Republic of Iraq

Ministry of Higher Education

and Scientific Research

University of Diyala

College of Science

Department of Mathematics

Solution Algorithms for Multicriteria optimization problem

A Thesis

Submitted to the College of Science, University of Diyala in Partial Fulfillment of the Requirements for the Master Degree of Science in Mathematics

By

Anmar Sabri Hasan

B.Sc. **Mathematics** / College of Science / University of Mustansiriyah, 2004

Supervised by

Assist.Prof.Dr. Adawyia Ali Mahmood Al-Nuaimi

2021 A.D. **1442 A.H.**

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

يَرۡفَعِ ٱللَّهُ ٱلَّذِينَ ءَامَنُواْ مِنكُمۡ
وَٱلَّذِينَ أُوتُواْ ٱلۡعِلۡمَ دَرَجَٰتٖۚ وَٱللَّهُ بِمَا تَعۡمَلُونَ
خَبِيرٞ ۞

صَدَّقَ الله العظيم

سورة المجادلة- الأية (١١)

Dedication

To the soul of my martyr father... May God have mercy on him and forgive him and put him in his spacious gardens.

To my mother... the tender symbol of love and giving.

To my dear wife... who bore me the trouble of studying.

To my brothers and sisters... in appreciation and respect.

To my children a source of love and tenderness.

To everyone who contributed to the completion of this work, even if it were complete.

I dedicate the fruit of this work.

Anmar

Acknowledgements

To begin with, I would like to thank God for giving me this chance, and giving me the strength, and the patience, to go through with this thesis.

My deepest gratitude and faithful thanks also go to my supervisor, **Dr. Adawyia Ali Mahmood** for the support, encouragement, continuous guidance throughout my work; words can never express my thanks.

Also, I would like to thank the **Dean**, the **Head of the Department of mathematics** and all the faculty of the Department of Mathematics at the College of Science at Diyala University for being helpful with me.

All my fellow **graduate students** thank you

Finally, this thesis become a reality with the help of many individuals. I would like to sincere thanks to all of them.

Anmar

(Scientific Amendment)

I certify that thethesisentitled**“SolutionAlgorithmsforMulticriteria optimization problem”**was prepared by **Anmar Sabri Hasan** has been evaluated scientifically ؛therefore, it is suitable for debate by examining committee.

.

Signature:

Name:

Date: / / 2021

(Scientific Amendment)

I certify that the thesis entitled "**Solution Algorithms for Multicriteria optimization problem**" was prepared by **Anmar Sabri Hasan** has been evaluated scientifically ؛therefore, it is suitable for debate by examining committee.

Signature:

Name:

Date: / / 2021

Abstract

In this thesis, multicriteria scheduling problem on a single machine is considered. The three criteria are total completion time $\sum C_j$, total tardiness $\sum T_j$ and maximum earliness E_{max}.

We aim in this study to find optimal and approximate schedule for the jobs j, j=1,2,…,n to minimize the multicriteria objective function $\sum C_j + \sum T_j + E_{max}$.

For solving this problem, we present a branch and bound (BAB) algorithm to find optimal solution. We derived a lower bound (LB) to be used in a branch and bound (BAB) algorithm. On a vast collection of test problem, computational experiments for the BAB algorithm are provided. The NP-hardiness of this problem demonstrate that finding an optimal solution immediately is not always achievable and the problem is solved for n =13. As a result, rather than spending a lot of time searching for the best solution, we employ local search algorithms to uncover approximation answers that are close to the best but take less time. The problem is solved using three local search algorithms: descent method (DM) , simulated annealing (SA) and Genetic algorithm (GA). The algorithms DM , SA and GA are compared with the BAB algorithm in order to evaluate effectiveness of the solution methods. Conclusions are formulated on the efficiency of the algorithms, based on findings of computational experiments.

List of symbols and abbreviation

Symbol	Description
BAB	Branch and Bound
$\alpha/\beta/\gamma$	Classification of problem
d_i	Due date of job i
DP	Dynamic programming
EDD	Earliest Due Date
ILB	Initial Lower bound
i	Job i
β	Jobs characteristics
Lex	Lexicographical
LB	Lower bound
α	Machine environment
C_{max}	Maximum completion time or makespan
E_{max}	Maximum earliness
L_{max}	Maximum lateness
T_{max}	Maximum tardiness
MST	Minimum slack time rule
NP	Non-deterministic Polynomial time
n	Number of jobs
pmtn	Preemption
p_i	Processing time of job i
r_i	Release date of job i
SPT	Shortest processing time
SWPT	Shortest weighted processing time
DM	Descent Method
SA	Simulated Annealing

TS	Tabu Search
GA	Genetic algorithm
s_i	Slack time of job i
C_i	The completion time of job i
F_i	The flow time of job i
L_i	The lateness of job i
f_{max}	The maximum function
γ	The objective function
$\sum C_i$	The sum of completion time
$\sum T_i$	The total tardiness
U_i	The unit penalty of job i
w_i	The weighting of job i
$\sum w_i C_i$	Total weighted completion time
$\sum w_i U_i$	Total weighted number of tardy jobs
UB	Upper bound

List of contents

Introduction

Introduction

The scheduling problem is one of the most studied problems in combinatorial optimization. It can be defined as a decision making process that is used on a regular basis in many manufacturing and services industries. It is concerned with allocating resources to tasks over certain time periods with the goal of minimizing one or more objectives [1]. The scheduling problem is defined as the challenge of assigning a group of machines in a given amount of time while adhering to certain constraints [2].

Real-world problems arising in various applications domains are usually strictly related to time [2].

Due dates or deadlines are commonly used in scheduling theory to simulate time constraints, and the quality of schedules is calculated using these factors [2]. Minimize or maximize $F(s) = (f_1(s), f_2(s),...,f_k(s)$ for multicriteria (multiple objective) scheduling problems, such that $s \in S$, where s stands for the solution, S is the set of feasible solutions, k is the number of objectives in the problem, and F(s) is the image of S in the k-objective scheduling problem. The goal of many issues is to determine the best arrangement of a group of discrete items that meets additional requirements and limitations. If the problem has many objectives, numerous criteria exist to assess the quality of the solution, and each of these criteria has an objective (min. or max.) linked to it [3].

There are three types of Multicriteria problems. The first type of these problems consists of identifying all sequences that minimize the first objective. The optimal sequence for that task is picked from among those that minimize a second objective. The hierarchical approach [3] is the name given to this method. The second of these Multicriteria problems, when the criteria are weighted differently

an objective function can be defined as the sum of weighted functions and transform the problem into a single criterion scheduling problem. This approach is called simultaneous optimization along with the third type of Multicriteria problems. The third one of these multicriteria problems is going to consider both criteria as equally important. This problem is to find a sequence that does well on both objectives. Note that this optimization is called a priority optimization, is clearly the most difficult variant of the three approaches; if we can solve this problem (finding the set of pareto optimal points), then we will solve the other two as well.

For a single machine scheduling with multicriteria, Emmons [4] addressed the hierarchical problem of minimizing $\sum_{j=1}^{n} C_j$ based on the constraint that f_{max} is minimal; this problem is denoted by $1/f_{max} \leq f^*/ \sum_{j=1}^{n} C_{j,}$ where f^* signifies the optimal solution value of the $1//f_{max}$ problem. The $1//f_{max}$ Lawler's algorithm solves the issue in $O(n^2)$ time algorithm [5].

The importance of multicriteria scheduling has been recognized in [6]. Hoogeveen [7] studied a number of bi-criteria scheduling problems, he proved strong NP-hardness of bi-criteria problems involving $\sum w_j C_j$ and L_{max}. Hoogeveen surveyed the most notable results on multicriteria scheduling [8]. Nagar et al. [9] provided a questionnaire of the multiple and bi-criteria scheduling research involving multiple machines. Van Wassenhove and Gelders [10] proposed a pseudo-polynomail algorithm for finding all efficient schedules with respect to $\sum C_j$ and T_{max} . Al-Nuaimi [11] proposed an algorithm to find efficient solutions for multicriteria scheduling problem of total completion time $\sum C_j$ with maximum late work ($V_{max)}$ and maximum lateness (L_{max}) on a single machine. In [12] Al-Nuaimi presented some methods to identify the most precise and best possible solutions to

the three-criteria problem maximum lateness L_{max} , maximum earliness E_{max} and sum of completion time $\sum C_j$ in hierarchical case. Also, Al-Nuaimi [13] proposed an algorithm to solve the problem $1//F(\sum C_j, \sum T_j, L_{max})$ to find the set of efficient solutions. Local search method proved that the led to significant better results than traditional heuristics if they are implemented carefully. Within a reasonable amount of time, local search algorithms can identify the best approximation solution [2].

In this thesis, we look at how to schedule n jobs on a single machine while keeping the total cost low completion time ($\sum C_j$) with total tardiness ($\sum T_j$) and maximum earliness (E_{max}).

This thesis is organized as follows:

Chapter one gives a description of machine scheduling problem, including the assumption for machines, jobs and optimality criteria. Classification and representation of scheduling problem are also mentioned. Chapter two considers basic concepts of multicriteria scheduling optimization and local search methods with some definitions and considers some models studied of multicriteria scheduling problems. In chapter three, we present the mathematical form for the simultaneous multicriteria problem, which is recognized NP-hard, we derive a good lower bound based on objective decomposition, in order to design a branch and bound the problem-solving algorithm. We also present computational experiments for the exact solution and local search algorithms in chapter three.

Chapter One

Scheduling Problem

1.1 Scheduling Problem (Definition and Classification)

1.1.1 Machine Scheduling

There has been a large number of researches on production scheduling problems since the original of mathematical formulations typically, this entails assigning machines to process tasks (jobs) over time in order to refine certain output parameters, either precisely or roughly.

The literature can be divided into two major categories:

a- Deterministic scheduling research: where all problem parameters are considered to be well-known.
b- Stochastic scheduling research: where at least some parameters are random variables.

In deterministic scheduling research a large view is taken and multiple machines are often modeled. The deterministic approach is to plan the work through the machines over a period of time in the best possible way, given a specific objective to optimize. The implicit assumption here is often that a schedule can be executed directly as developed.

However, several scholars have recently recognized that this unlikely scenario exists in many manufacturing settings, and have attempted to apply the deterministic approach to circumstances involving some complexity [14].

1.1.2 Basic Scheduling Concepts

We begin by adding some key notations, focusing on performance parameters without going into details about system environments, etc. We assume that there are n jobs, denoted by 1,...,n, that must be scheduled on a set of machines that are always available from time zero onwards and can only handle one job at a time.

We only state the notation used in this study for the single machine, jobs j (j=1,...,n):

p_j: This means the processing time.

r_j: A release date of job j, i.e. the earlier time of which the p_j of job begin.

d_j: This means the due date.

W_j: This means the weighted.

We can now compute for job j for a given sequence of jobs:

1- The completion time C_j.
2- The flow time $F_j=C_j-r_j$.
3- The lateness $L_j= C_j-d_j$.
4- The tardiness $T_j=\max\{ C_j -d_j,0\}$.
5- The earliness $E_j=\max\{ d_j-C_j,0\}$.
6- The unit penalty $U_j=1$ if $C_j> d_j$ and $U_j = 0$ if $C_j\leq d_j$.

The following performance criteria appear frequently in the literature [15].

For a given scheduling δ we compute:

1- $C_{max}(\delta) = \max_j (C_j)$ (maximum completion time).
2- $E_{max}(\delta) = \max_j (E_j)$ (maximum earliness).
3- $L_{max}(\delta) = \max_j (L_j)$ (maximum lateness).
4- $T_{max}(\delta) =\max_j (T_j)$ (maximum tardiness).
5- $\sum (W_j) C_j (\delta)$ = total (weighted) completion time.
6- $\sum (W_j) E_j (\delta)$ = total (weighted) earliness.
7- $\sum (W_j) T_j (\delta)$ = total (weighted) tardiness.
8- $\sum (W_j) U_j (\delta)$ = total (weighted) number of tardy jobs.

All these criteria except for E_{max} and $(\sum W_j) E_j$ are regular, i.e., the value of the objective function can not be decreased by inserting idle time into the schedule.

1.1.3 The problem classification

A notation which is commonly used to formulate scheduling problem is based on three fields: **α/β/γ** [7]. In this notation, α describes the machine environment, i.e. the structure of the,

-Single machine or multiple machines.

- Machines that are the same or that are different.

The field β explains the problem's constraints as well as other processing conditions. Among the constraints that can exist [14], [16]:

-Preemption allowed or not, i.e. whether the processing of jobs can be

Interrupted and resumed.

-Whether special processing conditions (release date, due date, setup times,

etc,) specified or not, and if these are not and if these are deterministic or

stochastic.

-Fixed or dynamic arrival of jobs, etc.

The criteria (**γ**) used to evaluate the equality of the scheduling include:

-Minimum completion time or make span C_{max}.

-Maximum earliness $E_{max} = \max (E_j)$ for j=1,…,n.

-Maximum tardiness $T_{max} = \max (T_j)$ for j = 1,…,n, etc.

1.1.4 A few examples of scheduling problems

We give a few examples on three fields classification of scheduling problems. The $1/r_i/\sum W_i\ C_i$ is the problem of minimizing total weighted completion time on one machine subject to non- trivial release dates. The $1//\sum W_i\ C_i+T_{max}$ is the problem of determining the best order for jobs to be processed on a single machine in order to minimize the amount of the overall weighted completion period and the maximum tardiness.

1.1.5 Some Types of Machine Scheduling Problems

Due to the large variety of machine scheduling problems, several classification schemes have been proposed based on different dimensions [17]. The number of available machines and how they are arranged (see figure (1-1)) is example of such dimension. The simplest problem is the one-machine sequencing problem: all jobs must be processed on a single machine, and no two jobs can be processed at the same time. There are four types of scheduling issues in a work store [5]:

a-Scheduling a single machine [5]

The single machine scheduling problem entails allocating a single resource to a collection of jobs. This is achieved by creating a series that involves each job and assigning the jobs to their respective sources. Each job should have a priority, a ready time, a processing time, and a due date assigned to it. On the basis of this data and the job series, the value of the performance measure can be computed. This problem grows in complexity at an exponential rate as the number of jobs to be scheduling increases.

b-Flow Shop Scheduling [5]

In each of the m machines, there are n jobs to process, i.e., each job consists of m steps or operations. Each job is processed in the same order through the processing stages, i.e., from the first to the last machine. The problem is to find the sequence in which the jobs should be processed so that the given objectives are achieved.

c-Job Shop Scheduling [5]

This is a general case of the flow shop scheduling problem, in which each job is not necessarily sequenced through the machines in the same way. As in a flow shop, there are n jobs with m operations each of which are predefined and fixed; for example, $J_m/d_j/C_{max}$ denotes a work shop configuration in which all jobs have a due date and the aim is to minimize the maximum completion time.

d-Scheduling of Open Shops [5]

The open shop is a more general case of the job shop scheduling problem as before, there are n jobs consisting of m steps to be processed in m machines. Each work is sequenced differently across the machine, and determining the best sequences for each of the n jobs is part of the problem. And, in addition to the work processing schedule, the sequence of steps for each job must be decided.

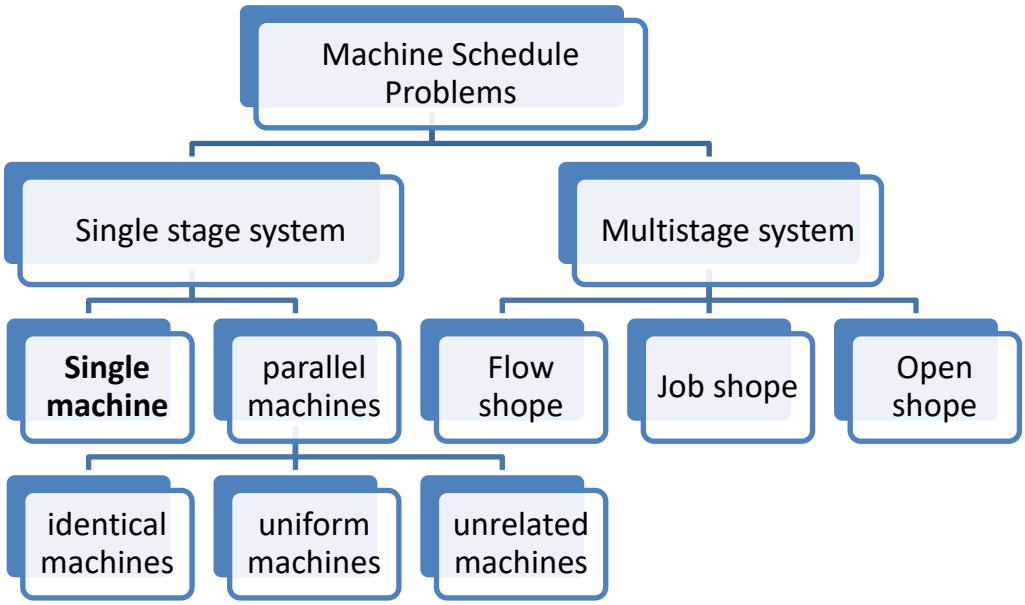


Figure 1-1: Classification of machine Schedule Problems [17]

Definition: P- type problem (1.1.6)

The problems for which the polynomial bounded algorithm for solving this problem, were found.

Definition: NP- type problem (1.1.7)

The problem that haven't a polynomial bounded algorithm to solve this problem.

1.2 Solution Approaches

1.2.1 Basic Rules and Main Results to Find Optimal Solution for p-Type Machine Scheduling Problem

The common basic rules which help us in finding a solution for scheduling problem:

1- Smith rule or **SPT** (shortest processing time) rule, that is, sequencing the jobs are listed in non-descending order of processing time. This rule solves the problem $1//\sum_j C_j$ [18] more general is the **SWPT** rule, that is sequencing the

jobs in non-decreasing order of their processing time to the weight ratio which solves the problem $1//W_jC_j$.

2- The earliest due date or the **EDD** rule, which solves the problem $1//L_{max}$ by sequencing the jobs in non-decreasing order of their due dates. For the $1//T_{max}$ query, this rule also minimizes T_{max} [19].

3- The longest processing time or the **LPT** rule, i.e., sequencing the jobs in a non-increasing order of processing time [20], which minimize $\sum_j Ej$ for the problem $1/C_j \leq d_j/ \sum_j E_j$.

4- The minimum slack time or **MST** rule, that is, sequencing the jobs in non-decreasing order of their slack times ($s_j=d_j-p_j$). In a single machine environment with ready time set at zero, which solves $1//E_{max}$ problem [6].

1.2.2 Mathematical Programming to Solve NP-hard Machine Scheduling Problems

There are some mathematical programming techniques used to solve the combinatorial optimization problem, these techniques are also used for scheduling problems [21].

Many scheduling problems can be formulated as a (mixed) programming problem; in that case, standard integer programming solution procedure can be used. Both Conway et al. [22], and Baker and Schrage [23] discussed integer-programming formulation of scheduling problems.

Complete enumeration method generates schedules one by one, searching for an optimal solution. This method lists all possible schedules and then eliminates the non-optimal schedules from the list, leaving those, which are optimal. Clearly searching for an optimal schedule among all possible schedules using complete enumeration is not suitable even for problems of small size.

The dynamic programming (DP) method is an implicit enumeration technique that can be used to solve any optimization problem that can be solved by solving the derived recurrence relation for this problem [21].There are some difficulties for this method, one of them is the difficulty of finding a good way for brake down problem into stages so that a convenient computation is rather large, which means that the computation grows to exponential rate with increasing in the size of problem.

Branch And Bound (BAB) method is a general method for solving many types of combinatorial optimization problem. BAB method is the most wildly solution technique that is used in scheduling [24]. This method is the typical example of the implicit enumeration approach, which can find an optimal solution by systematically examining subset of feasible solution. The procedure is usually described by means of search tree with nodes that correspond to these subset. From each node for a partially complete solution there grows a number of new branches which replaces the original one by set of new smaller problems that are mutually exclusive. There are two forms of branching that are commonly used:

1- The forward branching, that is the jobs are sequenced one by one from the beginning.
2- The backward branching, that is, the jobs are sequenced one by one from the end.

To minimize an objective function Z, for a particular scheduling problem, the BAB method successively partitions the problem into subsets by using a branching procedure and computes bound by using a lower bounding procedure. These methods are used to exclude subsets that are found to be devoid of any optimal solution. This eventually leads to at least one optimal solution. The lower bound (LB) on the solution to each created sub problem is calculated using the bounding method. For each node we calculate a (LB) which is the cost of the

scheduling jobs (depending on the objective function and the cost of the unscheduled jobs (depending on the derived lower bound)). If this node has a value (LB) greater than or equal to the upper bound (UB) then this node is can called the upper bound is usually defined as the minimum of the values of all feasible solutions currently found. If branching reaches a full sequence of jobs, that sequence is evaluated, and if its value is less than the current upper bound (UB), this (UB) is reset to that value. We repeat the procedure until all nodes have been considered, that is, LB≥UB for all nodes in the search tree. An optimal solution for this problem is a feasible solution with this (LB).

In BAB procedure one can introduce dominance rules (if possible) to specify whether a node can be eliminated before computing its (LB) which reduce the computation time by ignoring the calculations of the dominated nodes and their successors.

1.2.3 The Heuristic Method

It's evident (from the previous section) that utilizing mathematical programming algorithms to tackle a particular problem could take a lot longer than it usually does for large problems. Indeed, even for a minor issue, there is no guarantee that a solution will be found immediately. Sometimes we use a heuristic scheduling instead of the optimal schedule, that is, we can find near optimal solution. Reeves [25] the heuristic technique is defined as follows: A heuristic which seek good (i.e., near optimal) solution at a reasonable computational cost without being able to guarantee either feasibility or optimality, or even, in many case to state how close to optimality a particular feasible solution.

1.3 Multicriteria Scheduling

Many researchers have been working on multiple criteria scheduling with the majority of work being on bi-criteria scheduling. When two criteria are used instead of one, the problem becomes more realistic. One criterion can be chosen to represent the manufacturer's concern while the other could represent consumer's concern. Several studies evaluate the literature on multiple criteria scheduling. Nager et al.(1995) [9], and Tkindt and Billaut (1999) [26] review a special version of the problem, Lee and Vairaktarakis (1993) [27] review a particular variant of the problem in which one criterion is fixed to its greatest feasible value and the other criterion is attempted to be optimized under this restriction.

Hoogeveen (1992) [7] studied a number of bi-criteria scheduling problems. Most real-world optimization problems have several, often conflicting objectives. Therefore, the optimum for a multiobjective problem is typically not a single solution. It is a set of solutions that trade-off between objectives.

There are three different sorts of problems with many criteria that can be detected:

The first of these problems entails determining all sequences that minimize a first objective. The optimal sequence for that task is picked from among those that minimize a second objective. Assume we've decided on two performance criteria to consider, say f and g. If $\boldsymbol{f}$ is more important than $\boldsymbol{g}$, then the approach is to find the optimum value with respect to criterion $\boldsymbol{f}$, say $\boldsymbol{f}^*$, and choose from among the set of optimum schedules for $\boldsymbol{f}$ the one that performs best on $\boldsymbol{g}$, such an **approach** is called **hierarchical** optimization or **lexicographical** optimization, which is denoted by Lex ($\boldsymbol{f}$,$\boldsymbol{g}$) in the third field of the $\boldsymbol{\alpha/\beta/\gamma}$ notation scheme. This method is known as a hierarchical approach.

When the criteria are weighted differently, ***the second*** of these multiple criteria problems is an objective function that can be expressed as the sum of weighted functions and turns the difficulties into a single criterion scheduling problem. Simultaneous optimization is the name given to this method, as well as the ***third type*** of multiple criteria issues. Also in these multiple criteria problems, both criteria are going to be considered as equally important. The issue is determining a sequence that achieves both goals. To solve this problem, the main concept is that we select a subset of solutions from a larger set that contains efficient solutions.

Theorem (1.3.1) [8]

There is an extreme schedule that minimizes F (f,g) if the composite objective function F (f,g) is linear.

Chapter Two

Multi Criteria Scheduling Problem

2 Multicriteria Scheduling problem

2.1 Basic Concept of Multicriteria Scheduling

For several years, scheduling researchers focused on single, non-decreasing work completion period performance metrics. More than one efficiency metric is of concern in most real-world scheduling applications [28]. The three types of bi-criteria scheduling problems are as follows. Assume that we've chosen two success criteria to consider, say f and g [29].

1. In the event that output criterion, say f, is significantly more significant than the other, an easy strategy is to identify the optimal value with regard to criterion f, which is indicated by f*, and pick the one that performs the best on g from the set of optimum schedules for f, (hierarchical or lexicographical optimization). This is represented by Lex (f, g), where the criterion stated first in Lex's argument is the most relevant. The primary criterion is f, and the secondary criterion is g. The earliest work in this field is Smith [18]. Work on minimizing overall completion time with no tardy jobs.

Definition (2.1.1) [18]:

A feasible schedule for the problem 1//Lex (f, g) is one that meets the fundamental requirement f.

Definition (2.1.2) [18]:

An optimal schedule for 1//Lex (f,g) is a feasible schedule that minimizes the secondary criterion g.

If no criterion is dominant, then lexicographical optimization may lead to a schedule that is unbalanced. In this case, simultaneous optimization may be

better choice. Evans [3] and Fry [6] distinguish different approaches in simultaneous optimization.

2. Apriority optimization: For any given function F, both criteria are used to form a single composite objective function **F (f (σ), g (σ)),** where σ is the timetable taking into account, and an optimal solution to this problem is found. This function F, like δ f (σ) +g (σ), can be linear.

Where δ is that constant which shows the in relation to the significance of criterion **f** in relation to criterion **g**, However, it might also it could be a quadratic or even more exotic function.

3. Interactive optimization: in this case one or more obtained solutions are given, the decision maker must indicate which one is preferred, and if not satisfied yet, in which direction the search should continue.

2.2 Finding the Optimal Solution

It's difficult to find algorithms that have the best solution for most multi-objective optimization problems, such problems are called NP- hard.

Many authors pointed out the total completion time ($\sum C_i$), the total tardiness ($\sum T_i$), maximum completion time (C_{max}), maximum tardiness (T_{max}) and maximum earliness (E_{max}) are the most prominent measure among the scheduling objectives in industrial applications.

We address precise approaches in this subsection's reminder. In the problem of minimizing total tardiness is NP-hard, any problem that includes $1//\sum_j T_j$ as a sub problem is NP-hard as well. It has been shown that certain in polynomial time, particular instances can be solved. The EDD-order solves the problem $1/p_j = p/\sum_j (E_j+T_j)$ according to Garey et al.(1988) [30]. This result hold true for the problem $1/p_j = p/\sum_j (\alpha E_j+\beta T_j)$. Verma and Dessouky (1998) [31] showed that the problem $1/p_j = p/\sum_j (\alpha E_j+\beta T_j)$ is solvable in polynomial time if

the weights are agreeable, i.e., the works can be renumbered as $\alpha_1 \leq \alpha_2 \leq \ldots\ldots \leq \alpha_n$ and $\beta_1 \leq \beta_2 \leq \ldots\ldots \leq \beta_n$, this covers the case with symmetric weights $\alpha_j = \beta_j$.

Fry and Kenog Leong (1987) [32] made the initial try to tackle the problem in general solving the $1//\sum_j (\alpha E_j + \beta T_j)$ is a problem of integer linear programming; this was workable for cases with up to 12 jobs. Based on two lower bounds, Kim and Yano (1994) [33] applied Branch And Bound to solve the issue $1/ \sum_j (E_j + T_j)$; they can solve instances with up to 20 jobs. Hoogeveen and Van de Velde (1996) [34] reported a similar result when they apply Branch and Bound to solve the $1//(\alpha E_j + \beta T_j)$ problem. They suggest six lower boundaries and a set of dominance principles, however they are unable to tackle cases with more than 25 jobs. Recently, better results have been reported by Sourd and Kedad Sidhoum (2003) [35] and Bulbul et al.[36] (submitted for publication). In both papers the problem is formulated as an integer linear programming problem using a time-indexed formulation. Each task J_j is split into P_j unit duration segments, where one of the constraints signals that the segments forming job J_j (j=1...n) must be assigned to consecutive intervals.

The problem in which two maximum cost criteria are minimized, $1//F (f_{max}, g_{max})$, is solvable in $O (n^4)$ time, according to Hoogeveen (1996b) [15]. The ε-constraint technique is used in this algorithm: there are some $O (n^2)$ Pareto optimum points, and each sub problem is solved in $O (n^2)$. When it comes to the maximum cost criterion g_{max} is such that the elapsed time can be reduced to $O (n^3 \log n)$. After renumbering $g_1 (t) \leq g_2 (t) \leq .... \leq g_n (t)$ for all $t \in [0, \sum_j p_j]$, the running time can be reduced to $O (n^3 \log n)$; notice this L_{max} satisfies this constraints. Hoogeveen also demonstrates a generalization of the above algorithm can solve the problem of simultaneously minimizing three maximum cost criteria in $O (n^8)$ time. There is only one Pareto optimum point, and the number of them is limited $O(n^2)$, takes $O(n^2)$ time $O(n^6)$. This algorithm can be extended to find all Pareto optimal points for a set of **K** maximum cost

parameters, but since there is no polynomial upper bound on the number of Pareto optimal points for **K≥ 4**, it is ambiguous if this algorithm is polynomial for fixed K. (for arbitrary K the problem is strongly NP-hard). Sourd (2001) [37] considers the problem 1/ r_j, pmtn /F (f_{max}, g_{max}). He demonstrates that the number of Pareto optimal points is bounded by $O(n^2)$, and that each Pareto optimal point can be calculated in $O(n^2)$ time, resulting in an $O(n^4)$ algorithm overall.

Because each of the sub problems can be solved as responsibilities problem, Chen and Bulfin (1990) [38] show that the set of Pareto optimal points can be calculated by applying the ε-constraint approach for any combination of two performance parameters, one of which is T_{max}. They claim that instead of T_{max}, f_{max} or E_{max} can be achieved a similar result.

2.3 Local Search Heuristic Methods

2.3.1 Introduction

The term heuristic comes from the Greek word (heuriskein) which means to discover or find [39]. A heuristic, according to Reeves [25], is a procedure for finding really good (i.e., near optimal) solutions at a low computational cost without being able to guarantee feasibility or optimality, or even, in certain cases, to state how close to optimality a particular feasible solution. The best solution can be found using the local search approach in a fair amount of time.

2.3.2 Preliminaries

The first time local search was used to solve NP-hard problems was in the late 1950's and early 1960's [3]. Local search methods have the same feature: they iteratively move according to some, from one viable solution to another given rules, exploring the search space. Many of these methods are inspired from natural and they explore neighborhood of feasible solution. Local

search strategies vary in the problem representation they use, the neighborhood description they use on that representation, and the approach they use to search through that neighborhood. Evolution has influenced some local search methods. This suggests the need for the following definition:

Definition 2.3.2.1 [4]

An instance of a combinatorial optimization problem is a pair (S,f), where the solution set S is the set of all feasible solutions and the cost function f is a mapping f:s→R. The problem is to find a globally optimal (minimum) solution, i.e. an $s^* \in S$, such that $f(s^*) \leq f(s)$ for all $s \in S$.

2.3.3 Representation of the Solution [4]

Solution Representation depends on the problem specification. In a scheduling problem of n jobs, a solution is represented by a permutation of the integer 1,...,n.

Definition 2.3.3.1 [2]

A neighborhood function N* is a mapping $N^* : S \rightarrow P(S)$ which specifies for each $s \in S$ subset $N^*(s)$ of S neighbors of s.

For permutation, there are three traditional neighborhoods. They're described through making use of specific moves to a series of tasks [21].

1- Shift (insert): This neighborhood is obtained by removing a job from one position in the sequence (1,2,3,4,5,6,7,8) and insert it at another position either before (left insert) or after (right insert) the original position. For example the schedules (1,5,2,3,4,6,7,8) and (1,2,3,4,6,7,5,8) are booths neighborhoods.

2- Interchange (swap): Swap two jobs that aren't necessarily next to each other. For example the schedule (1,6,3,4,5,2,7,8) is a neighbor.

3- Insert a block: Insert a subsequence of jobs in a new position. The schedule (1,4,5,2,3,6,7,8) is an example of a neighbor.

Definition 2.3.3.2 [4]

Let N* be a neighborhood function and (S,f) be an example of a combinatorial optimization problem. A solution $s^* \in S$ is referred to as a local optimal (minimal) solution with respect to N^* if $f(s^*) \leq f(s)$ for all $s \in N^*(s^*)$. The neighborhood function N^* is called exact if every local minimum with respect to N^* is also a global minimum.

2.4 Algorithms for Local Search: The Fundamental Notation

The following is a feature that all local search methods have in common:

1. Initialization: As the present solution, the start feasible solution s is generated randomly or using a heuristics method or some known rule. The value of the objective function in the present solution is computed.
2. Neighborhood generation: A move is made through the solution space S from neighbor to neighbor to select a neighbor s' of s.
3. Acceptance test: Each local search method has its own acceptance test to determine whether s' replace s as the current solution.
4. Criteria for termination: The algorithm is repeated until some termination criteria are satisfied. The output will be the best solution generated.

2.4.1 Descent Method (DM)[2]

The Descent Method is a simple form of neighborhood search methods in which only improving moves are allowed. The resulting solution is a local optimum, not necessarily a global optimum.

The structure of a Descent Algorithm is presented in the figure (2-1)

Step (1): Choose a starting solution $s \in S$.

Step (2): Select a sequence $s' \in N^*(s)$; $\Delta = f(s') - f(s)$;

If $\Delta < 0$ then $s = s'$.

Step (3): If $f(s') \geq f(s)$, $\forall s' \in N^*(s)$, then stop; if no stopping criteria is meet, go back to step (2).

Figure (2-1) Structure of a Descent Algorithm

2.4.2 Algorithm of Simulated Annealing (SA)

Simulated annealing (SA) has its origin in statistical physics, where the process of cooling solids slowly until they reach a low energy state is called annealing. It was originally proposed by Metropolus et al. [21] and was first applied to combinatorial optimization problems by Kirkpatrick et al. [40].The sequence of the goal function values in such an algorithm does not have to decrease monotonically. A neighbor s' in a specific neighborhood is generated (typically randomly) from an initial sequence s.

Then the difference $\Delta = \mathbf{F(s') - F(s)}$, in the values of the objective function F is calculated. When it comes to $\Delta <= 0$, sequence s' is accepted as the new iteration's starting solution. In the case $\Delta > 0$, sequence s' is accepted as new starting solution with probability $\mathbf{exp(-\Delta/T)}$, where T is a temperature-related parameter. Typically, the current temperature is high in the early phases, making it relatively easy to escape from a local optimum in the first rounds.The temperature normally drops after a set of sequences have been created. Often this is done by a geometric cooling scheme which we will also apply.

In this case, the new temperature T^{new} has been chosen so that $T^{new} = \lambda T^{old}$, where $0 < \lambda < 1$ and T^{old} denoting the old temperature and T^{new} denoting the new temperature. A possible stopping criterion would then be a cycle of a final

temperature, which is sufficiently close to zero. might therefore be used as a stopping condition. As in [41] we ascertain on the basis of the initial temperature T=10.

Simulated Annealing Algorithm

Step (1): Choose an initial solution s ∈S, s*=s; a starting temperature t_0>0; k=0, G=1

Step (2): Define B; select s' ∈ N^*(s); Δ=f(s')-f(s);

$P(\Delta,t_k) = \exp(-\Delta/t_k)$;

If Δ≤ 0, then s=s', and if f(s)<f(s^*) , then s^*=s; else (Δ>0);

If a set of numbers is chosen at arbitrary, [0,1] ≤ p (Δ,t_k) , then s=s' ; G=G+1,

Step (3): If G≤ B return to the previous step (2),

Step (4): Return to step (2) until some Stopping condition are fulfilled; update temperature; k=k+1;

Figure (2-2) SA algorithm

2.4.3 Tabu Search (TS) method

The origins of Tabu Search (TS) can be traced back to the 1960s and 1970s, and was Gupta presented it in its current form by (1989) [42]. The vast majority of TS applications began in the late 1980s [25]. As the name implies, one of the major concepts of TS is the use of a flexible memory (tabu list) to tabu particular moves for a period of time. When a move is chosen to lead the search from the current solution to its neighbor solution, it is immediately allocated to the tabu list in every iteration of TS. For a lot of iterations after that, this move will not be chosen. The size of the tabu list is determined by the number of iterations, and it is limited to a particular length. When the list reaches its maximum length, the move with the oldest assignment is removed

from the list and the most recent assignment is inserted. With an appropriate design of the tabu list, TS is able to prevent cycling of the search and guide the search to the solution regions which have not been examined and approach to good solutions in the solution space. However, design of the tabu list may also prohibit the search to appealing solution regions. To compensate for this disadvantage, Gupta suggested the use of the concept of (aspiration criterion) defined as follows: if a specific move is currently tabued and has the potential to lead the search to good solution regions, that move should be removed from the tabu list (aspired). The most prevalent one is when a tabued move is removed from the tabu list if it can provide a better solution than the existing one [42].

2.4.3.1 Tabu Search (TS) algorithm

The (TS) algorithm discuss each of the following issues:

a) Initialization

Determine the initial solution by an effectively sequence.

b) Neighborhood Generation,

The stander methods can be used here (swap or insert); also one can use any modified method to generate a new neighborhood.

c) Termination criterion

Stopping criterion for (TS) chosen by time less then 10m or by end of all iteration.

The outline of (TS) is given by the structure below:

step 1. Initialization:

Initial solution should be chosen (or generated at random) (p)

put (p) as a present solution (s)

put (cost (s)) as a present solution (f)

create a tabu list that only contained (s)

step 2. If the termination requirement is not met,

step 3. Create (Ṕ) neighborhood for (s) (by insert or swap)

step 4. Calculate the cost (Ṕ)

step 5. If cost (Ṕ) < f

step 6. s = p′ (s ← p′) (accept the move)

step 7. f = cost (p′) (f ← cost (p′)) (modify the existing value)

step 8. tabu list= tabu list + p'

step 9. else if cost(p') = f

step 10. If p' $\notin$ tabu list

step 11. s= p' (s ← p′) (accept the move)

step 12. tabu list= tabu list + p'

step 13. end

step 14. end if

step 15. end while

Step 16. Solution =s

Figure (2-3) TS algorithm procedure

2.5 Genetic Algorithm (GA)

2.5.1 Introduction

John H. Holland [43] was the first to suggest Genetic Algorithms (GA).

They are search algorithms that simulate the biological evolution process by exploring a solution space.

Genetic algorithms work with the population of solution each solution is represented as a string the (GA) technique based on evolution's mechanism. The solution space is usually represented by a population. New structures are generated by applying simple genetic operators such as (select, cross-over, and mutation). Members of the existing population with greater fitness values (i.e., better objective function values) will have a larger chance of being chosen as parents, which is comparable to Darwin's concept of survival of the fittest. Because the beginning population is produced at random, the ultimate solution's optimality cannot be guaranteed. As a result, at least one solution with the shortest must be included in the initial population (objective function of our problem) is included applying (select, cross-over and mutation), to generate new population and save the best solution in every generation. The best one from saved solutions becomes GA solution [43]. A solution's fitness value is a vector that represents the function values. A parent is generated by selecting the best solutions from the current population. Then, in each generation, solutions with high fitness values in each population are chosen and recombined to create a new offspring after applying the genetic operators for each new offspring we get a new population. It's worth noting that the mutation operation, for example, is based on the pairwise swapping of two tasks in the relevant sequence. There are several applications of Genetic Algorithms (GA**)** have been widely applied to various fields since 1975. They are applied to business, scientific, and engineering areas including:

(Optimization of complex function system Classifier system, Machine learning, Pattern recognition, Error diagnoses, Scheduling**,** Partitioning objects and graphs, Self-adapting system, Clustering, design and Process control). Researchers have showed that (GA) have been used in a wide variety of

optimization tasks, including numerical optimization and such combinatorial optimization problem as shop–job scheduling [44].

2.5.2 Basic Stricture of Genetic Algorithm

The following are the main components of a genetic algorithm [45]:

1. Encoding of the solution

Solutions are represented on the chromosome through a chromosomal representation (solution encoding).The natural permutation form of a solution for the machine schedule problem is a permutation of the integers 1,...,n, which describes the processing order of n jobs. A scheduling solution, or the natural permutation representation of a solution, is used to represent each chromosome.

2. Initial Population

The first population of chromosomes is created (initial population). The initial population of chromosomes is formed using scheduling heuristic dispatching rules (heuristics methods), combined with random methods, in order to approximate an ideal solution as closely as feasible.

3. Fitness (evaluation)

The objective function is used to determine chromosomal fitness (fitness). Each chromosome is examined and its fitness is calculated for each chromosome when a population is produced. Finally, a fitness value is assigned to each chromosome based on the population size.

4. Selection

Natural selection of some chromosomes occurs when chromosomes (parents) are picked from the population for combining to form new

chromosomes (children) using selection procedures (typically based on fitness value).

5. Genetic Operators

Crossover and mutation operators are genetic operators that are applied to chromosomes with the goal of creating new members, i.e. offspring, in the population by crossing the genes of two chromosomes (crossover operators) or changing the genes of one chromosome (mutation operators):

a) Crossover

The role of a crossover operator is to combine elements from two parent chromosomes to generate one or more child chromosomes.

b) Mutation

A mutation operator's job is to ensure that a population's diversity is maintained so that other operators can continue to work.

6. Substitute

The natural selection of population members who will survive (replacement) is based on elitism. That is, to preserve the existing population's best chromosomes and their progeny. They'll build a new population to ensure the following generation's survival.

7. Parameter Selection

Natural population convergence that is improved globally at each stage of the algorithm. For determining appropriate parameter values such as population, size crossover, and mutation.

The design of the foregoing components, as well as the selection of factors like as population size, probability of genetic operators (i.e., crossover

and mutation), and the number of generations, all influence the performance of a (GA). The following steps give us the outline of (GA):

1. the beginning: make the first population

this population's worth

As a (GA) solution, save the best element from this population.

2. while the halting condition is not met, choose a good solution (parents) Using genetic operators, create a new population from the existing population (crossover and mutation) this population's worth. If you discover a new best individual element, save it as a (GA) solution.

3. end while

Figure (2-4) Genetic algorithm

The following cycle also give us the outline of (GA)

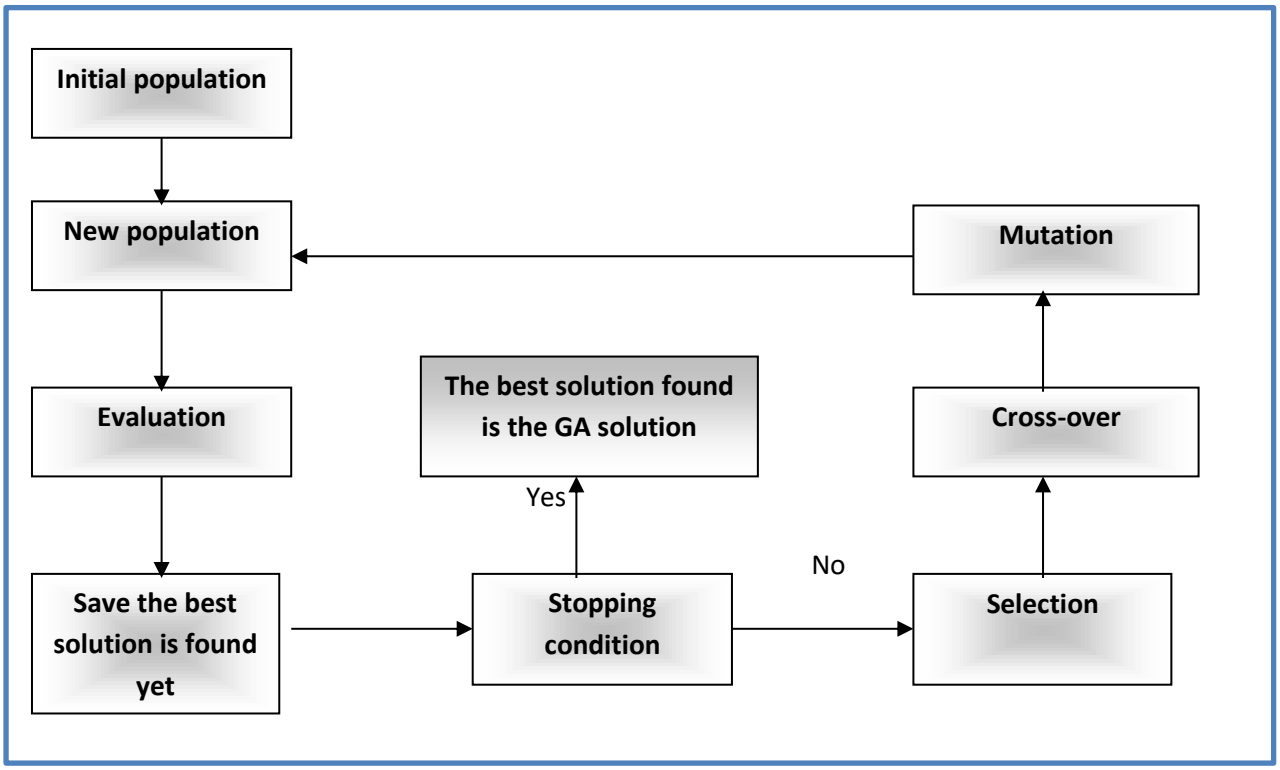


Figure (2-5) Genetic algorithm cycle

2.6.1 Initial population

The initial population can be generated at random or can be constructed by using problem-specific knowledge. Chen et al.[46] used specific construction heuristics for the flow shop problem to build their first population. They claim that a good initial population increases the efficiency of GA. Delia Croce et al. [47] select the solution for the initial population at random, but in order to speed up convergence, propose to choose an initial population partially produced with some quick heuristic. Reeves [48] compares the performance of GA with a completely random initial population and a population where one (or more) individual is obtained with a good heuristics rule and the remaining ones are generated randomly. The procedure with the specific-element in its initial population appears to arrive at its final solution more quickly, with no observed deterioration in solution quality. Inserting a high-fitness chromosome into the initial population is called (seeding) the success of the strategy is dependent on the availability of good starting solution; the large variation in the population size (m), used by different researches, ranging from a size of 20 Lee and Kim [16] to300 Delia Croce et al. [47]. Nordstrom and Tufekci experiment with different sizes concluded that increasing the population size seems to improve the quality of the solution. Large population does not show any significant improvement in the rate of convergence. Conversely, a population size of 20 seems to be too small, because it runs with this size yield somewhat poorer performance [49].

2.6.2 Selection

Selections to choose good candidate solutions from current population for the next generation i.e.(for generate the next population). The number of these candidate solutions (k) is controlled (determined) according to the population size (m), which is selected in the initial steps of (GA). Lee and Kim,

[16] compare the (s)-best reproduction operator with roulette wheel selection. The (s)-best scheme requires less computation time, but the quality of the solutions obtained with roulette wheel selection is better.

2.6.3 Genetic Operators

a) Crossover

The crossover serves to exchange information between chromosomes. Thus usually results in a useful combination of partial solutions on other chromosomes, and it speeds up the search process early on in the generation. However, the process of transferring gene information may result in some genes having redundant or missing properties. The crossover may result in an infeasible solution. Such events are most common in chromosomes that have been permuted. The reproduction in its first form is based on rank ordering. The first parent for the crossover is selected at random from among the best (s) individuals, where (s) is a parameter. The second parent is selected at a random from the rest of the population. This reproduction operator increases the greediness of the GA [49].

The classical 1-point crossover is used by Lee and Kim [16] and compared with 2-point crossover. In a 2-point crossover, two crossover points are chosen at random and the segments in between are exchanged. The results show that 2-point crossover is slightly better, but that it also requires more computation time. Every time a permutation representation is employed, an appropriate crossover operator must be designed, as described in the previous paragraph. Partially matched crossover (PMX) is such an operator. Two crossover points are generated at random and the segments in between define a matching section. This matching is used to affect a cross through position-by-position exchange operations. For example, with crossover points after the 3rd and 6th element:

	Parents:		Exchanging:		Restoring:	
P1	789|251|638	→	798|483|634	→	795|483|612	= Ch 1
P2	956|483|271		956|251|271		986|251|473	= Ch 2

In this example, the mapping is 2 ↔ 4, 5 ↔ 8 and 1 ↔ 3. The sections are swapped between the two crossover points. The elements 8, 3 and 4 outside the section are substituted according to the matching to restore feasibility in the first child. The elements 5, 2, and 1 are replaced in the second child. PMX is mentioned in a number of articles, including Chen et al. [46].

Delia Croce et al. [47] use linear order crossover **(LOX)**. This operator chooses two random crossover points. The parts from parent 1's cross section are deleted from parent 2, leaving some "holes" (marked with a '.'). The holes are moved inwards from the extremities until they reach the cross section. The parent 1 cross section is then swapped with the parent 2 cross section. The other child is obtained in a similar manner.

	Parents:	Holes	Sliding	Exchanging	
P1	798|251|634	79.|251|6..	792|…|516	792|483|516	Ch 1
P2	956|483|271	9.6|483|.7.	964|…|837	964|251|837	Ch 2

LOX prefers to honor relative positions between elements and, to the extent possible, absolute places in the string. For example, the ordering of the first cross section (2,5,1) is completely destroyed in the first offspring by PMX. In the first offspring produced by LOX, the relative order, 2 before 5 and 1 and 5 before 1, is preserved.

Other researches resolve conflicts (a child in which some elements appear more than once) in a random fashion. Reeves [48] also experiments with random choice to restore feasibility. But he reports that disruption in the offspring seems to be excessive. Therefore, he used the following 1-point

crossover: One crossover point is generated randomly. A child is composed of the subsections before the crossover point of the parent and filled up taking in order each "legitimate" element from the order parent. This 1-point crossover, denoted by LEGX.

Parents	Children
7982|51634	7982|56431
9564|83271	9564|78213

There is another cross over scheme: (homogeneous mixture crossover) **HMX** [41] this is determined by uniformly mixing the two parents by creating a set (m) of genes, with the odd position from the first parent and the even position from the second parent. Then, since we read the set (m) from the left, we keep the gene j if it does not exist in the child ch1 and put (0) in (m), else we keep the gene j in the second child ch2 and put (1) in (m), until the set (m) genes are exhausted. This method also results in the birth of two new offspring.

Parent	Mixture	Child
P1= 798251634	799586245813623741	Ch1 = 795862413
P2= 956483271	001000001100111111	Ch2 = 958623741

The rationale for this crossover is that it keeps the absolute positions of one parent while still preserving the relative locations of the other parent's children. However, after a number of generations, the population has converged and crossover alone cannot improve the population anymore. A diversifying component is necessary which can be offered by mutation.

Our crossover (QMX) we chose the 1st (crossover point at the end of) quart of parent no.1 and at the beginning of last quart of parent number 2. The jobs in the 1st quart of parent number 1. are deleted from parent 2. and put them

in start of p2 and called the resulting sequence child no. 1; by same meaning we remove the jobs in the last quart of p2 from p1 and then put them in the last of p1 and called the new sequence child 2; for example:

P1=13|245876 → 1st quart of p1 =[1 3]

P2= 486713|52 → last quart of p2 =[5 2]

→ P1= 13…4…876

→ P2= 4867… … 52

→ ch1= 13486752

→ ch2= 13487652

b) Mutation

Syswerda introduced a number of mutation operators in scheduling study [49]. Two elements are chosen at random in this operator. Order-mutation: swaps these two items around. The second element is placed before the first in a position-mutation. The order-mutation method outperforms the position-mutation method. Other studies employ the same techniques but refer to them by more traditional terms, such as "swap" and "shifting" in neighborhood search approaches. Chen et al. [46] change the order of the two elements. Reeves [48] conducts certain tests with both operators. "Shift" appears to be preferable to "swap," thus "shift" is used in his final version.

Tate and Smith [50] use another form for mutation. They pick two random points in a string and reorder all elements within the substring confined by the two chosen elements.

Before		After
7 9|8 2 5 1|6 3 4	→	7 9|1 5 2 8|6 3 4

2.6.4 Termination

Classically, When a certain number of generations (or iterations) have been completed, the method comes to an end. For example, Chen et al. [46] observe that the solutions become stable after twenty generations; therefore, they use 20 generations. Because of this fixed number of generations, it is possible that some generations at the end of the process are superfluous. To avoid this, the procedure can be terminated when the best solution in a population is not better than the previous population for a number of iterations. Lee and Kim [16] used this termination condition. There are other stopping criteria which terminate the procedure when the objective function values for the best and worst individuals in the population are equal [49]. The algorithm of Lee and Kim [16] stop when the improvement of the average fitness value in one generation is less than 0.01% of the average fitness value in the preceding generation.

2.7 Some applied examples of Multicriteria problems [26].

2.7.1 Some examples

Several criteria are involved in many scheduling challenges in the production domain. There are various works in the literature that deal with a category of difficulties that are ideally suited to a situation: the requirement to generate “Just-in-Time” products. This requirement translates into two wishes: the first is to avoid delivering late to the client, and the second is to avoid storing finished goods. As a result, generating "Just-in-Time" is a compromise between producing somewhat late and not too early. In the literature, there are numerous definitions of "Just-in-Time" scheduling. We now give a set of scheduling challenges that relate to real-world scenarios, regardless of their application field.

2.7.2 Chemical and electroplating industries

This category of problems returns us to the Hoist Scheduling Problem in the literature. A limited number of chemical-filled tanks are provided for the galvanization treatment of items. The arrival of items in the shop follows a cyclic pattern. These objects are transported from one thank to the next by a transportation robot (or a group of robots) that is normally suspended above the tanks. A variable of the problem is the processing time, or soaking time, of things in the tanks. Indeed, the chemical engineers specify a minimum and maximum soaking time for each soaking, allowing the analyst free to determine the ideal times. The primary goal is to find a cycle time that is as short as possible, i.e. a minimum value for the makespan requirement. Nonetheless, two aspects compel us to examine this issue from a Multicriteria perspective. To begin with, actual experience shows that the timing of transportation robot movement (handling and placing of things into the tanks) is the most challenging component to determine in order to efficiently reduce cycle time. Following that, for the majority of the tanks (and hence the chemical baths), observing the minimal soaking duration is the only thing that matters. In actuality, we can sometimes go above the maximum soaking time if it helps us better regulate the robots' movements. When minimizing cycle time and, for example, a weighted total of overtaking the soaking times relative to the permissible maximum soaking times, the problem becomes bi-criteria.

2.7.3 Steel hot rolling mill industry

The difficulty with steel hot rolling mills is that they produce steel coils from steel slabs. The shop can be divided into two sections in this problem: a large slab yard where steel slabs are stacked awaiting processing by the rolling mill, and the rolling mill in itself. Each slab has its own unique properties and can be used to treat a variety of steel coils. When a slab is chosen to be

processed, it is hauled to the rolling mill by cranes and placed in a furnace where it is heated to a high temperature. After leaving the furnace, the hot steel slab is rolled through a series of rolls under high pressure to produce the desired width, thickness, and hardness for the steel coil. It's worth noting that each shift of processed orders has an ideal sequencing form connected with it, which takes into account additional constraints such as the furnace and the fact that we can't change its temperature as much as we'd like. One of the planner's goals is to reduce pressure setting variations between two consecutively generated coils because this might have a significant impact on their quality. Furthermore, because the rolls come into touch with hot steel, they quickly wear out and must be replaced. As a result, coil production is scheduled in shifts of a few hours. There are also a few other limits to consider. The goal is to sequence the steel coils in such a way that the value of the coils rolled in the sequence is maximized, changes in characteristics between subsequent coils are minimized, non-essential crane movements are minimized, and departure from the optimum sequencing shape is minimized.

2.7.4 Car assembly

Subcontractors face multicriteria scheduling challenges as a result of car production lines. This is especially true when it comes to car seats. The automobile manufacturer and the car assembler are in sync, and a vehicle's manufacturing line sequence automatically instructs the manufacturer to produce seats. This establishes a deadline for their delivery. This is a Just-in-Time scheduling issue since early creation of a seat incurs additional storage expenses for the assembler (higher than storage costs of an engine). Late seat deliveries, on the other hand, cause the assembly process to come to a standstill. The car in question must therefore be moved to the head of the line, resulting in increased production costs.

ChapterThree

Minimizing Total Completion Time with Total Tardiness and Maximum Earliness

3.1 Formulation of the Problem

A set of n independent jobs N={1,2,...,n} are available for processing at time zero, each job j (j=1,2,...,n) is to be processed without interruption on a single machine that can be handle only one job at a time, requires processing time P_j and due date d_j. Completion time is calculated based on a specified job schedule δ, $C_{\delta(j)} = \sum_{i=1}^{j} p_{\delta(i)}$, total tardiness $\sum_{j=1}^{n} T_{\delta(j)}$, where $T_{\delta(j)} = \max \{C_{\delta(j)} - d_{\delta(j)} ,0\}$ and maximum earliness $E_{max}(\delta) = \max\{ E_{\delta(1)}, E_{\delta(2)},\ldots, E_{\delta(n)}\}$ can be computed where $E_{\delta(j)} = \max \{ d_{\delta(j)} - C_{\delta(j)} ,0\}$. The aim is to organize the jobs so that they can be completed in a timely manner objective function of three criteria $\sum_{j=1}^{n} C_j + \sum_{j=1}^{n} T_j + E_{max}$ is minimized. This problem is NP-hard since the $\sum_{j=1}^{n} T_j$ is NP-hard.

This problem is denoted by the letter (P), and it can be described as follows:

$$Z= \min_{\delta \in S}\{ \sum_{j=1}^{n} C_{\delta(j)} + \sum_{j=1}^{n} T_{\delta(j)} + E_{max}(\delta)\}$$

s.t.

$$\left.\begin{array}{ll}
C_{\delta(1)} = p_{\delta(1)} & \\
C_{\delta(j+1)} = C_{\delta(j)} + p_{\delta(j+1)} & j=1,2,\ldots,n-1 \\
C_{\delta(j)} \geq 0 & \\
T_{\delta(j)} \geq C_{\delta(j)} - d_{\delta(j)} & j=1,2,\ldots,n \\
T_{\delta(j)} \geq 0 & \\
E_{\delta(j)} \geq d_{\delta(j)} - C_{\delta(j)} & j=1,2,\ldots,n \\
E_{\delta(j)} \geq 0 &
\end{array}\right\} \quad \ldots(p)$$

Where S denotes the set of all schedules.

3.2 An Application of the Branch and Bound (BAB) Algorithm for the Problem (P)

The initial upper bound is determined by the specific problem at the start of the solution process. The shortest processing time (SPT) rule, which is sequencing the jobs in non-decreasing order of their processing time (P_j), j=1,2,...,n, is obtained by the heuristic method proposed and applied once at the root node of the (BAB) search tree to find the upper bound (UB) on the minimum value of ($\sum_{j=1}^{n} C_j + \sum_{j=1}^{n} T_j + E_{max}$).

To calculate a lower bound (LB) for each node, let δ be the sequencing jobs and δ' be the un sequencing jobs, hence.

LB(δ) =Exact cost of (δ) + cost of (δ').

Where cost of δ' is obtained by using lower bounding procedure.

Decomposing the problem into three sub problems (SP_1), (SP_2) and (SP_3) as follows:

$$Z_1 = \min_{\delta \in S}\{\sum_{j=1}^{n} C_{\delta(j)}\}$$

s.t.

$$C_{\delta(1)} = P_{\delta(1)}$$

$$C_{\delta(j)} = C_{\delta(j-1)} + P_{\delta(j)} \qquad j=2,3,\ldots,n$$

$$C_{\delta(j)} \geq 0$$

(SP_1)

This sub problem (SP_1) is solved by (SPT) rule.

$$Z_2 = \min_{\delta \in S}\{\sum_{j=1}^{n} T_{\delta(j)}\}$$

s.t.

(SP_2)

$T_{\delta(j)} \geq C_{\delta(j)} - d_{\delta(j)}$ j=1,2,…,n

$T_{\delta(j)} \geq 0$

This sub problem (SP_2) is NP-hard.

$Z_3=\min_{\delta \in S} \{E_{max}(\delta)\}$

S.t. (SP_3)

$E_{\delta(j)} \geq d_{\delta(j)} - C_{\delta(j)}$, j=1,2,…,n

$E_{\delta(j)} \geq 0$

This sub problem (SP_3) is solved using the minimum slack time (MST) rule, which involves sequencing jobs in non-descending order of slack time d_j-p_j, j=1,2,…,n.

Thus, the lower bound (LB) for the problem (p) is the sum of minimum value of the sub problems (SP_1), (SP_2) and (SP_3). We proposed that the minimum value for $\sum T_j$ is obtained by $\sum T_j$ (SPT) – T_{max} (EDD), Where EDD is the earliest due date value, i.e., sequencing the jobs in non-decreasing order of their due dates.

It is clear that $\sum T_j$ (SPT) – T_{max} (EDD) $\leq \sum T_j$

Let Z_1,Z_2 and Z_3 be the minimum values of (SP_1), (SP_2) and (SP_3), respectively, and use the following theorem to obtain a lower bound for (P).

Theorem (3.1) [51]:

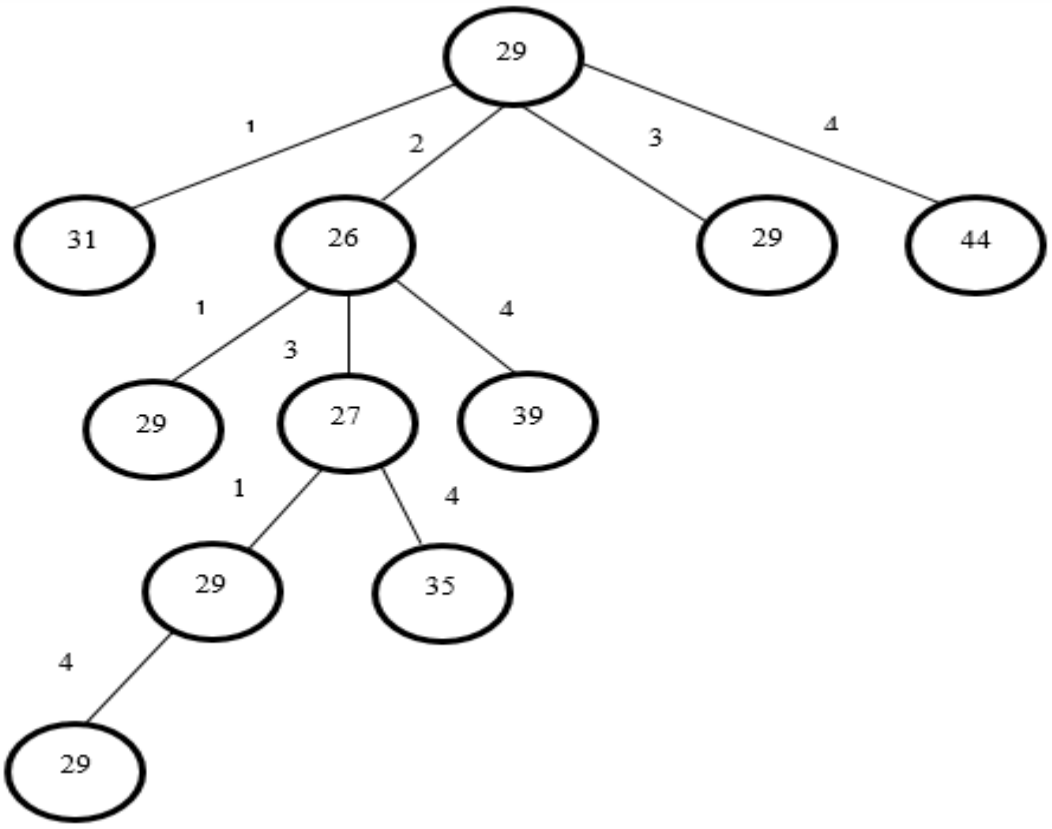
If Z_1,Z_2,Z_3 and Z are the minimum objective function values of (SP_1),(SP_2),(SP_3) and (P) correspondingly then $Z_1+ Z_2 +Z_3 \leq Z$.

$LB = Z_1 + Z_2 + Z_3$ is a lower bound (LB) for the problem (P) obtained by applying theorem (3.1).

An example : Suppose the problem (P) has the following data:

j	1	2	3	4
p_j	2	3	1	6
d_j	8	4	6	10

The BAB algorithm tree to find the optimal solution for the problem (P) is as follows:



The optimal sequence is (2,3,1,4) with the optimal value 29.

3.3 Practical Results

In the following subsections we interduce the practical results from determine the parameters and computational and the outcome tables.

3.3.1 Deterministic the Parameters for the Algorithms

A) BAB algorithm

we chose the forward branching technique as we mention in 1.2.2, and for the stopping criteria the algorithm will stop after fix period of time specially after (1800 second).

B) DM algorithm

we chose the swap method to generate new solution and for the stopping criteria the algorithm will stop after fix number of iterations here we chose (30,000) or a fix period of time specially after (600 second).

C) SA algorithm

we chose the swap method to generate new solution and for the stopping criteria We employ a predetermined number (30,000 in this thesis) of generated solutions as a stopping criterion for all heuristics because we need to be unbiased, or a fix period of time specially after (600 second). And we ascertain on the basis of the initial temperature T=10.

D) GA algorithm

i) Initial population generation

Reeves [25] compares the performance of GA with a completely random initial population and a population where one (or more) individual is obtained with a good heuristic and the remaining ones are generated randomly. The procedure with the specific-element in its initial population appears to arrive at its final solution more quickly. We use the above technique for our problem, and then we

construct the initial population by using some individual solutions found by some good known sequence like (SPT, EDD, MST and Lowler).

ii) Selection

Selections to choose good solutions from current population. The number of this selected solutions (k) is controlled (determined) according to the population size (m), moreover the population size (m) can be found by these number of selected solutions (k), as follows:

$\boldsymbol{m = 2k^2 + q}$, where $2k^2$ solutions come from the crossover operation, and the q solutions generated randomly to escape from the local optimum, these number of random solutions add at each iteration, the number q represent approximately 10% from the hole population size m, in this way we get the following table in which we list the population size (m) for some (k) selected candidate solutions and the number of random generated solutions:

k	5	6	7	10
q	10	8	12	20
m	60	80	110	220

iii) Genetic operator

- **Cross over:** Among the crossover rules that introduced in 2.6.3 {PMX, LOX, LEGX, HMX, QMX} we chose the our cross over (QMX) since it gives the best influence of others rules.

- **Mutation:** We chose (swap mutation), or (Order-mutation) since it performs better than position-mutation, as we see after test them by multiple runs on several examples.

iv) Population size with iterations and stopping condition

The efficiency of (GA) is dependent mainly on the (population size and stopping condition) parameters, since both of them are determine the speed of (GA) and the convergent to nearest optimal solution, so we should be determined them in more precisely. The most important question here is (how to determine these values?). We suggested that: the populating size (m) is chosen from the set A= {60, 80, 110, 220}, as we mention the range from a size of (20 to 300), and the number of iterations is chosen from the set B= {50, 100, 250, 500}. Then for each value of (A) solve same example along all values of (B), in this way we will have (4x4) values matrix and (4x4) times matrix for each example we solve it. So, we chose population size = 80 and iteration number =100; since we note that there is no good improve ness by increasing the population size and (or) iterations, while it consumes more time. For stopping condition, in this study we shall terminate the GA cycle after a fix number of generation (100 as we mention above) or after a fix period of time (600 second as the algorithms that used in this work).

3.3.2 Computational Results

The BAB algorithm and local search algorithms are put to the test by coding them in MATLAB R2019b and running them on a computer. Test problems are generated as follows: for each job j, an integer processing time p_j is generated from the discrete uniform distribution [1,10]. Also, for each job j, an integer due date is generated from the discrete uniform distribution [P(1-TF-RDD/2), P(1-TF+RDD/2)], where P= $\sum_{j=1}^{n} p_j$ depending on the relative range of due date (RDD) and on the average tardiness factor (TF). The values 0.2,0.4,0.6,0.8,1.0 are taken into account for both parameters. For each selected value of n, two problems are

generated for each of the five values of parameters producing 10 problems for each value of n, where the number of jobs n=from 3 to 40,000 .

3.3.3 Tables of Results

The following tables give the comparative of computational results and the time (in seconds) for the problem (P). When a problem cannot be solved to its optimality within the time constraint of 1800 seconds, the problem is abandoned. Symbols we have all we need in all of these tables:

Ex: Number of example.

Node: Number of nodes.

Optimal: The optimal value that is obtained by BAB algorithm.

No.of opt.: Number of examples that catches the optimal value.

No.of best: Number of examples that catches the best value.

DM: The value that is obtained by descent method.

SA: The value that is obtained by simulated annealing method.

GA: The value that is obtained by Genetic Algorithm.

Time: Time in seconds.

$$\text{Status} = \begin{cases} \text{1, if the example is solved} \\ \text{0, otherwise} \end{cases}$$

Table 3-1: A comparison between the optimal solutions obtained by BAB algorithm and the values result of local search algorithms at n=5

BAB						Local Search					
n	Ex	Optimal	Node	Time	Status	DM	Time	SA	Time	GA	Time
5	1	74	98	0.058544	1	74	0.431927	74	0.490253	74	0.339277
	2	101	52	0.003252	1	108	0.432393	101	0.484132	101	0.15029
	3	134	307	0.009876	1	134	0.42002	134	0.396143	134	0.141369
	4	74	73	0.002755	1	74	0.42304	74	0.410846	74	0.145799
	5	147	195	0.006027	1	147	0.378842	147	0.391997	147	0.15266
	6	94	156	0.004961	1	94	0.428725	94	0.420832	94	0.140788
	7	144	196	0.006223	1	144	0.414412	144	0.413769	144	0.141206
	8	126	110	0.003537	1	126	0.412513	126	0.421714	126	0.150554
	9	128	132	0.004153	1	128	0.443888	128	0.448604	128	0.160451
	10	128	140	0.004861	1	128	0.403901	128	0.429997	128	0.136119
		No. of optimal			.	9		10		10	

Table 3-2: A comparison between the optimal solutions obtained by BAB algorithm and the values result of local search algorithms at n=7

BAB						Local Search					
n	Ex	Optimal	node	Time	Status	DM	Time	SA	Time	GA	Time
7	1	149	576	0.074127	1	149	0.451028	149	0.434465	149	0.176869
	2	181	1649	0.049056	1	181	0.44275	181	0.451889	181	0.157082
	3	155	1865	0.054323	1	155	0.448604	155	0.435941	155	0.155311
	4	319	7296	0.215803	1	331	0.44135	319	0.436842	319	0.160151
	5	159	993	0.030626	1	159	0.447995	159	0.446451	159	0.158759
	6	181	1444	0.044574	1	181	0.424566	181	0.432911	181	0.153841
	7	163	1013	0.029366	1	163	0.42765	163	0.429353	163	0.154091
	8	185	1401	0.044981	1	185	0.488119	185	0.430049	185	0.183691
	9	108	5168	0.153238	1	108	0.436393	108	0.458326	108	0.154093
	10	227	1853	0.053289	1	227	0.441418	227	0.429537	227	0.155878
		No.of optimal			.	9		10		10	

.Table 3-3: A comparison between the optimal solutions obtained by BAB algorithm and the values result of local search algorithms at n=9

BAB						Local Search					
n	Ex	Optimal	node	Time	Status	DM	Time	SA	Time	GA	Time
9	1	298	1512	0.103231	1	300	0.522179	298	0.530969	298	0.17119
	2	206	19404	0.669483	1	206	0.52964	206	0.534949	206	0.168965
	3	200	46645	1.29119	1	200	0.549212	200	0.537272	200	0.168887
	4	341	47239	1.304335	1	341	0.576686	341	0.527742	341	0.16837
	5	196	16954	0.461582	1	196	0.528663	196	0.535685	196	0.167691
	6	304	24716	0.726086	1	304	0.538602	304	0.51874	304	0.167103
	7	140	16888	0.500476	1	140	0.534924	140	0.526103	140	0.168946
	8	378	98906	2.8047	1	378	0.527394	378	0.526531	378	0.16809
	9	322	27287	0.743506	1	322	0.500353	322	0.511782	322	0.165107
	10	301	30011	0.829983	1	301	0.520649	301	0.518089	301	0.166214
		No. of optimal			.	9		10		10	

In table3-1,2,3 the number of examples that gives local search values equal to the optimal value is 9 for DM, 10 for SA and 10 for GA when n =5 , n=7 and n =9

Table 3-4: A comparison between the optimal solutions obtained by BAB algorithm and the values result of local search algorithms at n=11

BAB						Local Search					
n	Ex	0ptimal	Node	Time	Status	DM	Time	SA	Time	GA	Time
11	1	380	862639	25.53819	1	383	0.561954	380	0.56491	380	0.193609
	2	614	4883141	146.780577	1	627	0.555486	618	0.565943	614	0.18624
	3	436	1138123	34.5071715	1	437	0.58563	436	0.60058	436	0.183437
	4	340	225082	7.0207728	1	346	0.561673	340	0.558479	340	0.189434
	5	393	666300	19.4370768	1	393	0.541898	393	0.558525	393	0.179975
	6	528	284273	8.5266774	1	528	0.555718	528	0.555971	528	0.233818
	7	451	722008	22.042973	1	452	0.588117	451	0.652591	451	0.183895
	8	284	483145	13.8482672	1	284	0.565228	284	0.557501	284	0.186993
	9	478	846894	25.8682167	1	478	0.565022	478	0.55208	478	0.184799
	10	588	3435594	103.426301	1	588	0.569821	588	0.549853	588	0.184909
		No. of optimal			.	5		9		9	

Table 3-5: A comparison between the optimal solutions obtained by BAB algorithm and the values result of local search algorithms at n=13

BAB						Local Search					
n	Ex	0ptimal	Node	Time	Status	DM	Time	SA	Time	GA	Time
13	1	792	43497668	1329.71367	1	797	0.72278	799	0.669855	793	0.2453
	2	569	25103067	774.251406	1	570	0.575887	569	0.61263	569	0.201711
	3	483	24079250	733.735107	1	483	0.6511	485	0.579348	485	0.246121
	4	490	9327944	342.161331	1	490	0.625585	491	0.590707	490	0.248464
	5	645	41447845	1800.00002	0	645	0.706026	649	0.698971	645	0.247236
	6	689	40341289	1800.00003	0	690	0.664823	690	0.699526	689	0.198438
	7	725	40336942	1800.00011	0	725	0.678124	729	0.694032	725	0.195141
	8	586	20875539	940.108954	1	586	0.673539	588	0.589377	586	0.244738
	9	485	20710738	876.02978	1	485	0.695372	486	0.697011	485	0.245802
	10	846	41276248	1800.00009	0	846	0.725083	851	0.706807	846	0.247795
		No. of optimal			.	2		2		6	

Table 3-6: The values result of local search algorithms at n=100

Ex	Best	DM	Time	SA	Time	GA	Time
1	28207	28260	1.842397	28778	1.856365	28207	1.729882
2	31853	32315	1.974563	32872	2.04155	31853	1.993012
3	32895	33109	1.926219	33791	2.516388	32895	2.018078
4	25340	25486	2.085834	26146	1.890639	25340	1.666724
5	26423	26439	1.922907	27173	1.805007	26423	1.684188
6	33295	33295	1.747778	33380	1.768575	33306	1.625732
7	33454	33484	2.086676	33610	1.73712	33460	2.485722
8	32158	32329	1.853215	32797	1.093407	32158	1.622862
9	29264	29396	2.168188	29738	1.151685	29264	1.651836
10	28352	28389	2.153944	28681	1.804689	28352	2.235184
	No. of best	2		0		8	

In this table, the number of examples that gives the best known solution yet is 2 for DM, 0 for SA and 8 for GA.

Table 3-7: The values result of local search algorithms at n=500

Ex	Best	DM	Time	SA	Time	GA	Time
1	714129	714129	8.0811966	721116	7.42512	715217	33.567855
2	658590	660702	9.442675	673628	8.334985	658590	31.376931
3	752400	752400	8.6926941	757187	7.284213	753253	32.96676
4	737029	737029	9.1847774	746586	8.450665	737878	31.50552
5	675646	675646	7.2728892	684939	7.187699	678507	34.525774
6	735378	735378	7.5283349	740941	7.185011	736329	35.142774
7	749799	749799	7.1421142	754688	7.104844	751863	29.485768
8	759633	759663	12.582026	764393	8.055614	761132	35.992534
9	889358	889358	8.2353456	890449	8.127139	889899	35.726727
10	932929	932929	8.053281	932948	8.207121	932938	33.884866
	No.of best	9		0		1	

Table 3-8: The values result of local search algorithms at n=1000

Ex	Best	DM	Time	SA	Time	GA	Time
1	2553790	2572352	15.882113	2578112	16.374763	2553790	128.369943
2	2946110	2951049	15.638138	2953360	15.594807	2946110	130.81592
3	2991605	2996568	15.91777	2999152	16.146398	2991605	132.545383
4	2838114	2842095	16.198844	2844309	16.044214	2838114	123.581505
5	2896048	2905992	14.980914	2910830	15.349176	28960448	123.017859
6	3173478	3174271	14.418465	3175709	14.149838	3173478	128.410823
7	3484831	3485135	15.917629	3486448	14.154318	3484831	130.947455
8	2886778	2895780	14.918354	2900328	14.784146	2886778	123.149516
9	3372625	3374208	14.817342	3376331	14.878303	3372625	120.304961
10	3216306	3217585	13.767239	3219015	13.988947	3216306	115.164895
	No. of best	0		0		10	

Table 3-9: The values result of local search algorithms at n=5000

Ex	Best	DM	Time	SA	Time	GA	Time
1	73524007	73536677	79.942571	73536865	71.185332	73524007	605.118035
2	63207491	63255141	75.938803	63255660	76.811952	63207491	612.660498
3	66903616	67113370	76.183674	67113824	78.896933	66903616	618.602769
4	79680417	79686076	75.849555	79686221	75.530779	79680417	615.510455
5	67804848	68036332	76.48351	68036831	77.186682	67804848	615.643608
6	70477608	70491804	77.321727	70492253	77.91059	70477608	606.714346
7	73442537	73448819	74.704688	73449028	74.50419	73442537	622.650479
8	91175307	91175790	61.24575	91175828	77.452048	91175307	616.703271
9	93941993	93941993	59.54845	93941995	71.766329	93941994	611.5997935
10	76716103	76721985	60.44667	76722261	74.890752	76716103	604.545843
	No. of best	0		0		10	

Table 3-10: The values result of local search algorithms at n=10000

Ex	Best	DM	Time	SA	Time
1	289606034	289606034	155.2902	289606135	147.3921
2	273653864	273653864	153.0772	273653987	145.3847
3	285225751	285225751	154.83	285225807	147.4262
4	284408782	284408782	162.3384	284408938	151.158
5	276890195	276890195	154.9541	276890298	162.6316
6	336633779	336633779	137.8737	336633828	144.5711
7	321570865	321570865	143.1081	321570917	135.8353
8	346921027	346921027	146.2184	346921063	145.0729
9	299196329	299196329	145.0793	299196441	147.0557
10	305059249	305059249	147.2527	305059347	146.8354
	No.of best	10		0	

In this table, the number of examples that gives the best solution is 10 for DM and 0 for SA.

Table 3-11: The values result of local search algorithms at n=20000

Ex	Best	DM	Time	SA	Time
1	1198810298	1198810298	237.7489	1198810352	240.5755
2	1166318774	1166318774	240.2549	1166318816	235.574
3	1035314962	1035314962	242.8797	1035315030	239.9499
4	1166864162	1166864162	238.9088	1166864226	240.2091
5	1234855812	1234855812	239.3137	1234855850	236.8713
6	1170262797	1170262797	239.1409	1170262828	237.8809
7	1146405539	1146405539	240.5716	1146405586	241.4909
8	1246434331	1246434331	237.116	1246434352	235.1815
9	1366225692	1366225692	235.0879	1366225709	233.0021
10	1545672927	1545672927	229.7283	154672927	228.244
	No.of best	10		1	

Table 3-12: The values result of local search algorithms at n=30000

Ex	Best	DM	Time	SA	Time
1	2316579635	2316579635	368.9691	2316579694	365.4173
2	2371616071	2371616071	7000.079	2371616117	367.3791
3	2754714509	2754714509	493.5022	2754714539	492.3983
4	2912393595	2912393595	477.308	2912393611	472.8284
5	2568730859	2568741545	10916.23	2568730859	1637.515
6	3082704178	3082704178	386.4808	3082704201	600.0024
7	3073066706	3073066706	372.3755	3073066715	371.7001
8	2838453378	2838453378	363.0306	2838453387	356.1615
9	2906051952	2906051952	470.5902	2906051976	356.743
10	3165132308	3165132308	481.673	3165132313	468.6973
	No. of best	9		1	

Table 3-13: The values result of local search algorithms at n=40000

Ex	Best	DM	Time	SA	Time
1	4517342015	4517342015	468.3425	4517342036	463.6928
2	4342910570	4342910570	483.8663	4342910604	468.9732
3	4697900554	4697900554	463.0457	4697900560	466.0637
4	4239059324	4239059324	480.22	4239059359	467.2866
5	5175147897	5175147897	466.3659	5175147903	461.4844
6	4470139888	4470139888	505.815	4470139898	465.3543
7	4819997499	4819997499	463.9291	4819997510	466.0675
8	4717213474	4717213474	477.5863	4717213490	461.3611
9	5441662736	5441662736	460.291	5441662746	459.9377
10	5058154973	5058154973	463.9767	5048154987	460.7368
	No. of best	10		0	

3.4 Conclusions and Future Work

3.4.1 Conclusions

This thesis proposes an effective branch and bound (BAB) algorithm to find the best solution to the problem of reducing a $\sum_{j=1}^{n} Cj + \sum_{j=1}^{n} Tj + E_{max}$. On a large number of test problems, the (BAB) method is used. The BAB algorithm is efficient for $n \in \{5,7,9,11,13\}$, as evidenced by the computed values. Finding approximation solutions for the problem can also be achieved by applying simulated annealing (SA) , local search algorithms descent method (DM) and Genetic algorithm (GA). On a broad set of test problems, a computational experiment for local search algorithms is presented. The Genetic algorithm (GA) is more effective for problem of size n= 100,500,1000,5000. The descent method (DM) is much more successful for problems of large size n=10000,20000,30000,40000. Where the computational time of DM is close to that of SA. This is the most important we can derive from our computational results.

3.4.2 Future Work

An interesting future research topic would include the development of the lower bound (LB), the improvement of upper bound (UB) by using the results of local search algorithms in order to improve the efficiency of BAB algorithm and experimentation with Tabu Search (TS) algorithms, and use multi-start SA algorithm.

References

References

[1] Pindeo, M., "Scheduling; Theory; Algorithms and Systems", Prentice Hill, Inc., Englewood diffs, New Jersey 2nd edition, (2016).

[2] Al- Nuaimi A. A. M. , "Local search algorithms for multiobjective scheduling problem", Journal of AI- Rafidain University College 36: 201-217,2015.

[3] Evans G.W., "An overview at techniques for solving miltiobjective Mathematical programs", management science 30, 1268-1282, (1984).

[4] Emmons H., "A note on a scheduling problem with dual criteria", Naval Research Logis. Quarterly, 22,515-516,(1975).

[5] Lawler, E.L., "Optimal sequencing of a single machine subject to precedence constraints", management science 19,544-546, (1973).

[6] Franch S. "Sequencing and Scheduling an Introduction to Mathematics of Job Shop" , John Wiley & Sons, New York (1982).

[7] Hoogeveen H., "Single – machine bi-criteria scheduling", PhD Dissertation, Center for mathematics and Computer science, Amsterdam. The Netherlands, (1992).

[8] Hoogeveen H., "Invited review of Multicriteria scheduling", European Journal of Operational Research 167,592-623,(2005).

[9] Nager A., Jorge H., and Sunderesh H., "Multiple and bi-criteria Scheduling: A literature survery", European Journal of Operational Research North-Holland, 81, 88-104, (1995).

References

[10] Van Wassenhove, L.N. and Gelders, F., "Solving a bicriteria scheduling Problems", European Journal of Operational research, 4, 42-48, (1980).

[11] Al- Nuaimi A. A. M. , " A proposed algorithm to find efficient solutions for Multicriteria problem", Journal of Engineering and Applied Sciences 14(2): 5547-5549,2019.

[12] Al- Nuaimi A. A. M. , "Minimizing three hierarchically Criteria on a single machine", Diyala Journal for pure sciences 13(1): 14-22, 2017.

[13] Al- Nuaimi A. A. M. , " An algorithm for solving three criteria scheduling problem on a single machine", Int. J. Agricult. Stat. Sci, 14(1): 271-273, 2018.

[14] Bagchi T.P., "Multiobjective Scheduling by genetic algorithm", Kluwer Academic publisher, (1999).

[15] Hoogeveen, J.A., "Single machine scheduling to minimize a function of Two or three maximum cost criteria", Journal of Algorithms 21,415-433, (1996b).

[16] Lee J.K. and Kim Y.D., "Search heuristic for resource constrained project scheduling", Journal of the operation research society 47, 678-689 (1996).

[17] Lenstra,J.K., Rinnooy Kan A.H. G and Brucker P., "Complexity of Machine scheduling problems", Annals of operation research 1,343-362,(1977).

[18] Smith W. E., "Various Optimizers for Single Stage Production", Naval Research Logistics Quarter 3, 59-66, (1956).

[19] Jackson J.R., "Scheduling a production line to minimize maximum tardiness" Research Report 43 management science, Res. Project, University of California, Loss Angles, CA, (1955).

References

[20] Kanet J.J., "Minimizing the Average Deviation of jobs Completion Time about a Common Due Dates", Naval Res. Logistics Quarter 28,643-65 (1981).

[21] Mahmood A.A, "Solution procedures for scheduling job families with setup and due dates" M.S.c. thesis Univ. of Al-Mustansiriyah, College of Science, Dep. of Mathematics (2001).

[22] Conway R.W., Maxwell W.L., and Miller L.W. "Theory of Scheduling Addison ", Wesley, reading, Mass (1967).

[23] Baker K.R., and Scharge LE., "Finding an optimal sequence by dynamic programming: an execution to precedence-related tasks ", European journal of operational res. 26,111-120, (1978).

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

[25] Reeves C.R. "Modern Heuristic techniques for combinatorial Problems". John Wiley and Sons, Inc, New York, (1993).

[26] Tkindt V. and J.-C. Billaut, " Multicriteria scheduling theory, models and Algorithms", Springer, Berlin. 35 (2006) 143-163.

[27] Lee C.V and Vairaktarakis GL., "Complexity of single machine Hierarchical scheduling: a survey complexity in numerical Optimization", 19,269-298, world scientific (1993).

[28] Al-Assaf S.S.,"Solving multiple objectives scheduling problems", M.Sc. thesis Univ. of Al- Mustansiriyah, College of Science, Dep. of Mathematics (2007).

[29] Hoogeveen J.A., "Minimizing maximum promptness and maximum Lateness on a single machine", Mathematics of operation research. 21,100-114, (1996a).

References

[30] Garey, M.R., Tarjan, R.E., Wilfong. G.T., " One-processor Scheduling with symmetric earliness and tardiness penalties". Mathematics of Operations Research 13,330-348 (1988).

[31] Verma, S., Dessouky, M., "Single- machine scheduling of unit-time jobs with earliness and tardiness penalties". Mathematics of Operations Research 23,930-943, (1998).

[32] Fry T.D., Keong Leong, G., "Single machine scheduling": A Comparison of two solution procedures", Omega 15,277-282,(1987).

[33] Kim, Y.-D., Yano, C.A., "Minimizing mean tardiness and earliness in single machine scheduling problems with unequal due dates", Naval Research Logistics 41,913-933,(1994).

[34] Hoogeveen J.A., and Van de Velde, S.L., "A branch-bound algorithm for Single-machine earliness-tardiness scheduling with idle time", INFORMS Journal on Computing 8, 402-412,(1996).

[35] Sourd L., Kedad-Sidhoum F., " The one machine problem with earliness and tardiness penalties". Journal of Scheduling 6,533-49, (2003).

[36] Bulbul L., K., Kaminsky, and P., Yano, C., "Submitted for publication. Preemption in single machine earliness tardiness scheduling", Naval Research Logistics, Department of IEOR, University of California at Berkeley (2005).

[37] Sourd L. Kedad-Sidhoum F., " The one machine problem with earliness and tardiness penalties". Journal of Scheduling 6,533-49, (2003).

References

[38] Chen L., and Bulfin R.L., "Scheduling unit processing time jobs on a Single machine with multiple criteria", Computers and Operations Research 17,1-7, (1990).

[39] Hummadi L.Z., "Using Genetic algorithm to solve (NP-complete)\ problems". M.Sc. Thesis. College of Science, University of Al- Mustansiriyah, (2005).

[40] Kirkpatrick S.,Gelatt Jr, CD., and Vecchi MP., "Optimization by simulated annealing", Science 220,671-80,(1983).

[41] Mohammed, H. A.A. "Using Genetic and local search algorithms as a tool for providing optimality for job scheduling", M.Sc. Thesis, College of Science, University of Al-Mustansiriyah, (2005).

[42] Gupta J.N.D., Hennig K., Werner F., "Local search heuristics for two-stage flow shop problems with secondary criterion", Ball stat university, Muncie, IN43,306, USA. Pergaamon computer and operation research 29,123-149, (2002).

[43] Holland J. H. "Adaptation in Natural and Artificial Systems", Ann Arbor, University of Michigan Press, 1975.

[44] Al-Samarii N.A.A., "Signatures verification using neural network", M.Sc. thesis, College of Engineering, University of Al-Mustansiriyah, April-2004

[45] Liu N., Mohamed A. Abdelrahman, and Srini Ramaswamy," A Genetic Algorithm for the Single Machine Total Weighted Tardiness Problem", Tennessee Technological University, Cookeville, TN 38505, USA, 2003.

[46] Chen C.L., Vempati V.S., and Aljaber N., ,"An application of genetic algorithms for flow shop problems", European Journal of operation research, 80 389-396, (1995).

References

[47] Delia Croce F., Tadi R. and Valta G., "A Genetic algorithm for job shop problems", Computer and operation research, 22, 15-40 (1995).

[48] Reeves C.R., " A Genetic algorithm for flow shop sequencing", computer and operation research, 22,5-13 (1995).

[49] Crauwels, H. " A comparative study of local search methods for one machine sequence problem". Ph. D. thesis Katholieke University, Heverlee. Belgium (1998).

[50] Tate D.M., and Simth A.E., "A genetic approach to the quadratic assignment problem". Computer and operation. research. 22,73-83 (1995).

[51] Al-Nuaimi A.A.M., Optimal solution for simultaneous multicriteria problem, Diyala Journal for pure Sciences 12(2): 18-27,2016.

المستخلص

في هذه الرسالة ، يتم النظر في مسألة الجدولة متعددة المقاييس على ماكنة واحده. المقاييس الثلاثة المستخدمة هي وقت الاتمام الكلي $\sum C_j$ والتأخير الكلي $\sum T_j$ والحد الاقصى للوقت المبكر E_{max}.

نهدف في هذه الدراسة الى ايجاد الجدول الزمني الامثل والتقريبي للوظائف $j=1,2,\ldots,n$ ، j لتقليل دالة الهدف متعدد المقاييس للدالة $\sum C_j + \sum T_j + E_{max}$.

لحل هذه المسالة نستخدم خوارزمية التقيد والتفرع (BAB) لايجاد الحل الامثل للمسالة. وقمنا باشتقاق القيد الادنى (LB) لاستخدامه في خوارزمية التفرع والتقيد (BAB) . يتم اجراء التجارب الحسابية لخوارزمية BAB على مجموعة كبيرة من مشكلات الاختبار. هنا تظهر صعوبة NP لهذه المسألة اي انه ليس من الممكن دائما العثور على الحل الامثل بسرعة والمسألة حلت الى $n=13$.

لذلك بدلا من البحث عن الحل الامثل بجهد حسابي هائل نستخدم خوارزميات البحث المحلية لايجاد حلول قريبة من الحل الامثل مع وقت حسابي اقل. تم تطبيق ثلاث خوارزميات بحث محلية ، طريقة النسب (DM) والتلدين المحاكي (SA) والخوارزمية الوراثية (GA) لهذه المسالة.

تتم مقارنة الخوارزميات DM و SA و GA مع خوارزمية BAB من اجل تقيم فعالية طرق الحل. تمت صياغة الاستنتاجات على كفاءة الخوارزميات، بناءً على نتائج التجارب الحسابية.

.

جمهورية العراق

وزارة التعليم العالي والبحث العلمي

جامعة ديالى

كلية العلوم

قسم علوم الرياضيات

خوارزميات حل لمسألة امثلية متعددة المقاييس

رسالة مقدمة إلى مجلس كلية العلوم– جامعة ديالى

وهي جزء من متطلبات نيل شهادة الماجستير في علوم الرياضيات

من قبل الطالب

انمار صبري حسن

بكلوريوس علوم الرياضيات / كلية العلوم/ الجامعة المستنصيرية 2004

إشراف

ا.م.د عدوية علي محمود النعيمي

1442هـ **2021 م**