Ibn Al Haitham Journal for Pure and Applied Science

Journal homepage: http://jih.uobaghdad.edu.iq/index.php/j/index

Exact Solutions for Minimizing cost Function with Five Criteria and Release Dates on Single Machine

Hanan Ali Chachan **Hussein Abdullah Jaafar**

Department of Mathematics, College of sciences, Al-Mustansireah University, Iraq.

Hanan-altaai@yahoo.com alhassen237@gmail.com

Article history: Received 28 September 2019, Accepted 4 December 2019, Published in July 2020.

Doi: 10.30526/33.3.2479

Abstract:

In this paper, we present a Branch and Bound (B&B) algorithm of scheduling (n) jobs on a single machine to minimize the sum total completion time, total tardiness, total earliness, number of tardy jobs and total late work with unequal release dates. We proposed six heuristic methods for account upper bound. Also, to obtain lower bound (LB) to this problem we modified a (LB) select from literature, with (Moore algorithm and Lawler's algorithm). And some dominance rules were suggested. Also, two special cases were derived. Computational experience showed the proposed (B&B) algorithm was effective in solving problems with up to (16) jobs, also the upper bounds and the lower bound were effective in restricting the search.

Key words: Branch and Bound method (B&B), Upper bound (UB), Lower bound (LB), Multi-Objective problems.

1. Introduction

A scheduling problem is defined as a problem of assigning a set of tasks or jobs to a set of resources or machines in a specific time. Common performance measures criteria are usually in the form $\sum f_j$ and $f_{max} = \max\{f_j\}$
Most of the studies that were introduced in the early years of the discovery of the theory of scheduling focused on a single performance measuring criterion. However, in practice, a manager may need to find an acceptable schedule which can simultaneously meet the requirements for several criteria [1].

In this paper, we examine the problem of scheduling with five criteria (referred to above) for measuring performance with the release times, and to the best of our knowledge, this problem was not studied before.

Here, we found it necessary to present some research from the literature which included machine scheduling problems with one or more of the performance criteria contained in our problem.

The Completion Time: This criterion has been extensively studied in the literature. For $1/r_j/\sum C_j$ is NP- hard in the strong sense by Lenstra, Rinnooy Kan and Bruker [2]. This problem with equal release date solved by "shortest- processing- time (SPT)-rule" of smith [3]. Also, the problem $1/r_j$,pmtn $/\sum C_j$ is possible to solve by the "shortest-remaining- processing- time (SRPT)-rule" of Schrage [4]. Several branch and bound algorithms for the problem $1/r_j/\sum C_j$, depend extensively on the "dominance rules (DR)" derived by Chandra [5]. Dessouky and Deogun [6]. And others. For this problem, Ahmadi and Bghchi [7]. Display that the lower bound which is symbolized by (SRPT-LB) obtained by using the (SRPT)- rule to solve the problem $1/r_j$, pumt$/\sum C_j$ best (LB) in comparison other know bounds. Also, Chu [8]. Uses this (SRPT-LB) in (B&B) algorithm that is effective in solving problems, when $n \leq 100$ jobs.

The Tardiness Time: Minimizing the total tardiness is the other one of the most important criteria in practice. For $1/r_j/\sum T_j$ is known to be NP- hard in the strong sense by Rinnooy [9]. Chu [10]. Proposes a (BAB) algorithm and proves some dominance properties. Also suggested, (LB) depends on constructing a schedule in which the jobs are scheduled according to (SRPT)-rule (Schrage [4].) by restful the problem under the presumption that the jobs are preemptive. The proposed algorithm contributed to solve problems, when $n \leq 260$ jobs. Philippe et al [11]. Presented a (B&B) algorithm with new lower bounds (NLB) depending on improve (LB) of Chu [10]. And generalization some well–known dominance properties to solve this $1/r_j/\sum Tj$ problem. The cases handled by this procedure are as 500 jobs.

The Earliness Time: In general, earliness criterion is one of the scheduling objectives studied by few number of researchers, because for many years, researchers focused on one-criterion regular performance measures (i.e. no decreasing in C_j for all j) [12]. And most research that considers earliness as an optimality criterion also includes a tardiness component. The first study on earliness and tardiness (E/T) penalties was by Sidney [13]. Who developed a polynomial-algorithm to minimize maximum of (E/T) for a single machine problem. One Of the studies presented in recent years was by Mehdi and Ghasem [14]. Who studied this problem $1/r_j$ / E_{max} +T_{max}. A (B&B) algorithm is proposed, the algorithm extensively uses efficient dominance rules. In the (B&B) algorithm, a (LB), is obtained by relaxing the assumption of the non-preemption, and divided the problem into two sub-problems of ($1/r_j$, pmtn/T_{max}) and ($1/r_j$, pmtn/E_{max}). The two problems are then resolved by applying some procedures derived from the two rules, (EDD) and (MST). Computational experiments showed the efficiency of the proposed procedure of solving problems with up to 1000 jobs.

The late work*:* Is the quantity of processing performed after its due date, and is denoted by (V_j). At first, this problem was known as information loss by Blazewicz [15]. Potts and Van [16**,** 17]. Suggested the term (Total late work). For the problem $1/pmtn/\sum V_j$, they showed that the minimum $(1//\sum V_j) \leq [(T_{max}$) for the (EDD)-rule sequence]. While the problem $1//\sum V_j$ is NP-hard by Potts and Van [17]. Bryan and Bahram [18]. In this research, considering $1//\sum W_j V_j$ the total weighted late work, with the condition that all jobs arrived at the same time (i.e. the release date r_j= 0 for all jobs). This function is a more general form than the formula Potts and Van [16, 17]. Abdul Razaq et al [19].

studied this $1/r_j/\sum W_jV_j$ problem, using unequal release dates, to solve this problem some special cases were proofed and using a branch and bound algorithm with up to 30 jobs. Also five local search methods to solve this problem were applied and performance is evaluated to it with up to 60000 jobs, and who showed this problem its NP-hard.

The unit Penalty: For the $1//\sum U_j$, Moore [20]. Study was among the earliest to consider scheduling to minimize this problem by an algorithm known as Moor algorithm “sometimes known as Hudgson's Algorithm" that solved the problem optimally. With release dates, the problem $1/r_j/\sum U_j$ is strongly NP- hard by Lenstra et al [2]. Dauzere-peres [21]. Studied this$1/r_j/\sum U_j$ problem, and determined a lower bound depending on the relaxation of a "Mixed-Integer- Linear- Programming" formulation, presented a heuristic method to solve the problem. A large sample of problems has been tested with up to 50 jobs. The calculations showed the efficiency of the proposed approach by comparison with the lower bound. Philippe et al [22]. To solved this $1/r_j/\sum U_j$ problem, who suggesting (B&B) algorithm, with lower bounds based on a "Lagrangian relaxation". Also, they used dominance rules to reduce the search space, suggested techniques are showed solve to optimality cases with up to 200 jobs. Cyril and Samia [23]. In this study showed how good – quality lower and upper bounds that can be calculated for the problem$1/r_j/\sum U_j$, using an original mathematical integer programming formulation. Numerical experiences showed the assessing of the proposed approach it up to 160 jobs. Al Zuwaini and mohanned [24, 25]. Studied the problem $1/r_j/\sum(F_j + U_j)$, and presented a (B&B) algorithm, and application of some dominance rules to solve this problem, finding lower bound by using (SPT)-rule and Moor's algorithm. Computational experience with instances having up to 40 jobs showed that the lower bound was effective in restricting the search.

Scheduling problems of multiple performance measures (three or more) with release date, to the moments of writing this paper. We did not find any study submitted to discuss this subject.

From the above, we can say that scheduling problem often increases complexity by increasing the number of performance measures with release date. Also, we can say that our problem (P) is the first study to address scheduling problem with five criteria and with release date.

In this paper, we describe the problem of scheduling of (n) jobs on one- machine with multiple performance measures and release date, with a view to minimizing the sum total completion time, total tardiness, total earliness, number of tardy jobs and total late work, this problem is denoted by $1/r_j/\sum_{j=1}^{n}(C_j + T_j + E_j + U_j + V_j)$ …(P), from used form 3- field $\alpha/\beta/\gamma$ by Graham et al [26].

This paper is organized as follows: In next section begins with some notation and basic concepts of one-machine scheduling. Formulation of the problem and decomposition into three sub-problem are given in section 3. Some algorithms are given in section 4. Also, three special cases were presented in section 5. in section 6, the (B&B) algorithm was discussed, an account the upper bound and the derivation of a lower bound, and some dominance rules in section 7. Computational results are presented in section 8.

2. Notation

The following notations are used in this paper:
j : Job index.
$\bar{N}$: The set of all n jobs .
n : Number of jobs .
p_j : Processing time for job j .
d_j : Due date for job j .
r_j : Release date for job j .
C_j : Completion time of job j .
$\sum C_j$: The total completion times .
T_j : The tardiness of job j.
$\sum T_j$: The total tardiness.
E_j : The earliness of job j .
$\sum E_j$: The total earliness .
V_j : The late work of job j .
$\sum V_j$: The total late work .
U_j : the unit penalty of job j .
$\sum U_j$: The number of tardy jobs .
(B&B) : Branch and Bound.
UP : Upper bound.
LB : Lower bound.
(SC) : Special cases.
(DR) : Dominance rules.

3. The Mathematical Formulation

The problem (P) considered in this paper is to schedule a set $\bar{N}$ of n jobs , $\bar{N}=\{1,\dots,n\}$ on an one- machine. Each job j , j ϵ $\bar{N}$ has integer processed time p_j , a release date r_j , and due date d_j. Given a schedule $\sigma=(1,\dots,n)$, then for each job j we calculate the completion time by $C_1=r_1+p_1$, $C_j=\max\{r_j, C_j\}+p_j$ for j=2,…,n.
The tardiness of job j is defined by $T_j=\max\{C_j-d_j, 0\}$, and earliness by $E_j=\max\{d_j-C_j, 0\}$. The unit penalty of job j is defined by $U_j=1$, if $C_j>d_j$; o.w, $U_j=0$.The late work of job j given by $V_j=\min\{T_j, p_j\}$. Let δ be a set of all feasible solutions , and σ is a schedule in δ . The mathematical form of our problem (P) can be written as :

$$\left.\begin{aligned}
&M=\text{Min } F(\sigma)=\text{Min}_{\sigma\epsilon\delta}\ \{\textstyle\sum_{j=1}^{n}(C_{\sigma(j)}+T_{\sigma(j)}+E_{\sigma(j)}+U_{\sigma(j)}+V_{\sigma(j)})\}\\
&\text{Subject to :}\\
&C_{\sigma(1)}=r_{\sigma(1)}+p_{\sigma(1)}\\
&C_{\sigma(j)}=\text{Max }\{r_j, C_{j-1}\}+p_j \qquad j=2,\dots,n\\
&T_{\sigma(j)}=\text{Max}\{C_{\sigma(j)}-d_{\sigma(j)}, 0\} \qquad j=1,\dots,n\\
&E_{\sigma(j)}=\text{Max}\{d_{\sigma(j)}-C_{\sigma(j)}, 0\} \qquad j=1,\dots,n\\
&U_{\sigma(j)}=\begin{cases}1 & \text{if } C_{\sigma(j)}>d_{\sigma(j)}\\ 0 & \text{o.w}\end{cases} \qquad j=1,\dots,n
\end{aligned}\right\}\quad \dots(P)$$

$V_{\sigma(j)} = \min \{ T_{\sigma(j)} , p_{\sigma(j)} \} \qquad j = 1,\dots, n$

The objective is to find a processing order σ =(σ(1),…,σ(n)) for the problem (P) to minimize the sum of the total completion times, the total tardiness, the total earliness, the number of tardy, and the total late work.

3.1 Decomposition of Problem (P)

In order to have a less complex structure of the problem (P), can be decomposed into three sub problems (p_1), (p_2) and (p_3), as following:

$$\left.\begin{array}{ll} m_1 = \text{Min}_{\sigma\epsilon\delta} \{ \sum_{j=1}^{n} (C_{\sigma(j)} + T_{\sigma(j)} + E_{\sigma(j)}) \} & \\ \text{Subject to :} & \\ C_{\sigma(1)} = r_{\sigma(1)} + p_{\sigma(1)} & \\ C_{\sigma(j)} = \max \{ r_{\sigma(j)} , C_{\sigma(j-1)} \} + p_{\sigma(j)} & j = 2,\dots,n \\ T_{\sigma(j)} = \max\{ C_{\sigma(j)} - d_{\sigma(j)}, 0\} & j = 1,\dots,n \\ E_{\sigma(j)} = \max\{ d_{\sigma(j)} - C_{\sigma(j)}, 0\} & j = 1,\dots, n \end{array}\right\} \dots\dots(p_1)$$

$$\left.\begin{array}{ll} m_2 = \text{Min}_{\sigma\epsilon\delta} \left(\sum_{j=1}^{n} U_{\sigma(j)}\right) & \\ \text{Subject to :} & \\ U_{\sigma(j)} = \begin{cases} 1 & \text{if } C_{\sigma(j)} > d_{\sigma(j)} \quad j = 1,\dots,n \\ 0 & \text{o.w} \end{cases} & \\ C_{\sigma(1)} = r_{\sigma(1)} + p_{\sigma(1)} & \\ C_{\sigma(j)} = \max \{ r_{\sigma(j)} , C_{\sigma(j-1)} \} + p_{\sigma(j)} & j = 2,\dots,n \end{array}\right\} \dots\dots(p_2)$$

$$\left.\begin{array}{ll} m_3 = \min_{\sigma\epsilon\delta} \left(\sum_{j=1}^{n} V_{\sigma(j)} \right) & \\ \text{Subject to :} & \\ V_{\sigma(j)} = \min \{ T_{\sigma(j)} , p_{\sigma(j)} \} & j = 1,\dots,n \\ T_{\sigma(j)} \geq 0 & j = 1,\dots,n \\ p_{\sigma(j)} \geq 0 & j = 1,\dots,n \end{array}\right\} \dots\dots (p_3)$$

Theorem 1 [26].

$m_1 + m_2 + m_3 \leq M$ where m_1, m_2, m_3 and M are the min-objective function values of (p_1), (p_2),(p_3) and (P) respectively .

4. Two Important Algorithms

4.1. Lawler Algorithm (LA) [27].

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

Step (2): let [j*] such that $[f_{j^*}(\Sigma_{j\in\bar{N}} Pj)= \min_{j\in M}\{ f_j (\Sigma_{j\in\bar{N}} P_j)\}]$, $j\in M$

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

Modify M to represent the new set of the schedule jobs.

Step (3): If $\bar{N} = \emptyset$ stop, o.w go to step (2).

This algorithm, which solves the [$1/prec/f_{max}$ p or $1//f_{max}$] problems, where $f_{max} \in \{C_{max}, L_{max}, T_{max}, V_{max}\}$.

4.2 Moore Algorithm (MA) [21].

Step 1: Order the jobs in (EDD)-rule, let E=L=Ø , and let t= k=0 .
Step 2: k=k+1 , if k > n then go to step (4).

Step 3: set $t=t+P_k$, $E=E\cup \{k\}$ $if\ t \le d_k$.

Then go to step (2).

Otherwise if ($t > d_k$) then find a job $j \in E$,

with p_j is as large as possible and set $E=E- (j)$, $L=L\cup (j)$,

and $t= t- p_j$, go to step (2).

Step 4: E is the step of early jobs , L is the step of late jobs.

This algorithm to solve the ($1//U_j$) problem.

5. Special Cases (SC)

A special cases (SC) for scheduling problem means getting an optimal schedule (optimal solution) directly without utilize (B&B) method or (DP) technique [28].
In this section we present Two (SC) of our problem (P), which are as follows:

Case 1. If $C_j= d_j$ $\forall$ j in a schedule S and the preemptive is allowed then S given an optimal schedule for the problem 1/ r_j, pmtn / $\sum_{j=1}^{n}(C_j + T_j + E_j + U_j + V_j)$.

Proof: From $C_j = d_j$ $\forall$ j in S, then $T_j = E_j = U_j = V_j = 0$ $\forall$ j, therefore problem (P) with preemptive reduced to $1/r_j$, pmtn/$\sum C_j$, but this problem solved in "shortest remaining processing time (SRPT)-rule" [6]. Then S given an optimal schedule for the problem

1/ r_j, pmtn / $\sum_{j=1}^{n}(C_j + T_j + E_j + U_j + V_j)$.

Case 2. If in SPT schedule $r_j=r$ $\forall$ j and satisfy, "(Just In Time) (JIT)" then (SPT)-rule given an optimal schedule for the problem (P).

Proof : From (JIT) we get $C_j = d_j$, then $T_j = E_j = U_j = V_j = 0$ $\forall$ j , therefore problem (P) reduced to $1/r_j/\sum_{j=1}^{n} C_j$, but this problem was solved by (SPT)-rule. Then (SPT)-rule given an optimal schedule for the problem (P).

6. Branch and Bound (B&B) Algorithm

In this section, we apply (B&B) to get an exact solution for our problem (P). The (B&B) method is strategy to explore the solution space based on the implicit enumeration of the solution. This method is based on the idea of calculate all feasible solutions by a special technique, which among them research tree technique, where helps to present the procedures of this method more clearly. Here, at the root node of the search tree, we suggest six heuristic methods to provide an upper bound (UB) on the cost of the optimal schedule. Also, we derive a especially formula to ensure an lower bound (LB), as following:

6.1. Six-Upper Bound (6-UB)

In this section, six heuristic techniques are used for arranging the jobs and valuation the cost problem (P).

1. The first upper bound (UB-1) is obtained by "short release time (SRT)-rule" i.e. ($r_1 \leq \ldots \leq r_n$), then find; UB-1
2. The second upper bound (UB-2) is obtained by, set $j_1 = \min\{r_j + p_j\}$, then order all jobs by (SRT) rule, where j=2,…,n. Then find; UB-2
3. The third upper bound (UB-3) is obtained by order the jobs in non-decreasing of the ($r_j + p_j$) i.e. ($r_1 + p_1 \leq \ldots \leq r_n + p_n$). Then find; UB-3
4. The fourth upper bound (UB-4) is obtained by, set $j_1=\min\{r_j + p_j\}$, then order all jobs by (SPT) rule, i.e. ($p_2 \leq \ldots \leq p_n$). Then find; UB-4
5. The fifth upper bound (UB-5) is obtained by (SPT)-rule. Then find; UB-5
6. The sixth upper bound (UB-6) is obtained by (EDD)-rule. Then find; UB-6

Among these six- heuristics, we select the lowest value to be the an upper bound, (i.e. UB= min {UB-1,..., UB-6}). This UB is then used in a root node of the search tree in (BAB) method.

6.2. Lower Bound (LB)

Finding a lower bound (LB) for our problem (P) is based on decomposing problem (P) into three sub-problems (p_1), (p_2) and (p_3). Then calculate m_1 to be (LB-1) for (p_1), m_2 to be (LB-2) for (p_2), and m_3 to be (LB-3) for (p_3). Then applying the theorem (1):

- To get a lower bound (LB) for problem (p_1) (LB-1). We modified the lower bound of Tariq and Hussam [29]. They proved that (Max{ $\sum_{j=1}^{n} d_{\sigma(j)}$, $\sum_{j=1}^{n} Max$ {$2C_{\sigma(j)} - d_{\sigma(j)}$, $C_{\sigma(j)}$ }) is (LB) for the $1//\sum_{j=1}^{n}(C_{\sigma(j)} + T_{\sigma(j)} + E_{\sigma(j)})$ problem, since this problem is a special case of our problem (p_1), then (Max{ $\sum_{j=1}^{n} d_{\sigma(j)}$, $\sum_{j=1}^{n} Max$ {$2C_{\sigma(j)} - d_{\sigma(j)}$, $C_{\sigma(j)}$ }) is also (LB) of our problem (p_1). But this (LB) is a weak. To improvement this (LB), we suggest the following :

(LB-1)= $nr^{*}+\alpha - R$; where:
n = number of jobs.
r^{*}= min r_j j=1,…,n.
α = (Max{ $\sum_{j=1}^{n} d_{\sigma(j)}$, $\sum_{j=1}^{n} Max$ {$2C_{\sigma(j)} - d_{\sigma(j)}, C_{\sigma(j)}$ }})
R = $\sum_{j=1}^{n} r_j$ j=1,…,n. Then for problem (p_1), the following algorithm obtained this (LB-1).

6.2.1. Algorithm (LB-1)

Step(1): Order the jobs by using (SPT)- rule**.**
Step (2): For each job j compute the completion time C_j, $2C_{\sigma(j)}$ and $\sum_{j=1}^{n} d_{\sigma(j)}$ j=1,…,n.
Step (3): Set n = number of jobs and r^{*}= min r_j j=1,…,n.
Step (4): Set α = (Max{ $\sum_{j=1}^{n} d_{\sigma(j)}$, $\sum_{j=1}^{n}$ Max {$2C_{\sigma(j)} - d_{\sigma(j)}, C_{\sigma(j)}$ }}) and R= $\sum_{j=1}^{n} r_j$ j=1,…,n.
Step (5): Find **(LB-1)= $nr^{*}+\alpha - R$**.
Step (6): Stop.

- For (p_2) we get a (LB-2) for p_2 by applying (Moore algorithm).
- For (p_3) we get a (LB-3) for p_3, firstly we calculate a minimum maximum cost of the late work V_{max} by applying (Lawler algorithm), since $V_{max} \le \sum_{j=1}^{n} V_{\sigma(j)}$, then V_{max} is (LB) of the problem (p_3) i.e.(LB-3). Then we find (LB) of problem (P), where:

(LB) = (LB-1) + (LB-2) + (LB-3) ……………… From Theorem (1).

Both of the (UB), (ILB) represent two values a root node in a search tree.

The (B.A.B) method include of essential procedures the following [24]:

- Branching is the procedure of dividing mother (original) problem into two or more sub-problems. In the search tree, the sub problems are expressed by nodes. The branching rule specified by use forward branching means the jobs are sequenced one by one from the beginning. Or backward branching i.e. (the jobs are sequenced one by one from the end).
- Bounding is procedure the of computing a lower bound on the optimal solution of a sub problems (nodes).
- Search strategy is a procedure reflect the method of choosing a node in the search tree to branching from it, usually the branch be from a node with the smallest (LB) in the search tree , with commitment to the following [30].
    - If (LB) ≥ (UB), then this sub problem cannot yield a better solution for problem. Thus, we need not continue to branch from the corresponding node in the branching tree.
    - If (LB) < (UB), then (UB) is reset to take (LB) value, (i.e. replace (UB) by (LB)). This procedure is repeated until all nodes (sub-sets) have been test.

The at first noteworthy feature of (B&B) is the use of "Dominance- rules" (DR) that try to exclude nodes prior to calculating (LB) to it [2]. These (DR) are computationally useful as they reduce storage requirements on the computer as well as reducing computation time .

7. Dominance- Rules (DR)

Dominance rules (DR) usually specify whether a node can be discarded in search tree before its lower bound (LB) is calculated, so it helps reduce search space. Clearly (DR) are particularly useful when a node can be eliminated which has (LB) that is less than the optimal solution.

To introduce the (DR) for our problem (P) consider schedules S = (σ, i, j, σ') and S' = (σ, j, i, σ') where σ, σ' are two a partial schedule of the remaining n-2 jobs. Let $t=\sum_{k\in\sigma,} p_k$ be the completion time of σ, with $r_j = r$, $d_i \le d_j$, and $p_i \le p_j$.

Define ($C_i(t) + T_i(t) + E_i(t) + U_i(t) + V_i(t)$ an the sum of total completion time, tardiness, earliness, number of tardy jobs and late work of job i) if scheduled at time t and let $F_{ij}= (C_{ij}(t) + T_{ij}(t) + E_{ij}(t) + U_{ij}(t) + V_{ij}(t)$) be the sum of total completion time, tardiness, earliness, number of tardy jobs and late work of job i and j, if i precedes j and their processing starts at time t.

The following interchange function $\Delta_{ij}(t)$ is used to specify the new dominance properties which gives the cost of interchanging adjacent jobs i and j whose processing start at time t.

$\Delta_{ij}(t) = F_{ij}(t) - F_{ji}(t)$.

Note that this cost $\Delta_{ij}(t)$ does not depend on how the jobs are arranged in σ and σ' but depends on start time t of the pair, and :

- If $\Delta_{ij}(t) < 0$ then i precede j at time t.
- If $\Delta_{ij}(t) > 0$ then j precede i at time t.
- If $\Delta_{ij}(t) = 0$ then it is indifferent to schedule i or j first.

First: If $r \le t$, we divide the situation into the following cases:

Case 1. If $d_i \le t + p_i$, $d_j \le t + p_j$ (i.e. both the jobs i, j are always tardy).

Proof. To show that S dominates S', it suffices to show that ($\Delta_{ij}= F_{ij}(S) - F_{ji}(S') \le 0$) and vice versa. Since the jobs i and j are both tardy, then $E_i=E_j=0$, $V_i=p_i$, $V_j=p_j$, and $U_i=U_j=1$. Now, let

- $F_{ij}(S)= [(t +p_i) + (t +p_i - d_i) +0 + 1 +p_i +(t +p_i +p_j) + (t +p_i +p_j -d_j) +0 +1 +p_j]=4t +5p_i +3p_j - d_i -d_j +2$ ….. (a)
- $F'_{ji}= [(t +p_j) + (t +p_j -d_j) +0 +1 +p_j + (t +p_j +p_i) + (t +p_j +p_i -d_i) +0 +1 +p_i]=4t +3p_i +5p_j -d_j -d_i +2$ ….. (b).

Δ_{ij}= (a) - (b)= $2p_i - 2p_j \le 0$, then job i precede j

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

Proof. Since the job i, is always tardy then $E_i=0$, $U_i=1$, and if job j scheduled first then $T_j=0$, and $V_j=\{0, C_j - d_j\}$ (i.e. j , is early or partial early).

- $F_{ij}= [(t +p_i) + (t +p_i - d_i) +0 + 1 +p_i +(t +p_i +p_j) + (t +p_i +p_j -d_j) +0 +1 +p_j]=4t +5p_i +3p_j - d_i -d_j +2$ ….. (a).
- $F'_{ji}= [(t +p_j) +0 + (d_j -t -p_j) +0 +0 +(t +p_j +p_i) + (t +p_j +p_i -d_i) +0 +1 +p_i]= 2t +3p_i +2p_j +d_j - d_i +1$ …. (b). **(when $V_j=0$).**
- $F'_{ji}= [(t +p_j) +0 + (d_j - t -p_j) +0 +(t +p_j -d_j) +(t +p_j +p_i) + (t +p_j +p_i -d_i) +0 +1 +p_i]= 3t +3p_i +3p_j - d_i +1$ …. (c). **(when $V_j=C_j -d_j$).**

1. **Δ_{ij}=(a) – (b)= $2t + 2p_i + p_j - 2d_j + 1 > 0$,** then job j precede i.
2. **Δ_{ij}=(a) – (c)= $t + 2p_i - d_j + 1 > 0$,** then job j precede i.

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

Proof. Since the job i, is always tardy then E_i=0, U_i=1, and if job j is always early, then T_j=0, and V_j={0, $C_j - d_j$} (i.e. j , is early or partial early).

☒ **(when V_j=0).**

- F_{ij}= [$(t + p_i) + (t + p_i - d_i) + 0 + 1 + p_i + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0 + 0$]= $2t + 3p_i - d_i + d_j + 1$ ….. (a).
- F'_{ji}= [$(t + p_j) + 0 + (d_j - t - p_j) + 0 + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + 1 + p_i$]= $2t + 3p_i + 2p_j + d_j - d_i + 1$ …. (b).

Δ_{ij}=(a) – (b)= $-2p_j < 0$, then job i precede j

☒ When **(when V_j=** $C_j - d_j$).

- F_{ij}= [$(t + p_i) + (t + p_i - d_i) + 0 + 1 + p_i + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0 + t + p_i + p_j - d_j$]= $3t + 4p_i + p_j - d_i + 1$ ….(c).
- F'_{ji}= [$(t + p_j) + 0 + (d_j - t - p_j) + 0 + (t + p_j - d_j) + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + 1 + p_i$] = $3t + 3p_i + 3p_j - d_i + 1$….. (e).

Δ_{ij}=(c) – (e)= $p_i - 2p_j \le 0$, then job i precede j

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

Proof. Since the job j, is always tardy then E_j=0, U_j=1, and if job i, scheduled first then T_i=0, and V_i={0, $C_i - d_i$} (i.e. i , is early or partial early).

- **(when V_i=0).** F_{ij}= [$(t + p_i) + 0 + (d_i - t - p_i) + 0 + 0 + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + 1 + p_j$]= $2t + 2p_i + 3p_j - d_j + d_i + 1$ ….. (a).
- F'_{ji}= [$(t + p_j) +$ $(t + p_j - d_j) + 0 + 1 + p_j + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + 1 + p_i$]=$4t + 3p_i + 5p_j - d_j - d_i + 2$ ….. (b).

Δ_{ij}=(a) – (b)= $-2t - p_i - 2p_j + 2d_i - 1 \le 0$, then job i precede j

- **(when V_i= $C_i - d_i$).** F_{ij}= [$(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i - d_i) + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + 1 + p_j$] = $3t + 3p_i + 3p_j - d_j + 1$…. (c).
- F'_{ji}= [$(t + p_j) +$ $(t + p_j - d_j) + 0 + 1 + p_j + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + 1 + p_i$]=$4t + 3p_i + 5p_j - d_j - d_i + 2$ ….. (e).

Δ_{ij}=(c) – (e)= $-t - 2p_j + d_i - 1 \le 0$, then job i precede j

Case 5. If $t + p_i \le d_i$, $t + p_j \le d_j \le t + p_i + p_j$ (i.e. each of the two jobs i and j are tardy if not scheduled first).

Proof:

- **(when V_i=0).** F_{ij}= [$(t + p_i) + 0 + (d_i - t - p_i) + 0 + 0 + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + 1 + p_j$]= $2t + 2p_i + 3p_j - d_j + d_i + 1$ ….. (a).
- **(when V_j=0).** F'_{ji}= [$(t + p_j) + 0 + (d_j - t - p_j) + 0 + 0 + (t + p_j + p_i) + (t + p_j + p_i - d_i) + 0 + 1 + p_i$]= $2t + 3p_i + 2p_j + d_j - d_i + 1$ …. (b).

  Δ_{ij}=(a) – (b)= $-p_i + p_j + 2d_i - 2d_j$ >0, then job j precede i.

- **(when V_i= $C_i - d_i$).** F_{ij}= [$(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i - d_i) + (t + p_i + p_j) + (t + p_i + p_j - d_j) + 0 + 1 + p_j$] = $3t + 3p_i + 3p_j - d_j + 1$…. (c).

- **(when V_j=** $C_j - d_j$). F'_{ji}= $[(t+p_j)+0+(d_j-t-p_j)+0+(t+p_j-d_j)+(t+p_j+p_i)+(t+p_j+p_i-d_i)+o+1+p_i] = 3t+3p_i+3p_j-d_i+1$….. (e).

Δ_{ij}=(c) – (e)= $-d_j + d_i \le 0$, then job i precede j.

- **(when V_i=0).** F_{ij}= $[(t+p_i)+0+(d_i-t-p_i)+0+0+(t+p_i+p_j)+(t+p_i+p_j-d_j)+0+1+p_j] = 2t+2p_i+3p_j-d_j+d_i+1$ ….. (a').
- **(when V_j=** $C_j - d_j$). F'_{ji}= $[(t+p_j)+0+(d_j-t-p_j)+0+(t+p_j-d_j)+(t+p_j+p_i)+(t+p_j+p_i-d_i)+0+1+p_i] = 3t+3p_i+3p_j-d_i+1$….. (e').

Δ_{ij}=(a') – (e')= $-t-p_i+2d_i-d_j$ >0, then job j precede i.

- **(when V_i= $C_i - d_i$).** F_{ij}= $[(t+p_i)+0+(d_i-t-p_i)+0+(t+p_i-d_i)+(t+p_i+p_j)+(t+p_i+p_j-d_j)+0+1+p_j] = 3t+3p_i+3p_j-d_j+1$…. (c').
- **(when V_j=0).** F'_{ji}= $[(t+p_j)+0+(d_j-t-p_j)+0+0+(t+p_j+p_i)+(t+p_j+p_i-d_i)+0+1+p_i] = 2t+3p_i+2p_j+d_j-d_i+1$ …. (b).

Δ_{ij}=(c') – (b')= $t+p_j+d_i-2d_j \le 0$, then job i precede j.

Case 6. If $t+p_i \le d_i \le t+p_i+pj \le d_j$ (i.e. the job i, is tardy if not scheduled first, and j, is always early).

Proof.

- **(when V_i, V_j=0).** Fij= $[(t+p_i)+0+(d_i-t-p_i)+0+0+(t+p_i+p_j)+0+(d_j-t-p_i-p_j)+0+0] = d_i+d_j$ …..(a).
- **(when V_j=0).** F'_{ji}= $[(t+p_j)+0+(d_j-t-p_j)+0+0+(t+p_j+p_i)+(t+p_j+p_i-d_i)+0+1+p_i] = 2t+3p_i+2p_j+d_j-d_i+1$ …. (b).

Δ_{ij}=(a) – (b)= $-2t-3p_i-2p_j+2d_i-1 \le 0$, then job i precede j.

☒ **(when V_i= $C_i - d_i$, and V_j = $C_j - d_j$).** F_{ij}= $[(t+p_i)+0+(d_i-t-p_i)+0+(t+p_i-d_i)+(t+p_i+p_j)+0+(d_j-t-p_i-p_j)+0+t+p_i+p_j-d_j] = 2t+2p_i+p_j$…..(c).

☒ **(when V_j = $C_j - d_j$).** F'_{ji}= $[(t+p_j)+0+(d_j-t-p_j)+0+(t+p_j-d_j)+(t+p_j+p_i)+(t+p_j+p_i-d_i)+0+1+p_i] = 3t+3p_i+3p_j-d_i+1$….. (e).

Δ_{ij}=(c) – (e)= $-t-p_i-2p_j+d_i-1 \le 0$, then job i precede j.

Case 7. If $t+p_i+pj \le d_i \le d_j$ (i.e. both the jobs i, j are always early).

Proof.

☒ When **V_i, V_j** =0.

- F_{ij} = $[(t+p_i)+0+(d_i-t-p_i)+0+0+(t+p_i+p_j)+0+(d_j-t-p_i-p_j)+0+0] = d_i+d_j$ …..(a).
- F'_{ji} = $[(t+p_j)+0+(d_j-t-p_j)+0+0+(t+p_j+p_i)+0+(d_i-t-p_j-p_i)+0+0] = d_j+d_i$….(b).

Δ_{ij}=(a) – (b)=0, then it is indifferent to schedule i or j first.

☒ When **V_i= $C_i - d_i$, V_j = $C_j - d_j$.**

- F_{ij} = $[(t+p_i)+0+(d_i-t-p_i)+0+(t+p_i-d_i)+(t+p_i+p_j)+0+(d_j-t-p_i-p_j)+0+(t+p_i+p_j-d_j)] = 2t+2p_i+p_j$…..(c).
- F'_{ji}= $[(t+p_j)+0+(d_j-t-p_j)+0+(t+p_j-d_j)+(t+p_j+p_i)+0+(d_i-t-p_j-p_i)+0+(t+p_j+p_i-d_i)] = 2t+p_i+2p_j$ ….(e).

Δ_{ij}=(c) – (e)= $p_i - p_j$ **≤ 0**, then job i precede j.

☒ When **V_i= $C_i - d_i$, V_j = 0.**

- $F_{ij} = [(t + p_i) + 0 + (d_i - t - p_i) + 0 + (t + p_i - d_i) + (t + p_i + p_j) + 0 + (d_j - t - p_i - p_j) + 0 + 0] = t + p_i + d_j$ ….(n).
- $F'_{ji} = [(t + p_j) + 0 + (d_j - t - p_j) + 0 + 0 + [(t + p_j + p_i) + 0 + (d_i - t - p_j - p_i) + 0 + (t + p_j + p_i - d_i) = t + p_j + p_i + d_j$ …..(m).

Δ_{ij}=(n) – (m)= $-p_j$ **≤ 0**, then job i precede j.

Second: if r > t , then in the same way above we show that the theorem is true (integral).

Theorem 2

For the problem (P), if $r_i = r$, $d_i = d$, for every $(i \epsilon N)$, and if $p_i \leq p_j$ and $d_j = d$, where jobs (i), and (j) are adjacent jobs, then job (i) must precede job (j) in at least one optimal sequence.

Proof. Using the same Previous methodology the we can prove the validity of this theorem with the following cases:

First: $r \leq t$:

Case 1: If $d \leq t + p_i \leq t + p_j$ (i.e. both the jobs i, j are always tardy).

Proof. To show that S dominates S', it suffices to show that (Δ_{ij}= F_{ij} (S)- F_{ji} (S') ≤ 0). Since the jobs i and j are both tardy, then $E_i=E_j=0$, $V_i=p_i$, $V_j=p_j$, and $U_i=U_j=1$. Now, let

- $F_{ij}(S) = [(t + p_i) + (t + p_i - d) + 0 + 1 + p_i + (t + p_i + p_j) + (t + p_i + p_j - d) + 0 + 1 + p_j] = 4t + 5p_i + 3p_j - 2d + 2$ ….. (a)
- $F'_{ji} = [(t + p_j) + (t + p_j - d) + 0 + 1 + p_j + (t + p_j + p_i) + (t + p_j + p_i - d) + 0 + 1 + p_i] = 4t + 3p_i + 5p_j - 2d + 2$ ….. (b).

Δ_{ij}= (a) - (b)= $2p_i - 2p_j \leq 0$, then job i precede j.

Case 2. If $d \leq t + p_i$, $t + p_i + p_j \leq d$ (i.e. the job i, is always tardy and the job j, is always early).

Proof. Since the job i, is always tardy then $E_i=0$, $U_i=1$, and if job j is always early, then $T_j=0$, and $V_j=\{0, C_j - d_j\}$ (i.e. j , is early or partial early).

☒ **(when V_j=0).**

- $F_{ij} = [(t + p_i) + (t + p_i - d) + 0 + 1 + p_i + (t + p_i + p_j) + 0 + (d - t - p_i - p_j) + 0 + 0] = 2t + 3p_i + 1$… (a)
- $F'_{ji} = [(t + p_j) + 0 + (d - t - p_j) + 0 + 0 + (t + p_j + p_i) + (t + p_j + p_i - d) + 0 + 1 + p_i]$
  $= 2t + 3p_i + 2p_j + 1$…(b)

Δ_{ij}=(a) – (b)= $-2p_j < 0$. then job i precede j.

☒ When **(when V_j=** $C_j - d$).

- $F_{ij} = [(t + p_i) + (t + p_i - d) + 0 + 1 + p_i + (t + p_i + p_j) + 0 + (d - t - p_i - p_j) + 0 + t + p_i + p_j - d] = 3t + 4p_i + p_j - d + 1$ ….(c).
- $F'_{ji} = [(t + p_j) + 0 + (d - t - p_j) + 0 + (t + p_j - d) + (t + p_j + p_i) + (t + p_j + p_i - d) + 0 + 1 + p_i] = 3t + 3p_i + 3p_j - d + 1$….. (e).

Δ_{ij}=(c) – (e)= $p_i - 2p_j \leq 0$, then job i precede j.

Case 4. If $t + p_i + p_j \leq d$ (i.e. both the jobs i, j are always early).

Proof.

☒ When **V_i, V_j** =0.

- $F_{ij} = [(t + p_i) + 0 + (d - t - p_i) + 0 + 0 + (t + p_i + p_j) + 0 + (d - t - p_i - p_j) + 0 + 0] = 2d$ …..(a).
- $F'_{ji} = [(t + p_j) + 0 + (d - t - p_j) + 0 + 0 + (t + p_j + p_i) + 0 + (d - t - p_j - p_i) + 0 + 0] = 2d$….(b).

  Δ_{ij}=(a) – (b)=0, then it is indifferent to schedule i or j first.

☒ When $\mathbf{V_i = C_i - d}$, $\mathbf{V_j = C_j - d}$.

- $F_{ij} = [(t + p_i) + 0 + (d - t - p_i) + 0 + (t + p_i - d) + (t + p_i + p_j) + 0 + (d - t - p_i - p_j) + 0 + (t + p_i + p_j - d)] = 2t + 2p_i + p_j$…..(c).
- $F'_{ji} = [(t + p_j) + 0 + (d - t - p_j) + 0 + (t + p_j - d) + (t + p_j + p_i) + 0 + (d - t - p_j - p_i) + 0 + (t + p_j + p_i - d)] = 2t + p_i + 2p_j$ ….(e).

$\mathbf{\Delta_{ij} = (c) - (e) =}$ $p_i - p_j \le \mathbf{0}$, then job i precede j.

Second: if r > t , then in the same way above we show that the theorem is true (integral).

8. Computational Experience

An intensive work of numerical experimentations has been performed subsection (8.1) shows how instances (test problems) can be randomly generated.

8.1. Test problems: We created (10) problems randomly, for each problem, n$\in$ {5, …, 16} jobs, and for each job j has the following data:

- The processing time p_j is generated from the discrete uniform distribution [1,10].
- Integer due date d_j is generated from the uniform distribution [1, (1-Tf+Rrdd/2)Sp], where Sp= $\sum_{j=1}^{n} p_j$, (Tf) is the "tardiness factor", and (Rrdd) is the "relative range of the due dates". For the two parameters (Tf) and (Rrdd), the values (0.2, 0.4, 0.6, 0.8, 1.0) are considered.
- Integer release date r_j is generated for each j from the uniform distribution [1, 5].

8.2. Computational Experience with the (UB) and (LB) of (B&B) Algorithm

The (B&B) algorithm was tested by coding in (Mat lab 2018) and running on a personal computer Dell Core i7 with Ram 8 GB. **Tables 1, 2.** Shows the results to problem (P) obtained by (B&B) algorithm, when n $\in${5,6,…,10} and n$\in${11,12,…,16} respectively. The first column (n) indicate to the number of jobs, the second column (EX) indicate to the number of examples for each instance n, the third column (CEM) complete enumeration method only in **Tables 1.** The fourth column (optimal) indicate to the optimal solution obtained by(B&B) method, the fifth and sixth columns indicate to upper bound (UB) and initial lower bound(ILB) respectively, and the other columns (NON) are indicate to number of nodes, time (CEM), and time(B&B) , finally, column (status) indicate to the problem solved (0) or not (1). The symbols (*) indicate to the (UB) given an optimal value and (**) indicate to the (ILB) given an optimal value. The (B&B) algorithm was stopped when the sum of (status column ≥3). A condition for stopping the (B&B) algorithm was determined and considering that the problem is unsolved (state is 1). Here, the (B&B) algorithm is stopped, after (1800) second. From **Tables 1, 2.** We are noticed that the six heuristic of upper bound given good results, it gives the value for objective function equal to optimal or near optimal value.

We also have two other **Tables 3, 4.** Are the summary of the two previous **Tables 1, 2.** That show the average computational time of (CEM) and (B&B), the average of nodes, and the unsolved problems.

Table 1. The performance of CEM, optimal of (B&B), (UB), (ILB), number of node and (CPU) in seconds of (CEM) and (B&B), for n = (5,7,10).

n	EX	CEM	B&B	UB	LB	NON	T.(CEM)	T.(B&B)	Status
	1	163	163	166	123	41	0.0217488	0.110176	0
	2	160	160	160*	102	51	0.0040399	0.025487	0
	3	207	207	219	177	57	0.0043776	0.016705	0
5	4	93	93	93*	44	69	0.003325	0.009982	0
	5	87	87	87*	55	24	0.0113214	0.016442	0
	6	95	95	95*	71	25	0.002008	0.00461	0
	7	166	166	166*	129	94	0.002042	0.007144	0
	8	135	135	136	98	57	0.0020626	0.004578	0
	9	168	168	171	115	51	0.0019062	0.003848	0
	10	145	145	145*	113	38	0.0018841	0.003119	0
	1	244	244	286	183	235	0.1016655	0.1174261	0
	2	152	152	159	113	362	0.0814766	0.0421561	0
	3	188	188	188*	131	211	0.0738479	0.0200314	0
	4	158	158	181	122	256	0.073096	0.0223296	0
	5	311	311	317	270	226	0.0808035	0.0301934	0
7	6	146	146	149	109	226	0.0697635	0.0211181	0
	7	254	254	254*	192	583	0.0714399	0.0397211	0
	8	230	230	260	175	291	0.0715506	0.014869	0
	9	306	306	308	245	731	0.0696867	0.0437375	0
	10	199	199	209	162	271	0.0700934	0.0178577	0
	1	540	540	595	478	6799	62.948728	0.5378211	0
	2	377	377	377*	314	8800	57.021325	0.5824576	0
	3	525	525	540	423	11431	57.77288	0.7884419	0
	4	325	325	363	261	23298	60.377976	1.3742526	0
	5	402	402	411	324	7183	57.3601	0.4897919	0
10	6	322	322	369	239	13592	61.739524	0.5339548	0
	7	370	370	394	307	3135	56.726814	0.2065081	0
	8	377	377	383	304	7338	59.551903	0.4800463	0
	9	372	372	393	288	4955	60.867124	0.293253	0
	10	526	526	556	442	29039	59.035872	1.846271	0

Table 2. The performance of optimal of (B&B), (UB), (ILB), number of node and (CPU) in seconds of (B&B) for n=(11, 13, 16).

n	EX	B&B	UB	LB	NON	T.(B&B)	Status
11	1	504	559	404	28164	1.798678	0
	2	597	597*	494	16666	1.1426506	0
	3	412	412*	339	37043	2.4015924	0
	4	315	384	230	14400	0.6151017	0
	5	626	665	541	66156	4.2628177	0
	6	599	599*	518	12403	0.8402274	0
	7	416	416*	332	88787	5.5034052	0
	8	550	578	438	33125	1.968353	0
	9	446	446*	364	52656	3.3493212	0
	10	563	580	469	77098	4.981989	0
13	1	758	786	678	966958	68.04239	0
	2	782	797	664	384969	26.9411	0
	3	869	871	773	1799803	122.9181	0
	4	551	568	426	764902	55.11314	0
	5	726	735	616	803127	54.33091	0
	6	855	866	726	461218	34.64196	0
	7	544	588	431	70847	5.386437	0
	8	767	824	658	1013911	100.976	0
	9	725	795	578	583650	28.91363	0
	10	845	889	687	2107275	120.6244	0
16	1	1150	1157	1022	12015015	976.3893616	0
	2	1193	1280	1101	15004109	1141.920575	0
	3	914	990	793	14240404	767.3393739	0
	4	1226	1242	1107	24168411	1800.001399	1
	5	789	864	677	6028307	513.3604747	0
	6	940	989	828	22692450	1800.000598	1
	7	950	1092	836	8260306	504.8303402	0
	8	1182	1183	1061	23951871	1800.000418	1
	9	1067	1086	951	21206817	1800.000022	1
	10	1004	1059	879	20173233	1676.25391	0

In the Tables 1, 2. We have:
n : Number of jobs.
EX : Example number.
CEM: The optimal solution obtained by (CEM).
B&B: The optimal solution obtained by (B&B) method.
UB: The upper bound obtained from (section 6.1).
LB: The lower bound obtained from (section 6.2).
NON: Number of nodes.
T.CEM: The time (in seconds) which is required for (CEM), only in Table 1.
T.BAB: The time (in seconds) which is required for (B&B).

Status$=\begin{cases} \mathbf{0} & \boldsymbol{if\ the\ problem\ is\ solved} \\ \mathbf{1} & \boldsymbol{o.w.} \end{cases}$

***= The upper bound givens the optimal value.**
****= The (ILB) given an optimal value.**

The following tables summarize **Tables 1, 2.**

Table 3. Summary of the **Table 1.**

n	Av.T.(CEM)	Av. T.(B&B)	Av. NON	N. Uns. P
5	0.005472	0.0202091	50.7	0
7	0.076342	0.036944	339.2	0
10	59.34022	0.7132798	11557	0

Table 4. Summary of the **Table 2.**

n	Av. T.(B&B)	Av. NON	Uns. P
11	2.6864136	42650	0
13	61.78880509	895666	0
16	1278.009647	17052164	4

In Table 3, 4. We have:
n : Number the jobs.
Av.T. (CEM) :Average computational time of (CEM).
Av.T. (B&B):Average computational time of (B&B).
Uns. P: The unsolved problem

8. Conclusion and Future Work

In this paper, we been developed exact solutions for the problem of scheduling (*n)* jobs on one- machine to minimize the sum total completion time, total tardiness, total earliness, number of tardy jobs and total late work with unequal release dates. A branch

and bound (B&B), is used to solve to our problem. Computational experience showed the proposed (B&B) algorithm is effective in solving problems with up to (16) jobs.
The study of this problem opens up new horizons for future research and here we can refer to the most important ideas that we want to work on in the future:

1. Develop the suggested lower bound (LB) for problem, as well as dominance rules (DR), in order to reduce the search space.
2. Use local search methods to solve our problem.
3. from of the topics of interesting to us in the future are to examine the following two problems:
   a) $1 / r_j / F(\sum C_j, \sum T_j, \sum E_j)$
   b) $1 / r_j / F(\sum C_j, \sum U_j, T_{max})$

References

1. Chen, W.Y.; Sheen, A. Pareto- optimal solution procedure for the single- machine scheduling problem with release time and multiple performance measures. *Journal of the Chinese Institute of Industrial Engineers*.**2011**, *28*, *5*, 346-359
2. Lenstra, J.K.; Kan, A.R.; Brucker, P. Complexity of machine scheduling problems. Annals of discrete mathematics. *Elsevier*.**1977**, *1*, 343-362.
3. Smith, W.E. various optimizers for single-stage production. Naval Research Logistics Quarterly. **956**, *3*, *1-2*, 59-66.
4. Schrage, L. Letter to the editor—a proof of the optimality of the shortest remaining processing time discipline. *Operations Research*.**1968**, *16*, *3*, 687-690.
5. Chandra, R. On n/1/F dynamic deterministic problems. *Naval Research Logistics Quarterly*.**1979**, *26*, *3*, 537-544.
6. Dessouky, M.I.; Jitender, S.D. Sequencing jobs with unequal ready times to minimize mean flow time. *SIAM Journal on Computing*.**1981**, *10*, *1*, 192-202.
7. Ahmadi, R.H.; Uttarayan, B. Lower bounds for single-machine scheduling problems. *Naval Research Logistics (NRL)*.**1990**, *37*, *6*, 967-979.
8. Chu, C. A branch-and-bound algorithm to minimize total flow time with unequal release dates. *Naval Research Logistics (NRL)*.**1992**, *39*, *6*, 859-875.
9. Rinnooy Kan, A.H.G. Machine sequencing problem: Classification. *Complexity and Computation*, **1976**.
10. Chu, C. A branch-and-bound algorithm to minimize total tardiness with different release dates. *Naval Research Logistics (NRL)*.**1992**, *39*, *2*, 265-283.
11. Baptiste, P.; Jacques, C.; Antoine, J. A branch-and-bound procedure to minimize total tardiness on one machine with arbitrary release dates. *European Journal of Operational Research*.**2004**, *158*, *3*, 595-608.
12. Mohammed, H.A. Approximation algorithms for minimizing the total weighted earliness on machines scheduling. *Journal of Karbala university*.**2009**, *7*, *1*, 23-33.
13. Sidney, J.B. Optimal single-machine scheduling with earliness and tardiness penalties". Operations Research.**1977**, *25*, *1*, 62-69.
14. Mahnam, M.; Ghasem, M. A branch-and-bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times. *Engineering Optimization*. **2009**, *41*, *6*, 521-536.

15. Błażewicz, J. Scheduling preemptible tasks on parallel processors with information loss. *Recherche Technique et. Science Informatiques*.**1984**, *3*, 415–420.
16. Potts, C.N.; Van Wassenhove, L.N. Approximation algorithms for scheduling a single machine to minimize total late work. *Operations Research Letters*.**1992**, *11*, *5*, 261-266.
17. Potts, C.N.; Van Wassenhove, L.N. Single machine scheduling to minimize total late work. *Operations Research*.**1991**, *40*, *3*, 586-595.
18. Kethley, R.B.; Bahram, A. Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms. *Computers & Industrial Engineering*.**2002**, *43*, *3*, 509-528.
19. Abdul Razaq, T.S.; Al Saidy, S.K.; Al Zuwaini, M.K. Single Machine Scheduling to Minimize Total Weighted Late Work with Release Date. *J. of Al-Qadisyah for pure science*.**2008**, *13*, *4*, 91-112.
20. Moore, J.M. one machine sequencing algorithm for minimizing the number of late jobs. *Management science*.**1968**, *15*, *1*, 102-109.
21. Dauzère, P.S. Minimizing late jobs in the general one machine scheduling problem. *European Journal of Operational Research*.**1995**, *81*, *1*, 134-142.
22. Baptiste, P.; Laurent, P.; Eric, P. A branch and bound to minimize the number of late jobs on a single machine with release time constraints. *European Journal of Operational Research*.**2003**, *144*, *1*, 1-11.
23. Briand, C.; Samia, O. Minimizing the number of tardy jobs for the single machine scheduling problem: MIP-based lower and upper bounds. *RAIRO-Operations Research*. **2013**, *47*, *1*, 33-46.
24. Al-Zuwaini, M.K.; Mohanned, M.K. One Machine Scheduling Problem with Release dates and Tow Criteria. *Journal of Thi-Qar University*.**2011**, *6*, *2*, 1-14.
25. Graham, R.L.; Lawler, E.L.; Lenstra, J.K.; Kan, A.R. Optimization and approximation in deterministic sequencing and scheduling. a survey. In Annals of discrete mathematics. *Elsevier*.**1979**, *5*, 287-326.
26. Mahmood, A. A. Solution procedures for scheduling job families with setups and due dates. Diss. M. Sc. Thesis, University of AL-Mustansiriyah, College of Science, Dept. of Mathematics, **2001**.
27. Lawler, E.L. Optimal sequencing of a single machine subject to precedence constraints. *Management science*.**1973**, *19*, *5*, 544-546.
28. Hanan, A.C. Exact Approximation Algorithms for Scheduling with and without Setup times Problems. PH. D. Thesis. University of Mustansiriya, college of Science, **2007**.
29. Tariq, S.A.; Hussam, A.M. Simulation Annealing and Genetic Algorithms for the Single Machine Scheduling Problem. AL- Mustansiriya J. Sci. **2010**, *21*, *7*, 24-33.
30. Baúto, J.; Rui, N.; Nuno, H. Parallel Genetic Algorithms for Financial Pattern Discovery Using GPUs. *Springer International* Publishing, **2018**.