalties for any aircraft when their Target Landing Time (TLT) is less than their Scheduling Landing Time (SLT). The objective of a second BOALP is to lower the total SLT of all aircraft and the associated penalty charges.

In SOALP, two significant functions are shown.

In order to maximize airport runway use (or throughput), $Z_1 = \min \sum_i^p SLT_i$ is one of the most important criteria, since it dictates decreasing the total of the SLT.

The leading and trailing aircraft types must be carefully chosen in order to maintain a safe gap between them. Idris et al. [20],[21] present an airport system where the runway is a key flow restriction. The first and most significant analytical approach developed by Blumstein to assess the carrying capacity of an arrival runway was published in 1959[22].

The maximum throughput, also known as the expected hourly rate of aircraft operations that may be sustained by a single or group of runways, can be used to determine the runway capacity. The dynamic programming method described by Balakrishnan and Chandran (2007) [23] optimizes runway throughput while lowering the Landing Time (LT) of the final aircraft or makespan using a fixed set of aircraft (the static case).

The objective function (Z)states that for static and dynamic ALP, Ernst et al. (1999) [24] provided heuristic and exact methods, respectively. Z Beasley et al. (2000) [25] offered a variant of the problem that had an integer range of zero to one. An aircraft will have a TLT (Target Lading Time) as long as it is within its Time Window (TW). A cost is charged when an aircraft lands before or after its TLT. The objective is to keep TLT deviations as affordable as feasible. Beasley and colleagues (2004) [26] looked at the dynamic form of the problem. Pinol and Beasley (2006) [27] proposed two heuristic methods: Scatter Search and the Bionomic Algorithm, for the available test circumstances, which included as many as 500 aircraft and five runways.

There are usually many runways available at large international airports. Since there may be more than one available runway, we must choose the best one and assign a landing time to each aircraft in such a scenario. The problem of the separation time between aircraft that land on separate runways must be taken into account. In 2020, Amir Salehipour [28]studied the problems of landing aircraft in large airports that contain several runways and proposed several effective algorithms to solve these problems, which were compared with the algorithms of Awasthi et al. (2013) and Sabar and Kendall (2015), which showed more efficiency .

1.1 Introduction :

The foundational ideas and methods for MSP and ALP solving are introduced in this chapter . First, some definitions for the general and the general optimization problems are provided . Then, by grouping the strategies into two classes exact and heuristic solution approaches we outline the various approaches to the problem.

1.2 Some Basic Concepts for the MSP and ALP:

We begin by making some critical remarks, focusing on performance requirements rather than detailing the machine contexts. These jobs (jobs), labeled $j_1, j_2, \ldots j_n$, are to be scheduled on an available set of computers (resources) that can only handle one work at any one moment, from time zero onwards.

We state the observation that is used for the machine scheduling, jobs j_i (i=1,…,n) has :

Processing time (P_{ij}) [29]: The p_{ij} indicates how long it took for task j to run on the machine i. If task j can be completed in the same amount of time regardless of the machine, or if job j is only going to be processed on that one machine, then the subscript i may be removed.

Due date (d_j) [29]: The committed shipment or completion date dj of work *j* (i.e. the date of the job's promised completion to the client) is denoted by the symbol dj. Jobs that are completed over their due date are permitted, although penalties are assessed. For jobs where the deadline is absolute, $d_j = d$, it is known as the common due date, while for those where the deadline is flexible, it is marked by d_j.

Now for a given sequence of jobs (1, 2, … , n) the following can be computed for job j.

- Time of flow $F_j = C_j - r_j$.

- The lateness $L_j = C_j - d_j$.

- The tardiness $T_j = \{L_j, 0\}$.

- The earliness $E_j = \{-L_j, 0\}$.

- The completion time C_j, $C_j = \sum_{i=1}^{j} P_i$

The following performance requirements for a specific sequence occur often in the literature [30].

$C_{max}(s) = max\ j \in \{C_j\}$ (makespan or maximum completion time).

$E_{max}(s) = max\ j \in \{E_j\}$ (maximum earliness).

$L_{max}(s) = max\ j \in \{L_j\}$ (maximum lateness).

$T_{max}(s) = max_j\ j \in \{T_j\}$ (maximum tardiness).

$R_L(s) = L_{max}(s) - L_{min}(s)$ (range of lateness).

$\sum j \in s\ w_j C_j(s)$ (total weighted completion times).

$\sum j \in s\ w_j E_j(s)$ (total weighted earliness).

$\sum j \in s\ w_j T_j(s)$ (total weighted tardiness).

$\sum j \in s\ w_j U_j(s)$ (weighted number of tardy jobs).

1.3 Classification Problem ($\alpha|\beta|\gamma$):

The categorization of MSPs in this dissertation uses the language used by Graham et al. [33]. If n jobs j_j (j = 1,..., n) need to be processed by m machines M_i (i = 1,..., m), then this is the case. An optimality criterion, task characteristics, and machine environment make up the three fields of a three-field classification ($\alpha|\beta|\gamma$) that may be used to define an MSP type.

- For the first field α is the structure of the:
  - Single machine or multiple machines.
  - Identical or different machines.

1.3.1 Types of the MSP :[38]

Single-machine, parallel-machine, flow-shop, job-shop, and open-shop problem modules are only few examples in industrial scheduling. Each of these concerns is addressed in a separate module, with its own set of objectives and constraints [29]. In the following paragraphs, we'll go into greater detail about each problem type.

1. Single Machine Scheduling (SMS):

A single machines models are significant for several reasons. The environment of a single machine is a relatively straightforward and unique example of every other environment. Models of one machine often have characteristics that don't parallel models of other machines or machines in the chain. The findings that may be obtained for single machine models serve as a foundation for a heuristic that can be used in situations with more complex machine interactions in addition to offering insights into a single machine. In real-world applications, scheduling problems are often divided into smaller, more manageable machine settings.[29]

2. Parallel Machine Scheduling:

There is just one machine in this environment, m. In the event that preemption is permitted, the task j with processing requirement pj may be begun on one of the machines, and if preempted, the work can be continued on another of the machines. There is a limit to the number of machines that can process a single task at once.[31]

Types of parallel machines:[32]

a. Identical parallel machines(Pm): There are m identical machines in parallel. Job j requires a single operation and may be processed on any one of the m machines or on any one that belongs to a given subset.

b. Uniform parallel machines (Qm): There are m machines in parallel with different speeds. The speed of machine i is denoted by vi. The time pij that job j spends on machine i is equal to pi=vi (assuming job j receives all its processing from machine i)

.**c. Unrelated parallel machines(Rm):** This environment is a generalization of the previous one. There are m different machines in parallel. Machine i can process job j at speed vij (assuming jobj receives all its processing from machine i).

3.Flow Shop Scheduling (F_m):

A flow shop schedule has m chains of machines. On each of the m machines, each work must be completed. Every work must be processed in the same order on the first machine, the second machine, the third machine, and so on through machine m. Next completion on one machine, a task enters the queue on the following computer. All lines are typically expected to follow the First In First Out principle, which states that no one may "pass" another while they are in line.[29]

4.Job Shop Scheduling (J_m):

Every work in a job shop with m machines has a preset route to follow. Differentiating between work shops where each job only visits each machine once and job shops where a job may visit each machine more than once.[29]

5.Open Shop Scheduling (O_m):

Every work must be processed again on each of the m machines in an open shop with m machines. Some of these processing durations, however, could be zero. Regarding how each work is routed across the machine environment, there are no limits. Every work may be assigned a path using the scheduler, and distinct jobs may have various routes.[29]

Table 1.1 Notations for Common Machine Environment (α)

Environment Name	Symbol / Notation	Description
Single Machine	1	One machine
Identical Machines in Parallel	P_m	P : Parallel machines "m" : number of machines
Parallel machines with different speeds	Q_m	Q : Parallel machines with different speeds "m" : number of machines
Flow Shop	F_m	F : Flow shop "m" : number of machines
Job Shop	J_m	J : Job shop "m" : number of machines
Open Shop	O_m	O: Open shop "m" : number of machines

1.3.2 The second field β, the indicators "pmtn, r_j, prec" denote specific job characteristics:[38]

Preemptions are only permitted if pmtn is present; otherwise, they are not permitted.

Each work could have a distinct release date if r_j is present. If not, all jobs start at time zero.

If prec is present, then there is a relationship of precedence between the occupations.

Table 1.2 Notations for Common Processing Characteristics / Constraints (β)

Term	Notation	Description
Release Date	r_j	This term if present indicates the dynamic shop; a job cannot start its processing on a machine prior to its r_j value.
Preemptions	*Prmp*	A job may interrupted during its processing due to arrival of a high priority job
Precedence constraints	*Prec*	When one job depends on the completion of another job, this implies precedence constraint
Breakdowns	*Brkdwn*	This implies that machines are not continuously available for processing
Recirculation	*Recrc*	When a job visits a machine more than once
Permutation	*Prmu*	The processing order of all jobs on one machine is maintained throughout the shop

1.3.3 The third field is objective function (γ):

There are two types of objective functions f_{max} and $\sum f_i$. The combination between the function is called multi-objective function , otherwise they are called a single objective function.

Table 1.3 Notations for Common scheduling objective functions (γ).

Makespan	C_{max}
Maximum Lateness	L_{max}
Total weighted completion time	$\Sigma\omega_j C_j$
Total weighted Tardiness	$\Sigma\omega_j T_j$
Total completion times	ΣC_j

1.4 Multi-Objective Scheduling Problem (MOSP):

There was a single objective issue being studied by most researchers until the 1980s. However, scheduling decisions should take into account several factors in order to present the decision maker with more realistic answers. To model and solve Scheduling Problems (SPs) more than one objective (criterion) must be met.

In a lot of cases, it is improbable that different objectives would be best served by the same set of decision factors. In other words, a loss from a criteria lowering frequently cancels out a gain from a criterion increase. The several aims must thus be traded off. According to T'kindt and Billaut [33] and Hoogeveen[34], this sort of issue is referred to as a MOSP.

Jobs must typically be completed with the intention of maximizing, reducing, or achieving a mix of many objectives. One objective would be to increase customer pleasure as much as possible, while another might be to reduce delays.

Definition (1.1) (optimize) [35]: 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 (1.2) (efficient) [35]: It is claimed that a schedule s is effective if there is no alternative schedule s′ that meets $f_j(s') \leq f_j(s)$, $j = 1,\ldots,k$ with a strict inequality holding if and only if one of the above requirements is met. If not, then s′ is said to be more important than s.

1.5 Sequencing and Scheduling Problems:

In 2012, Pinedo [29] defined scheduling as a process of selecting choices in order to maximize some objective or goals. It is concerned with the scheduling of jobs over a range of time intervals. Sequencing focuses on the correct sequence of things, items, events, etc., whereas scheduling explains when a work could begin in more detail.

Runways at airports, factory equipment, medical professionals, computer memory, and central processing units are all examples of resources.

Common examples of such actions are software installations, airport landings and takeoffs, medical care for patients, and manufacturing procedures. Each activity may have a different level of urgency, preparedness, and deadline. Many different approaches may be used to solve the problem.

The following some theorems are used in this dissertation:

Theorem (1.1) [36]: The $1||\sum_j C_j$ issue is reduced to a minimum by arranging the jobs in a queue in which pj does not decrease according to the Shortest Processing Time (SPT-rule).

Note: The LPT-rule, or longest processing time rule, specifies a non-increasing pj order.

Theorem (1.2) [37]: The $1||T_{max}$ issue is reduced to a minimum by arranging jobs in accordance with the Earliest Due Date (EDD-rule), i.e., by non-decreasing dj.

1.5.1.Johnson's Algorithm:

Step 1: Let X = $\{j|p_{j1} \leq p_{j2}\}$ and $Y = \{j|p_{j1} > p_{j2}\}$.

Step 2: Sort the items of set X in non-decreasing order of p_{j1} and items of set Y in non-increasing order of p_{j2}.

Step 3: The optimal sequence is the ordered set X followed by the ordered set Y.

1.5.2.Palmer's Heuristic :[38]

The two stages of this heuristic are as follows.

Step 1: Determine the incline for the n-job, m-machine static flow shop issue A_j for j^{th}job as follows;

$$A_j = -\sum_{i=1}^{m} \{m - (2i - 1)\} p_{ij}$$

Step 2: Sequence the jobs with the Aj values decreasing from highest to lowest.

1.5.3.Campbell, Dudek, and Smith (CDS) Algorithm :[38]

The method divides an issue with m machines and n jobs into p surrogate problems with n jobs and 2 machines (where p = m-1). The Johnson rule is used to find the best solution to each surrogate issue. Cmax is determined for each surrogate issue by using the Johnson rule. Job scheduling on the

machines is done by choosing the sequence of the surrogate issue that results in the smallest value of Cmax when Johnson's rule is applied.

The CDS algorithm begins by transforming the original issue into a set of surrogate problems. Below is an example of a three-machine, four-job scenario. Two substitutes will be used in the 3-machine. $F_2 \mid\mid C_{max}$ problems.

1.6 Aircraft Landing Problems (ALPs)[39]

The timing of airplane landings has piqued the curiosity of many avionics experts. Finding the ideal arrival or departure times for aircraft subject to operational constraints is the crux of the fundamental aviation scheduling issue. A comprehensive analysis of the available literature on ALP is offered. Using a resolution strategy that relies on relaxing binary variables and imposing extra constraints, the authors have presented the issue mathematically as a mixed linear program.

ALP is difficult to solve since it may be thought of as a job MSP with release times and sequence dependent processing time, claims Beasley. The ALP is NP-hard since the job MSP has been shown to be NP-hard.[25]

1.6.1 Basic ALP Concepts:

We begin by outlining some of ALP's most crucial points. After determining the most viable sequence for the P aircrafts, we next apply SLT to each of those aircraft based on a set of restrictions we have imposed.

1.6.2. Decision Problems[40]

A decision problem is a problem whose output is a single boolean value: Yes or No. Let me define three classes of decision problems:

• **P** is the set of decision problems that can be solved in polynomial time. Intuitively, P is the set of problems that can be solved quickly.

• **NP** is the set of decision problems with the following property: If the answer is Yes, then there is a proof of this fact that can be checked in polynomial

time. Intuitively, NP is the set of decision problems where we can verify a Yes answer quickly if we have the solution in front of us.

• **Co-NP** is essentially the opposite of NP. If the answer to a problem in co-NP is No, then there is a proof of this fact that can be checked in polynomial time.

The MSP and ALP are NP-hard problems .

ALP usually includes the runway assignment, scheduling, and sequencing decision-making challenges. In order to ensure the necessary level of safety, scheduling assigns SLT to each aircraft in the sequence once the sequencing algorithm selects one from the set of viable sequences. Every aircraft must be allocated to a specific runway when more than one is available for landing. When demand for landing is at its highest, it may even surpass the runway's capacity, exacerbating problems with resolution.

1.6.3. Dynamic and Static Models:[40]

Although some studies are considered in a dynamic state, the ALP model explored in the majority of the literature deals with a static state (off-line) (on-line or in real-time). In the static version, the model is solved using a predetermined set of aircraft, with the assumption that all relevant data is accessible. As more planes arrive and more data becomes available, solutions may be considered in the future. However, there are various degrees of uncertainty related to information about the aircraft, operating environment (runway state, weather, etc.), and taxiways. This ambiguity decreases with time. ALPs should thus be addressed in a dynamic context in real-world scenarios.

1.6.4. Landing Time (*LT*):[41]

Clearly, there is an ELT if the airplane continues to go toward the runway at its maximum speed, i.e. (ELT_j). The airplane j will eventually run out of fuel and will be unable to continue spinning around the airport, hence there is also a latest landing time (LLT_j). These hours specify the Time Window (TW) [ELT_j, LLT_j] during which each aircraft must land. For every aircraft, there is a target landing time TLT_j. The time it arrives if it maintains the aircraft's cruising speed until landing, conserving fuel, or the time it should have arrived based on the flight schedule, if economic concerns are a major problem.

1.6.5. Separation :[40]

The safety of flights is the air traffic controllers' primary duty. One of the primary Air Traffic Control (ATC) safety problems is the standard horizontal and vertical separations that prevent airplanes from becoming too close together. The horizontal separations between aircraft vary based on factors including the location of the aircraft, its type, its velocity, the accessible ground facilities, and other circumstances. The common practice is to establish a 1,000-foot minimum vertical separation between civilian aircraft flying in restricted airspace.

1.7 ALP Objectives :[39]

The ATC must establish the landing authorization, LT, and suitable runway if more than one runway is available when an aircraft enters the airport's radar range (or radar horizon). Each early arrival or late arrival relative to the specified time has a cost in terms of the penalty. Regarding the following limitations, the objective is to reduce the overall penalty cost:

1- There is a safety interval between two consecutive runway landings;

2- Every plane must arrive inside a certain TW [Earliest LT, Latest LT].

The following objective functions are as a result:

$$Z = min \sum_{i=1}^{p} (g_i \alpha_i + h_i \beta_i) \qquad (1.1)$$

1.7.1.Constraints:

The airplane must comply with several complicated and composite limitations connected to ALP in order to make a safe landing. There are constraints governing the overlap between the TWs of the aircraft, including constraints imposed on the same aircraft, such as the requirement that he land the aircraft during the TW and aircraft-specific constraints, such as the requirement that there be a separation time between the aircraft if it is necessary for their safety. In Chapter 5, we'll examine some more restrictions.

1.7.2.Other Constraints:

The ability to schedule the aircraft is greatly influenced by the weather as well. Landings may be postponed due to wind, particularly for smaller aircraft. Fog may reduce vision to the point that it is deemed unsafe to allow flights to land at the regular intervals. The aircraft may have problems due to lightning, torrential rain, snow, or ice. Last but not least, turbulence caused by an airplane landing on one runway affects planes landing on adjacent runways.

2.1 Introduction:

Exact methods for MSP and ALP involve algorithms that guarantee to find the optimal solution in a finite amount of time. In contrast, heuristic methods are algorithms that do not provide such guarantees but often provide near-optimal solutions in a shorter time than exact methods.

2.2 Methods of Solution:

This section discusses some of the most common approaches used to solve MSPs and ALPs. The MSP and ALP are solved using a combination of approximation techniques and enumerative methods (such as BAB and DP).

For mathematical optimization problems, the Complete Enumeration Method (CEM) considers and ranks every plausible solution. As a brute force approach, CEM creates and ranks all solutions equally, regardless of their efficacy.

The objective function to minimize or maximize and the constraints that set the bounds on the feasible area are both required for successful application of CEM to an optimization issue. Following this, we compute the objective function for each candidate solution and then choose the one that best meets our optimization criteria.

While CEM may be used to problems of any scale, solving really big problems using the method is often computationally prohibitive. If there are n choice variables and m potential values for each, then there are $(n!)^m$ distinct outcomes to consider.This figure increases exponentially with the number of decision factors, rendering CEM unworkable for problems with many independent variables.

However, CEM is helpful for situations with a limited viable area and a controllable amount of solutions. Furthermore, various optimization techniques' effectiveness is often compared to CEM.

2.2.2. Branch and Bound (BAB) Algorithm:[42]

Many different kinds of MSPs and ALPs can be solved using the BAB method. These methods are representative of an implicit enumeration strategy that searches for an OS by investigating subsets of solutions that are technically possible. The most often used strategy in scheduling, which may be used to detect an OS by the systematic analysis of its structure. Most often, a search tree is used to represent the connections between these categories. As more and more branches branch out from each node, it's clear that we're nowhere near a full answer. These variants restate the primary problem as a set of identical and self-contained subproblems. Those forks give rise to new nodes, and so on, until nodes representing the ultimate solutions have been generated.

The next steps in determining this BAB technique are:

1 - The bounding procedure:

Lower Bound (LB) values should be determined for each created subproblem. There is no need to continue examine the associated subproblem when a full solution with objective function value Z is already known (a trial solution) and LB Z. However, the enumeration procedure is limited if it does not diverge from such a branch. More often, one must decide between limits that are not too tight but take a long time to compute and bounds that are rather narrow but can be determined rapidly. Using dominance qualities is a further method to prune the tree's limbs. A subset of any solution is determined by these characteristics, and it is guaranteed to include an operating system. In this way, we may eliminate any branches or subproblems in the tree that don't meet these criteria.

2 - The branching procedure:

The method that sorts problems according to their solutions. In backward branching, jobs are sequenced in order from the finish, while in forward branching, jobs are sequenced in order from the start..

3 - The search strategy:

With best first search (also known as jump tarcking), the search tree always branches out from the node that has the lowest lower bound at the time. When looking for an experimental solution, depth-first search (also known as backtracking) skips through everything except the lowest levels. When it reaches a level where further advancement is impossible, it goes back to the first level where it is feasible to do so and takes the best branch available there.

2.3.Approximation Solution Methods:

Developing computationally efficient algorithms that find solutions that are, in most situations, optimal or near-optimal is one technique to take on the apparently insurmountable task of many MSPs and ALPs. This may be possible using a heuristic strategy.

2.3.1. Variable Neighborhood Descent (VND) Method:

VND was proposed by Hansen and Mladenovic (2003) [43], who also recommended doing a local search for the local optimum in each neighborhood as changes are being made. Only the most optimal solutions are approved by all research neighborhood. In particular, we kept tabs on the VND initiative.

Initial Solution:

Since the VND rapidly converges to the local optimum of a low-quality region, starting with a randomized solution was a bad idea. Keep in mind that the VND is only a very basic version of local search. It is also typical practice in the scheduling literature to utilize heuristic algorithms as a first solution followed by a meta-heuristic approach.

Description VND**:**

To optimize the answer found in the previous phase, the VND algorithm always begins with that solution. The algorithm will keep looking so long as the NH is getting better. When no better answer can be found in the current Neighborhood, we revert to the prior Neighborhood. As a result, the VND algorithm will exhaustively look at all possible options before settling on the ideal one for the Neighborhood.

Intensification Algorithm (IA)**:[44]**

We introduce a novel class of local algorithms that relies on the initial solutions of the PRs implemented in the ISA, and we do so by splitting the sequence s into two partial subsequences, S_T and S_{NT}, where S_T contains the maximum number of aircrafts that landed at the TLT, and S_{NT} contains all of the aircrafts that landed on non-target LTs.

2.3.2.Tabu Search Algorithm:[45]

In the 1980, Fred W. Glover devised a metaheuristic optimization technique called Tabu Search. It is a strategy for global optimization that involves iteratively altering the present solution via a series of steps and then assessing the results.

The Tabu Search algorithm avoids becoming stuck in local optimum solutions by following a strict set of constraints throughout the search process. These guidelines require remembering a list of "tabu" motions, which are actions that are now prohibited to prevent re-visiting previously studied solutions. To prevent becoming caught in a rut when searching for a solution, the algorithm also uses diversification tactics to broaden its scope.

Scheduling, network design, transportation, and manufacturing are just few of the areas where Tabu Search has been effectively used as a means of improving upon initial solutions. It has been shown as a reliable method for rapidly finding optimal answers.

Beginning in 1985, Glover published many articles describing various applications of the tabu search. The development of this technique has led to its quick adoption in a variety of different fields, including sequencing, scheduling, oil drilling, and routing.

The tabu search may help prevent other methods from becoming stuck in local minima because to its unique properties. Using its memory, tabu search avoids rapidly revisiting previously explored regions of the solution space. Examples of possible solutions are kept in a database for this purpose. The word "taboo" is used to describe the societal disapproval that surrounds these sorts of interventions. The size of the tabu list is taken into account throughout the search process.

Mechanisms for controlling the search are included into the taboo search as well. At least one solution will be unpleasant thanks to the tabu list, but the algorithm may become stuck on a local maximum if the tabu list's limits are too stringent in certain cases. The tabu search suggests using idealistic criteria to go around this problem. The taboo limits are replaced by the aspiration criteria, enabling a broader search for the global optimum.

2.3.3.Genetic Algorithm (GA):[46]

Natural selection and genetics provide as inspiration for the famous search heuristic known as the Genetic Algorithm (GA). It's an optimization technique that's found widespread usage in areas like science, engineering, and business when more conventional approaches have proven ineffective.

In order to solve a problem, GA uses a predetermined set of rules and operations to "evolve" a population of candidates. The parameters of each solution in the population are encoded in a string of genes that make up the chromosome's representation.

The chromosomal population is subjected to a series of genetic operators, including mutation, crossover, and selection, throughout the GA's operational phase. Each solution is then scored for how well it fits the issue specifications using a fitness function that has already been established.

In evolutionary processes, if a solution is not discovered or a termination requirement is not fulfilled, the process will keep going until it does. Many other industries have found success with GA, including machine learning, data mining, robotics, image processing, and many more.

The GA, or Genetic Algorithm, is a well-known optimization method that takes its cues from the principles of evolution and natural selection. GA is a population-based algorithm that models the process of natural selection, in which those with higher fitness values are more likely to survive and propagate their qualities to future generations. It's no secret that GA has been utilized to solve many optimization challenges, including the pairing of parallel workstations. The goal of job assignment when matching parallel machines is to reduce the overall time, early time, and delay time. This is a difficult combinatorial optimization issue that may be difficult to tackle using common optimization tools. However, GA may be a useful tool for resolving this issue by establishing a pool of candidate solutions and refining them via successive generations.[47]

Randomly produced populations of solutions (chromosomes) provide the basis for GA. Each chromosome represents a potential answer, which is a specific combination of machine jobs. The objective function, which is the sum of the early

time and the delay time, is then used to determine the fitness of each chromosome. The chromosomes with the highest fitness values will be picked for the following generation, and their characteristics will be passed on to the children through crossover and mutation operators. As time passes, the population improves its ability to provide solutions that will lead to a reduction in the goal function. This repetitive process is repeated until a stopping requirement is satisfied, such as after a certain number of generations have passed or when a certain fitness value has been reached in the GA algorithm.

To reduce total time, early time, and delay time, GA may be used to optimize task assignment and scheduling for pairs of parallel processors. GA may aid in shortening the time needed to match parallel machines and enhancing the system's efficiency by creating a wide range of alternative solutions and iteratively refining them over successive generations.

Overall, GA is a potent optimization method that may be used for the matching of parallel machines and other difficult optimization problems. An organization's productivity may be increased and the total time, early time, and delay time of its machines can be decreased by utilizing GA to optimize job assignment and scheduling.

2.3.4.Memetic Algorithm (MA):[48]

The following pseudo code demonstrates the overall process that led to the creation of the memetic algorithm (MA). Creating the first population is the first step in the memetic process. Parameters like as population size, the frequency of selection, and the rate of genetic mutation, among others, are set in order to achieve this goal. A collection of solutions, encoded as chromosomes, serves as the basis for the first population. After this, a searching technique is used to these solutions to provide neighbor solutions. Decoding and evaluating each member of the original population and the selected neighbors is the next step in calculating the makespan.

The software exits and displays the best-found solution if the algorithm's termination requirement has been met; otherwise, a new population is formed by applying the genetic operators (selection, crossing, and mutation) to the existing population. Each fresh batch of people is called a generation, and the process described above is repeated until the limiting factor is met.

2.4 Special Heuristic Methods for ALP : [49]

2.4.1 Priority Rules (PRs) for Sequencing ALP:

The following heuristics will be utilized to determine an FS (which will serve as starting solutions for the subsequent heuristics) for sorting the planes by PRs according to variables:

1. : ELT_i: If we want to get the most out of the runway, we should give preference to the plane that will arrive with its earlier ELT.

2. : TLT_i: If we want to limit the amount of time planes are early or late, we may set this priority such that the planes with the earliest TLT take off first.

3. : LLT_i: To prevent the situation where the plane with the earliest LLT takes precedence, we place LT an after each plane's LLT.

4. ELT_i:/g_i : The airplane with the earliest ELT and the largest early arrival penalty is given preference.

5. LLT_i:/h_i : The plane with the earliest LLT and the largest lateness penalty is given preference.

6. $TLT_i/(g_i + h_i)$: Priority is given to the plane with the earliest TLT and the largest early and late penalty.

7. $1/(g_i + h_i)$: In the event of a tie, the plane with the larger penalty for being early or late would take off first.

2.4.2.Parallel Improving Algorithm (*PIA*):[49]

To reduce the overall penalty cost (Z), Bencheikh et al.[49] used a novel algorithm to enhance the automated aircraft LT. The most crucial part of the algorithm is adjustment LT (aircraft scheduling), where the goal is to reduce the overall penalty cost produced by each aircraft. The LT of the chosen aircraft j is adjusted in accordance with the landing slot periods available. If aircraft j lands early (late), the LT for aircraft j is increased by one unit of time; if this does not ensure that the safety interval between aircraft j and the one landing after it is maintained, the LTs of aircraft j and the one landing before it are adjusted accordingly. If the new LT is found to fall beyond the landing window, the increase (or decrease) is canceled and the previous flight speed is maintained.

This algorithm (PIA) modifies the LT of each aircraft at the time of its assignment based on whether or not the LT is increased (decreased) during the startup phase.

Let s be the sequence of aircraft setup according to a priority rule and $O = \emptyset$.

Algorithm Parallel Improving Algorithm (PIA) [49]

1. $O \rightarrow {}^{*}s_1{}^{+}$;
2. $SLT_1 \rightarrow TLT_1$;
3. for j $\rightarrow 2$ to P
4. $SLT_j \rightarrow max\,((ELT_j\ or\ TLT_j), max_{i\in 0}(SLT_i + S_{ij}))$;
5. end
6. while $there$ is $decrease\ of\ penalty\ cost$
7. if $SLT_j > TLT_j$
8. $Reduce\ the\ SLT_j\ for$ j by 1 $unit\ of$ time;
9. $else$
10. $Increase\ the\ SLT_j\ for$ j by 1 $unit\ of$ time;
11. end
12. if $the\ solution$ is $non-feasible$
13. $Reject\ the\ change\ and\ keep\ the\ last\ feasible\ solution$;
14. $break$;
15. end
16. end

2.4.3.Modified Parallel Improving Algorithm $(MPIA)$:[50]

The PIA, which is given above, implements the sequencing using one of the PRs and scheduling represented by increases (decreases) the SLT_i value by one unit if it less (larger) than TLT_i. Abdul-Razaq and Ali modified the PIA to improve the efficiency of PIA and called the new $PIA$by Modified PIA ($MPIA$).

The 2 modification is represented by testing the feasibility of the intervalsolution to improve aircraft landing time during the initialization which must respect the safety period between the next (previous) ones before increasing (reducing) the aircraft LT by one unit and calculating the penalty cost (say Z^*) (Z is the penalty cost) then increasing (reducing) the aircraft LT by one unit as mentioned in ALP, then calculating the penalty cost (Z), if $Z > Z^*$, then ignoring Z and keeping Z^* otherwise we take Z.

Let s be the sequence of aircraft set up according to a PR and $O = \emptyset$

Algorithm Modified Parallel Improving Algorithm ($MPIA$) [50]

1. $O \rightarrow {}^{*}s_1{}^{+}$;
2. $SLT_1 \rightarrow TLT_1$;
3. for j $\rightarrow 2$ to P
4. $SLT_j \rightarrow (TLT_j, max_{i \in 0}(SLT_i + S_{ij}))$;
5. end
6. while $there$ is $decrease$ of $penalty$ $cost$
7. $Test$ the $feasibility$ of the $solution$ interval of improving LT;
8. $Calculate$ $penalty$ $Cost$ Z;
9. if $SLT_j > TLT_j$
10. $Reduce$ the SLT_j for j by 1 $unit$ of $time$;
11. $else$
12. $Increase$ the SLT_j for j by 1 $unit$ of $time$;
13. end
14. if the $solution$ is $non-feasible$
15. $Reject$ the $change$ and $keep$ the $last$ $feasible$ $solution$;
16. $break$;
17. end
18. end

2.4.2. Improvement Scheduling Aircraft (ISA) Algorithm:[44]

The ISA algorithm improves the actual computed aircraft scheduling LT to minimize the total penalty cost Z.

This improvement is based on the reduction of the total cost of the penalty incurred by all aircraft. To get this object, we have to solve two problems: the first is a sequencing problem that determines the sequence of aircraft landing, and the second is a scheduling problem that determines the SLT for all aircraft in the sequence, subject to all constraints. The first problem (sequencing problem) can be solved by implementing any heuristic for ALP. Then, for the second problem (scheduling problem) to schedule the aircraft in order to minimize the total cost of the penalty caused by each aircraft and to improve the SLT, we recalculate the LTs foreach aircraft. Since the LTs assigned to aircraft, then it is possible to minimize the deviation from the target time for all aircraft.

Now the ISA algorithm can be described. Start with any sequence as a heuristic for ALP. We change the LT of a selected aircraft i, as follws: if the aircraft i land in advance, then we increase its LT toward the TLT, in this case, we may have got a zero contribution to the objective function for the aircraft i, if it is not, i.e. if the aircraft i lands in tardy, then we decrease LTs by one unit of time for aircraft i and the adjacent aircrafts for it (that the time between them is only the separation time), when the difference between the $SLTs$ for adjacent aircrafts equal to the separation times, in this case we have to verify the viability of the solution: if the new LTs respects the period of security between the previous periods and it is FS, we continue to decrease its LT by one unit of time, otherwise reject the change and keep the last FS. This means that we organizing the LT for the aircraft that increases the penalty in order to reduce the total cost of penalties for all aircraft.

Algorithm Improvement Scheduling Aircraft (ISA) Algorithm for Z [44]

1. $O \rightarrow \{s_1\}$
2. $SLT_1 \rightarrow TLT_1$;
3. for j $\rightarrow 2$ to P
4. $SLT_j \rightarrow (TLT_j, max_{i \in 0}(SLT_i + S_{ij}))$; % i < j
5. $O \rightarrow O \cup \{s_j\}$;
6. if $SLT_j = TLT_j$
7. $Return$ $step3$;
8. $else$ % $SLT_j > TLT_j$
9. $Reduce$ the SLT_j for j and $adjecent$ $aircrafts$ by 1 $unit$ of $time$;
10. if (The $solution$ is $feasible$ i. $e.$ $there$ is no $increasing$ of $penalty$ $cost$)
11. $Repeat$ $step$ 9 $until$ $SLT_j = TLT_j$;
12. $Repeat$ $step$ 9 $until$ $SLT_j = TLT_j$;
13. $Reject$ the $change$ and $keep$ the $last$ $feasible$ $solution$;
14. $Return$ $step3$;
15. end
16. end
17. end

3.1.Introduction:

It is a flow shop problem if every job runs through the identical sequence of operations on every machine. In a flow shop, the work must move sequentially between the machines because to technological constraints. Therefore, the technical constraints of each work have an effect on the order (sequence) in which the machines process it in the flow shop. It's common for real-world scheduling issues to revolve on just two key choices:

How to determine the optimal sequence in which to split tasks across several computers to be handled in parallel; and loading schedules for machines describe when different jobs will start and finish on each unit.

3.2 Problem Formulation:

An rise in computing complexity may be caused by an expansion.

For a sequence of jobs $s \in S$ where S is the set of all feasible solutions, the criteria (f_1) and (f_2) are computed in the following by :

Let $C_{max} = C_{n3}$ and $R_L = L_{max}(s) - L_{min}(s)$, $L_{max} = \max_{1 \le i \le n}\{L_i\}$ and $L_{min} = \min_{1 \le i \le n}\{L_i\}$ where $L_i = C_{i3} - d_i,\ i = 1, \dots, n.$

Our problem FP can formally by stated for any schedule $s \in S$ as:

$$
Min\begin{cases} f_1(s) = C_{max} & \dots (f1) \\ f_2(s) = R_L & \dots (f2) \\ s.t. & \end{cases}
\left.\begin{array}{ll}
C_{i1} = \sum_{k=1}^{i} p_{k1} & i = 1, \dots, n, \\
C_{12} = p_{11} + p_{12}, & \\
C_{i2} = max\{C_{i1}, C_{i-1,2}\} + p_{i2}, & i = 2, \dots, n, \\
C_{13} = p_{11} + p_{12} + p_{13}, & \\
C_{i3} = max\{C_{i2}, C_{i-1,3}\} + p_{i3}, & i = 2, \dots, n, \\
L_i = C_{i3} - d_i, & i = 1 \dots, n.
\end{array}\right\} (FP)
$$

Where ,

P_{ij} is the processor time of job j in machine i

C_{max} is a maximum of completion time ,

R_L is a range of lateness ($L_{max}(s) - L_{min}(s)$),

C_{ij} is the completion time of job j in machine i

L is the lateness of time $_j = C_j - d_j$

$L_{max}(s) = max\ j \in \{L_j\}$ (maximum lateness),

FP is the flow shop problem.

In the problem that is addressed, the idle time of the machine is not allowed. The multi-criteria problem that we consider concerns the simultaneous minimization of the performance measures. The flow shop problem P is NP-hard since the problem $(F3 \| (C_{max,}\ R_L)$ is NP-hard. The problem $F3 \| (C_{max,}\ R_L)$ is a special case of problem P. In this section, we are presented some lemmas and corollaries, which are the base of the proposed modify BAB algorithm. We define a special cases of flow shop studied are called Proportional Flow Shop (PFS) .

In $PFS1$, assume that $p_{iA} = p_{i1} + p_{i2}$ and $p_{iB} = p_{i2} + p_{i3}$, the processing time of any job on machine A is less than or equal those of every jobs on machine B and $PFS2$, we assume $p_{iB} = p_{i1} + p_{i2}$ and $p_{iA} = p_{i2} + p_{i3}$, the processing time of any job on machine B is less than or equal those of every jobs on machine A

Lemma (3.2.1.): In PFS1, where the objective is to minimize C_{max}, the SPT_A-sequence gives optimal solution for the objective C_{max} and the minimum value of the objective function C_{max} is:

$p_{1A} = p_{11} + p_{12}$,

$$C_{max} = p_{1A} + \sum_{i=1}^{n} p_{i3}.$$

Proof: From the definition of PFS1, the processing time of any job on machine A is less than or equal those of every jobs on machine B.

We get , the minimum value of the objective function C_{max} is

$p_{1A} = p_{11} + p_{12}$, and
$C_{max} = p_{1A} + \sum_{i=1}^{n} p_{i3}$. ■

Corollary (3.2.2): In PFS1, if a job j has a minimum processing time on machine A, then must be scheduled first and any sequence with remaining jobs gives optimal solutions for the objective function C_{max}.

Proof: By lemma 3.2.1, we will be satisfied with taking the smallest p_{iA} on the first machine without having to arrange all the work on SPT_A -sequence

We get, the optimal solutions for the objective function $_{\max}$. ■

Lemma (3.2.3): In PFS1, If reduction in cost is the objective function R_L, for the two adjacent jobs i and j, if $d_i \leq d_j$ and $d_i - p_{iB} \leq d_j - p_{jB}$, then job i precedes job j in the optimal schedule.

Proof: Let $p_{iA} = p_{i1} + p_{i2}$ and $p_{iB} = p_{i2} + p_{i3}$, consider a schedule $s_1 = s'ijs''$ and a schedule $s_2 = s'jis''$ which is obtained by interchanging the jobs i and j in s_1, where s' and s'' are partial sequences of jobs and the two neighboring jobs i and j with $d_i \leq d_j$ and $d_i - p_{iB} \leq d_j - p_{jB}$. Let $t = p_{1A} + \sum_{k=1}^{i-1} p_{k3}$ when the final task was finished of s'.

To prove this lemma, it is sufficient to show that the values of L_{max} and L_{max} for the two jobs i and j do not increase and do not decrease, respectively, when i and j are replaced. This is because the replacement does not affect the completion times of other jobs and, hence, their lateness. The following relations are established on the basis of the aforementioned ones:

(1) To prove $L_{max}(s_1) \leq L_{max}(s_2)$ using the condition $d_i \leq d_j$, let L is the maximum lateness for $(n-2)$ jobs of s' and s''.

For the schedule s_1, we have:

$L_{max}(s_1) = max\{L, L_i(s_1), L_j(s_1)\}$

$= max\{L, p_{1A} + \sum_{k=1}^{i-1} p_{k3} + p_{i3} - d_i, p_{1A} + \sum_{k=1}^{i-1} p_{k3} + p_{i3} + p_{j3} - d_j\}$

Now, consider the schedule s_2, we have:

$L_{max}(s_2) = max\{L, L_j(s_2), L_i(s_2)\}$

$= max\{L, p_{1A} + \sum_{k=1}^{j-1} p_{k3} + p_{j3} - d_j, p_{1A} + \sum_{k=1}^{j-1} p_{k3} + p_{j3} + p_{i3} - d_i\}$

Compare $L_i(s_2)$ with $L_i(s_1)$ and $L_j(s_1)$ where $d_i \leq d_j$

It is clear that $L_i(s_2) > L_i(s_1)$ since $p_{j2} > 0$

Also, it is clear that $L_i(s_2) \geq L_j(s_1)$ since $d_i \leq d_j$

Hence $L_i(s_2) \geq max\{L_i(s_1), L_j(s_1)\}$

That implies $L_{max}(s_1) \leq L_{max}(s_2)$.

(2) To prove $L_{min}(s_1) \geq L_{min}(s_2)$ with condition $d_i - p_{iB} \leq d_j - p_{jB}$, let L is the minimum lateness for $(n-2)$ jobs of s' and s''.

For the schedule s_1, we have:

$L_{min}(s_1) = min\{L, L_i(s_1), L_j(s_1)\}$

$= min\{L, p_{1A} + \sum_{k=1}^{i-1} p_{k3} + p_{i3} - d_i, p_{1A} + \sum_{k=1}^{i-1} p_{k3} + p_{i3} + p_{j3} - d_j\}$

Now, consider the schedule s_2, we have:

$L_{min}(s_2) = min\{L, L_j(s_2), L_i(s_2)\}$

$= min\{L, p_{1A} + \sum_{k=1}^{j-1} p_{k3} + p_{j3} - d_j, p_{1A} + \sum_{k=1}^{j-1} p_{k3} + p_{j3} + p_{i3} - d_i\}$

Compare $L_j(s_2)$ with $L_j(s_1)$ and $L_i(s_1)$ where $d_i - p_{iB} \leq d_j - p_{jB}$

It is clear that $L_j(s_2) < L_j(s_1)$ since $p_{iB} > 0$

Also, it is clear that $L_j(s_2) \leq L_i(s_1)$ since $d_i - p_{iB} \leq d_j - p_{jB}$

Hence $L_j(s_2) \leq min\{L_j(s_1), L_i(s_1)\}$

That implies $L_{min}(s_1) \geq L_{min}(s_2)$.

From (1) and (2) we have:

$L_{max}(s_1) - L_{min}(s_1) \leq L_{max}(s_2) - L_{min}(s_2)$

Then $R_L(s_1) \leq R_L(s_2)$.

Therefore the sequence s_1 is better than the sequence s_2 and job

i precedes job j . ■

Corollary (3.2.4): In PFS2, In the event that job i has the earliest schedule feasible and the same amount of idle time, then B $(d_i - p_{iB})$, then the sequence with order EDD and MST_B minimizes the objective function R_L.

Proof: By lemma 3.2.3 and the definition of PFS2 (the processing time of any job on machine B is less than or equal those of every jobs on machine A) , we get , the sequence with order EDD and MST_B minimizes the objective function R_L.

Lemma (3.2.5): In PFS2, where the objective is to minimize C_{max}, the LPT_B-sequence gives optimal solution for the objective C_{max} and the minimum value of the objective function C_{max} is:

$p_{nB} = p_{n2} + p_{n3}$,

$$C_{max} = p_{nB} + \sum_{i=1}^{n} p_{i1}.$$

Proof: From the definition of PFS2, the processing time of any job on machine A is less than or equal those of every jobs on machine B.

We get , the minimum value of the objective function C_{max} is:

$p_{nB} = p_{n2} + p_{n3}$,

$$C_{max} = p_{nB} + \sum_{i=1}^{n} p_{i1}.$$

Lemma(3.2.6): In PFS2, if a job j has a minimum processing time on machine B, then must be scheduled last and any sequence with remaining jobs give OSs for the objective function C_{max}.

Proof: By lemma 3.2.5 and Jonson rule on machine B , we have remaining jobs give OSs for the objective function C_{max}. ■

Theorem (3.2.7)[45] : There is an efficient sequence in the multi-criteria problem $P1$ that satisfies Johnsons-rule with condition either $min(p_{i1}) \geq max\ (p_{i2})$ or $min(p_{i3}) \geq max\ (p_{i2})$

Corollary (3.2.8): In the optimal solution for the objective C_{max}, if jobs $i \in s' \subset s \in S$ satisfy the Johnsons-rule condition with $min(p_{i3}) \geq max\ (p_{i2})$, then the subsequence s' sequencing by the SPT_A-sequence.

Proof: By theorem 3.2.7 and Jonson rule on machine B with condition $min(p_{i3}) \geq max\ (p_{i2})$, we get subsequence s' sequencing by the SPT_A-sequence in optimal solution for the objective C_{max} ■

3.3. Genetic Algorithm (GA):

Using a population of people with potential solutions (chromosomes), Genetic Algorithms (GAs) are a search and optimization tool.

The following steps describe the structure of GA :

Step (1) : Initialization

A random sample of (m) chromosomes may be used to seed the initial population. In this dissertation, we formulate the issue with m=randomly. $\left(\sum_j^n C_j, \sum_j^n\left(\alpha_j E_j + \beta_j T_j\right)\right)$.

Step (2): New population

The following procedures are repeated until a new population is formed.

(a) Selection: Either the parents are chosen at random from the whole population, or they are picked from the already existing population based on how fit they are.

(b) Crossover : The most crucial operator in the (GA) we use is the crossover operation, which is performed on each set of parent solutions in order to produce two offspring. (PMX)

(c) Mutation: pairwise (swap) mutation is applied on each pair solutions to generated two new solutions (children).

(d) Replacement : It's important to check each of the newly created chromosomes to make sure no flawed ones sneak into the following generation. In cases when multiple solutions are found, the best one is kept as the current one and the process restarts at (2)

(e) Step (3): Termination Condition

When the best solution in a population is not better than that of the previous population, the method is ended when $\mathscr{L}$ generations have passed $\mathscr{L}$ = (1000).

3.4.Memetic Algorithm:

Memetic algorithms may be thought of as the union of a global, population-based approach with a personalized, local search. They are a kind of genetic algorithms known for their use of a hill-climbing local environment. Memetic algorithms are a kind of population-based AI, much as genetic ones. In various problem spaces, they have shown to be several times quicker than standard genetic algorithms. A ternary tree-based population structure was selected for use in this memetic algorithm. It separates the people into groups and limits the opportunities for mixing, in contrast to a random population[48].

3.5.Computational Experiments:

All the algorithms were run on an Intel Core i5 @ 2.53GHz with 4GB of RAM using Matlab R2015a, and their respective results are compared below. Both methods had the same cutoff period in terms of CPU time before they were terminated. This depends on the size of the instance being examined and is expressed in terms of n seconds (n = number of tasks). The MBAB algorithm was performed for 30 minutes (or 1800 seconds), and if the instance took longer than that to solve, the method was considered to have failed.

3.6.Comparison of Results:

For the PFS, the MBAB ,GA and MA algorithms ,were run for each the 10 instances for each n of the problem. The sets S_i (i = 1, . .10), contain the $NDSs$ found by every run. The cardinal measure is calculated for these sets.

Table 3.1 Comparison Results among CEM and MBAB for FP

		CEM			MBAB			
n	**Iter**	**EES**	**Time**	**N**	**EES**	**Time**	**N**	**Nodes**
5	**1**	**(143,101), (146,93)**	**0.0030**	**2**	**(143,101), (146,93)**	**0.0496**	**2**	**25**
	2	**(148,89), (158,86)**	**0.0032**	**2**	**(148,89),(158,86)**	**0.0039**	**2**	**25**
	3	**(139,62)**	**0.0016**	**1**	**(139,62)**	**0.0023**	**1**	**15**
	4	**(143,110),(148,88), (149,75),(150,54)**	**0.0017**	**3**	**(143,110),(148,88), (149,75), (150,54)**	**0.0062**	**4**	**45**
	5	**(133,91)(139,87),(143,55)**	**0.0018**	**3**	**(133,91),(139,87),(143 ,55)**	**0.0047**	**3**	**35**
	6	**(131,80)**	**0.0021**	**1**	**(131,80)**	**0.0020**	**1**	**15**
	7	**(136,62)**	**0.0020**	**1**	**(136,62)**	**0.0018**	**1**	**15**
	8	**(154,77)**	**0.0019**	**1**	**(154,77)**	**0.0022**	**1**	**15**
	9	**(147,47)**	**0.0020**	**1**	**(147,47)**	**0.0024**	**1**	**15**
	10	**(146,96),(149,61),(155,51)**	**0.0034**	**3**	**(146,96),(149,61), (155,51)**	**0.0047**	**3**	**35**

6	1	(168,122), (173,121), (174,116)	0.0138	3	(168,122),(173,121), (174,116)	0.0095	3	51
	2	(163,100), (164,94)	0.0108	2	(163,100),(164,94)	0.0052	2	36
	3	(176,131), (180,94)	0.0106	2	(176,131),(180,94)	0.0052	2	36
	4	(163,82), (168,64)	0.0106	2	(163,82),(168,64)	0.0047	2	36
	5	(167,81), (178,74)	0.0103	2	(167,81),(178,74)	0.0047	2	36
	6	(158,102)	0.0115	1	(158,102)	0.0039	1	21
	7	(165,92)	0.0112	1	(165,92)	0.0037	1	21
	8	(165,116), (168,10), (171,89)	0.0111	3	(165,116),(168,110) ,(171,89)	0.0072	3	51
	9	(170,80;172,56)	0.0112	2	(170,80),(172,56)	0.0065	2	36
	10	(162,83), (164,58)	0.0112	2	(162,83),(164,58)	0.0053	2	36
7	1	(185,139), (189,134), (191,133), (196,131)	0.0815	4	(185,139),(189,134), (191,133),(196,131)	0.0165	4	91
	2	(189,128),(201,122), (203,117)	0.0801	3	(189,128),(201,122), (203,117)	0.0115	3	70
	3	(188,146),(195,132), (198,91)	0.0812	3	(188,146),(195,132), (198,91)	0.0126	3	70
	4	(185,111), (187,69)	0.08304	2	(185,111),(187,69)	0.0065	2	49
	5	(199,74)	0.08442	1	(199,74)	0.0052	1	28
	6	(184,132), (190,126)	0.08387	2	(184,132),(190,126)	0.0062	2	49
	7	(194,135) ,(195,126), (204,112)	0.08649	3	(194,135),(195,126), (204,112)	0.0094	3	70
	8	(188,105), (196,97)	0.08810	2	(188,105),(196,97)	0.0062	2	49
	9	(195,149), (202,130), (204,111)	0.08524	3	(195,149),(202,130), (204,111)	0.0090	3	70
	10	(201,82),(207,69), (215,52)	0.08371	3	(201,82),(207,69),(215,52)	0.0088	3	70
8	1	(218,158), (222,157)	0.7827	2	(218,158),(222,157)	0.0095	2	64
	2	(212,145)	0.7980	1	(212,145)	0.0049	1	36
	3	(220,175), (226,128), (228,109)	0.7353	3	(220,175),(226,128), (228,109)	0.01339	3	92
	4	(208,144), (209,102), (211,99); (212,88)	0.7491	4	(208,144),(209,102), (211,99),(212,88)	0.0155	4	120

	5	**(228,168), (229,102), (241,90)**	**0.6444**	**3**	**(228,168),(229,102), (241,90)**	**0.0123**	**3**	**92**
	6	**(217,171),(218 ,170), (219,158)**	**0.6399**	**3**	**(217,171),(218,170), (219,158)**	**0.0120**	**3**	**92**
	7	**(224,182), (227,142), (234,130)**	**0.6502**	**3**	**(224,182),(227,142), (234,130)**	**0.0129**	**3**	**92**
	8	**(223,187), (225,158), (226,141), (228,121)**	**0.6274**	**4**	**(223,187),(225,158), (226,141),(228,121)**	**0.0159**	**4**	**120**
	9	**(220,161), (221,150), (222,115)**	**0.6321**	**3**	**(220,161),(221,150), (222,115)**	**0.0130**	**3**	**92**
	10	**(233,122), (240,118)**	**0.6528**	**2**	**(233,122),(240,118)**	**0.0090**	**2**	**64**

In Table 3.1, n = 5 to 8 was taken for each (i = 1,2,...,10) between the complete solution (CEM) and the BAB algorithm, so the results showed very close in terms of the number of solutions and their values, so we conclude that the BAB algorithm is a exact algorithm.

Figure (3.1) the chart of applying BAB algorithm , where n=5 only and number of iteration =10.

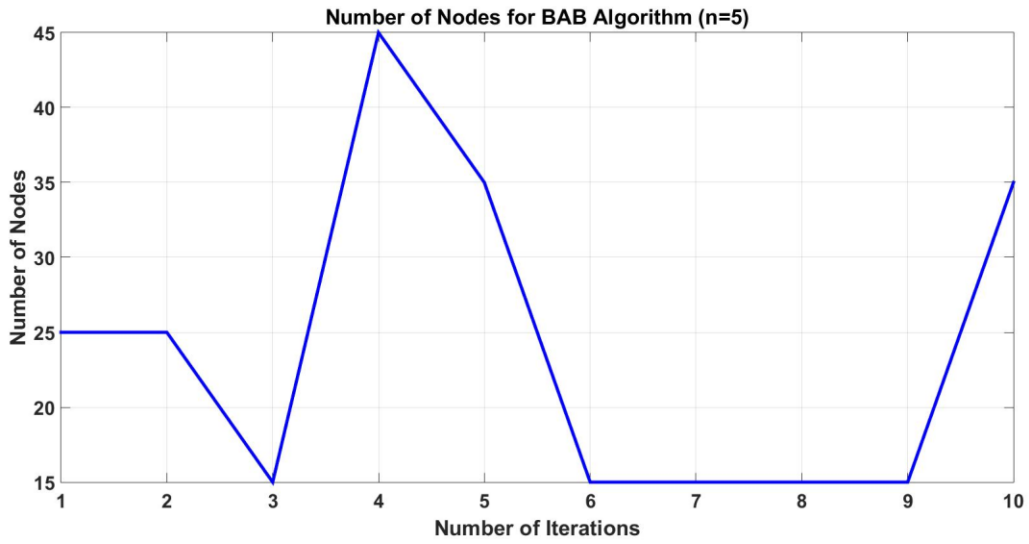


Figure 3.1 .The number of nodes of BAB where n=5

Table 3.2 Comparison results between BAB, GA and MA for FP where $5 \leq n \leq 200$

n	BAB		GA		MA	
	Eff	Time	Eff	Time	Eff	Time
5	19	0.00798	19	0.096545	19	0.110606
6	20	0.00559	19	0.096545	20	0.066379
7	26	0.00919	26	0.062949	26	0.066121
8	28	0.01183	25	0.065905	28	0.069294
9	26	0.01419	23	0.070522	23	0.071937
10	28	0.01824	22	0.076651	28	0.076276

15	**30**	**0.04492**	**20**	**0.095055**	**28**	**0.095471**
20	**28**	**0.07997**	**22**	**0.112917**	**23**	**0.116034**
30	**31**	**0.21748**	**10**	**0.15431**	**10**	**0.155799**
40	**47**	**0.62989**	**10**	**0.192842**	**10**	**0.192815**
50	**45**	**1.04377**	**8**	**0.228183**	**8**	**0.234993**
75	**49**	**3.61365**	**9**	**0.331251**	**9**	**0.334316**
100	**48**	**7.60006**	**8**	**0.468497**	**8**	**0.435716**
200	**46**	**58.0193**	**11**	**1.423861**	**11**	**0.832528**

Table 3.2shows the comparison between the three algorithms BAB, GA and MA, where the number of solutions for each n and the average time were taken (i=1,2,...10).

Figure (3.2) describes comparison chart which shows the relation between values of problem (FP) and number of iterations when applying BAB,GA and MA where n=200.

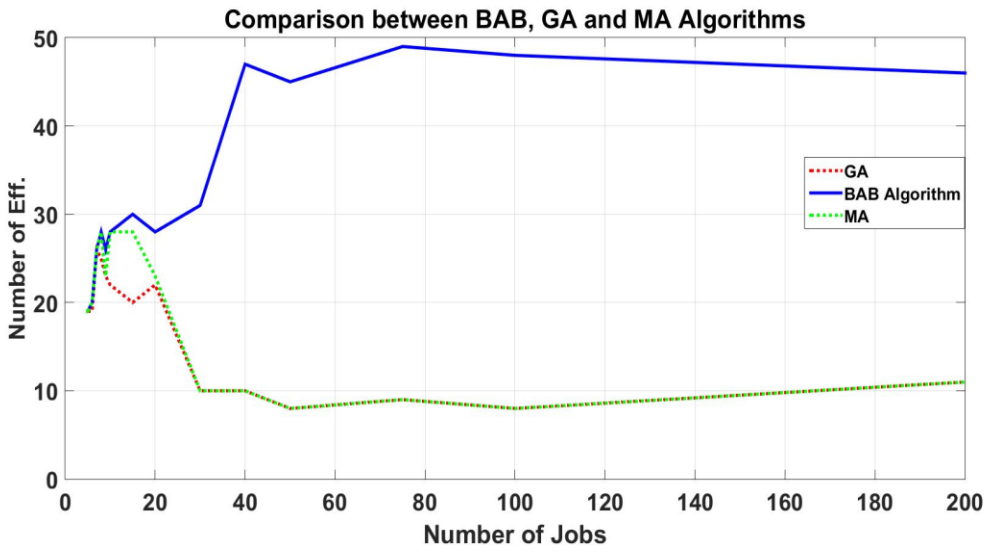


Figure (3.2) comparison chart of applying BAB ,GA and MA for FP, n=200.

Table 3.2 Comparison results between GA and MA for FP where

$$5 \leq n \leq 2000$$

n	GA		MA	
	EES	Time	EES	Time
5	19	0.096545	19	0.110606
6	19	0.096545	20	0.066379
7	24	0.062949	26	0.066121
8	25	0.065905	28	0.069294
9	23	0.070522	23	0.071937
10	22	0.076651	28	0.076276
15	20	0.095055	28	0.095471
20	22	0.112917	26	0.116034
30	10	0.15431	23	0.155799
40	10	0.192842	33	0.192815
50	8	0.228183	22	0.234993
75	9	0.331251	22	0.334316
100	8	0.468497	22	0.435716
200	11	1.423861	14	0.832528
500	37	3.570555	37	2.069793
1000	40	4.695768	37	7.777485
2000	35	10.64744	39	10.85984

Table 3.3 shows the comparison between the GA and MA algorithms, where the number of solutions per n and the average time (i = 1,2, ...10) were taken. Table 3.3 displays the numbers of solutions obtained by the GA and MA algorithms is close, as both algorithms showed similar results.

Figure (3.3) describes comparison chart which shows the relation between values of problem (FP) and number of iterations when applying GA and MA where $5 \leq n \leq 200$

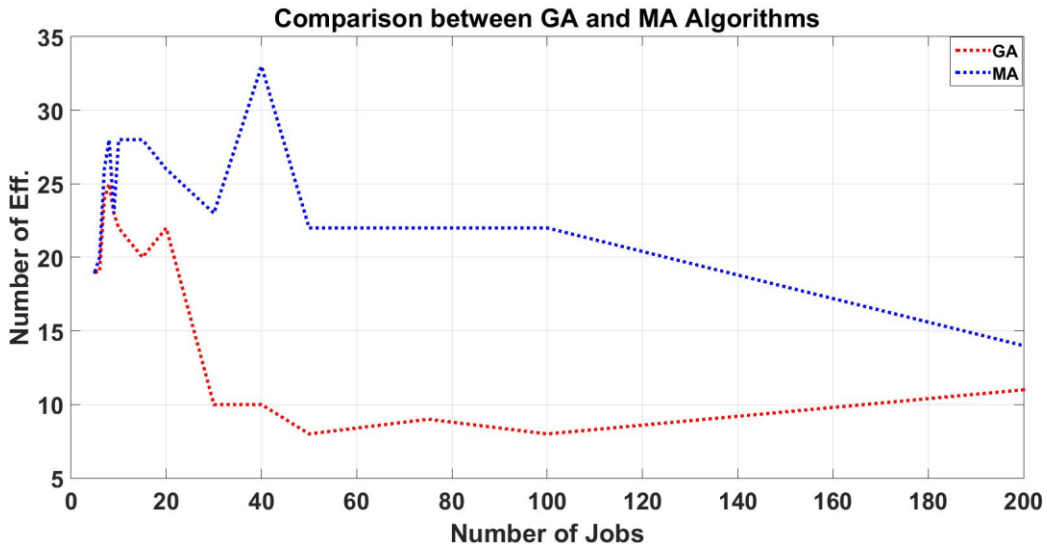


Figure (3.3) Comparison chart of applying GA and MA for FP, n=200.

4.1.Introdction :

In this chapter, we will explore the different approaches to solving the parallel machine scheduling problem, with a particular focus on genetic algorithms and tabu search. We will review the relevant literature, including the studies mentioned above, and provide insights into the strengths and limitations of each method.

4.2 Problem Formulation

Suppose we have two identical parallel machines, the goal is to minimize of the completion time cost as well as the earliness time cost with the lateness time cost . We will have the following objective function:

$$Min\left(\sum_{j}^{n} C_j, \sum_{j}^{n}\left(\alpha_j E_j + \beta_j T_j\right)\right) \qquad \text{... (IPP)}$$

The constraints for the objective function of two identical parallel machines are:

1. Capacity constraint: The total processing time of all jobs assigned to each machine cannot exceed its capacity. Mathematically, it can be expressed as:

$\sum_j^n p_{ij} \leq C_j$, where p_{ij} is the processing time of job j on machine i and Cj is the capacity of machine i.

2. Job assignment constraint: Each job must be assigned to exactly one machine. Mathematically, it can be expressed as:

$\sum_j^n x_{ij} = 1$, where x_{ij} is a binary decision variable that equals 1 if job j is assigned to machine i and 0 otherwise.

3. Non-preemptive constraint: Once a job is assigned to a machine, it cannot be interrupted or moved to another machine. Mathematically, it can be expressed as:

$\sum_j^n x_{ij}.p_{ij} = \sum_j^n x_{ik}.p_{ik}$, for all $i \in \{1,2\}$, where x_{ij} and x_{ik} are binary decision variables that equal 1 if job j and k are assigned to machine i, respectively, and 0 otherwise.

4. Objective function constraint: The objective function is the sum of the completion times on each machine and the weighted sum of earliness and tardiness penalties. Mathematically, it can be expressed as:

$$Min\left(\sum_j^n C_j + \sum_j^n (\alpha_j E_j + \beta_j T_j)\right) \quad \ldots(4.1)$$

Or

$$Min\left(\sum_j^n (C_j + \alpha_j E_j + \beta_j T_j)\right) \quad \ldots(4.2)$$

where E_j is the earliness of job j on machine i (i.e., the difference between its completion time and its due date if it is completed before the due date, and 0 otherwise), T_j is the tardiness of job j on machine i (i.e., the difference between its completion time and its due date if it is completed after the due date, and 0 otherwise), α_j and β_j are weighting factors for earliness and tardiness, respectively.

These constraints ensure that the jobs are assigned to the machines in a way that minimizes the objective function while respecting the capacity and assignment constraints.

4.3.Exact and Local Search Algorithms

4.3.1.BAB Algorithm

The Branch and Bound (BAB) algorithm is a widely used optimization technique in computer science and operations research. It is particularly useful for solving combinatorial optimization problems such as the matching of parallel machines .

In the context of matching parallel machines, the goal is to assign a set of tasks to a set of machines in such a way that the total time, early time, and delay time are minimized. The BAB algorithm is well-suited for this problem because it systematically explores the solution space by dividing it into smaller and smaller sub-problems until the optimal solution is found .[50]

At a high level, the BAB algorithm works by creating a search tree that represents all possible assignments of tasks to machines. The root node of the tree represents the initial state of the problem, and each child node represents a possible assignment of a single task to a machine. The algorithm then evaluates each child node to determine its quality and selects the most promising node to explore further. This process continues until the optimal solution is found or all nodes have been evaluated.[47]

The BAB algorithm is particularly effective for solving matching parallel machines problems because it can quickly eliminate large portions of the solution space that are unlikely to contain optimal solutions. By using various heuristics to guide the search process, the BAB algorithm can efficiently find near-optimal solutions even for large problem sizes.

Overall, the BAB algorithm is a powerful tool for optimizing the assignment of tasks to parallel machines. By reducing the total time, early time, and delay

time, it can help organizations save time and resources while improving their productivity and efficiency.

4.3.2.Lower Bound of BAB

From Eq. (4.2)

$$Min\, f(\sigma) = \underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(C_{\sigma j} + \alpha_{\sigma j}E_{\sigma j} + \beta_{\sigma j}T_{\sigma j}\right)\right)$$

Assume $\alpha_{\sigma j} = \beta_{\sigma j} = 1$ for all jobs j.

$$f(\sigma) \geq \underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j} - C_{\sigma j}, 0\right\} + Max\left\{C_{\sigma j} - d_{\sigma j}, 0\right\} + C_{\sigma j}\right)\right)$$

$$= \underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j} - C_{\sigma j}, C_{\sigma j} - d_{\sigma j}, 0\right\} + C_{\sigma j}\right)\right)$$

$$= \underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j}, 2C_{\sigma j} - d_{\sigma j}, C_{\sigma j}\right\}\right)\right)$$

It is clear from above inequality function $f(\sigma)$ a lower bound (LB) is obtained by sequencing the jobs by SPT rule for all machine i.

Hence, we can prove that:

$$Min\, f(\sigma) \geq \underset{\sigma \in S}{Min}\left(Max\sum_{j=1}^{n} d_{\sigma j}\,, \sum_{j=1}^{n}\left(Max\left\{2C_{\sigma j} - d_{\sigma j}, C_{\sigma j}\right\}\right)\right)$$

It is a LB of our problem, since

$$\underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j}, 2C_{\sigma j}-d_{\sigma j}, C_{\sigma j}\right\}\right)\right)$$

$$\geq \underset{\sigma \in S}{Min}\left(Max\sum_{j=1}^{n} d_{\sigma j}, \sum_{j=1}^{n}\left(Max\left\{2C_{\sigma j}-d_{\sigma j}, C_{\sigma j}\right\}\right)\right)$$

Let $v_{\sigma j}=Max\left\{2C_{\sigma j}-d_{\sigma j}, C_{\sigma j}\right\}$

To show $\underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j}, v_{\sigma j}\right\}\right)\right) \geq \underset{\sigma \in S}{Min}\left(Max\left(\sum_{j=1}^{n} d_{\sigma j}, \sum_{j=1}^{n} v_{\sigma j}\right)\right)$

when $d_{\sigma j}$ and $v_{\sigma j}$ are positive integers, then

$$\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j}, v_{\sigma j}\right\}\right) \geq Max\left(\sum_{j=1}^{n} d_{\sigma j}, \sum_{j=1}^{n} v_{\sigma j}\right)$$

Hence, it is clear that

$$\underset{\sigma \in S}{Min}\left(\sum_{j=1}^{n}\left(Max\left\{d_{\sigma j}, 2C_{\sigma j}-d_{\sigma j}, C_{\sigma j}\right\}\right)\right)$$

$$\geq \underset{\sigma \in S}{Min}\left(Max\sum_{j=1}^{n} d_{\sigma j}, \sum_{j=1}^{n}\left(Max\left\{2C_{\sigma j}-d_{\sigma j}, C_{\sigma j}\right\}\right)\right)$$

Hence, LB = $\underset{\sigma \in S}{Min}\left(Max\left(\sum_{j=1}^{n} d_{\sigma j}, \sum_{j=1}^{n}\left(Max\left\{2C_{\sigma j}-d_{\sigma j}, C_{\sigma j}\right\}\right)\right)\right)$

4.3.3.Genetic Algorithm (GA)

Genetic Algorithm (GA) is general search and optimization method work on a population of feasible solutions (chromosomes) individuals to a given problem.

The following steps describe the structure of GA :

Step (1) : Initialization

The initial population can be generated at random of (m) individual chromosomes (solutions). In this dissertation we start with m randomly for the our problem $\left(\sum_j^n C_j,\ \sum_j^n\left(\alpha_j E_j + \beta_j T_j\right)\right)$.

Step (2): New population

A new population is created by repeating the following steps until the new population is completed .

(a) Selection: The parents are selected at a random from the rest of the population or they are selected from a current population according to their values the batter fitness, the bigger chance to be selected

(b) Crossover : The most crucial operator in the (GA) we use is the crossover operation, which is performed on each set of parent solutions in order to produce two offspring.

(c) Mutation: pairwise (swap) mutation is applied on each pair solutions to generated two new solutions (children).

(d) Replacement :Evaluate each of the new generated chromosomes and replace bad individuals will skive into the next generation. However , duplicated solutions may occur , save the best individual of the new population as the current solution and return to step (2).

Step (3): Termination Condition

We terminate the algorithm when the number of generations reaches the value $\mathscr{L}$ where $\mathscr{L}$ = (10000), when the best solution in a population is not better than that of the pervious population then procedure is terminated.

4.3.4.Tabu search

There is a procedure to be considered in order to wake computational using TS.

A general out line of a TS procedure as follows:

Given a feasible initial solution s* with objective function value z*, let s= s*

And z(s)= z*., while stopping criterion is not fulfilled do the following steps:

Step(1): Selected best adissible move that s into s′ with objective function value z(s′) and add its sequence to the tabu list.

Step(2): Perform tabu list management compute move to be set as tabu , i.e. update the tabu list.

Step(3): Perform exchange

s= s′ and z(s)= z(s′) ..(4.3)

if z(s)≤ z* then z* = z(s) and s*=s. …(4.4)

Step(4): We terminate the algorithm when the number of generations reaches the value $\mathscr{L}$ =(10000) and s* is the best of all determined solution with objective function value z*.

4.4.Computational experience

This section reports the results of computational test to assess the effectiveness heuristics algorithms. These algorithms are coded in in Matlab 2021 and executed on an Cpu Core i7 5720mq with a 2.7 GHz and RAM 16 GA. The algorithms were run with the same stopping criterion (Stoping Criterion) based on an amount of CPU time.

4.4.1 Problems Instances

The algorithms are compared on 10 problems instances. For the complete enumeration (CE) method and the reference set the sizes of these instances are (3, 4, 5, 6, 7), for the BAB method are(3,4,5,6,7), and the sizes of more jobs instances are (5, 10, 15, 20, 25, 30, 40, 50, 100, 200, 500, 1000, 2000) jobs. The problems were generated randomly. For each job j, the processing times p_{ij} was uniformly generated in [1,10].

4.4.2 Comparative computational results

This section will report the results of our computational test to show the effectiveness for the Branch and Bound (MBAB) method the local search methods (genetic algorithms (GA) and tabu search (TS)), we present tables of results which shows the importance of each of the methods. In each tables the first column gives the number of jobs the second column gives the number .The third column describes . The fourth, fifth and sixth columns.

Table 4.1 Comparison Results among CEM and MBAB for IPP ,$3 \leq n \leq 7$

Comparison between Complete Solution and BAB Algorithm					
N	CEM		MBAB		
	No. of Sol	Time(s)	No. of Sol	Time(s)	Nodes
3	15	0.3143539	10	0.0356346	29
4	25	0.0021941	21	0.0091321	132.91
5	38	0.0048883	31	0.0103229	1189.4

6	79	0.0923632	59	0.0460734	6671.6
7	142	9.98810817	106	2.85608785	76648

Table 4.1 shows the comparison between the complete solution of CEM and the MBAB algorithm, where the number of solutions was taken for each ten examples in n and the average time for these examples, where similar results appeared for the algorithm with CEM .

Figure (4.1) describes comparison chart which shows the relation between values of problem (IPP) and number of iterations when applying CEM and GA and MBAB where $3 \leq n \leq 7$

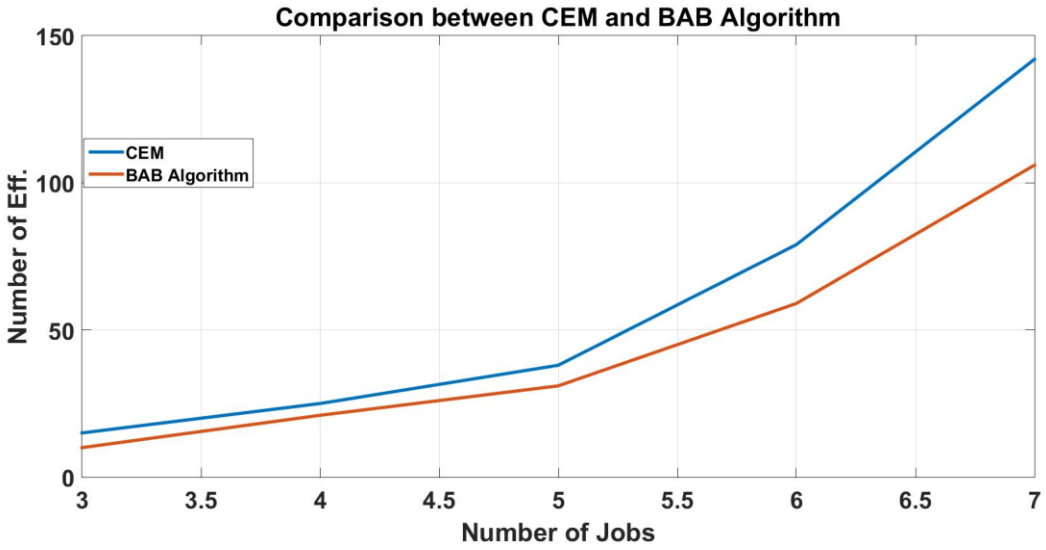


Figure (4.1) Comparison chart of applying CEM and MBAB for IPP

Table 4.2 Comparison results between MBAB,GA and TS for IPP where $5 \leq n \leq 10$

Comparison between MBAB , Genetic and Tabu Search						
n	MBAB		GA		TS	
	No. of Sol	Time(s)	No. of Sol	Time(s)	No. of Sol	Time(s)
5	31	0.0103229	17	0.5155333	12	0.0562403
6	59	0.0460734	37	0.5063162	14	0.0568675
7	106	2.85608785	80	0.5411111	14	0.0634693
8	188	2107.329602	124	0.5431343	33	0.072857
9	259	4662.98984	202	0.5834485	27	0.0867459
10	255	5469.844022	252	0.4828122	54	0.0753819

Table 4.2 shows the comparison between the three algorithms MBAB, GA, and TS, where the number of solutions was taken for each ten examples in n, and the average time for these examples, as the results of TS showed the lowest possible solutions, and the remaining two algorithms had similar results.

Figure (4.2) describes comparison chart which shows the relation between values of problem (IPP) and number of iterations when applying MBAB ,GA and TS where $\mathbf{5 \leq n \leq 10}$

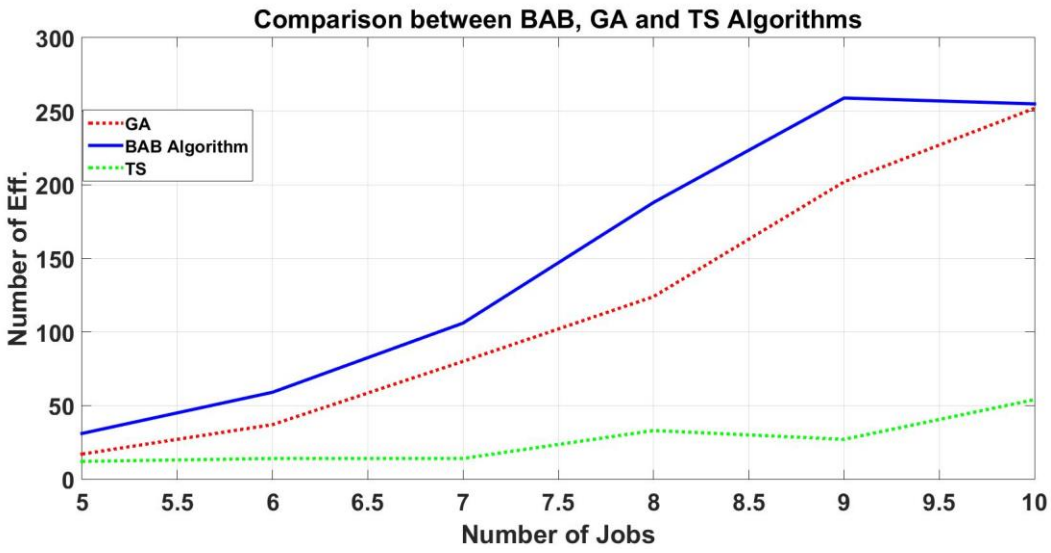


Figure (4.2) Comparison chart of applying MBAB ,GA and TS where $\mathbf{5 \leq n \leq 10}$

Table 4.3 Comparison results between GA and TS for IPP where

$$5 \leq n \leq 2000$$

n	Comparison between Genetic Algorithm and Tabu Search			
	GA		TS	
	No. of Sol.	Time(s)	No. of Sol.	Time(s)
5	17	0.5155333	12	0.0562403
10	253	0.6339888	56	0.0988966
15	236	0.7376676	54	0.1483479
20	251	1.0125075	89	0.2070926
25	297	1.0125075	132	0.2699514
30	294	1.1297793	253	0.6339888
40	292	1.3990928	235	0.4980252
50	324	1.6517448	238	0.6873296
100	339	2.9507885	350	2.3416946
200	293	5.7969116	382	8.2839836
500	376	15.874865	1177	51.3586491
1000	373	36.229470	1577	243.1260402

2000	307	97.018255	1424	600.1819503

Table 4.3 shows the comparison between the two algorithms GA and TS, where the number of solutions was taken for every ten examples in n and the average time for these examples, as the results of TS showed the lowest possible solutions of the GA algorithm.

Figure (4.3) describes comparison chart which shows the relation between values of problem (IPP) and number of iterations when applying GA and TS where $5 \leq n \leq 2000$

.

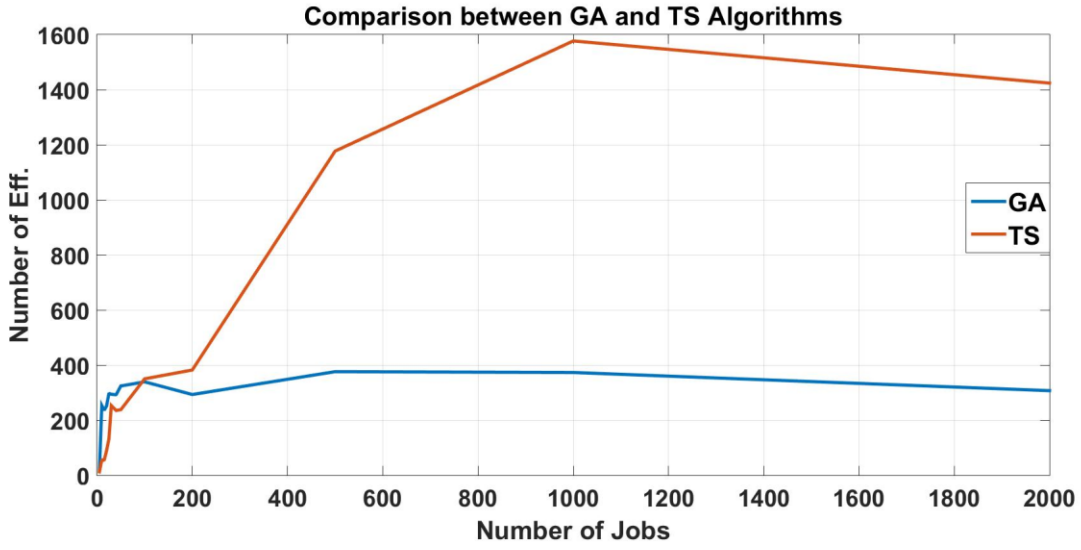


Figure (4.3) Comparison chart of applying GA and TS for IPP, $5 \leq n \leq 2000$

5.1.Introduction:

The Aircraft Landing Problem (ALP) is a significant class of scheduling issues that offers significant benefits to systems facing extended delays. The ALP is the issue of deciding which incoming planes get which runways and when they get to land. The penalty is based on the spread in landing timings from the ideal between the earliest and later possible periods. The aim of the ALP is to reduce the number of times landing timings deviate from the ideal. The objective function has two parts since it punishes both early and late landings.

There are usually many runways available at major international airports. In cases when more than one runway is available, we must thus choose the most suitable runway for flights to land on and assign a landing time to each jet.

The problem of the separation time between flights is relevant if we have planes landing on various runways.

n this work , we have assumed, essentially, that aircraft landing on various runways have the same separation time.

5.2. Classification of ALP

ALP may be decomposed into single-runway and multi-runway issues based on the foundational concept. In order to apply ALP to issues in a multi-airport setting, it is necessary to create sophisticated models that take airport resources into account.

5.2.1 Problems for single runways

Airports may be classified as either "single runway" or "multiple runway" depending on the number of runways they have. In the event of a single runway issue (SRP), all planes must land at the same airport. Runway assignment and sequencing are both required in the multi-runway problem (MRP). In queuing theory, both single-server resource planning (SRP) and multi-server resource planning (MRP), which includes sequencing and distribution, may be abstracted as single-server queues. As most countries airports only have access to a single runway, SRP research is still relevant. As the foundation of research into the multi-runway issue, its accuracy and

efficiency are subject to steady development. MRP is the major topic of the present study. The development and refinement of heuristic algorithms has led to an increase in the complexity of problems addressed and an enhancement of the accuracy and efficiency of computation. The present research focuses primarily on the parallel runways layout of multi-runway airports because of the significant impact it has on airport capacity. Although most of some countries airports with more than one runway employ a layout where the runways run in parallel to one another, the new airport in Chengdu has a configuration where the runways run in parallel to one another and a lateral runway, which will affect the wake interval. Independent or relative parallel instrument approach, and so on, are only a few of the runways' operational modes. The multiple runways function separately from one another in independent parallel instrument approach mode. Landing planes on the two runways next to each other must maintain a certain radar spacing in the applicable parallel instrument approach mode. In this study, we primarily used a parallel instrument strategy that relied on autonomy.

5.2.2. Problems for multiple runways

P : numbers of planes .

ELTi : planes may touch down as early as i (i=1,...,P)

LLTi : airplane's latest possible arrival time i (i=1,...,P)

xi : the landing time for plane i (i=1,...,P)

αi : how soon plane i (i=1,...,P) lands before Ti

βi : how soon plane i (i=1,...,P) lands after Ti

δij : 1 if plane i lands before plane j (i=1,...,P; j=1,...,P; i$\neq$j)

0 otherwise

TLTi : time of plane's planned (preferred) landing i (i=1,...,P)

Sij : needed landing interval (0) between flights I and J, assuming Flight I lands first and Flight J lands second on separate runways. To simplify, let's write: i=1,...,P; j=1,...,P; i$\neq$j

gi : plane i (i=1,...,P) has a penalty cost of (0) per unit of time if it lands before the goal time Ti.

hi : cost (in zeros) for aircraft i (i=1,...,P) to land for every second it is later than the goal time Ti.

R : total number of aircraft

yir = 1 aircraft i (i=1,...,P) lands on runway r (r=1,...,R)

= 0 otherwise

zij = 1 If flights i and j use the same strip, then (i=1,...,P; j=1,...,P; i≠j)

= 0 otherwise

= 0 otherwise

The following restrictions are then added to the preceding statement of the problem:

$\text{Min} \sum_{i=1}^{p}(g_i\alpha_i + h_i\beta_i)$ ……..(ZP)

s.t.

$E_i \leq x_i \leq L_i \; \forall i \in P; \; i \neq j$

$\delta_{ij} + \delta_{ji} \leq 1 \; \forall i, j \in P$

$\sum_{r=1}^{R} y_{ir} = 1 \qquad i=1,...,P$

$z_{ij} = z_{ji} \quad i=1,...,P; \; j=1,...,P; \; i \neq j$

$z_{ij} \geq y_{ir} + y_{jr} - 1 \quad i=1,...,P; \; j=1,...,P; \; j>i; \; r=1,...,R$

$\alpha_i \geq T_i - x_i \; \forall i \in P$

$0 \leq \alpha_i \leq T_i - E_i \quad \forall i \in P$

$\beta_i \geq x_i - T_i \quad \forall i \in P$

$0 \leq \beta_i \leq L_i - T_i \; \forall i \in P$

$x_i = T_i - \alpha_i + \beta_i \quad \forall i \in P$

It's also possible that we'll shift our focus to equitably dividing traffic across runways. This is simple to put into words. In order to articulate the issue of balancing workload, we may modify the goal of the multiple runway model described above as follows, with Zmin representing the workload on the least busy runway and Zmax representing the burden on the busiest runway:

$\min Z_{max} - Z_{min}$

s.t.

$Z_{max} \geq \sum_{i=1}^{p} w_i r y_i r \quad r=1,...,R$

$Z_{min} \leq \sum_{i=1}^{p} w_i r y_i r \quad r=1,...,R$

5.3. Improvement Scheduling Aircraft (ISA) Algorithm:

A heuristic algorithm for the static ALP is based on constraint position shifting concept is given by Dear and Sherif [51]. Abdul-Razaq and Ali [2] attempted and they succeed to modify the PIA to improve the efficiency of PIA, which introduced by Bencheikh et al. [39] and calledthe new PIA by Modified PIA ($MPIA$). The PIA and $MPIA$ are described by Algorithms 2.1 and 2.2, respectively (see Chapter Two). Now we introduce a new development for the PIA. The ISA algorithm improvesthe actual computed aircraft scheduling LT to minimize the total penalty cost Z. This improvement is based on the reduction of the total cost of the penalty incurred by all aircraft. To get this object, we have to solve two problems: the first is a sequencing problem that determines the sequence of aircraft landing, and the second is a scheduling problem that determinesthe SLT for all aircraft in the sequence, subject to all constraints. The first problem (sequencing problem) can be solved by implementing any heuristic for ALP. Then, for the second problem (scheduling problem) to schedule the aircraft in order to minimize the total cost of the penalty caused by each aircraft and to improve the SLT, we recalculate the LTs foreach aircraft. Since the LTs assigned to

aircraft, then it is possible to minimize the deviation from the target time for all aircraft. Now the ISA algorithm can be described. Start with any sequence as a heuristic for ALP. We change the LT of a selected aircraft i, as follws: if the aircraft i land in advance, then we increase its LT toward the TLT, in this case, we may have got a zero contribution to the objective function for the aircraft i, if it is not, i.e. if the aircraft i lands in tardy, then we decrease LTs by one unit of time for aircraft i and the adjacent aircrafts for it (that the time between them is only the separation time), when the difference between the $SLTs$ for adjacent aircrafts equal to the separation times, in this case we have to verify the viability of the solution: if the new LTs respects the period of security between the previous periods and it is FS, we continue to decrease its LT by one unit of time, otherwise reject the change and keep the last FS. This means that we organizing the LT for the aircraft that increases the penalty in order to reduce the total cost of penalties for all aircraft.

5.4. Techniques to Improve the Solution and Reduce the Computations

We show three different approaches that help the solution overall and get us closer to the optimal one faster.

The scheduling indicated by raises (lowers) the ti value by one unit if it is less (greater) than Ti, as was described before; the PIA is located which performs the sequencing using one of the priority rules's. In this part, we'll discuss the ways in which we've tried to enhance the PIA's effectiveness. We gave the revised PIA the catchy acronym MPIA. The change is represented by comparing the penalty cost for a given interval of landing aircraft time improvement (let's call it Z_0) at the beginning of the modification to the penalty cost for that same interval at the end of the modification (let's call it Z), and if $Z>Z_0$, ignoring Z and keeping Z_0, and otherwise taking Z.

5.4.1 A Variable Neighborhood Descent (VND) Algorithm:[43]

The VND meta-heuristic, which belong to the class of the VNS algorithms. To expand the space of the solution to be used more comprehensive change, which increases the possibility of finding the best solutions. The following Algorithm 4.3 shows the VND algorithm, which starts by initial solution s of the PRs using the ISA with aim to improve it. As long as NH s^* is improved, the algorithm will continue the search. Otherwise, it returns to the previous NH s when there is no best solution in the current NH. Thus, the algorithm VND accurately searching at all until the NH chooses the best solution. We either stop at a given time (3600 second) or if the NH s is not improved for 100 iteration then stop. The best solution is to be the last NH

Algorithm 5.1 Variable Neighborhood Descent (VND) Algorithm for Z

1. $s \leftarrow$ *Be an initial solution;*
2. *while until time is finished in sec*
3. $s^* \leftarrow$ *Neighbor of* s;
4. *Evaluate* s^* *by ISA algorithm;*
5. *if* s^* *be a best solution*
6. $s \leftarrow s^*$;
7. *else*
8. $s \leftarrow$ *Repeat previous neighborhood* s;
7. *end*
8. *if* s *is not improved for* 100 *iteration*
9. *Stop;*
10. *end*
11. *end*
12. end

5.4.2 Intensification Algorithm (IA):

In this section, we introduce a new type of local algorithms depending on the initial solutions of the PRs using the ISA, divide the sequence s into four partial subsequences $s_{T\mathrm{i}}$ and $s_{NT\mathrm{i}}$ (i=1,2), the first subsequence $s_{T\mathrm{i}}$ contains maximum number of the aircrafts that landed at the TLT and the second subsequence $s_{NT\mathrm{i}}$ contains all the aircrafts that landed on non-target LTs. The algorithm has used parameter which is the number of aircraft in s_{NT}. The Intensification Algorithm (IA) is composed of two phases: destruction and construction. In the destruction phase, d aircraft (choose from all aircraft that landing in non-target LT) are deducted from s and a partial solution $s_{T\mathrm{i}}$ (of size $P_{\mathrm{i}} - d_{\mathrm{i}}$) is obtained. The d aircrafts are stored in $s_{NT\mathrm{i}}$ ($s_{NT\mathrm{i}}(\mathrm{j})$, $\mathrm{j} = 1, \dots, d$, are the deducted aircraft). The construction phase has d steps. In step $\mathrm{j} = 1$, inserting aircraft $s_{NT\mathrm{i}}(1)$ in all possiblepositions of s^* (is a partial scheduling in which not necessary that all aircrafts are scheduled on the TLT), while maintaining the issue restrictions, to get $(P_{\mathrm{i}} - d_{\mathrm{i}} + 1)$ partial solutions. The best s^* is selected (it is less cost for the penalty) as new s^* . In this technique, we suggest intensification procedure to improve a FS, best chosen from set D_α (set of partial FSs, which is less than or equal to $(P_{\mathrm{i}} - d_{\mathrm{i}} + 1)$ solutions).

Algorithm 5.2 Intensification Algorithm (IA) for Z

1. $s \leftarrow$ *Be an initial solution*;
2. *while until time* is *finished in sec*
3. $s_{NT} \leftarrow$ *Set of all non − target landing d aircrafts from s and* s_T *be a set of all target landing*;
4. *for* j $\leftarrow$ 1 *to* d
5. $D_\alpha \leftarrow$;
6. *for* $k_1 \leftarrow 1$ *to* $P - d +$ j
7. if $ELT_{k1} + S_{k1,\,k1+1} \leq TLT_{k+1}$
8. *Insert aircraft* (j) *in k position of* s^*;
9. *Evaluate each sequence* s^* *by ISA algorithm*;
10. $D_\alpha \leftarrow$ *Set of partial solutions of* $D_\alpha \cup \{s^*\}$;
11. *end*
12. *end*
13. $s^* \leftarrow$ *Best select solution of* D_α;
14. *end*
15. *for* $k_2 \leftarrow 1$ *to* $P - d +$ j
16. if $ELT_{k2} + S_{k2,k2+1} \leq TLT_{k+1}$
17. *Insert aircraft* (j) *in k position of* s^*;
18. *Evaluate each sequence* s^* *by ISA algorithm*;
19. $D_\alpha \leftarrow$ *Set of partial solutions of* $D_\alpha \cup \{s^*\}$;
20. *end*
21. *end*
22. $s^* \leftarrow$ *Best select solution of* D_α;
23. *end*
24. $D_\beta \leftarrow D_\alpha$;
25. $s \leftarrow$ *Best select solution of* D_β;
26. if s *be detected previously or* it is *not improved for* 100 *iteration*
27. *Stop*;
28. *end*
29. *end*

5.4.3 Algorithm: Modified Parallel Improving Algorithm (MPIA)

Let $s = (s_1, \dots, s_P)$ be a sequence of aircrafts obtained by any heuristicand $O = \{\emptyset\}$ be the set of schedule aircrafts.

1. $O \leftarrow \{ s_1 \}$
2. $SLT_{11} \leftarrow TLT_{11}$
3. For j← 2 to P
    a. $SLT_{1j} \leftarrow \max (TLT_{1j}, \max_{i \in O1} (SLT_{1i} + S_{ij})$; % i<j
    b. If $SLT_{1j} = TLT_{1j}$
        i. $O_1 \leftarrow O_1 \cup S_{ij}$
        ii. Return to step (3)
    c. else % $SLT_{1j} > TLT_{1j}$
        i. $SLT_{2j} \leftarrow \max (TLT_{2j}, \max_{i \in O1} (SLT_{2i} + S_{ij})$; % i<j
        ii. if $SLT_{2j} = TLT_{2j}$
            1. $O_2 \leftarrow O_2 \cup S_{ij}$
        iii. else % $SLT_{1j} > TLT_{1j}$ & $SLT_{2j} > TLT_{2j}$
            1. *Reduce the* SLT_j *for* j *and adjecent aircrafts by* 1 *unit of time*;
                a. i*f* (*The solution is feasible* i. *e. there is no increasing of penalty cost*)
                b. *Repeat step* 9 *until* $SLT_{ij} = TLT_{ij}$;
            2. *else*
                a. *Reject the change and keep the last feasible solution*;
                b. *Return step*3;
            3. *end*

4. Idx =min{SLT_{1j}, SLT_{2j}}
5. if Idx = 1
    a. $O_1 \leftarrow O_1 \cup S_{ij}$
6. *else*
    a. $O_2 \leftarrow O_2 \cup S_{ij}$
7. *end*

iv. *end*

d. *end*

4. *end*

Algorithm 5.4.-Intensification Algorithm (IA) for Z

1. $s \leftarrow$ *Be an initial solution*;
2. *while until time is finished in sec*
3. $s_{NT} \leftarrow$ *Set of all non − target landing d aircrafts from and* s_T *be a set of all target landing*;
4. *for* j $\leftarrow$ 1 *to* d
5. $D_\alpha \leftarrow$;
6. *for* $k \leftarrow 1$ *to* $P - d +$ j
7. if $ELT_k + S_{k,k+1} \leq TLT_{k+1}$
8. *Insert aircraft* (j) *in* k *position of* s^*;
9. *Evaluate each sequence* s^* *by ISA algorithm*;
10. $D_\alpha \leftarrow$ *Set of partial solutions of* $D_\alpha \cup \{s^*\}$;
11. *end*
12. *end*
13. $s^* \leftarrow$ *Best select solution of* D_α;
14. $D_\beta \leftarrow D_\alpha$;
15. $s \leftarrow$ *Best select solution of* D_β;
16. if *s be detected previously or* it *is not improved for* 100 *iteration*
17. *Stop*;
18. *end*
19. *end*

5.5 Results for instances with two runways:

Table 5.1 Comparison Results Priority Rules (*PRs*) for Sequencing *ALP*

P	ELT_i	TLT_i	LLT_i	$\frac{ELT_i}{g_i}$	$\frac{LLT_i}{h_i}$	$\frac{TLT_i}{g_i + h_i}$	$\frac{ELT_i + LLT_i}{g_i + h_i}$	$Best$ Z
10	90	90	670	190	400	190	250	90
15	270	210	270	580	580	520	520	210
20	300	240	1810	270	160	210	270	160
20	1520	640	1660	760	1050	640	640	640
20	1040	1230	1000	900	840	750	840	750
30	1008	1008	1008	17798	24360	17798	24557	1008
44	0	0	0	26704	29776	28342	28342	0
50	10530	405	30045	71550	91725	64355	85570	405
100	7864.53	634.900	7864.5	84930.0	166538.	59927.4	77238.09	634.900
150	13332.1	1821.78	13332.	305572.	377217.	248167.	256157.9	1821.78
200	16014.7	2006.53	16014.	472819.	675901.	408254.	434476.8	16014
250	20826.9	2430.36	20826.	846361.	940839.	623004.	718416.1	2430.36
500	46641.7	5267.26	46641.	368399	467544	308810	2922689.	5267.26
	3	8	2	0	1	2	1	

From Table 5.1 the best priorities and best results are the numbers of last row.

The best objective function (Z) is TLT_1.

Table 5.2 Comparison Results among GA , DS and IA without *TWT*

P	GA	DS	IA	$Best\ Z$
10	90	90	90	90
15	210	210	210	210
20	60	60	160	60
20	640	640	640	640
20	690	690	750	690
30	636	636	1008	636
44	0	0	0	0
50	185	335	405	185
100	568.29	576.45	634.9	568.29
150	1477.75	1410.48	1821.78	1410.48
200	1543.53	1549.21	2006.53	1543.53
250	1923.33	2173.78	2430.36	1923.33
500	4833.76	4757.01	5267.26	4757.01

In comparison to other commonly used algorithms, Table 5.2 shows that the GA algorithm provides the best solutions to the needed objective function.

The following figure shows the comparison between among GA , DS and IA algorithms . The figure show that shows that the GA algorithm provides the best solutions to the needed objective function.

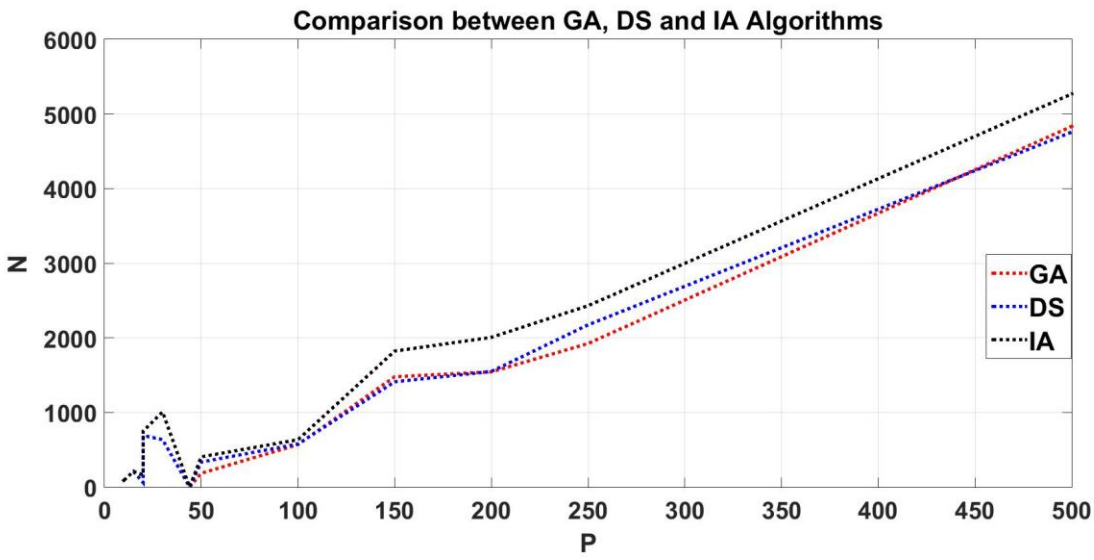


Figure 5.1 Comparison results among GA , DS and IA algorithms.

The following figure shows the comparison between among GA , DS , IA and Pinol and Beasley algorithms.

By comparing the results of the various algorithms, Figure 5.2 demonstrates that the results produced by the employed algorithms are extremely similar to the results of the global algorithm.

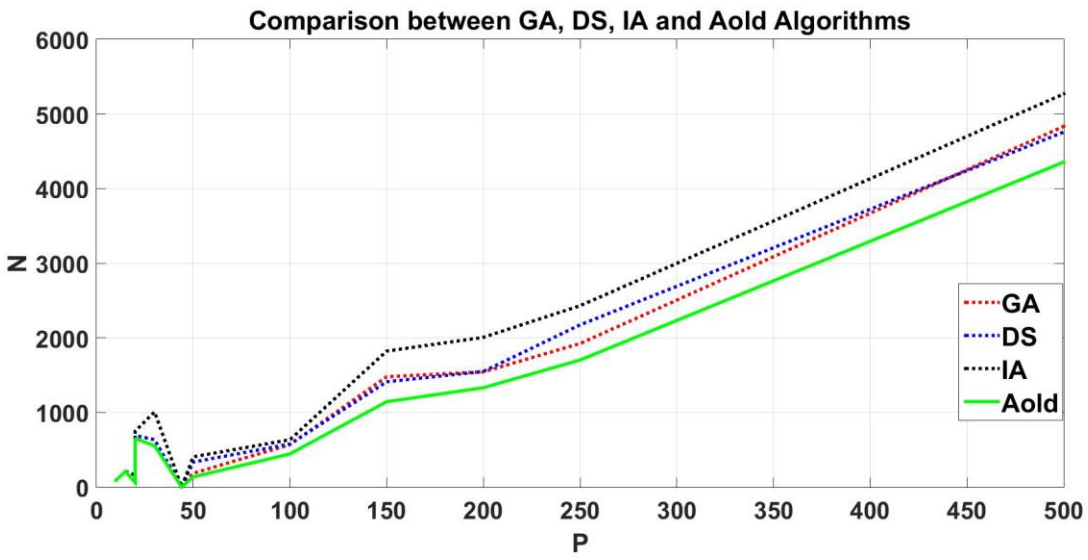


Figure 5.2 Comparison results among GA , DS , IA and Pinol - Beasley algorithms.

6.1 Conclusion

In this dissertation, we have provided a formal definition of both the MSP and ALP issues that we studied it. It is possible to derive various conclusions about general, MSP, and ALP difficulties from this dissertation.

1- Some NP-hard MSP and ALP issues can be solved for tolerable sizes by the certain rules, and these rules may be acquired directly from the information of the problem.

2- Because it may be modeled as a task MSP with release times and sequence that vary with processing time, the ALP is hard to solve. Since it has been shown that is NP-hard, it follows that.

6.1.1 *Machine Scheduling Problem (MSP):*

This dissertation investigates the FP issue of multi-objective bi-objective flow shop scheduling (BOFSS) and the PMS problem of multi-objective bi-objective scheduling of parallel machines. The goal is to create a number of methods, both accurate and approximate, to address these problems. F3(Cmax,RL) (FP) to discover optimal, approximate, and precise solutions, and P||($\sum_j C_j$, ET_{max}) problem.

6.1.2 A*ircraft Landing Problem (ALP):*

The purpose of this dissertation is to explore the issue of landing aircraft (ALPs) at two-runway aerodromes in an effort to lessen the penalty cost of deviating from the standard landing time (SLT) for all aircraft and the target landing time (TLT). We make an effort to find ALP on the steady-state plane of the binary runway, where P is less than 500.

6.2 *Future Works*

Several potential follow-up projects are recommended by this dissertation to address the aforementioned issues with the studied cases.

6.2.1 *Machine Scheduling Problem* :

Finding close OSs in an acceptable amount of time is one technique to cope with NP-hard MSP. The following issues need algorithms capable of finding such answers.

- $P2||(C_{\max}, ET_{max})$.
- $F2||(T_{max}, R_L)$.

6.2.2 Aircraft Landing Problem

1- In this field of ALP for $BOALP$, further investigation and study are required. An intriguing possibility would be to employ meta-heuristic techniques to refine the results of the algorithms suggested in this thesis, bringing them closer to OSs.

2- We have to study plane landings in the dynamic state, particularly in challenging conditions like dense fog, torrential rain, etc.

REFERENCES

[1] Nagar, A., Haddock, J., and Heragu, S.S., (1995), Multiple and Bicriteria Scheduling: A Literature Survey, European Journal of Operational Research, 81:88–104.

[2] Hoogeveen, H., (2005), Invited Review Multicriteria Scheduling,European Journal of Operational Research, 167:592–623.

[3] Sidney, J.B., (1977), Optimal Single Machine Scheduling with Earlinessand Tardiness Penalties, Operational Research, 25:62–69.

[4] Li, C.L., and Cheng, T.C.E. (1994), The Parallel Machine Min–Max Weighted Absolute Lateness Scheduling Problem, Naval Research Logistics, 41:33–46.

[5] Mosheiov, G., and Oron, A., (2007), Minmax Scheduling with Job– Classes and Earliness–Tardiness Costs, European Journal of Operational Research, 177:612–622.

[6] Tavakkoli-Moghaddam, R., Moslehi, G., Vasei, M., and Azaron, A., (2005), Optimal Scheduling for a Single Machine to Minimize the Sum of Maximum Earliness and Tardiness Considering Idle Insert, Applied Mathematics and Computation, 167:1430–1450.

[7] Tavakkoli-Moghaddam, R., Moslehi, G., Vasei, M., and Azaron, A., (2006), A Branch-and-Bound Algorithm for a Single Machine Sequencing to Minimize the Sum of Maximum Earliness and Tardiness with Idle Insert,Applied Mathematics and Computation, 17:388–408.

[8] Tapan, S., Farhad, M.E., and Parthasarati, D., (1988), A Branch and Bound Approach to the Bicriteria Scheduling Problem Involving Total Flowtime and Range of Lateness, Management Science, 34(2):254–260.

REFERENCES

[9] Liao, C., and Huang, G., (1991), An Algorithm for Minimizing the Range of Lateness on a Single Machine, Journal of the Operational Research Society, 42(2):183–186.

[10] Geiger, M.J., (2004), Randomized Variable Neighborhood Search for Multi-Objective Optimization, 4th EU/ME: Design and Evaluation of Advanced Hybrid Meta–Heuristic, 34–42.

[11] Arroyo, J.E.C., Ottoni, R.S., and Oliveira, A.P., (2011), Multi-Objective Variable Neighborhood Search Algorithms for a Single Machine Scheduling Problem with Distinct Due Windows, Electronic Notes in Theoretical Computer Science, 281:5–19.

[12] Baker, K.R., (1974), Introduction to Sequencing and Scheduling, Wiley, New York.

[13] French, S., (1982), Sequencing and Scheduling an Introduction to the Mathematics of the Job Shop, Ellis Horwood Series in Mathematics & Its Applications, John Wiley and Sons, New York.

[14] Johnson, S.M., (1954), Optimal Two and Three Stage Production Schedule with Set-Up Time Included, Naval Research Logistics Quarterly, 1:61–68.

[15] Blum, C., and Roli, A., (2003), Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, ACM Computing Surveys, 35(3):268–308.

[16] Bertrand, M.T., (2001), Scheduling in the Two Machine Flow Shop with Due Date Constraint, Introduction Journal of Production Economics, 70: 117–123.

[17] Tapan, S., and Dilpen, P., (1991), Job Lateness in a Two Machine Flow Shop with Setup Time Separated, Computers and Operations Research, 18:549–556.

[18] Ladhari, T., and Hauari, M., (2000), Minimizing Maximum Lateness in aTwo Machine Flow Shop, Journal of the Operational Research Society, 51:1100–1106.

[19] Enbin Liu, Yong Peng, Yongqiang Ji, Mohammadamin Azimi, and Laimin Shi (2000), Energy Consumption Optimization Model of Large Parallel Natural Gas Pipeline Network: Using Compressors with Multiple Operating Modes.

REFERENCES

[20] Idris, H.R., Delcaire, B., Anagnostakis, I., Hall, W.D., Pujet, N., Feron, E., Hansman, R.J., Clarke, J.-P., and Odoni, A.R., (1998), Identification of Flow Constraint and Control Points in Departure Operations at Airport Systems, American Institute of Aeronautics and Astronautics, Guidance, Navigation and Control Conf., Boston, AIAA-1998–4291, American Institute of Aeronautics and Astronautics, Reston, VA.

[21] Idris, H.R., Delcaire, B., Anagnostakis, I., Hall, W.D., Pujet, N., Feron, E., Hansman, R.J., Clarke, J.-P., and Odoni, A.R., (1999), Observations of Departure Processes at Logan Airport to Support the Development of Departure Planning Tools, Air Traffic Control Quarterly, 7(4):229–257.

[22] Blumstein, A., (1959), The Landing Capacity of a Runway, Operations Research, 7(6):752–763.

[23] Balakrishnan, H., and Chandran, B., (2007), Efficient and Equitable Departure Scheduling in Real-Time: New Approaches to Old Problems, the Seventh USA/Europe Air Traffic Management Research and Development Seminar, Barcelona, Spain.

[24] Ernst, A.T., Krishnamoorthy, M., and Storer, R.H., (1999), Heuristicand Exact Algorithms for Scheduling Aircraft Landings, Networks, 34(3): 229–241.

[25] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y.M., and Abramson, D., (2000), Scheduling Aircraft Landings – the Static Case, Transportation Science, 34(2):180–197.

[26] Beasley, J.E., Krishnamoorthy, M., Sharaiha, Y. M., and Abramson, D., (2004), Displacement Problem and Dynamically Scheduling Aircraft Landings, Journal of the Operational Research Society, 55(1):54–64.

[27] Pinol, H., and Beasley, J.E., (2006), Scatter Search and Bionomic Algorithms for the Aircraft Landing Problem, European Journal of Operational Research, 171(2):439–462.

[28] Amir Salehipour ; An algorithm for single-and multiple-runway aircraft landing problem;MathemticComputersSimulation,Volume175,Semtemper2020,

P:179-191.

[29] Pinedo, M.L., (2012), Scheduling Theory, Algorithms, and Systems, Springer Science + Business Media, Fourth Edition, LLC., New York.

[30] Hoogeveen, H., (2005), Invited Review Multicriteria Scheduling,European Journal of Operational Research, 167:592–623.

[31]David, K., Cliff, S., and Joel, W., (1997), Scheduling Algorithms, CRC Handbook of Computer Science.

[32]Dear, R.G., and Sherif, Y.S., (1991), An Algorithm for Computer Assisted Sequencing and Scheduling of Terminal Area Operations, Transportation Research (Part A), Policy and Practice, 25:129–139.

[33] T'kindt, V., Billaut, J.-C., (2006), Multicriteria Scheduling: Theory,Models and Algorithms, Second Edition, Springer, Berlin.

[34] Hoogeveen, J.A., (1996), Single Machine Scheduling to Minimize aFunction of Two or Three Maximum Cost Criteria, Journal of Algorithms, 21:415–433.

[35] Hoogeveen, J.A., (1996), Single Machine Scheduling to Minimize aFunction of Two or Three Maximum Cost Criteria, Journal of Algorithms, 21:415–433.

[36] Smith, W.E., (1956), Various Optimizers for Single-Stage Production, Navel Research Logistics Quarterly, 3:59–66.

[37] Jackson, J.R., (1955), Scheduling a Production Line to Minimize Maximum Tardiness, Research Report 43, Management Science Research Project, University of California, Los Angeles.

[38] Alharkan I. M., "Algorithms for Sequencing and Scheduling", Industrial Engineering Department, College of Engineering, King Saud University, Riyadh, Saudi Arabia, (2007).

[39] Bencheikh, G., El Khoukhi, F., Baccouche, M., Boudebous, D., Belkadi,A., and Ouahman, A.A., (2013), Hybrid Algorithms for the Multiple Runway Aircraft Landing Problem, International Journal of Computer Science and Applications, 10(2):53 – 71.

REFERENCES

[40] Potts, C.N., Mesgarpour, M. and Bennel, J.A. (2009), A Review ofAirport Runway Optimization, University of Southampton.

[41] Veidal, E., (2007), Scheduling Aircraft Landings–the Dynamic Case. Master's thesis, Department of Informatics and Mathematical Modelling, Technical University of Denmark.

[42] Puchinger, J., Raidl, G.R., (2005), Combining Metaheuristics and Exact Algorithms on Combinatorial: A Survey and Classification. In: Mira, J., Álvarez, J.R. (eds.) IWINAC 2005, LNCS, 3562:41–53, Springer,Heidelberg.

[43] Hansen, P., and Mladenović, N., (2003), Variable Neighborhood Search, Handbook of Metaheuristics, Glover F. and Kochenberger G.A. (Ed.), International Series in Operation Research and Management Science, 57:145–184.

[44] Hussam, A.A.M., (2017), Exact and Heuristic Algorithms for Solving Combinatorial OptimizationProblems ,Ph.D. Thesis Dept.of Mathematics, College of Sciences, AL-Mustansiriayah University.

[45] Glover F., (1989), Tabu Search Part I, ORSA Journal on Computing, 1: 190–206.

[46] Hoogeveen, H., (2005), Invited Review Multicriteria Scheduling,European Journal of Operational Research, 167:592–623.

[47] Holland, J.H., Adaptation in Natural and Artificial Systems, Cambridge, MA: MIT Press. Second edition (1992), (First edition, University of Michigan Press, 1975), 1975/1992.

[48] Murata T., Ishibuchi H. and Tanaka H., (1996): Multi-objective genetic algorithm and its
applications to flowshop scheduling. Computers and Industrial Engineering 30:957–968.

[49] Bencheikh, G., Boukachour, J., and El Hilali Alaoui, A., (2011), Improved

REFERENCES

Ant Colony Algorithm to Solve the Aircraft Landing Problem, International Journal of Computer Theory and Engineering, 3(2):224–233.

[50] Abdul-Razaq, T.S., and Ali, F.H., (2015), Hybrid Bees Algorithm to Solve Aircraft Landing Problem, Journal of Zankoi Sulaimani (Part A), 17(1):71–89.

[51] Dear, R.G., and Sherif, Y.S., (1991), An Algorithm for Computer Assisted Sequencing and Scheduling of Terminal Area Operations, Transportation Research (Part A), Policy and Practice, 25:129–139.

[52] Intensification Algorithm Hejazi, S.R., and Saghafian, S., (2005), Flowshop-Scheduling Problems with Makespan Criterion: A Review, International Journal of Production Research, 43(14):2895–2929.

REFERENCES

الخلاصة:

تتضمن مشاكل التحسين التوافقي (COPs)، التي لها عدد محدود من الحلول الممكنة (FS)، آلات الجدولة وهبوط الطائرات. يعد التحقق من جميع FSs في منطقة البحث هو أفضل أسلوب لحل مشكلة COP. ليس من العملي دائمًا التحقق من كل FS، خاصة إذا كانت منطقة البحث واسعة النطاق. ولمعالجة هذه المشكلات، تم تطوير وتحسين خوارزميات البحث المحلية أو الاستدلال الفوقي. تعد مشكلة جدولة الآلة (MSP) ومشكلة هبوط الطائرة (ALP) مسألتين للتحسين تمت دراستهما في هذه الأطروحة، إلى جانب استخدام كل من استراتيجيات التحسين الدقيقة والإرشادية لحلهما. تمت دراسة متجر التدفق التناسبي (PFS) والآلات المتوازية المتطابقة (IPM) في مشكلة جدولة الآلة (MSP) وتقليل كلفة التبكير والتاخير في هبوط الطائرات (ALP) في هذه الاطروحة.

تم إجراء كل هذه الحسابات باستخدام طرق دقيقة مثل طريقة التعداد الكامل (CEM) والفرع والحدود (BAB). كما تم استخدام الأساليب الفوقية الإرشادية، مثل الخوارزمية الجينية (GA)، والخوارزمية الميميتية(MA) وغيرها.

في مسائل هبوط الطائرات (ALP)، نصف العديد من الخوارزميات متعددة الأهداف لـ BOALP التي تحسب كلاً من أوقات الهبوط الإجمالية المجدولة (SLTs) لجميع الطائرات وإجمالي تكلفة عقوبة الاختلاف بين SLT لكل طائرة ووقت الهبوط المستهدف (TLT)).

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة بابل

كلية التربية للعلوم الصرفة

قسم الرياضيات

الطرق المضبوطة والتقريبية والفوق تنقيبية لحل مسائل التحسين التوافقي

اطروحة

مقدمة الى مجلس كلية التربية للعلوم الصرفة / جامعة بابل كجزء من متطلبات نيل درجة الدكتوراه في التربية / الرياضيات

من قبل

عادل هاشم نوري جاسم

بإشراف

أ. د. كريمة عبد الكاظم

أ.م.د. حسام عبد علي محمد

1445 هـ

2023 م