DOI: 10.24996/ijs.2019.60.8.17

ISSN: 0067-2904

Exact Methods for Solving Multi-Objective Problem on Single Machine Scheduling

Hanan Ali Chachan*, Alaa Sabah Hameed
Department of Mathematics, College of Sciences, Al-Mustansireyah University, Baghdad, Iraq

ABSTRACT
In this paper, one of the Machine Scheduling Problems is studied, which is the problem of scheduling a number of products (n-jobs) on one (single) machine with the multi-criteria objective function. These functions are (completion time, the tardiness, the earliness, and the late work) which formulated as $1//\sum_{j=1}^{n}(C_j + T_j + E_j + V_j)$. The branch and bound (BAB) method are used as the main method for solving the problem, where four upper bounds and one lower bound are proposed and a number of dominance rules are considered to reduce the number of branches in the search tree. The genetic algorithm (GA) and the particle swarm optimization (PSO) are used to obtain two of the upper bounds. The computational results are calculated by coding (programing) the algorithms using (MATLAP) and the final results up to (18) product (jobs) in a reasonable time are introduced by tables and added at the end of the research.

Key words: Multi-Objective Problem (MOP), Branch and Bound (BAB) method, Genetic Algorithm (GA), Particle Swarm Optimization (PSO).

طرق تامة لحل مسألة جدولة متعددة الاهداف على ماكنة واحدة

حنان علي جيجان*،علاء صباح حميد
قسم الرياضيات، كلية العلوم، الجامعة المستنصرية، بغداد، العراق

الخلاصة
في هذا البحث, تمت دراسة إحدى مشاكل جدولة المكائن ألا وهي مشكلة جدولة عدد من النتاجات (n من الاعمال) على ماكنة واحدة بدالة هدف متعددة المعايير, و هذه الدوال هي (وقت الاتمام , التأخير اللاسلبي , التبكير اللاسلبي و العمل المتأخر) المصاغة بالشكل $1||\sum_{j=1}^{n}(C_j + T_j + E_j + V_j)$. و تم اعتبار طريقة التفرع و التقييد (BAB) كطريقة رئيسية لحل المشكلة و التي تم فيها اقتراح أربعة قيود عليا , قيد أدنى و بعض قواعد الهيمنة لتخفيض عدد التفرعات في شجرة البحث. كما تم استخدام الخوارزمية الجينية (GA) و أمثلية سرب الجزيئات (PSO) لاستحصال أثنين من القيود العليا. أما النتائج الحسابية فقد تم حسابها عن طريق ترميز (برمجة) الخوارزميات على برنامج (MATLAP) و اعطاء النتائج النهائية للحل الامثل لغاية 18 نتاج (عمل) ضمن وقت مقبول بجداول تم إضافتها بنهاية البحث.

1. Introduction:
1.1. Scheduling Importance:
The scheduling problem is a collection of one or more machines (**the resources**), a number of jobs (**the tasks**) and the function as an (**objective criteria**) need to be combined under some constraints to get mathematical formula with data that helps us to evaluate the problem and obtain a solution, and it

*Email: Hanan_altaai@yahoo.com

can be defined as a decision making practicability deals with an organized basis in many manufacturing industries, where it is a distribution (allocation) of machines (**single** or **multiple**) to jobs over given time intervals and the aim is to maximize or minimize (**one** or **more**) objectives criteria [1]. Since the scheduling problem is an (optimization problem) it has to face (directly and practically) the real world matters [2]. And we will show the importance of the scheduling theory in our real world by the following simple example :

A man who needs to do more than one job at every morning before he goes to his work, these jobs are (takes his children crossing the street to school bus, eating his breakfast, buying house supplies from a near supermarket, showering and dressing). Every job needs a processing time to be done, and he can't do more than one job at a time . He must be ordering these disjoint jobs in such a way that he doesn't be late for his work.

This is an optimization problem, such that the man needs to do kind of a specific arrangement for these jobs that decreases the loss of time and this must be done in an order such that the best possible arrangement (sequence) which is minimizes the given objective function whatever it was and this (sequence) is said to be **"the schedule"** [3].

1.2. Scheduling History:

The first papers on scheduling problems have reached the surface in the years (1954, 1955 and 1956) [4-6] respectively, while in the year (1973), Lawler [7] introduced an algorithm to minimize the maximum cost (f_{max}). After that, the scheduling problems took a large amount of interesting that gave us a fine number of papers which introduced such good procedures for finding the optimality. In the next years, many researchers deal with scheduling problems as one objective criterion. But the evolution of the manufacturing industries, airlines, health care, military duties (missions), artificial intelligence and so many examples of our real world need more than one criteria to deal with. The complexity and the variety of resources in the real world matters pushed the specialists to inter a new area in the scheduling (the area of more than one criteria). The objective criteria (in this case) has been classified into two types; minimization of a maximum function **(minimax)** criteria which describes the minimizing of the maximum value of a set of functions, and minimization of a sum function (**minisum**) criteria where it describes the minimizing of the sum of the number of functions [8]. So, about the $\mathbf{80_s}$ of the $\mathbf{20^{th}}$ century until this moment of the time, the area of multi-criteria objective has been expanding.

1.3. Previous Work:

The researchers began to deal with (k) criteria where (k≥2), where Nelson et al. (1986) [9] has worked on three two-criteria problems employing the regular functions **(F, $\mathbf{T_{max}}$and $\mathbf{n_T}$)** the mean flow time, the maximum tardiness and the number of tardy jobs respectively, and presented four algorithms with computational results showed that the case of all three two-criteria the number of the efficient solutions are smaller than the permutation schedules and humbly larger in the case of three-criteria. Kim and Yano (1994) [10] developed both of the (optimal and heuristic) algorithms to minimize the function of the total tardiness and earliness function $\mathbf{1}||\sum_{\mathbf{j=1}}^{\mathbf{n}}(\mathbf{T_j}+\mathbf{E_j})$ up to 20 jobs. The problem of minimizing $\mathbf{1}||\sum_{\mathbf{j=1}}^{\mathbf{n}}\left(\mathbf{E_j}+\mathbf{T_j}+\mathbf{C_j}\right)$ has shown to be NP-hard problem by Ali (2005) [11] and he derived a lower bound for it, while Tariq S. and Chachan (2009) [12] used local search algorithms to find a solution for the same problem with $n \leq 150$ jobs. A Branch and Bound (BAB) algorithm was used to solve the problem $\mathbf{1}||\sum_{\mathbf{j}}\mathbf{C_j}+\mathbf{ET_{max}}$ for $n \leq 40$ by Ali (2017) [13] and his Tree Type Heuristic (TTH) gave excellent times for $n \leq 1000$ jobs. The problem $\mathbf{1}||(\sum\mathbf{C_j}+\sum\mathbf{T_j}+\sum\mathbf{E_j}+\mathbf{T_{max}}+\mathbf{E_{max}})$ is considered to be strongly NP-hard by Tariq S. and Akram (2017) [14], but they applied two local search algorithms; descent method (DM) and simulated annealing method (SA) to solve the problem for $n \leq 5000$ jobs. In this paper, the problem $\mathbf{1}||\sum_{\mathbf{j=1}}^{\mathbf{n}}(\mathbf{C_j}+\mathbf{T_j}+\mathbf{E_j}+\mathbf{V_j})$, where in section 2 the problem representation takes place and the methodology is put in section 3 while section 4 contains the dominance rules. The final sections 5 and 6 contain the experimental Results and the conclusion respectively.

2. Problem Representation:

In this paper, because of the importance of the **single-machine** applications such as they sometimes take place in practice, like in the airlines scheduling when there is just one runway, or computers scheduling with one processor, and it acts as an access to the entire system when we deal with multi-

machines problems, for this reason, we chose the single machine problem. A multi-objectives problem is considered, and the formal description of this problem is set as follows:
Scheduling **n** jobs on a single machine which is always available can do them, where each one of these jobs can be executed on that machine at its special time (i.e. only one job can be executed at a time), and the machine can do only one job at a time.
For $j = 1,2,3,\dots n$ we will denote $\mathbf{p_j}$ as the **processing time** of the job j, and $\mathbf{d_j}$ as the **due date** of the job j (i.e. the promised date for delivering (or finishing) the execution of the job j). The schedule will define for each job j a completion time $\mathbf{C_j = \sum_{i=1}^{j} p_i}$. The penalties (the tardiness and the earliness) will be given as: $\mathbf{T_j = max\{C_j - d_j, 0\}}$, $\mathbf{E_j = max\{d_j - C_j, 0\}}$ respectively, and the late work will be $\mathbf{V_j = min\{T_j, p_j\}}$. Where every job j will be ready to be processed at time zero, no preemption is allowed and our objective is to find a feasible schedule that gives the minimum value for the multi-objective function $F = \sum_{j=1}^{n}(C_j + T_j + E_j + V_j)$ by using a **(BAB)** method. Using the standard scheduling problem classification notation, the main problem (P) is denoted by $1||\sum_{j=1}^{n}(C_j + T_j + E_j + V_j)$ and formulated as ;

$$\left.\begin{array}{l}
\text{Min } F = \text{Min } \sum_{j=1}^{n}(C_j + T_j + E_j + V_j) \\
\textbf{subject to}: \\
C_1 = p_1 \,; \\
C_j = C_{j-1} + p_j \,; \qquad j = 2, \dots, n \\
T_j = \max\{C_j - d_j, 0\}, \quad j = 1, \dots, n \\
E_j = \max\{d_j - C_j, 0\}, \quad j = 1, \dots, n \\
V_j = \begin{cases} 0 & d_j \le C_j \\ C_j - d_j & d_j < C_j < d_j + p_j \\ p_j & d_j + p_j \le C_j \end{cases}
\end{array}\right\} \dots\dots\dots(P)$$

3. Methodology:

In this section, there will be a brief of the methods which are used for the problem (P). From the exact methods, the branch and bound (BAB) method is used as the main method for solving the problem, while the genetic algorithm (GA) and the particle swarm optimization (PSO) are used to obtain an upper bounds.

3.1. (BAB) Method:

The branch and bound (BAB) finds the optimal solutions by doing an implicit enumeration of all solutions in the solution set (i.e. testing smaller subsets of the solutions set increasingly), and these subsets can be treated as sets of solutions of corresponding sub-problems of the main problem. So, the (BAB) method is used as an exact method to find a solution that optimizes (minimizes) our problem [15].

In this paper, by proposing a number of upper bounds and a lower bound, a procedure for the (BAB) is introduced and a number of dominance rules are introduced to reduce the branching size.

3.1.1: Upper bounds:

In this subsection, we will introduce four upper bounds which are given as follows:
1- $\mathbf{UB_1}$: Where the (n) jobs are ordered in (SPT) rule (i.e. ordering the (n) jobs by their processing times increasingly ($p_1 \le p_2 \le \dots \le p_n$)) and then the cost is calculated.
2- $\mathbf{UB_2}$: Where the (n) jobs have been sorted in (EDD) rule (i.e. ordering the (n) jobs by their due dates increasingly ($d_1 \le d_2 \le \dots \le d_n$)) and then the cost is calculated.
3- $\mathbf{UB_3}$: This upper bound was found by applying the genetic algorithm (GA) where:

3.1.1(a): Genetic Algorithm (GA):

In general speaking, the (GA) is an evolutionary search technique which is used for the optimization problems to identify (near optimal) solutions. The algorithm starts with a (complete or partial) randomly generated population where the evolution is simulated in generations. Each individual in this population has attached an objective function called the (fitness function) which represents the individual performance that based on a number of criteria [16]

(GA) Algorithm:

Step 1: Create an initial population of (50) chromosomes (solutions) and evaluate the objective (fitness) function of each chromosome.

Step 2: Select two parents from the current population (randomly) to create children by using the (crossover operator) ;

Step 3: Apply the (mutation operators) for small changes in the results.

Step 4: Repeat Steps (2) and (3) until all parents are selected.

Step 5: Replace the old population of solutions with the new one.

Step 6: Evaluate the (fitness) of each solution in the new population.

Step 7: Terminate if the number of generations meets, else go to Step (2).

4- $\mathbf{UB_4}$: Finally, this upper bound is found by applying (PSO) The particle swarm optimization.

3.1.1(b): Particle Swarm Optimization (PSO):

The (PSO) is a very simple stochastic optimization algorithm and effective optimizer for functions from wide range, it was developed by Eberhart [17], which is based on social simulation models. The algorithm assigns a collection (population) of search points (or solutions) moves randomly in the search space. Concurrently, the best position found by every individual experience is saved in memory. The swarming behavior is produced by employing main rules in which are the velocity matching and acceleration by the distance applying by every individual in the swarm in their moving (searching of food) [18].

(PSO) algorithm

Let (i) be the location of the particle, (l) the dimension, α & β are constants, while $\mathcal{F}_1$ & $\mathcal{F}_2$ refer to random functions of [0,1] range, the variable (x) will represent the current position of the particle and (g) represents the global version of the swarm. Then, the algorithm is:

Step 1: Initialize a particles population with random (positions).

Step 2: Evaluate the objective (fitness) function,

Step 3: Compare evaluation with particle's (previous best value= pbest) and choose the minimum,

Step 4: Compare evaluation with group's previous best (pbest[gbest]) and choose the minimum,

Step 5: Update (velocity) and (position) by :

$$\mathbf{v_{ij}(t+1) = \omega * v_{ij}(t) + \alpha\, \mathcal{F}_1 \left(p_{ij}(t) - x_{ij}(t)\right) + \beta\, \mathcal{F}_2 \left(p_{gj} - x_{ij}(t)\right)} \quad \ldots \textbf{(3.1)}$$

$$\mathbf{x_{ij}(t+1) = x_{ij}(t) + v_{ij}(t+1), \quad j = 1, 2, \ldots, l.} \quad \ldots \textbf{(3.2)}$$

Step 6: Go to step 2.

Now, the main upper bound (UB) of the problem (P) will be obtained by:

$\mathbf{UB = min}\ \{\mathbf{UB_1, UB_2, UB_3, UB_4}\}$.

3.1.2: Lower bound:

For deriving the lower bound, the problem $(\mathbf{P})$ can be decomposed into two sub-problems $(\mathbf{P_1})$ and $(\mathbf{P_2})$, where:

$$\left.\begin{aligned} &\mathrm{Min}\ F_1 = \mathrm{Min} \sum_{j=1}^{n} \left(C_j + T_j + E_j\right) \\ &\textbf{subject to}: \\ &C_1 = p_1, \\ &C_j = C_{j-1} + p_j, && \text{for } j = 2, \ldots, n \\ &T_j = \max\{0, C_j - d_j\}, && \text{for } j = 1, \ldots, n \\ &E_j = \max\{0, d_j - C_j\}, && \text{for } j = 1, \ldots, n \end{aligned}\right\} \ldots\ldots\ldots (P_1)$$

For this sub-problem, the lower bound of Ali [11] is used to obtain the first lower bound $(\mathbf{LB_1})$, where:

$\sum_{j=1}^{n}\left(C_j + T_j + E_j\right) = \mathrm{Max}\left\{\sum_{j=1}^{n} d_j, \sum_{j=1}^{n} \mathrm{Max}\left\{2C_j - d_j\,, C_j\right\}\right\}$………(3.3)

$$\left.\begin{array}{ll} \mathrm{Min}\ F_2 = \mathrm{Min}\ \sum_{j=1}^{n} V_j & \\ \textbf{subject to}: & \\ C_1 = p_1, & \\ C_j \geq p_j, & \text{for } j = 2, \dots, n \\ T_j = \max\{0, C_j - d_j\}, & \text{for } j = 1, \dots, n \\ V_j = \min\{p_j, T_j\}, & \text{for } j = 1, \dots, n \end{array}\right\} \dots\dots\dots (P_2)$$

Here, the lower bound in the theorem (3.1) below is used for $(\mathbf{P_2})$ to get the second lower bound $(\mathbf{LB_2})$ **.**

Theorem(3.1)[19]**:** For $p_{max} = \max\{p_j\}$, and $T_{max} = \max\{T_j\}$ where $j = 1, 2, \dots, n$, then we have that; $\mathbf{T_{max} \leq \sum_{j=1}^{n} V_j \leq \min\ \{nT_{max}, T_{max} + p_{max} - 1\}}$.

Now, from the next lemma we can obtain the lower bound (LB) of the problem (P) :

Lemma (3.1) [20]**:** If $(\mathbf{LB_1})$ and $(\mathbf{LB_2})$ are lower bounds for the problems $(\mathbf{P_1})$ and $(\mathbf{P_2})$ respectively, then $(\mathbf{LB = LB_1 + LB_2})$ is the lower bound of the main problem $(\mathbf{P})$.

3.1.2(a): Lower bound procedure:

For $\mathbf{N}, \mathbf{S}$ and $\mathbf{U}$ where (**N** represents the set of all jobs, **S** is equal to the set of the scheduled jobs and **U** is the set of the un-scheduled jobs) the procedure is:

1. Starting with an empty set of the scheduled jobs (i.e. $S = \emptyset$), and begin to sort the jobs (one by one) until we have $|S| = n - 1$, and the final sequence (i.e. when we insert the n^{th} job) is solved by the complete enumerate method (CEM). At every step, we calculate the cost $\sum_{j \in S}(C_j + T_j + E_j + V_j)$.

2. For the set **U**, the jobs have been sorted in two rules for calculating the costs for the two sub-problems $(\mathbf{P_1})$ and $(\mathbf{P_2})$ by applying the following steps ;

Step(1): Sorting the jobs in the set **U** by the (**SPT**) rule, and then calculate $\sum_{i \in U}(C_i + T_i + E_i)$ by using the equation (3.3).

Step(2): Re-sorting the jobs in the set **U** by the (**EDD**) rule, and calculate $(\sum_{i \in U} V_i)$ by using theorem (3.1).

Step(3): Calculate the total cost $\sum_{\mathbf{j \in N}}(\mathbf{C_j + T_j + E_j + V_j})$ as follows:

Total cost $= \sum_{\mathbf{j \in S}}(\mathbf{C_j + T_j + E_j + V_j}) + \sum_{\mathbf{i \in U}}(\mathbf{C_i + T_i + E_i}) + \sum_{\mathbf{i \in U}} \mathbf{V_i}$

4. Dominance Rules [21]:

In the (BAB) method, especially in the branching scheme, the size of the search tree (the number of nodes) is getting bigger and bigger whenever (n) increase. So, we need to reduce this size by eliminating the un-interesting solutions, or selecting the interesting ones. The matter is that one subset of solutions is rejected while the complementary subset is stored. The interest of dominance rules is to reduce (either statically or dynamically) the search space of scheduling problems which are being solved. Therefore, as a procedure for reducing the search area size and decreasing the time, we state a number of the dominance rules below:

Theorem (4.1):

For the $1||\sum_{j=1}^{n}(C_j + T_j + E_j + V_j)$ problem, if $p_i \leq p_j$ and $d_i \leq d_j$, where the jobs (i) and (j) are adjacent jobs, then job (i) must precede job (j) in at least one optimal sequence:

Proof:

Let $s_1 = (\sigma_1\ i\ j\ \sigma_2)$ and $s_2 = (\sigma_1\ j\ i\ \sigma_2)$, where σ_1 and σ_2 are disjoint subsequences, and let (t) be the completion time of σ_1. We examine the change in $\Delta_{ij}(t) = f_{ij}(t) - f'_{ji}(t)$ with the following cases:

Case (1): If $\mathbf{d_i \leq t + p_i}$, $\mathbf{d_j \leq t + p_j}$, (i.e. the jobs i and j always tardy);

Proof:

Since the jobs (i and j) are both tardy, then $E_i = E_j = 0$ and $V_i = p_i$, let

$f_{ij} = [(t + p_i) + (t + p_i - d_i) + 0 + p_i + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,

$f'_{ji} = [(t + p_j) + (t + p_j - d_j) + 0 + p_j + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$.

Then;

$$\Delta_{ij} = f_{ij} - f'_{ji} = [4t + 5p_i + 3p_j - d_i - d_j] - [4t + 5p_j + 3p_i - d_j - d_i]$$

$= 2p_i - 2p_j = 2(p_i - p_j) \leq 0.$

Case (2): If $\mathbf{d_i \le t + p_i}$, $t + p_j \le \mathbf{d_j} \le \mathbf{t} + p_i + \mathbf{p_j}$, (i.e. the job (i) always tardy and (j) is tardy if not scheduled first);

Proof:

Since (i) is always tardy then $E_i = 0$, and for (j) if scheduled first then $T_j = 0$ and $V_j = \{0, C_j - d_j\}$, (i.e. V_j is (**early**) or (**partial early**)) ,

(a) When ($\mathbf{V_j = 0}$) :

$f_{ij} = [(t + p_i) + (t + p_i - d_i) + 0 + p_i + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,

$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$

Then;

$\Delta_{ij} = f_{ij} - f'_{ji} = 2t + 2p_i + p_j - 2d_j = (t + p_i + p_j - d_j) + (t + p_i - d_j) \le 0$.

(b) When : $\mathbf{V_j = C_j - d_j}$:

$f_{ij} = [(t + p_i) + (t + p_i - d_i) + 0 + p_i + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,

$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + (t + p_j - d_j) + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,

Then;

$$\Delta_{ij} = f_{ij} - f'_{ji} = [4t + 5p_i + 3p_j - d_i - d_j] - [3t + 3p_j + 3p_i - d_i]$$
$$= t + 2p_i - d_j \le 0 .$$

Case (3): If $\mathbf{d_i \le t + p_i}$, $\mathbf{t + p_i + p_j \le d_j}$, (the job (i) is always tardy and the job (j) is always early):

Proof:

Since (i) is always tardy, then ($E_i = 0$), and (j) is always early, then ($T_j = 0$), and ($V_j = \{0, C_j - d_j\}$):

(a) When $\mathbf{V_j = 0}$;

$f_{ij} = [(t + p_i) + (t + p_i - d_i) + 0 + p_i + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0]$,

$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,

Then;

$\Delta_{ij} = f_{ij} - f'_{ji} = [2t + 3p_i - d_i + d_j] - [2t + 2p_j + 3p_i + d_j - d_i] = -2p_j < 0$.

(b) When : $\mathbf{V_j = C_j - d_j}$;

$f_{ij} = [(t + p_i) + (t + p_i - d_i) + 0 + p_i + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + (t + p_j - d_j)]$,

$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + (t + p_j - d_j) + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,

Then;

$\Delta_{ij} = f_{ij} - f'_{ji} = [5t + 3p_i + p_j - d_i] - [3t + 3p_j + 3p_i - d_i] = 2t - 2p_j \le 0.$

Case (4): If $\mathbf{t + p_i \le d_i \le d_j \le t + p_j}$, (i.e. the job (j) is always tardy and the job (i) is tardy if not scheduled first);

Proof

$\mathbf{V_i = \{0, C_i - d_i\}}$

(a) When $\mathbf{V_i = 0}$;

$f_{ij} = [(t + p_i) + (d_i - t - p_i) + 0 + 0 + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,

$f'_{ji} = [(t + p_j) + (t + p_j - d_j) + 0 + p_j + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,

Then;

$$\Delta_{ij} = f_{ij} - f'_{ji} = [2t + 2p_i + 3p_j + d_i - d_j] - [4t + 5p_j + 3p_i - d_j - d_i]$$
$$= -2t - p_i - 2p_j + 2d_i \le 0 .$$

(b) When $\mathbf{V_i = C_i - d_i}$;

$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + (t + p_i - d_i) + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,

$f'_{ji} = [(t + p_j) + (t + p_j - d_j) + 0 + p_j + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,

Then;

$\Delta_{ij} = f_{ij} - f'_{ji} = [3t + 3p_i + 3p_j - d_j] - [4t + 5p_j + 3p_i - d_j - d_i] = -t - 2p_j + d_i < 0$

Case (5): $\mathbf{t + p_i \le d_i}$, $\mathbf{t + p_j \le d_j \le t + p_i + p_j}$, (i.e. each of the two jobs i and j are tardy if not scheduled first) ;
Proof
(a) When $(\mathbf{V_i = 0})$ and $(\mathbf{V_j = 0})$:
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [2t + 2p_i + 3p_j + d_i - d_j] - [2t + 2p_j + 3p_i + d_j - d_i]$
$= -p_i + p_j + 2d_i - 2d_j \le 0$.
(b) When ($\mathbf{V_i = C_i - d_i}$) and ($\mathbf{V_j = C_j - d_j}$) ;
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + (t + p_i - d_i) + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + (t + p_j - d_j) + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [3t + 3p_i + 3p_j - d_j] - [3t + 3p_j + 3p_i - d_i] = -d_j + d_i \le 0$.
(c) When ($\mathbf{V_i = 0}$) and ($\mathbf{V_j = C_j - d_j}$) ;
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + (t + p_j - d_j) + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [2t + 2p_i + 3p_j + d_i - d_j] - [3t + 3p_j + 3p_i - d_i]$
$= -t - p_i + 2d_i - d_j \le 0$.
(d) When ($\mathbf{V_i = C_i - d_i}$) and ($\mathbf{V_j = 0}$) ;
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + (t + p_i - d_i) + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + p_j]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,
Then;
$$\Delta_{ij} = f_{ij} - f'_{ji} = [3t + 3p_i + 3p_j - d_j] - [2t + 2p_j + 3p_i + d_j - d_i]$$
$$= t + p_j - 2d_j + d_i \le 0 .$$

Case (6): If $\mathbf{t + p_i \le d_i \le t + p_i + p_j \le d_j}$, (i.e. (i) is tardy if not scheduled first and (j) is always early) ;
Proof
(a) When $(\mathbf{V_i = 0})$ and $(\mathbf{V_j = 0})$:
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [d_i - d_j] - [2t + 2p_j + 3p_i + d_j - d_i]$
$= -2t - 2p_j - 3p_i + 2d_i - 2d_j \le 0$
(b) When : $\mathbf{V_i = t + p_i - d_i}$ and $\mathbf{V_j} = \begin{cases} \mathbf{t + p_j - d_j}, & \textbf{if (j)is 1}^{\textbf{st}} \\ \mathbf{t + p_i + p_j - d_j}, & \textbf{if (j)is 2}^{\textbf{nd}} \end{cases}$
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + (t + p_i - d_i) + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + (t + p_i + p_j - d_j)]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + (t + p_j - d_j) + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + p_i]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [2t + 2p_i + p_j] - [3t + 3p_j + 3p_i - d_i]$
$= -t - p_i - 2p_j + d_i \le 0$.
Case (7): If $\boldsymbol{t + p_i + p_j \le d_i \le d_j}$, (i.e. both i and j are early) ;
Proof
(a) When $\boldsymbol{V_i = 0}$ *and* $\boldsymbol{V_j = 0}$:
$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0]$,

$f'_{ji} = \left[(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + 0 + (d_i - t - p_j - p_i) + 0\right]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [d_i + d_j] - [d_j + d_i] = 0$.
(b) When :

$$V_i = \begin{cases} t + p_i - d_i, & if\ (i) is\ 1^{st} \\ t + p_j + p_i - d_i, & if\ (j) is\ 2^{nd} \end{cases} \quad and \quad V_j = \begin{cases} t + p_j - d_j, & if\ (j) is\ 1^{st} \\ t + p_i + p_j - d_j, & if\ (j) is\ 2^{nd} \end{cases}$$

$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + (t + p_i - d_i) + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + (t + p_i + p_j - d_j)]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + (t + p_j - d_j) + (t + p_j + p_i) + 0 + (d_i - t - p_j - p_i) + (t + p_j + p_i - d_i)]$,
Then;
$\Delta_{ij} = f_{ij} - f'_{ji} = [2t + 2p_i + p_j] - [2t + 2p_j + p_i] = p_i - p_j \le 0$.

(c) When : $V_i = \begin{cases} t + p_i - d_i, & if\ (i) is\ 1_{st} \\ t + p_j + p_i - d_i, & if\ (j) is\ 2_{nd} \end{cases}$ **and** $V_j = 0$ **;**

$f_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + (t + p_i - d_i) + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0]$,
$f'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j + p_i) + 0 + (d_i - t - p_j - p_i) + (t + p_j + p_i - d_i)]$,
Then;

$$\begin{aligned}\Delta_{ij} = f_{ij} - f'_{ji} &= [t + p_i - d_j] - [t + p_j + p_i + d_j] \\ &= -p_j - 2d_j \le 0 .\end{aligned}$$

5. Experimental Results

In this section, the results are reported in two tables. Table-1 contains the comparison results between (BAB) with the complete enumeration method (CEM) for $n \le 10$, while the Table-2 contains the results of the (BAB) for $n \le 18$.

5.1. Analysis and Problems Instances:

The performance of the (BAB) procedure is compared to 10 problems examples, the sizes of these examples are n =[5,18]. The problems are generated randomly, and for each job j, where $j \in \{1, \dots, n\}$, the processing time p_j was uniformly generated in [1, 10], while the due date d_j was uniformly generated in the interval $[(1 - T - RDD/2)TP, (1 - T + RDD/2)TP]$, where $T = \sum_{j=1}^{n} p_j$ and the two parameters (TP) and (RDD) are said to be Tardiness Factor and Related Range of Due Dates respectively, and have the values:
$RDD = 0.2, 0.4, 0.6, 0.8, 1$ and $TP = 0.2, 0.4$ as it has been showed in the literature [22].

5.2. Computational results:

In this subsection, the computational results are given in tables, each table of them gives the results (i.e. optimal values by (CEM) and (BAB), the upper bound of the problem, the initial lower bound, the number of the nodes and the execution time). For each n, 10 problems are tested, and the stopping condition is 1800 seconds as the maximum execution time. The symbols which used in the tables are:
n: no. of jobs,
Ex: no. of examples,
Opt (BAB): The optimal value of the function using (BAB),
Opt (CEM): The optimal value of the function using (CEM) the complete enumeration method.
UB: Refers to the upper bound of the problem.
ILB: Refers to the initial lower bound of the problem.
NOD: no. of the nodes.
Time: The execution time of the problem (by seconds).

5.3. Tables of results:

Table 1-A comparison results between (CEM) and (BAB) with (n=5, 6, 7, 8, 9, 10)

	Ex	Opt(CEM)	Opt(BAB)	UB	ILB	NOD	Time
n=5	1	221	221	221	222	133	0.1787
	2	117	117	117	115	36	0.0707
	3	97	97	97	91	46	0.0524
	4	112	112	112	103	69	0.0572
	5	74	74	74	67	59	0.0487
	6	97	97	97	91	45	0.0584
	7	51	51	51	47	67	0.0520
	8	77	77	77	62	108	0.0676
	9	142	142	142	135	93	0.0444
	10	147	147	147	145	43	0.0348
n=6	1	82	82	82	74	284	0.1879
	2	154	154	154	147	105	0.0745
	3	108	108	108	100	241	0.0529
	4	156	156	156	150	219	0.0568
	5	225	225	225	222	217	0.0549
	6	189	189	189	182	191	0.0617
	7	258	258	258	254	247	0.0708
	8	158	158	158	155	113	0.0581
	9	147	147	147	140	218	0.0406
	10	312	312	312	307	312	0.0523
n=7	1	198	198	198	188	408	0.2080
	2	239	239	239	233	243	0.0518
	3	220	220	220	213	992	0.1372
	4	156	156	156	151	3697	0.2925
	5	109	109	109	84	1968	0.2091
	6	212	212	212	208	308	0.0710
	7	166	166	166	149	303	0.0657
	8	131	131	131	127	199	0.0422
	9	278	278	278	274	320	0.0726
	10	144	144	144	139	597	0.0878
n=8	1	445	445	445	437	1358	0.2707

	2	239	239	239	225	978	0.1410
	3	370	370	370	372	430	0.0848
	4	415	415	415	414	1975	0.2023
	5	246	246	246	216	4367	0.2339
	6	268	268	268	266	517	0.0902
	7	159	159	159	153	1172	0.1415
	8	440	440	440	443	816	0.1093
	9	255	255	255	248	923	0.1076
	10	328	328	328	324	434	0.0821
n=9	1	297	297	297	269	4765	0.4512
	2	239	239	239	230	4575	0.3567
	3	296	296	296	282	2271	0.1551
	4	340	340	340	331	1225	0.1598
	5	466	466	466	456	2166	0.1967
	6	365	365	365	353	7111	0.4191
	7	394	394	394	384	2458	0.1418
	8	373	373	373	362	5168	0.3010
	9	288	288	288	283	1634	0.1313
	10	426	426	426	411	3709	0.1592
n=10	1	323	323	323	271	75008	3.4992
	2	544	544	544	541	2823	0.1670
	3	367	367	367	362	9359	0.4995
	4	580	580	580	572	12222	0.4480
	5	346	346	346	329	3067	0.2002
	6	411	411	411	404	5765	0.3618
	7	402	402	402	385	15795	0.5750
	8	428	428	428	418	1325	0.0805
	9	568	568	568	571	6552	0.2578
	10	431	431	431	398	3798	0.1610

Table 2-The results of applying (BAB) on (10) examples and (n=8, 11, 14, 17, 18)

	Ex	Opt (BAB)	UB	ILB	NOD	Time
n=8	1	445	445	437	1358	0.2707
	2	239	239	225	978	0.1410
	3	370	370	372	430	0.0848
	4	415	415	414	1975	0.2023
	5	246	246	216	4367	0.2339
	6	268	268	266	517	0.0902
	7	159	159	153	1172	0.1415
	8	440	440	443	816	0.1093
	9	255	255	248	923	0.1076
	10	328	328	324	434	0.0821
n=11	1	565	565	546	12272	0.8145
	2	343	343	328	67871	3.1457
	3	519	519	488	109022	3.8834
	4	383	383	374	40379	1.5113
	5	538	538	525	6982	0.2902
	6	381	381	374	11883	0.5564
	7	514	514	526	240	0.0437
	8	514	514	497	20157	0.7339
	9	328	328	323	46758	1.7423
	10	361	361	315	12748	0.4520
n=14	1	663	663	614	1523176	55.5153
	2	951	951	945	100425	3.8629
	3	796	796	770	1061072	39.6465
	4	800	800	786	71255	2.6237
	5	656	656	620	1020333	36.2558
	6	615	616	558	1186639	42.5126
	7	676	676	673	32082	1.19180
	8	580	584	555	1402176	51.7485
	9	753	753	740	372985	13.4106
	10	702	702	695	110204	3.9956
n=18	1	1232	1232	1167	44779464	1578.6889
	2	1638	1638	1591	4673204	175.9560
	3	1119	1120	1082	50150935	1768.2448
	4	1339	1339	1330	29700616	1082.9119
	5	/	/	/	/	1800.0001
	6	/	/	/	/	1800.0005
	7	1229	1229	1197	47075597	1800.0000
	8	1071	1072	1038	31991807	1268.6863
	9	1266	1266	1258	11712512	493.1440
	10	/	/	/	/	1800.0006

6. Conclusion

For solving the problem (P), a (BAB) method is used and for several different examples for each n. The obtained results from the comparison between (BAB) and (CEM) methods show that the optimal values are equal in for $n \leq 10$, while the (BAB) reach $n \leq 18$ jobs with 10 different example for each n except in case of job $n = 18$ where there are three examples failed because they take a long executing time (i.e. more than 1800 seconds)

References

1. Pinedo, M., L. **2008**. *Scheduling: Theory, Algorithms, and Systems*. Third Edition. Stern School of Business, New York University, New York. Springer.
2. RAO, S.S. **1977**. *Optimization theory and applications. School of mechanical Engineering,* Purdue University, West Lafayette, USA.
3. Hoogeveen, H. **2005**. Invited Review Multicriteria scheduling. *European Journal of Operational Research* **167**: 592–623.
4. Johnson, S.M. **1954**. Optimal two and three stage production schedules with set-up times included. *Naval Res. Logist Quart*. **1**: 61-68.
5. Jackson, J.R. **1955**. Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Sciences Research Project, UCLA.
6. Smith W.E. **1956** Various Optimizers for single-stage production. *Naval. Res. Logist Quart*. **3**: 59-66.
7. Lawler E.L. **1973**. Optimal sequencing of a single machine subject to precedence constraints. *Management Science* **19**: 544-546.
8. T'kindt, V. and Billaut, J. **2002**. *Multi-criteria Scheduling: Theory, Models and Algorithms*. Second edition. Springer Berlin Heidelberg New York.
9. Nelson, R.T. , Sarin, R.K. and Daniels, R.L. **1986.** Scheduling with Multiple Performance Measures: The One-Machine Case. *Management Science* **32**(4): 464-479.
10. Yeong-Dae Kim, Y. and Yano, C.A. **1994**. Minimizing Mean Tardiness and Earliness in Single-Machine Scheduling Problems with Unequal Due Dates. *Naval Research Logistics*, **41**(7): 913-933.
11. Ali, H. **2005**. Genetic and Local Search Algorithms as Robust and Simple Optimization Tools. M.Sc. Thesis. Department of Mathematics. College of Science. University of Al-Mustansiriyah.
12. Abdul-Razaq, T.S. , Chachan, H.A. and Sadkhan, S.B. **2009**. Scheduling Job Classes on Single Machine to Minimize the Multi-Objective Functions. Proceeding of 3rd scientific conference of the College of Science. University of Baghdad.
13. Ali, H. **2017**. Exact and Heuristic Algorithms for Solving Combinatorial Optimization Problems. PH.D. Thesis. Department of Mathematics. College of Science. University of Al-Mustansiriyah.
14. Abdul-Razaq, T.S. and Akram, A.O. **2017.** Local Search Algorithms for Multi-criteria Single Machine Scheduling Problem. *Ibn Al-Haitham Journal for pure and Applied science* (456-472). https://doi.org/ 10.30526/2017.IHSCICONF.1817.
15. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G. and Weglarz, J. **2007**. *Handbook on Scheduling: From Theory to Applications*. Springer. Berlin. Heidelberg. New York.
16. Muthiah, A., Rajkumar, R. and Muthukumar, B. **2015**. Minimizing Makespan in Job Shop Scheduling Problem Using Genetic Algorithm. Applied Mechanics and Materials. ISSN: 1662-7482, Vols. 813-814: 1183-1187.
17. Eberhart, R. C., and Kennedy, J. **1995**. A new optimizer using particle swarm theory. In Proceedings of the 6th Symposium on Micro Machine and Human Science, Nagoya, Japan (pp. 39-43).
18. Konstantinos, E. and Vrahatis, M.N. **2010**. Particle Swarm Optimization and Intelligence: Advances and Applications. Information science reference, Hershey, New York.
19. Potts, C.N. and Van Wassenhove, L.N. **1991**. Single Machine Scheduling To Minimize Total Late Work. **40**(3): 586-595.
20. Ramadhan, A.M. **1998**. Single machine scheduling using branch and bound techniques. M.Sc. Thesis. Department of Mathematics. College of Science. University of Al-Mustansiriyah.
21. Jouglet, A. and Carlier, J. **2011**. Dominance rules in combinatorial optimization problems**.** *European Journal of Operational Research* **212**: 433–444.
22. Arroyo, J.E.C., Ottoni, R.S. and Oliveira, A.P. **2011**. Multi-Objective Variable Neighborhood Search Algorithms for a Single Machine Scheduling Problem with Distinct Due Windows. *Electronic Notes in Theoretical Computer Science* **281**: 5–19.