Journal of Applied Mathematics and Physics, 2024, 12, 661-682
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Algorithms for Multicriteria Scheduling Problems to Minimize Maximum Late Work, Tardy, and Early

Karrar Alshaikhli[1], Aws Alshaikhli[2]

[1]Department of Engineering and Mathematics, St. Philip's College, San Antonio, USA
[2]Department of Biology, Northeastern Illinois University, Chicago, USA
Email: karar88k@yahoo.com, alshaikhliaws4@gmail.com

How to cite this paper: Alshaikhli, K. and Alshaikhli, A. (2024) Algorithms for Multicriteria Scheduling Problems to Minimize Maximum Late Work, Tardy, and Early. *Journal of Applied Mathematics and Physics*, **12**, 661-682.
https://doi.org/10.4236/jamp.2024.122043

Received: December 28, 2023
Accepted: February 26, 2024
Published: February 29, 2024

Abstract

This study examines the multicriteria scheduling problem on a single machine to minimize three criteria: the maximum cost function, denoted by maximum late work ($V_{\max}$), maximum tardy job, denoted by ($T_{\max}$), and maximum earliness ($E_{\max}$). We propose several algorithms based on types of objectives function to be optimized when dealing with simultaneous minimization problems with and without weight and hierarchical minimization problems. The proposed Algorithm (3) is to find the set of efficient solutions for $1//F(V_{\max}, T_{\max}, E_{\max})$ and $1//(V_{\max} + T_{\max} + E_{\max})$. The Local Search Heuristic Methods (Descent Method (DM), Simulated Annealing (SA), Genetic Algorithm (GA), and the Tree Type Heuristics Method (TTHM) are applied to solve all suggested problems. Finally, the experimental results of Algorithm (3) are compared with the results of the Branch and Bound (BAB) method for optimal and Pareto optimal solutions for smaller instance sizes and compared to the Local Search Heuristic Methods for large instance sizes. These results ensure the efficiency of Algorithm (3) in a reasonable time.

Keywords

Scheduling, Single Machine, Hierarchical, Simultaneous Minimization, Algorithms, Branch and Bound, Local Search Heuristic Methods

1. Introduction

The multicriteria scheduling problem has gotten much attention lately [1]. The fundamentals of multicriteria are as follows. On a single machine, there are n jobs to be processed. Each job has a processing time (p_i) and due date (d_i) at which it should be finished. When a task is finished ahead of schedule or after,

penalties are assessed. Therefore, the problem is a multicriteria scheduling problem. Let Ω be a schedule, $V(\Omega)$, $T(\Omega)$, and $E(\Omega)$ be functions of late work, tardiness, and earliness, respectively. The problem is to find a schedule Ω to optimize $V(\Omega)$, $T(\Omega)$, and $E(\Omega)$ or a composite objective function of $V(\Omega)$, $T(\Omega)$, and $E(\Omega)$. In the literature [2], there are mainly three classes of approaches that are applicable to the multicriteria scheduling problem. C1: In the hierarchical approach, one of the criteria (more important) is regarded as a constraint (primary) criterion that must be satisfied, and the other one is considered as a (secondary) criterion to optimize. This means optimizing the primary criterion while breaking ties in favor of the schedule that has minimum secondary criterion [3]. C2: Minimizing a weighted sum of the bicriteria (objectives) and converting the bicriteria to a single criterion problem, several bicriteria scheduling problems studied belong to this class [4] [5]. C3: One typically generates all efficient (Pareto optimal) schedules and selects the one that yields the best composite objective function value of two criteria [6]. For the bicriteria that concern the simultaneous minimization of $(\Sigma C_i, f_{\max})$ for $1//F(\Sigma C_i, f_{\max})$ problem in C3, which is solved by Hoogeveen and Vand de Velde [4] in a polynomial time, Vanwassenhove and Gelder [7] solved the $1//F(\Sigma C_i, T_{\max})$ problem.

This paper will extend the bicriteria $1//(V_{\max}, E_{\max})$ [6] to multicriteria problem $1//(V_{\max}, T_{\max}, E_{\max})$, which is an NP-hard problem, to get past these limitations, we can apply heuristic approaches. We suggest algorithm (3) to find optimal solutions, and we will study the problem in classes C1, C2, and C3. The rest of the paper is organized as follows:

Section (2), gives notations, basic concepts, and mathematical forms. In sections (3) and (4), we formulate the multicriteria problem according to the classes of the approaches, we propose algorithms for each problem and their particular cases, and we study the Branch and Bounded Method for $1//(V_{\max} + T_{\max} + E_{\max})$. In section (5), we suggested Local Search Heuristic methods to find approximate solutions, and the experimental results are given in section (6).

2. Notation, Basic Concepts and Mathematical Forms

2.1. Notation and Basic Concepts

The following notation will be used in this paper:

n = number of jobs.

p_i = processing time of job i.

d_i = due date of job i.

C_i = The completion time, the time at which the processing of job j is completed s.t.

$C_i = \sum_{j=1}^{i} p_i$

E_i = the earliness of the job i.

T_i = the tardiness of the job i.

V_i = the late work penalty for job i.

$E_{\max}$ = Max$\{E_i\}$, the maximum early work.

T_{max} = Max{T_i}, the maximum Tardy work.

V_{max} = Max{V_i}, the maximum late work.

f_{max} = Max{f_i}, the maximum function.

LB = lower bound.

UB = upper bound.

In this paper, we shall use the following sequencing rules and concepts:

MST: Jobs are sequenced in nondecreasing order of ($s_i = d_i - p_i$), this rule is well known to minimize E_{max} for $1//E_{max}$ problem [8].

EDD: Jobs are sequenced in nondecreasing order of (d_i), this rule is well known to minimize T_{max} for $1//T_{max}$ problem [9].

Definition (1): The term" optimize" in a multiobjective decision making problem refers to a solution around which there is no way of improving any objective without worsening at least one other objective [10].

Definition (2):

A feasible schedule σ is **Pareto optimal**, or non-dominated (efficient), with respect to the performance criteria f and g if there is no feasible schedule σ such that both $f(\sigma) \le f(\sigma)$ and $g(\sigma) \le g(\sigma)$, where at least one of the inequalities is strict [7].

Lawler algorithm (LA): [11]

Step (1): let $N = \{1, \cdots, n\}$, $\Omega = (\varnothing)$ and M be the set of all jobs with no successors.

Step (2): let j^* such that $fj^*(\Sigma p_i) = \min\{fj(\Sigma p_i)\}$, $j \in M$

Set $N = N - \{j^*\}$ and sequence the job j^* in the last position of Ω.

Modify M to represent the new set of scheduled jobs.

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

E^*_{max} = Minimum E_{max} by MST rule.

T^*_{max} = Minimum T_{max} by EDD rule.

V^*_{max} = Minimum V_{max} by LA.

2.2. The Mathematical Forms and Their Algorithms

2.2.1. Hierarchical Problems

We present the mathematical forms and the algorithms for generating solutions when one of three criteria (V_{max}, T_{max}, E_{max}) is more important than the others. These hierarchical problems are called secondary criteria problems, where the secondary criteria refer to the less important ones. The formulation for multicriteria problems is similar to that for the single criteria problems, which require that the optimal value of the primary objective is not violated.

Let us first consider the formulations for multicriteria hierarchical problems, say

1//Lex ($\gamma 1$, $\gamma 2$, $\gamma 3$). There are three parts of the formulations

- Primary objective function ($\gamma 1$)

  Subject to: Secondary objective function ($\gamma 2$)
- Secondary objective function ($\gamma 2$)

  Subject to: Primary objective function ($\gamma 1$)

- Secondary objective function ($\gamma 3$)

Subject to: Primary objective function ($\gamma 1$)

Hence the algorithm for solving the multicriteria problem needs two steps:

Step (1): We optimize $\gamma 1$, followed by

Step (2): The optimization of $\gamma 2$, and $\gamma 3$ subject to the primary objective value $\gamma 1$. For example, if $V_{\max}$ is more important than $T_{\max}$ and $E_{\max}$, then the 1//Lex ($V_{\max}$, $T_{\max}$, $E_{\max}$) problem section (3.1) can be written as Min($E_{\max}$)

s.t. $V_{\max} = \Delta$, where $\Delta = V_{\max}(\mathrm{LA})$

$T_{\max} \le T^*, T^* \in \left(T_{\max}(\mathrm{LA}), T_{\max}(\mathrm{MST})\right)$

2.2.2. Simultaneous Problems

Many algorithms can solve multicriteria scheduling problems to find efficient solutions or at least approximations of them [12]. The running time for the algorithm often increases with the increase of the instance size. Any algorithm process aims to find an optimal solution for each instance that minimizes the objective function. This usual meaning of the optimum makes no sense in the multicriteria case because it does not exist, in most cases, as a solution optimizing all objectives simultaneously. Hence, we search for feasible solutions yielding the best compromise among objectives constituting a so-called efficient solution set. These efficient solutions can only be improved in one objective by decreasing their performance in at least one of the others. This efficient solution set is challenging to find. Therefore, approximating that set in a reasonable time could be preferable.

3. Problem Formulation and Analysis

The problem of scheduling a set $N = \{1, \cdots, n\}$ of n jobs on a single machine to minimize multicriteria may be stated as follows. Each job $i \in N$ is to be processed on a single machine that can handle only one job at a time, job i has a processing time p_i and due date d_i. All jobs are available for processing at a time zero.

If a schedule $\Omega = (1, \cdots, n)$ is given, then a completion time $C_i = \sum_{j=1}^{i} p_i$ for each job i can be computed and consequently an earliness $E_i = \max\{d_i - C_i, 0\}$, $E_{\max} = \max\{E_i\}$ for each i and $E_{\max}^w = \max\{w_i E_i\}$, where w_i is the important of the job i with respect to other jobs. The tardiness $T_i = \max\{C_i - d_i, 0\}$, $T_{\max} = \max\{T_i\}$ for each i. The late work $V_i(\Omega)$ for the job $i \in N$ which is the amount of processing performed on job i after its due date d_i is easy to compute,

- If $V_i = 0$, then job i is early with $C_i \le d_i$
- If $0 < V_i < p_i$, then job i is partially early
- If $V_i = p_i$, then job i is late with $C_i \ge d_i + p_i$

This means that

$$V_i = \begin{cases} 0 & \text{if } C_i \le d_i,\ i = 1, \cdots, n \\ C_i - d_i & \text{if } d_i < C_i < d_i + p_i,\ i = 1, \cdots, n \\ p_i & \text{if } C_i > d_i + p_i,\ i = 1, \cdots, n \end{cases}$$

Hence $V_{\max}^{w} = \max\{w_i V_i\}$, w_i is the important of job i with respect to other jobs.

Our object is to find a schedule that minimizes bicriteria for the following problems:

1) 1//Lex ($V_{\max}$, $T_{\max}$, $E_{\max}$) problem (P1) $\in$ C1

2) 1//Lex ($V_{\max}^{w}$, $T_{\max}$, $E_{\max}$) problem (P2) $\in$ C1

3) 1//F($V_{\max}$, $T_{\max}$, $E_{\max}$) problem (P3) $\in$ C3

4) 1//F($E_{\max}^{w}$, $T_{\max}$, $E_{\max}$) problem (P4) $\in$ C3

5) 1//($V_{\max}$ + $T_{\max}$ + $E_{\max}$) problem (P5) $\in$ C2

3.1. [1//Lex ($V_{\max}$, $T_{\max}$, $E_{\max}$) Problem (P1)]

This problem can be written as:

$\text{Min}\left(E_{\max}\right)$

s.t. $V_{\max} = \Delta$, where $\Delta = V_{\max}\left(\text{LA}\right)$

$T_{\max} \le T^{*}, T^{*} \in \left(T_{\max}\left(\text{LA}\right), T_{\max}\left(\text{MST}\right)\right)$

Algorithm (1) for problem (P1):

Step (0): Using Lawler algorithm (LA) to find optimal $V_{\max}$, and set $\Delta = V_{\max}$ (LA).

Step (1): Set $N = \{1, \cdots, n\}$, $t = \Sigma p_i$, $\forall i \in N$.

Step (2): Solve $1/V_i \le \Delta/V_{\max}$ problem to determine job j to be the job completed at time t, such that: 1) $V_j \le \Delta$

2) $S_j \ge S_i$, $\forall j, i \in N$ and $V_i \le \Delta$.

Step (3): Schedule j in the interval $[t - p_j, t]$

Step (4): Set $N = N - \{j\}$, $t = t - p_j$

Step (5): If $t > 0$, then go to step (2)

Step (6): Stop.

Example (1): Consider the problem (P1) with the following data:

$p_i = (2, 3, 5, 7)$, $d_i = (11, 7, 18, 9)$ and $i = 1, 2, 3, 4$

Lawler algorithm (LA) gives the sequence (2, 4, 1, 3), with $V_{\max} = 1$, $T_{\max} = 1$ & $E_{\max} = 4$. Set $\Delta = 1$, we get the sequence (2, 4, 1, 3) gives $V_{\max} = 1$, $T_{\max} = 1$ & $E_{\max} = 4$. This sequence is optimal since the optimal sequence (2, 4, 1, 3) with $V_{\max} = 1$, $T_{\max} = 1$ & $E_{\max} = 4$ is obtained by the complete enumeration method (CEM).

3.2. [1//Lex ($V_{\max}^{w}$, $T_{\max}$, $E_{\max}$) Problem (P2)]

This problem can be written as:

$\text{Min}\left(E_{\max}\right)$

s.t. $V_{\max}^{w} = \Delta$, where $\Delta = V_{\max}^{w}\left(\text{LA}\right)$

$T_{\max} \le T^{*}, T^{*} \in \left(T_{\max}\left(\text{LA}\right), T_{\max}\left(\text{MST}\right)\right)$

Algorithm (2) for problem (P2):

Step (0): Using Lawler algorithm (LA) to find $V_{\max}^{w}$ and set $\Delta = V_{\max}^{w}\left(\text{LA}\right)$.

Step (1): Set $N = \{1, \cdots, n\}$, $t = \Sigma p_i$, $\forall i \in N$

Step (2): Solve $1/V_i^{w} = \Delta/V_{\max}^{w}$ problem where $V_i^{w} = W_i V_i$ to determine job j to be the job completed at time t, such that: 1) $W_j V_j \le \Delta$

2) $S_j \geq S_i$, $\forall j,i \in N$ and $W_i V_i \leq \Delta$

Step (3): Schedule j in the interval $[t-p_j, t]$

Step (4): Set $N = N - \{j\}$, $t = t - p_j$

Step (5): If $t > 0$, then go to step (2)

Step (6): Stop.

Example (2): Consider the problem (P2) with the following data: p_i = (4, 6, 2, 5), d_i = (20, 9, 4, 7), W_i = (4, 6, 2, 5), and i = 1, 2, 3, 4 Lawler algorithm (LA) gives the sequence (4, 2, 3, 1) With $V_{\max}^{w}$ = 12, $T_{\max}$ = 9, and $E_{\max}$ = 3, ($V_{\max}^{w}$, $T_{\max}$, $E_{\max}$)= (12, 9, 3), Set Δ = 12, we get the sequence (4, 2, 3, 1) gives $V_{\max}^{w}$ = 12, $T_{\max}$ = 9, and $E_{\max}$ = 3, ($V_{\max}^{w}$, $T_{\max}$, $E_{\max}$)= (12, 9, 3) is optimal.

3.3. [1//F ($V_{\max}$, $T_{\max}$, $E_{\max}$) Problem (P3)]

Multicriteria scheduling refers to the scheduling problem in which advantages of a particular schedule are evaluated using more than one performance criterion. Several scheduling problems considering the simultaneous minimization of various forms of sum completion time, earliness and tardiness costs have been studied in the literature [13], also solves 1//F ($f_{\max}$, $g_{\max}$) and solves the general problem 1//F ($f_{\max}^{1},\cdots, f_{\max}^{k}$), k is finite integer number and each one of these functions is assumed to be non-decreasing in the job completion time. Now consider the multicriteria problem 1//F ($V_{\max}$, $T_{\max}$, $E_{\max}$) in which $E_{\max}$ is not decreasing in job completion time. This problem belongs to C3 and is written as:

$$\text{Min}\{V_{\max}, T_{\max}, E_{\max}\}$$

$$\text{s.t. } V_i = \text{Min}\{p_i, T_i\}, \quad i = 1,\cdots,n$$

$$E_i \geq d_i - C_i, \quad i = 1,\cdots,n$$

$$E_i \geq 0, \quad i = 1,\cdots,n$$

$$T_i \geq C_i - d_i, \quad i = 1,\cdots,n$$

$$T_i \geq 0, \quad i = 1,\cdots,n$$

The following algorithm (3) is used to solve the problem (P3)

Algorithm (3) for the problem (P3):

Step (0): Determine the point ($V_{\max}^{*}$, $T_{\max}$, $E_{\max}$), ($V_{\max}$, $T_{\max}^{*}$, $E_{\max}$), and ($V_{\max}$, $T_{\max}$, $E_{\max}^{*}$) by solving 1//$V_{\max}$ by Lawler algorithm (LA), 1//$T_{\max}$ by EDD and 1//$E_{\max}$ by MST rule. Let SE be the set of efficient (Pareto) solutions, set SE = {($V_{\max}^{*}$, $T_{\max}$, $E_{\max}$), ($V_{\max}$, $T_{\max}^{*}$, $E_{\max}$), and ($V_{\max}$, $T_{\max}$, $E_{\max}^{*}$)} if each point is not dominated by the other. Set SUM = min {$V_{\max}^{*} + T_{\max} + E_{\max}$, $V_{\max} + T_{\max}^{*} + E_{\max}$, $V_{\max} + T_{\max} + E_{\max}^{*}$}.

Step (1): Set Δ = $V_{\max}$ (MST).

Step (2): Solve 1/ $V_i \leq \Delta$/ $V_{\max}$ problem by using LA (break tie to schedule the job j last with maximum $s_j = d_j - p_j$; let ($V_{\max}^{L}$, $T_{\max}^{L}$, $E_{\max}^{L}$) denote the outcome. Add ($V_{\max}^{L}$, $T_{\max}^{L}$, $E_{\max}^{L}$) to the set of Pareto optimal points (SE), unless it is dominated by the previously obtained Pareto optimal points. If SUM is greater than $V_{\max}^{L} + T_{\max}^{L} + E_{\max}^{L}$, then set SUM = $V_{\max}^{L} + T_{\max}^{L} + E_{\max}^{L}$. Let Δ = $V_{\max}^{L} - 1$, if Δ > 0 repeat step (2), otherwise go to step (3).

Step (3): The Pareto optimal set SE has been obtained and SUM which is the

minimum of values for the Pareto points in the set SE.

Step (4): Stop.

Example (3): Consider the problem (P3) with the following data: p_i = (1, 4, 8, 5), d_i = (20, 7, 11, 9) and i = 1, 2, 3, 4 MST gives the sequence (2, 3, 4, 1) with $E^*_{\max}$ = 3, $T_{\max}$ = 8 & $V_{\max}$ = 5; ($V_{\max}$, $T_{\max}$, $E^*_{\max}$) = (5, 8, 3). EDD gives the sequence (2, 4, 3, 1) with $E_{\max}$ = 3, $T^*_{\max}$ = 6 & $V_{\max}$ = 6; ($V_{\max}$, $T^*_{\max}$, $E_{\max}$) = (6, 6, 3). Lawler algorithm gives the sequence (4, 3, 2, 1) with $E_{\max}$ = 4, $T_{\max}$ = 10 & $V^*_{\max}$ = 4; ($V^*_{\max}$, $T_{\max}$, $E_{\max}$) = (4, 10, 4). Set SE = {(5, 8, 3), (6, 6, 3), (4, 10, 4)} & SUM = 15, set Δ = 5, we get the sequence (3, 2, 4, 1) which $E_{\max}$ = 3, $T_{\max}$ = 8 & $V_{\max}$ = 5; ($V_{\max}$, $T_{\max}$, $E_{\max}$) = (5, 8, 3), then the set SE remains the same, let Δ = 5-1=4, we get the sequence (3, 4, 2, 1) gives $E_{\max}$ = 3, $T_{\max}$ = 10 & $V_{\max}$ = 4; ($V_{\max}$, $T_{\max}$, $E_{\max}$) = (4, 10, 3), then the set SE = {(5, 8, 3), (6, 6, 3), (4, 10, 3)}. Let Δ = 4 − 1 = 3, There is no $V_j \le \Delta$, then we stop. The Set of efficient solutions is SE = {(5, 8, 3), (6, 6, 3), (4, 10, 3)} & SUM = 15.

Note that algorithm (3) does not find all the efficient solutions, but it finds most of them as shown in the following example.

Example (4): Consider the problem (P3) with the following data: p_i = (7, 14, 3, 1), d_i = (16, 20, 8, 2) and i = 1, 2, 3, 4 The algorithm (4) gives SE = {(7, 9, 4), (3, 17, 8), (5, 5, 5)}, but the exact set of efficient solutions which is obtained by complete enumeration method is SE = {(7, 9, 4), (3, 17, 8), (5, 5, 5), (4, 23, 6)}.

Note: We can use the BAB method to find the set of all efficient solutions.

3.4. [1//F ($E^w_{\max}$, $T_{\max}$, $V_{\max}$) Problem (P4)]

This problem is denoted by:

$$\text{Min}\left\{E^w_{\max}, T_{\max}, V_{\max}\right\}$$

$$\text{s.t. } V_i = \min\left\{p_i, T_i\right\},\ i = 1,\cdots,n$$

$$W_i E_i \ge W_i\left(d_i - C_i\right),\ i = 1,\cdots,n$$

$$W_i E_i \ge 0,\ i = 1,\cdots,n$$

$$T_i \ge C_i - d_i,\ i = 1,\cdots,n$$

$$T_i \ge 0,\ i = 1,\cdots,n$$

The following algorithm (4) is used to solve the problem (P4)

Algorithm (4) for problem (P4):

Step (0): Determine the point ($E^w_{\max}$, $T_{\max}$, $V^*_{\max}$), ($E^w_{\max}$, $T^*_{\max}$, $V_{\max}$), and ($E^{w*}_{\max}$, $T_{\max}$, $V_{\max}$) by solving 1// $V_{\max}$ by Lawler algorithm (LA), 1// $T_{\max}$ by EDD, and 1// $E^w_{\max}$ by WMST rule. Let SE be the set of efficient (Pareto) solutions, set SE = {($E^w_{\max}$, $T_{\max}$, $V^*_{\max}$), ($E^w_{\max}$, $T^*_{\max}$, $V_{\max}$), ($E^{w*}_{\max}$, $T_{\max}$, $V^*_{\max}$)}.

Step (1): Set Δ = $V_{\max}$ (WMST)

Step (2): Solve 1/ $V_i \le \Delta$/ $V_{\max}$ problem by using Lawler algorithm (break tie to schedule the job j last with maximum $S_j W_j = (d_j - p_j) W_j$; let ($E^{w(L)}_{\max}$, $V^L_{\max}$) denote the outcome. Add $E^{w(L)}_{\max}$, $V^L_{\max}$ to the set of Pareto optimal Points (SE), unless it is dominated by the previously obtained Pareto optimal points. Let Δ = $V^L_{\max}$ − 1, if Δ > 0 repeat step (2), otherwise go to step (3).

($E^{w(L)}_{\max}$, $V^L_{\max}$) to the set of Pareto optimal Points (SE), unless it is dominated by

the previously obtained Pareto optimal points.

Let $\Delta = V_{\max}^{L} - 1$, if $\Delta > 0$ repeat step (2), otherwise go to step (3).

Step (3): The Pareto optimal set SE has been obtained with values for the Pareto points.

Step (4): Stop.

Note Since the $1//E_{\max}^{w}$ problem cannot always be solved to optimality by the WMST $\{S_i W_i = (d_i - p_i) W_i\}$ rule, hence the algorithm (4) does not give the set of all efficient solutions [9].

Example (5): Consider the problem (P4) with the following data:

$p_i = (7, 3, 2, 7)$, $d_i = (15, 9, 4, 16)$, $W_i = (6, 3, 12, 1)$ and $i = 1, 2, 3, 4$. The WMST gives the sequence (4, 2, 3, 1) with $E_{\max}^{w} = 9$, $T_{\max} = 8$ & $V_{\max} = 4$, $(E_{\max}^{w}, T_{\max}^{*}, V_{\max}) = (9, 8, 4)$, EDD gives the sequence (3, 2, 1, 4) with $E_{\max}^{w} = 24$, $T_{\max} = 3$ & $V_{\max} = 3$, $(E_{\max}^{w*}, T_{\max}, V_{\max}) = (24, 3, 3)$, and Lawler algorithm (LA) gives the sequence (2, 1, 4, 3) with $E_{\max}^{w} = 30$, $T_{\max} = 15$ & $V_{\max}^{*} = 2$, $(E_{\max}^{w}, T_{\max}, V_{\max}^{*}) = (30, 15, 2)$.

Then SE = {(9, 8, 4), (24, 3, 3), (30, 15, 2)}.

Set $\Delta = V_{\max}$(WMST) = 4, we get the sequence (3, 4, 1, 2) gives $E_{\max}^{w} = 24$, $T_{\max} = 10$ & $V_{\max} = 3$, $(E_{\max}^{w}, T_{\max}, V_{\max}) = (24, 10, 3)$. Then SE remains the same SE = {(9, 8, 4), (30, 15, 2), (24, 3, 3)}

Set $\Delta = 2$, we get the sequence (4, 2, 1, 3) gives $E_{\max}^{w} = 9$, $T_{\max} = 15$ & $V_{\max} = 2$, $(E_{\max}^{w}, T_{\max}, V_{\max}) = (9, 15, 2)$. Then SE = {(9, 8, 4), (9, 15, 2), (24, 3, 3)}.

Set $\Delta = 1$, There is no $V_j \le \Delta$, then we stop. The set of efficient solutions is SE = {(9, 8, 4), (9, 15, 2), (24, 3, 3)}.

4. [$1//(V_{\max} + T_{\max} + E_{\max})$ Problem (p5)]

The aim for the problem (P5) is to find processing order σ of the jobs on a single machine to minimize the sum of maximum late work, maximum tardiness, and maximum earliness (*i.e.* to minimize $V_{\max}(\sigma) + T_{\max}(\sigma) + E_{\max}(\sigma)$, $\sigma \in S$ (where S is the set of all feasible solutions)). It is clear that the problem (P5) is a special case of the problem (P4).

In this section we decompose the $1//V_{\max} + T_{\max} + E_{\max}$ problem into two subproblems with a simpler structure. For this problem let:

$$M = \min_{\sigma \in S} \left\{ V_{\max}(\sigma) + T_{\max}(\sigma) + E_{\max}(\sigma) \right\}.$$

The problem (P5) can be decomposed into three subproblems (SP_1), (SP_2) and (SP_3).

$$\left.\begin{aligned} & M_1 = \min\left\{\max\left\{V_{\sigma(i)}\right\}\right\} \\ & \text{s.t.} \\ & V_{\sigma(i)} = \begin{cases} 0 & \text{if } C_{\sigma(i)} \le d_{\sigma(i)},\ i = 1, \cdots, n \\ C_{\sigma(i)} - d_{\sigma(i)} & \text{if } p_{\sigma(i)} \le C_{\sigma(i)} \le d_{\sigma(i)} + p_{\sigma(i)},\ i = 1, \cdots, n \\ p_{\sigma(i)} & \text{if } C_{\sigma(i)} \ge d_{\sigma(i)} + p_{\sigma(i)},\ i = 1, \cdots, n \end{cases} \end{aligned}\right\} \quad SP_1$$

$$\left.\begin{array}{l} M_2 = \min\left\{\max\left\{T_{\sigma(i)}\right\}\right\} \\ \text{s.t.} \\ T_{\sigma(i)} = C_{\sigma(i)} - d_{\sigma(i)},\ i = 1,\cdots,n \\ T_{\sigma(i)} \geq 0,\ i = 1,\cdots,n \end{array}\right\} \quad \text{SP}_2$$

$$\left.\begin{array}{l} M_3 = \min\left\{\max\left\{E_{\sigma(i)}\right\}\right\} \\ \text{s.t.} \\ E_{\sigma(i)} = d_{\sigma(i)} - C_{\sigma(i)},\ i = 1,\cdots,n \\ E_{\sigma(i)} \geq 0,\ i = 1,\cdots,n \end{array}\right\} \quad \text{SP}_3$$

4.1. Derivation of Lower Bound (LB) for Problem (P5)

The lower bound (LB) is based on decomposing problem (P5) into three subproblems (SP_1), (SP_2) and (SP_3). Then calculate M_1 to be the minimum value for (SP_1), M_2 to be the minimum value for (SP_2), and M_3 to be the minimum value for (SP_3), then applying the following theorem:

Theorem (1): [5]

$M_1 + M_2 + M_3 \leq M$ where M_1, M_2, M_3, and M are the minimum objective function values of (SP_1), (SP_2), (SP_3), and (P5) respectively.

To get a lower bound LB for the problem (P5):

- For the subproblem (SP_1), we compute M_1 as a lower bound by sequencing the jobs using Lawler's algorithm (LA) to find the minimum maximum late work $V_{\max}$.
- For the subproblem (SP_2), we compute M_2 as a lower bound by sequencing the jobs using EDD order (*i.e.*, sequencing the jobs in non-decreasing order of d_i) to find the minimum maximum tardiness $T_{\max}$.
- For the subproblem (SP_3), we compute M_3 to be a lower bound by sequencing the jobs by MST order (*i.e.*, sequencing the jobs in non-decreasing order of $s_i = d_i - p_i$) to find the minimum maximum early job $E_{\max}$.

then applying theorem (1) to obtain:

$$\text{LB} = M_1 + M_2 + M_3$$

4.2. Heuristic Method to Calculate Upper Bound (UB) for the Problem (p5)

- A simple heuristic is obtained by sequencing the jobs using Lawler's algorithm (LA) to find $V^*_{\max}$, $T_{\max}$, and $E_{\max}$, then $UB_1 = V^*_{\max}$ (LA) + $T_{\max}$(LA) + $E_{\max}$(LA).
- UB_2 is obtained by ordering the jobs in EDD order, that is, sequencing the jobs i, ($i = 1,\cdots,n$) in non-decreasing order of d_i to find $V_{\max}$, $T^*_{\max}$, and $E_{\max}$, then $UB_2 = V_{\max}$(EDD) + $T^*_{\max}$ (EDD) + $E_{\max}$(EDD).
- UB_3 is obtained by ordering the jobs in MST order, that is, sequencing the jobs i, ($i = 1,\cdots,n$) in non-decreasing order of $s_i = d_i - p_i$ to find $V_{\max}$, $T_{\max}$, and $E^*_{\max}$, then $UB_3 = V_{\max}$(MST) + $T_{\max}$(MST) + $E^*_{\max}$ (MST). Then UB =

$\min\{UB_1, UB_2, UB_3\}$.

4.3. Branch and Bound (BAB) Method

Our BAB method is based on the forward sequencing branching rule for which nodes at level k of the search tree correspond to the initial partial sequence in which jobs are sequenced in first k positions [6].

The LB at any node is the cost of scheduling jobs (this cost depends on the objective function) and the cost of unsequenced jobs (this cost depends on the derived lower bound (LB)). At any level of the BAB method, if a node has LB ≥ UB, then this node is dominated.

If the branching ends at a complete sequence of jobs, then this sequence is evaluated, and if its value is less than the current (UB), this (UB) is reset to take that value. The procedure is then repeated until all nodes have been considered using backtracking. The backtracking procedure is the BAB method's movement from the lowest to the upper level. The (UB) at the end of this procedure is the optimum for our scheduling problem (P5). Hence, we get at least one optimal solution using the BAB method. The BAB method is improved by using efficient (LB), good (UB), and dominance rules. If it can be shown that an optimal solution can always be generated without branching from a particular node of the search tree, then that node is dominated and can be eliminated. Dominance rules usually specify whether a node can be eliminated before its (LB) is calculated. Dominance rules are beneficial when a node that has a (LB) that is less than the optimal solution can be eliminated.

Example (6): Consider the problem (P5) with the following data:

$p_i = (2, 5, 7, 4)$, $d_i = (6, 21, 10, 9)$ and $i = 1, 2, 3, 4$

Lawler's algorithm gives a sequence (4, 3, 1, 2) where $V^*_{\max} = 2$, $T_{\max} = 12$ & $E_{\max} = 5$, then $UB_1 = V^*_{\max}(LA) + T_{\max}(LA) + E_{\max}(LA) = 2 + 12 + 5 = 19$, and

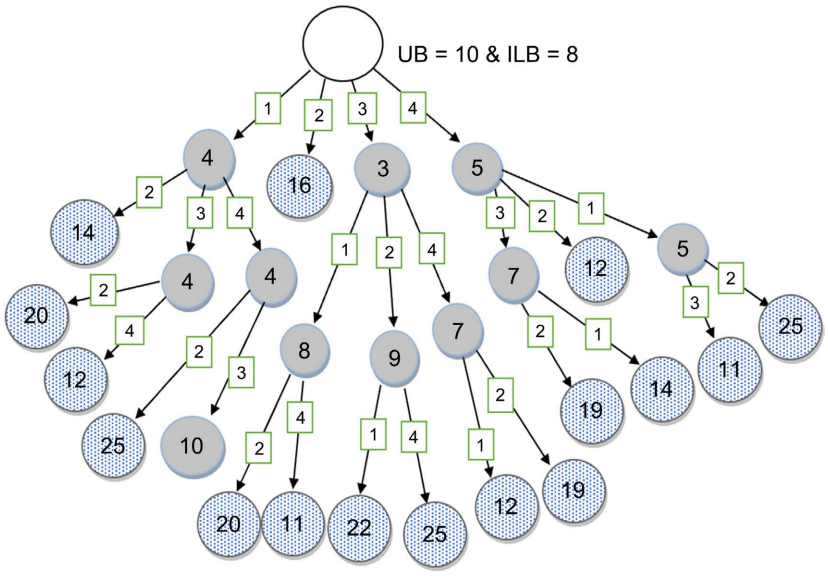

Figure 1. Branch and Bound (BAB) Method.

EDD rule gives a sequence (1, 4, 3, 2), where V_{max}(EDD) = 3, T^*_{max} (EDD) = 3 & E_{max}(EDD) = 4, then $UB_2 = V_{max}$(EDD) + T^*_{max} (EDD) + E_{max}(EDD) = 3 + 3 + 4 = 10, and MST rule gives a sequence (3, 1, 4, 2), where V_{max}(MST) = 4, T_{max}(MST) = 4 & E^*_{max} (MST) = 3, then $UB_2 = V_{max}$(MST) + T_{max}(MST) + E^*_{max} (MST) = 4 + 4 + 3 = 11

Hence the minimum upper bound is UB = min {UB_1, UB_2, UB_3} = 10

$LB_1 = V^*_{max}$ (LA) = 2, $LB_2 = T^*_{max}$ (EDD) = 3 & $LB_3 = E^*_{max}$ (MST) = 3

$ILB = LB_1 + LB_2 = 2 + 3 + 3 = 8$

We now give the BAB tree algorithm to find an optimal solution for (P5) in **Figure 1**.

5. Local Search Heuristic

In this section, several local search methods are implemented on the problem of scheduling n jobs on a single machine to minimize the sum of the maximum late work, maximum tardiness, and maximum earliness, *i.e.*, to minimize (V_{max} + T_{max} + E_{max}) for the problem (p5).

The local search provides approach high-quality solutions to NP-hard problems of realistic size in a reasonable time, and it's been widely used recently. Since it's simpler to construct algorithms using these techniques and get reasonable approximation results, several researchers employ them. The local search methods start with an initial solution and then continually try to add better solutions by searching neighborhoods. We proposed several local search methods and compared the results of these methods with the results of Algorithm (3) for 1//(V_{max} + T_{max} + E_{max}).

Definition: [14]

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

Definition: [15]

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

Glass and Potts (1995) [15] gave four possible neighborhoods below; each is illustrated by considering a typical neighbor of the sequence (1, 2, 3, 4, 5, 6, 7, 8) in a problem where there are eight jobs labeled 1, ···, 8.

(a) Transpose: Swap two adjacent jobs. Thus (1, 2, **4**, **3**, 5, 6, 7, 8) is a robhgien.

(b) Insert: Remove a job from one position in the sequence and insert it at another position (either before or after the original position). Thus, (1, **5**, 2, 3, 4, 6, 7, 8) and (1, 2, 3, 4, 6, 7, **5**, 8) are both neighbors.

(c) Swap: Swap two jobs that may not be adjacent. Thus, (1, **6**, 3, 4, 5, **2**, 7, 8) is a neighbor.

(d) Block Insert: Move a subsequence of jobs from one position in the sequence and insert it at another position. Thus, (1, **4**, **5**, 2, 3, 6, 7, 8) is a neighbor.

Definition: [2]

Let (*S*, *f*) be an instance of a combinatorial optimization problem, and let N^* be a neighborhood function. A solution $s^* \in S$ is called a local optimal (minimal) solution with respect to N^* if $f(s^*) \le f(s)$ 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.

5.1. Descent Method (DM)

This method is a simple form of a local search method [16]. It can be executed as follows:

1) Initialization:

In this step, a feasible solution $\sigma = (\sigma(1), \cdots, \sigma(n))$, obtained from the MST rule is chosen to be the initial current solution for descent method, with objective function value *Z*.

2) Neighborhood Generation:

In this step, a feasible neighbor $\sigma' = (\sigma'(1), \cdots, \sigma'(n))$ of the current solution is generated by randomly choosing two jobs from σ, (not necessarily adjacent), transposing their positions, and computing the function value denoted by Z'.

3) Acceptance Test:

Now consider the test of whether to accept the move from σ to σ' or not, as follows:

- If $Z' < Z$: then σ' replace σ as the current solution, and we set $Z' = Z$, and then return to step (2).
- If $Z' \ge Z$: then σ is the current solution and return to step (2).

4) Termination condition:

After a number of iterations, the algorithm is stops at a near optimal-solution.

5.2. The Simulated Annealing (SA) Method

In this method, improving and neutral moves are always accepted. While deteriorating actions are accepted according to a given probability acceptance function.

The following steps describe the SA method [17]:

1) Initialization:

Is the same as initialization in DM and with its objective function value *Z*.

2) A feasible neighborhood of σ is generated by the same technique described in DM to get σ' and Z'.

3) Acceptance Test:

In this step we calculate the difference value between the current initial solution *Z* and the new value Z', $\Delta = Z' - Z$ then we have:

- If $\Delta \le 0$, then Z' is accepted as a new current solution, and set $Z' = Z$, and go to step (2).
- If $\Delta > 0$, then Z' is accepted with $P(\Delta) = \exp(-\Delta/t)$, which is the probability of accepting a move, where *t* is known as temperature. The temperature *t* starts

at a relatively high value and then gradually decreases slowly as the algorithm progresses. We chose (40˚) as an initial temperature value for the (SA). Let m be the number of iterations, for each iteration j, $1 \le j \le m$ a temperature t_j is drive from [16] as:

$$t_j = t_{j-1} \Big/ \left(1 + B \times t_{j-1}\right), \quad j = 2, \cdots, m$$

where $B = t_1 - 1/m \times t_1$, $t_1 = 40$, m = is the number of iterations.

4) The algorithm is stopped after a number of iterations at the near-optimal solution.

5.3. Genetic Algorithm (GA)

A genetic algorithm is a general search and optimization method that works on a population of feasible solutions (individuals) to a given problem. The following steps have described the structure of GA [18]:

1) Initialization:

The initial population can be constructed by using heuristic methods. In this paper, we generated the initial population starting with (m = 30) two of them by using MST rule and Lawler algorithm, and the remaining ones are generated randomly.

2) New population:

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

a) Selection:

Selecting the individuals according to fitness value will usually form the next generation's parents.

b) Crossover:

Homogeneous mixture crossover (HMX) [16] is defined in the following:

The mixture of the two parents uniformly by making a set of (m) genes, the odd position from the first parent and the even position from the second parent. Then separate genes without repetition of the gene. If the gene (k) does not exist in the child, then keep it and put (0) in (m). otherwise, we keep gene k in the second child and put (1) in (m) until the genes are exhausted. For example

		Parent		Mixture
Parent (1)		1 3 2 5 4 9 6 8 7	→	1 4 3 5 2 9 5 6 4 7 9 3 6 1 8 2 7 8
Parent (2)		4 5 9 6 7 3 1 2 8		0 0 0 0 0 0 1 0 1 0 1 1 1 1 0 1 1 1
Exchanging	→	1 4 3 5 2 9 6 7 8		Child (1)
		5 4 9 3 6 1 2 7 8		Child (2)

c) Mutation:

The mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosomes (solutions) to the next. Pair-wise (swap) mutation is applied on each pair of parent solutions to generate two new solutions (children). For example:

1 3 **2** 5 4 9 **6** 8 7 swap → 1 3 **6** 5 4 9 **2** 8 7

d) Termination Condition:

The algorithm is terminated after a number of iterations.

5.4. The Tree Type Heuristics Method (TTHM)

The branch and bounded (BAB) method can be used to obtain the upper bound on the optimal value of the objective function if some of the possible optimal partial schedules have not been explored. The tree-type heuristic method [16] uses a (BAB) method without using a backtracking procedure. In the primary step of this method, the lower bound (LB) is evaluated at all nodes in each level of the search tree, then some of the nodes within each level of the search tree are chosen from which to branch. Usually, one node is selected with each level and stops at the first complete sequence of the jobs to be the solution.

6. Experimental Results

6.1. Computational Results

Algorithm (3), BAB, and all local search algorithms Decent Method (DM), Simulation Annealing (SA), Genetic Algorithm, and Tree Type heuristics method, are coded in MATLAB 9.12 (R2022a) and implemented on 11th Gen Intel(R) Core (TM) i7-1185G7 @ 3.00GHz 3.00 GHz, with RAM 32.0 GB personal computer.

6.2. Test Results for All Algorithms

Table 1 displays the outcomes of using the algorithm (3) to get a set of efficient solutions and minimum sum of $V_{\max}$, $T_{\max}$, and $E_{\max}$ for the problem (P3) on samples of different jobs with five experiments for each. The results of efficient solutions compare with those obtained from the BAB method for $n < 10$, and the complete enumeration method (CEM), which generates all solutions for $n < 7$.

Table 2 displays the minimum sum of $V_{\max}$, $T_{\max}$, and $E_{\max}$ using Local Search Heuristic methods on the same data in **Table 1** and compares the results with the SUM obtained from Algorithm (3), and BAB.

Table 3 displays the minimum sum of $V_{\max}$, $T_{\max}$, and $E_{\max}$ obtained by Local Search Heuristic methods, and the SUM from Algorithm (3) $n \geq 10$.

Table 1. Displays the outcomes of using the algorithm (3) to get a set of efficient solutions and minimum sum of $V_{\max}$, $T_{\max}$, and $E_{\max}$ for the problem (P3).

n	EX	Algorithm 3	CEM	SUM	BAB
3	1	(11, 12, 3), (8, 13, 3), (7, 16, 3)	(11, 12, 3), (8, 13, 3), (7, 16, 3)	24	24
	2	(4, 5, 2)	(4, 5, 2)	11	11
	3	(3, 13, 2), (5, 5, 2)	(3, 13, 2), (5, 5, 2)	12	12
	4	(4, 6, 1)	(4, 6, 1)	11	11
	5	(6, 6, 3)	(6, 6, 3)	15	15

Continued

4	1	(2, 19, 4), (3, 3, 4)	(2, 19, 4), (3, 3, 4)	10	10
	2	(8, 9, 0)	(8, 9, 0)	17	17
	3	(4, 6, 0), (3, 9, 1)	(4, 6, 0), (3, 9, 1)	10	10
	4	(5, 15, 1), (6, 6, 4)	(5, 15, 1), (8, 11, 3), (6, 6, 1)	16	13
	5	(5, 13, 1)	(5, 13, 1)	19	19
5	1	(4, 19, 1), (5, 9, 0)	(4, 19, 1), (5, 9, 0)	14	14
	2	(10, 18, 5), (6, 19, 1)	(10, 18, 1), (6, 19, 1)	26	26
	3	(6, 22, 3), (9, 14, 0), (10, 13, 0), (8, 22, 1)	(6, 21, 3), (9, 14, 0), (10, 13, 0), (8, 21, 1)	23	23
	4	(7, 19, 1)	(7, 19, 1)	27	27
	5	(4, 9, 5)	(4, 9, 5)	18	18
6	1	(5, 19, 3), (10, 16, 3)	(5, 19, 3), (10, 16, 3)	27	27
	2	(5, 16, 3), (7, 9, 2)	(5, 16, 3), (7, 9, 2)	18	18
	3	(11, 25, 0), (10, 25, 3)	(11, 25, 0), (10, 25, 3)	36	36
	4	(7, 9, 8), (3, 12, 2)	(7, 9, 2), (3, 12, 2)	17	17
	5	(4, 13, 11), (5, 5, 8)	(4, 13, 11), (5, 5, 8)	18	18
7	1	(6, 25, 3)		34	34
	2	(2, 28, 6), (6, 6, 2), (3, 12, 2)		14	14
	3	(10, 15, 2), (7, 16, 2)		25	24
	4	(8, 48, 3), (1, 40, 8)		49	49
	5	(3, 29, 4), (7, 10, 2), (5, 17, 2), (3, 21, 5)		19	19
8	1	(7, 7, 13)		27	27
	2	(11, 46, 2), (10, 54, 1)		59	58
	3	(10, 27, 0), (9, 33, 0), (8, 33, 1)		37	37
	4	(10, 38, 1), (11, 35, 1), (9, 38, 3)		47	46
	5	(4, 14, 9), (8, 9, 15)		27	26
9	1	(2, 41, 12), (6, 8, 10), (6, 6, 12), (5, 14, 10), (4, 16, 1) (3, 40, 13)		24	22
	2	(0, 0, 23)		23	23
	3	(7, 24, 1)		32	32
	4	(7, 28, 3), (8, 27, 5)		38	37
	5	(7, 17, 3)		27	27
10	1	(8, 26, 0), (11, 25, 0), (6, 35, 0)		34	33
	2	(10, 28, 3), (11, 25, 6), (9, 31, 3), (7, 31, 11)		41	38
	3	(5, 8, 6), (6, 7, 8), (4, 16, 6), (3, 45, 9)		19	19
	4	(10, 39, 0), (9, 46, 0), (8, 46, 3)		49	48
	5	(11, 30, 3), (9, 31, 2)		42	41

Table 2. Displays the minimum sum of V_{max}, T_{max}, and E_{max} using Local Search Heuristic methods on the same data in **Table 1** and compares the results with the sum obtained from Algorithm (3), and BAB.

n	EX	optimal	DM	SM	GA	TTHM	Algorithm 3	BAB
	1	24	24	24	24	26	24	24
	2	11	11	11	11	11	11	11
3	3	12	12	12	12	18	12	12
	4	11	11	11	11	11	11	11
	5	15	15	15	15	15	15	15
		No of opt.	5	5	5	3	5	5
		Av. time in Sec.	0.1875	0.0399	0.1800	0.0053	0.0163	0.0241
	1	10	10	10	10	25	10	10
	2	17	17	17	17	22	17	17
4	3	10	10	10	10	10	10	10
	4	13	13	13	13	21	16	13
	5	19	19	19	19	19	19	19
		No of opt.	5	5	5	2	4	5
		Av. time in Sec.	0.1933	0.0407	0.1839	0.0058	0.0176	0.0317
	1	14	14	14	14	14	14	14
	2	26	29	26	26	26	26	26
5	3	23	23	23	23	30	23	23
	4	27	27	27	27	28	27	27
	5	18	18	18	18	19	18	18
		No of opt.	4	5	5	2	5	5
		Av. time in Sec.	0.1930	0.0438	0.1821	0.0057	0.0950	0.0441
	1	27	29	27	27	27	27	27
	2	18	18	18	18	24	18	18
6	3	36	36	36	36	36	36	36
	4	17	17	17	17	17	17	17
	5	18	18	18	18	18	18	18
		No of opt.	4	5	5	4	5	5
		Av. time in Sec.	0.1908	0.0480	0.1824	0.0053	0.0258	0.0688

Continued

	1	34	34	34	34	34	34	34
	2	14	14	14	14	34	14	14
7	3	24	25	24	24	25	25	24
	4	49	49	49	49	50	49	49
	5	19	19	19	19	31	19	19
		No of opt.	4	5	5	1	4	5
		Av. time in Sec.	0.1968	0.0509	0.1911	0.0059	0.0277	0.3385
	1	27	27	27	27	27	27	27
	2	58	58	58	58	65	59	58
8	3	37	38	37	37	42	37	37
	4	46	46	46	46	49	47	46
	5	26	26	26	26	27	27	26
		No of opt.	4	5	5	1	2	5
		Av. time in Sec.	0.1965	0.0607	0.1973	0.0057	0.0688	2.6539
	1	22	22	22	22	30	24	22
	2	23	23	23	23	23	23	23
9	3	32	32	32	32	32	32	32
	4	37	37	37	37	38	38	37
	5	27	27	27	27	27	27	27
		No of opt.	5	5	5	3	3	5
		Av. time in Sec.	0.2035	0.06218	0.2018	0.0064	0.0273	7.8249
	1	33	34	33	33	41	34	33
	2	38	38	38	38	43	41	38
10	3	19	19	19	19	26	19	19
	4	48	48	48	48	55	49	48
	5	41	42	41	41	42	42	41
		No of opt.	3	5	5	0	1	5
		Av. time in Sec.	0.1977	0.0671	0.1954	0.0055	0.0238	2654.8582

Table 3. displays the minimum sum of V_{max}, T_{max}, and E_{max} obtained by Local Search Heuristic methods, and the sum from Algorithm (3) $n \geq 10$.

n	EX	optimal	DM	SM	GA	TTHM	Algorithm 3	BAB
	1	759	764	774	774	1573	759	x
	2	610	610	624	614	4327	624	x
100	3	863	863	876	863	1000	872	x
	4	596	596	596	596	2834	596	x
	5	874	894	895	894	1868	874	x
		No of opt.	3	1	2	0	3	
		Av. time in Sec.	0.3011	0.4940	0.2915	0.0597	0.0249	
	1	2889	2903	2903	2902	11,154	2889	x
	2	1566	1572	1572	1572	2262	1566	x
500	3	1612	1617	1617	1612	4503	1617	x
	4	1979	1979	1989	1979	8486	1988	x
	5	1138	1139	1139	1139	3977	1138	x
		No of opt.	1	0	2	0	3	
		Av. time in Sec.	0.6981	2.8196	0.6891	2.4845	0.0810	
	1	2001	2001	2001	2001	2813	2001	x
	2	2640	2648	2648	2648	29,837	2640	x
1000	3	1811	1811	1811	1811	1894	1811	x
	4	2678	2732	2732	2732	10,057	2678	x
	5	4013	4015	4015	4013	12,992	4013	x
		No of opt.	2	2	3	0	5	
		Av. time in Sec.	1.2873	6.9351	1.2615	19.0697	0.17836	
	1	2372	2376	2376	2376	34,793	2372	x
	2	2734	2746	2746	2746	3885	2734	x
1500	3	2624	2651	2651	2651	36,126	2624	x
	4	3175	3202	3202	3202	13,788	3175	x
	5	2143	2164	2164	2164	23,346	2143	x
		No of opt.	0	0	0	0	5	
		Av. time in Sec.	1.8198	13.5740	1.8171	51.8916	0.3083	

Continued

2000	1	3518	3532	3531	3531	3531	3518	x
	2	3601	3601	3602	3602	12,034	3602	x
	3	2456	2468	2468	2468	8609	2456	x
	4	2541	2567	2567	2567	20,922	2541	x
	5	2720	2747	2747	2747	5000	2720	x
		No of opt.	1	0	0	0	4	
		Av. time in Sec.	2.2309	20.0777	2.4222	115.0507	0.4138	
2500	1	3741	3755	3755	3755	70,285	3741	x
	2	4151	4189	4189	4189	89,556	4151	x
	3	3453	3455	3455	3455	32,784	3453	x
	4	3387	3387	3387	3387	57,419	3387	x
	5	3362	3398	3398	3398	7220	3362	x
		No of opt.	1	1	1	0	5	
		Av. time in Sec.	2.9920	30.9195	2.9153	245.7569	0.5907	
3000	1	3374	3381	3381	3381	113,080	3374	x
	2	3891	3925	3925	3925	82,982	3891	x
	3	2760	2765	2765	2765	12,794	2760	x
	4	5084	5086	5086	5086	22,620	5084	x
	5	4106	4107	4107	4107	28,783	4106	x
		No of opt.	0	0	0	0	5	
		Av. time in Sec.	3.4263	40.9182	3.4889	393.2098	0.7786	
3500	1	5327	5361	5361	5361	17,022	5327	x
	2	4774	4780	4780	4780	21,512	4774	x
	3	3025	3025	3025	3025	4200	3025	x
	4	3053	3053	3053	3053	64,133	3053	x
	5	2853	2872	2872	2872	31,780	2853	x
		No of opt.	2	2	2	0	5	
		Av. time in Sec.	4.9491	55.2326	4.9758	847.2746	1.2653	

Continued

4000	1	6124	6147	6147	6147	6147	6124	x
	2	4196	4202	4202	4202	106,519	4196	x
	3	4453	4499	4499	4499	5350	4453	x
	4	4693	4694	4694	4694	107,738	4693	x
	5	7584	7589	7589	7589	59,895	7584	x
		No of opt.	0	0	0	0	5	
	Av. time in Sec.		5.4975	68.4537	5.5505	1201.5244	1.5200	
	1	4758	4760	4760	4760	39,727	4758	x
	2	4867	4874	4874	4874	5933	4867	x
4500	3	3921	3929	3929	3929	36,110	3921	x
	4	5577	5579	5579	5579	77,961	5577	x
	5	4416	4418	4418	4418	51,170	4416	x
		No of opt.	0	0	0	0	5	
	Av. time in Sec.		5.8256	81.7492	8.8406	1691.8303	2.2209	
	1	7452	7452	7452	7452	27,028	7452	x
	2	6479	6479	6479	6479	81,149	6479	x
5000	3	4704	4704	4704	4704	35,999	4704	x
	4	8576	8612	8612	8612	94,709	8576	x
	5	7834	7834	7834	7834	159,234	7834	x
		No of opt.	4	4	4	0	5	
	Av. time in Sec.		6.3790	96.7463	6.8763	2351.9938	2.5285	

From our Computational results using random data we conclude that:

1) The number of efficient solutions (points) of an algorithm (3) is less than the number of jobs n.

2) Algorithm (3) can find most of efficient points and this clear from the results of **Table 1** from the 50 test problems for $n = 3, 4, \cdots, 10$ only for:

- $n = 4$, the experiment (4) gives SUM = 16, and the exact SUM = 14 which is obtained by (com) and BAB method, $n = 7$ experiment (3) gives SUM = 25 and the exact SUM = 24 which is obtained by BAB method.
- $n = 8$ experiment (2) gives SUM = 59 and the exact SUM = 58 which is obtained by BAB method.
- $n = 8$ experiment (4) gives SUM = 47 and the exact SUM = 46 which is ob-

tained by BAB method.

- $n = 8$ experiment (5) gives SUM = 27 and the exact SUM = 26 which is obtained by BAB method.
- $n = 9$ experiment (1) gives SUM = 24 and the exact SUM = 22 which is obtained by BAB method.
- $n = 9$ experiment (4) gives SUM = 38 and the exact SUM = 37 which is obtained by BAB method.
- $n = 10$ experiment (1) gives SUM = 34 and the exact SUM = 33 which is obtained by BAB method.
- $n = 10$ experiment (2) gives SUM = 41 and the exact SUM = 38 which is obtained by BAB method.
- $n = 10$ experiment (4) gives SUM = 49 and the exact SUM = 48 which is obtained by BAB method.
- $n = 10$ experiment (5) gives SUM = 42 and the exact SUM = 41 which is obtained by BAB method.

3) The algorithm (3) can be used for solving problems of the form $1//F(V_{\max}, T_{\max}, E_{\max})$.

4) **Table 2** compares DM, SA, GA, TTHM, Algorithm (3), and BAB. We found that the Local Search Heuristic methods give more efficient solutions than algorithm (3) for $n \leq 10$.

5) **Table 3** for $n \geq 10$, Algorithm (3) gives more efficient solutions than Local Search Heuristic methods when $n \to \infty$.

6) The average time (in seconds) for five experiments when $n = 5000$ is 2.5285 for an algorithm (3), DM is 6.3790, GA is 6.8763. SA is 96.7463, and TTHM is 2351.9938.

7) Since problem (P5) is a particular case of problem (P3), hence algorithm (3) can be used to find near-optimal solutions without using the BAB method and in a reasonable time for large n.

8) Local Search Heuristic approaches (DM, SA, and GA) are effective means of obtaining efficiency in a fair amount of time. **Table 3** demonstrates the effectiveness of these approaches and shows a small variation in the results when compared to algorithm (3).

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Azizoglu, M., Kondakci, S. and Kokslan, M., (2003) Single Machine Scheduling with Maximum Earliness and Number Tardy. *Computers & Industrial Engineering*, **45**, 257-268. https://doi.org/10.1016/S0360-8352(03)00034-2

[2] Celia, A.G. and Potts, C.N. (1996) A Comparison of Local Search Methods for Flow Shop Scheduling. *Annals Operations Research*, **63**, 489-509. https://doi.org/10.1007/BF02156631

[3] Chang, P. and Su, L. (2001) Scheduling Jobs on One Machine to Minimize the Maximum Lateness with a Minimum Number of Tardy Jobs. *Computer & Industrial Engineering*, **40**, 49-60. https://doi.org/10.1016/S0360-8352(01)00035-3

[4] Hoogeveen, J.A. and van de Velde, S.L. (1995) Minimizing Total Completion Time and Maximum Cost Simultaneously Is Solvable in Polynomial Time. *Operations Research Letters*, **17**, 205-208. https://doi.org/10.1016/0167-6377(95)00023-D

[5] Moslehi, G., Moghaddam, R., Vasei, M. and Azaron, A. (2005) Optimal Scheduling for a Single Machine to Minimize the Sum of Maximum Earliness and Tardiness Considering Idle Insert. *Applied Mathematics and Computation*, **167**, 1430-1450. https://doi.org/10.1016/j.amc.2004.08.022

[6] Abdul-Razaq, T.S. and Abdul-Razaq, K.F. (2016) Algorithms for Multicriteria Scheduling Problems. *Basrah Journal of Science* (*A*), **34**, 1-12.

[7] Van Wassenhove L.N. and Gelders, F. (1980) Solving a Bicriterion Scheduling Problem. *European Journal of Operational Research*, **4**, 42-48. https://doi.org/10.1016/0377-2217(80)90038-7

[8] Hoogeveen, J.A. (1992) Singlemachine Bicriteria Scheduling. Ph.D. Thesis, Center for Mathematics and Computer Science, Amsterdam.

[9] Jackson, J.R. (1955) Scheduling a Prodection Line to Minimize Maximum Tardiness. Ph.D. Thesis, University of California, Loss Angles.

[10] Jouni, L. (2000) Multi-Objective Nonlinear Pareto-Optimization. Ph.D. Thesis, Lappeenranta University of Technology, Lappeenranta.

[11] Lawler, E.L. (1973) Optimal Sequencing of a Single Machine Subject to Precedence Constraint. *Management Science*, **19**, 544-546. https://doi.org/10.1287/mnsc.19.5.544

[12] Hoogeveen, J.A. (2005) Invited Review Multicriteria Scheduling. *European Journal of Operational Research*, **167**, 592-623. https://doi.org/10.1016/j.ejor.2004.07.011

[13] Hoogeveen, J.A. (1996) Single Machine Scheduling to Minimize a Function of Two or Three Maximum Cost Criteria. *Journal of Algorithms*, **21**, 415-433. https://doi.org/10.1006/jagm.1996.0051

[14] Arats, E.H.L. and Lenstra, J.K. (1997) Local Search in Combinatorial Optimization. John Wiley and Sons, Chichest.

[15] Anderso, E.J., Glass, C.A. and Potts, C.N. (1995) Applications of Local Search in Machine Scheduling. Ph.D. Thesis, University of Southampton, Southampton.

[16] Salman, F.M. (2008) Exact and Local Search Methods for Single Machine Problem. Master's Thesis, University of Al-Mustansiriyah, Baghdad.

[17] Kirkpatrick, S., Gelat Jr. C.D. and Vecchi, M.P. (1983) Optimization by Simulated Annealing. *Science*, **220**, 671-680. https://doi.org/10.1126/science.220.4598.671

[18] Hummady, L.Z. (2005) Using Genetic Algorithm to Solve (NP-Compete) Problem. Master's Thesis, University of Al-Mustansiriyah, Baghdad.