Int. J. Nonlinear Anal. Appl. 13 (2022) No. 1, 2075-2085
ISSN: 2008-6822 (electronic)
http://dx.doi.org/10.22075/ijnaa.2022.5904

Theoretical approaches and special cases for a single machine with release dates to minimize four criterion

Omar Osama Daowd[a], Hanan Ali Chachan[a,*], Hazim G. Daway[a]

[a]Department of Mathematics, College of Sciences, Al-Mustansireah University, Baghdad, Iraq

(Communicated by Madjid Eshaghi Gordji)

Abstract

We propose a multi-objective machine scheduling problem (MSP) in this study. The sum of total flow time, total tardiness, total earliness, and total late work is the topic under discussion. With an arbitrary release date, This paper offers a theoretical analysis, discussion, and proofs for a number of special instances that apply to our topic.

Keywords: Multi-objective problems (MOP), single machine release date.
2010 MSC: Primary 90C33; Secondary 26B25.

1. Introduction

This article describes the challenge of scheduling (n) tasks on a single machine with several performance measurements and a release date with the goal of reducing the sum total flow time, total tardiness, total earliness, and total late work. This problem is represented by $1/\boldsymbol{r_j}/\sum_{\boldsymbol{j=1}}^{n}(\boldsymbol{F_J}+\boldsymbol{T_J}+\boldsymbol{E_J}+\boldsymbol{V_J})$, from used froṭ 3-filed $\alpha/\beta/\gamma$ by Graham et al [12].

Various investigations that were introduced in the early years of the discovery of the theory of scheduling focused on a single performance measuring criterion. But because of the increasing role of scheduling theory in the decision-making process in different fields for the real-world environments, whether in the field of industrial production or service, which often require more than one criterion to measure the quality of performance [8]. In addition to the spread of production, philosophy was timely which is referred to by (just-in-time (JIT)), which is based on an idea both earliness and tardiness are can be detrimental, for example, tardiness causes loss of customers or delayed payments. while earliness causes inventory carrying cost and insurance and so on [3]. These and

*Corresponding author
Email addresses: omarmaysan@yahoo.com (Omar Osama Daowd), hanomh@uomustansiriyah.edu.iq (Hanan Ali Chachan), hazimdo@uomustansiriyah.edu.iq (Hazim G. Daway)

Received: September 2021 *Accepted:* November 2021

other reasons have attracted researchers attention to scheduling problems with multiple criteria (two criteria or more). In this paper, we examine the problem of scheduling with four criteria (referred to above) for measuring performance with the release time, and to the best of our knowledge, this problem is not studied before. Before talking about our problem (P), we find 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 time requiled for completion

There has been much research done on this criterion in the literature For $1/r_j/\sum C_j$ is NP- hard in the strong sense by Lenstra, Rinnooy Kan and Bruker [14]. The "shortest- processing- time (SPT)-rule" of smith was used to address this problem with an equal release date. Also, the problem $1/r_j$, pmtn $/\sum C_j$ is amenable to resolution via the "shortest remaining processing- time (SRPT)-rule" of Schrage [20]. Chandra's "dominance rules (DR)" play a significant role in several branch and bound algorithms for the issue $1/r_j/\sum C_j$ [7, 11]. For such issue, Ahmadi and Bghchi [2]. Display that the lower bound which is symbolized by (SRPTLB) obtained by using the (SRPT)- rule to solve the problem 1/ r_j, put $/\sum C_j$ best (LB) in comparison other know bounds, also, see [9]. Uses this (SRPT-LB) in (B&B) algorithm that is effective in solving problems, when n $\leq$ 100 jobs.

The time of tardiness

The second one of the most significant requirements in practice is to keep the overall amount of tardiness to a bare minimum. Rinnooy proved that the problem of $1/r_j/\sum T_j$ is NP- difficult in the strict sense [19]. The (BAB) algorithm is proposed, and several dominant qualities are demonstrated. According to Schrage [20],(LB) is dependent on the construction of the timetable for which the jobs are scheduled in accordance with the (SRPT)-rule (by resting the issue under the assumption that the jobs are preemptive). When $n \leq 260$ tasks were involved, the suggested method made a significant contribution to issue solving. A(B& B) method with new lower limits (NLB) was presented, which was based on the improved (LB) of Chu [9]. In addition, several well-known dominance characteristics were generalized in order to address the $1/r_j/\sum T_j$ issue. The number of instances handled by this technique is around 500 jobs.

The beginning of time

Early arrival is one of the scheduling objectives that few researchers have studied since for long years academics focused on single-criterion regular performance measures (i.e. no reducing in Cj for all j). Most studies that use punctuality as an optimality criterion also include a component of tardiness. Sidney [21] published the first research on earliness and tardiness (E/T) fines.

Who invented a polynomial method to reduce the maximum of (E/T) in a single machine problem? Mahnam and Ghasem presented one of the most current investigations in recent years [15]. Who studied this problem $1/r_j/\ E_{\max} + T_{\max}\cdot$ 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 $(1/r_j, p\,\mathrm{mtn}\,/T_{\max})$ and $(1/r_j, p\,\mathrm{mtn}\,/E_{\max})$. 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 lateness of the work

Processed after the due date, as shown by (Vj). Initially, Blazewicz called this "information loss" [5]. The word was proposed as (Total late work). For the problem $1/pmtn/\sum V_j$, the researchers displayed that the minimum $(1//\sum V_j) \leq [(T_{\max})$ for the (EDD)-rule sequence]. Potts and Van Wassenhove [17, 18] have determined that the issue $1//\sum V_j$ is NP-hard. Kethley and Bahram are two of the most talented in machine scheduling.

In this study, we took into account $1//\sum W_j V_j$ the whole weighted late work under the assumption that all jobs arrived simultaneously (i.e., the release date $\mathrm{r_j} = 0$ for all jobs) and calculated the total weighted late work. Potts and Van Wassenhove [17, 18] developed a more broad version of the function than the formula they used. Abdul Razaq et al [1] investigated the $1/r_j/\sum W_j V_j$ problem with mismatched release dates. In order to solve this problem, various specific instances were proofed, as well as a branch and bound a method with as many as 30 tasks was used to solve the problem. In addition, five local search strategies to address such issues were used, and their performance was tested with up to 60000 tasks, which demonstrated that this problem is NP-hard to solve.

The unit penalty

Moore [16] provides an example of the $1//\sum U_j$. The study was one of the first to explore scheduling in order to reduce this difficulty, using a method known as the Moor Algorithm (also known as Hudgson's Algorithm) that handled optimally solving the problem. With release dates, the problem $1/r_j/\sum U_j$ is strongly $-NP-$ hard by Lenstra et al [14]. Dauzere [10] studied this $1/r_j/\sum U_j$ problem, and determined a lower bound based on relaxation of a "Mixed-Integer- Linear-Programming" formulation presented a heuristic method for problem resolution. Up to 50 tasks have been evaluated on a wide range of issues. When compared to the lower limit, the proposed method was shown to be more efficient. To solved this $1/r_j/\sum U_j$ problem, who suggesting $(B\&B)$ algorithm, based on lower bounds on a "Lagrangian relaxation". With the use of these strategies, they were able to solve optimality instances with up to 200 tasks using dominance criteria [6]. The current investigation displayed how good-lower and upper bounds on quality that can be calculated in relation to the problem $1/r_j/\sum U_j$, using an original mathematical integer programming formulation. Up to 160 jobs might be saved as a result of the proposed method's evaluation. Al-Zuwaini and Mohanned [4] investigated the problem $1/r_j/\sum (F_j + U_j)$, and presented a $(B\&B)$ algorithm, and to solve this problem, apply some dominance rules, finding lower bound by using (SPT) - rule & Moor's algorithm. The lower limitation was shown to be useful in limiting the search in cases with up to 40 jobs.

2. The formulation in mathematics

The problem (P) regarded in the present study is to schedule a set N of n jobs, $N = \{1, \ldots, n\}$ on an one $-$ machine. Each job, $j \in N$ has integer processed time pj, a release date rj, and due date dj. Given a schedule $\sigma = (1, \ldots, \mathrm{n})$, the flow time of job j, $\mathrm{F_j}$ can be define as $\mathrm{F_j} = \mathrm{C_j} - \mathrm{r_j}$ where C_j be completion time for job j, given by relationship : $C_1 = r_1 + p_1, C_j = \max\{r_j, C_j\} + p_j$ for $j = 2, \ldots, n$ The tardiness of job j is referred to by $\mathrm{T_j} = \max\{\mathrm{C_j} - \mathrm{d_j}, 0\}$, and earliness by $\mathrm{E_j} = \max\{\mathrm{d_j}- \ C_j, 0\}$

The late work of job j given by $\mathrm{Vj} = \min\{\mathrm{Tj}, \mathrm{pj}\}$. Let δ be a collection of all conceivable answers, and σ is a schedule in δ. The mathematical formulation of our issue (P) is as follows:

$$\left.\begin{array}{ll} M = \operatorname{Min} F(\sigma) = & \\ \quad \operatorname{Min}_{\sigma\in\delta}\left\{\sum_{\mathrm{j}=1}^{\mathrm{n}}\left(\mathrm{C}_{\sigma(\mathrm{j})} + \mathrm{T}_{\sigma(\mathrm{j})} + \mathrm{E}_{\sigma(\mathrm{j})} + \mathrm{V}_{\sigma(\mathrm{j})}\right)\right\} & \\ \text{Subject to:} & \\ C_{g(1)} = r\sigma(1) + p\sigma(1) & \\ C_{q(j)} = \operatorname{Max}\{r_j, C_{j-1}\} + p_j & j = 2,\dots,n \\ T_{g(j)} = \operatorname{Max}\left\{C_{\sigma(j)} - d_{\sigma(j)}, 0\right\} & j = 1,\dots,n \\ E_{g(j)} = \operatorname{Max}\left\{d_{\sigma(j)} - C_{\sigma(j)}, 0\right\} & j = 1,\dots,n \\ V_{\sigma(j)} = \operatorname{Min}\left\{T_{\sigma(j)}, p\sigma(j)\right\} & j = 1,\dots,n \end{array}\right\} ..(P)$$

This task's goal is to identify a processing order $\sigma = (\sigma(1),\dots,\sigma(n))$ for the problem (P) that minimizes t he sum of all flow times, all lateness, all earliness, all lateness, all tardiest, all tardy workers, and all late work.

3. Problem (P) decomposition

To simplify the problems structure, (P), it may be divided into two subproblems (P_1) and (P_2) as follows:

$$\left.\begin{array}{ll} m_1 = \operatorname{Min}_{\sigma\in\delta}\left\{\sum_{j=1}^{\mathrm{n}}\left(C_{\sigma(j)} + T_{\sigma(j)} + E_{\sigma(j)}\right)\right\} & \\ \text{Subject to:} & \\ C_{\sigma(1)} = r_{\sigma(1)} + p_{\sigma(1)} & \\ C_{\sigma(j)} = \operatorname{Max}\left\{r\sigma(j), C_{\sigma(0-1)}\right\} + p\sigma(j) & j = 2,\dots,n \\ T_{\sigma(j)} = \operatorname{Max}\left\{C_{\sigma(j)} - d_{\sigma(j)}, 0\right\} & j = 1,\dots,n \\ E_{g(j)} = \operatorname{Max}\left\{d_{\sigma(j)} - C_{\sigma(j)}, 0\right\} & j = 1,\dots,n \end{array}\right\} ..(P)$$

$$\left.\begin{array}{ll} \mathrm{m}_2 = \operatorname{Min}_{\sigma\in\delta}\left(\sum_{\mathrm{j}=1}^{\mathrm{n}} \mathrm{V}_{\sigma_{(j)}}\right) & \\ \text{Subject to:} & \\ \mathrm{T}_{\sigma(\mathrm{j})} \geq 0 & \mathrm{j} = 1,\dots,\mathrm{n} \\ \mathrm{p}_{\mathcal{Q}(\mathrm{j})} > 0 & \mathrm{j} = 1,\dots,\mathrm{n} \\ \mathrm{V}_{g(\mathrm{j})} = \operatorname{Min}\left\{\mathrm{T}_{g(\mathrm{j})}, \mathrm{p}_{\alpha(\mathrm{j})}\right\} & \mathrm{j} = 1,\dots,\mathrm{n} \end{array}\right\} ..(P)$$

4. Special cases (SC)

A special case (SC) is defined for scheduling problems as obtaining an ideal schedule (optimal solution) without utilizing either the *B&B* approach or the differential programming methodology. Two (SC) of our problem's (P) solutions are shown in this part. They are as follows:

Case 1. Suppose that in a schedule S, the condition $C_j = d_j \quad \forall$ is met, and the preemptive option is permitted. Then S is provided an optimum solution to the issue.

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

Proof . Since $C_j = d_j \forall \mathrm{j}$ in S, then $T_j = E_j = V_j = 0$, then the problem P with preemptive reduced to $1/r_j, pmtn/\sum C_j$, but this problem solved in *SRPT* rule, then S given an optimal solution for the problem

$$1/rj, pmtn/\sum_{\mathrm{j}=1}^{\mathrm{n}}\left(\mathrm{C}_{\mathrm{j}} + \mathrm{T}_{\mathrm{j}} + \mathrm{E}_{\mathrm{j}} + \mathrm{V}_{\mathrm{j}}\right).$$

Case (2). If the SPT schedule $r_j = r \quad \forall$j and the just-in-time (IIT) requirement are met, then SPT will provide an optimum solution for the problem P. □
Proof . From (JIT) we get $C_j = \mathrm{d}_j$ and $T_j = E_j = V_j = 0, \forall$j then the problem H reduced to $1/\mathrm{r}/\sum_{\mathrm{j}=1}^{\mathrm{n}} C_{\mathrm{j}}$, while the SPT rule was able to resolve this issue. As a result, SPT provides an optimum solution to the issue P. □

5. Dominance-rules(DR)

Dominance Rules (DR) are rules that indicate whether a node in a search tree can be removed before its lower bound (LB) is determined, which helps to decrease the amount of search space required. (DR) are clearly more effective when a node can be deleted that has (LB) that is smaller than the best solution, as seen in the following example.Let us examine two partial schedules of the remaining $S = (\sigma, i, j, \sigma')$ and $S' = (\sigma, j, i, \sigma')$ where I and j are two integers representing the number of jobs remaining in the schedule. Let t= $\sum_{k\in\sigma,} p_k$ be the completion time of, with $r_i = r_j = r, d_i \le d_j$, and $p_i \le p_j$ being the initial and final positions, respectively.

Assume that job I is scheduled at time t and that $F_i = C_i(t) + T_i(t) + E_i(t) + V_i(t)$ is the total completion time, tardiness, earliness, number of tardy jobs, and late work of job I and that if I comes before j and their processing begins at time t. In order to express the new dominant qualities, the interchange function $\boldsymbol{\Delta}_{ij}(t)$ is utilized. This function calculates the cost of interchanging two nearby jobs, I and j, whose processing begins at the same time as time t.

$$\boldsymbol{\Delta}_{ij}(t) = \mathcal{F}_{ij}(t) - \mathcal{F}_{ji}(t).$$

Keep in mind that this cost $\Delta_{ij}(t)$ is not dependent on the order in which the jobs are organized in σ and σ', but rather dependent on the start time to f the pair, and that:

1. If $\Delta_{ij}(t) < 0$ then should precedes j at time t.

2. If $\boldsymbol{\Delta}_{ij}(t) > 0$ then j should precedes i at time t.

3. If $\boldsymbol{\Delta}_{ij}(t) = 0$ then it is no matter to schedule i or j first.

First: We can categorize the situation into the following categories if $r \le t$ is true.

Case 1. If $d_i \le t + p_i, d_j \le t + p_j \quad$ (i.e. both the jobs i, j are always tardy).
Proof . If we prove that S dominates S', we just need to display that $(\Delta_{ij} = \mathcal{F}_{ij}$. Because jobs I and j are both late, $E_i = E_j = 0, Vi = pi$, and $Vj = pj$, and now, let us assume that $E_i = E_j = 0, V_i = p_i, V_j = p_j$

$$* \quad \mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + (\mathrm{t} + p_i - d_i) + 0 + p_i + (\mathrm{t} + p_i + p_j) - r + (\mathrm{t} + p_i + p_j - d_j) + 0 + p_j] = 4\mathrm{t} + 5p_i + 3p_j - d_i - d_j - 2r \ldots (\mathrm{a})$$
$$* \, \mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + (\mathrm{t} + p_j - d_j) + 0 + p_j + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 4\mathrm{t} + 3p_i + 5p_j - d_j - d_i - 2r \quad \ldots . (\mathrm{b})$$

$\Delta_{ij} = (\boldsymbol{a}) - (\boldsymbol{b}) = 2\boldsymbol{p}_i - 2p_j \le \mathbf{0}$, then job icomes before 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 . There is no job i thus $E_i = 0$, and $T_j = 0$, and if the first planned work j is delayed and $V_j = \{0, C_j - d_j\}$ (i.e. j, is early or partial early).

$$*\,\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + (\mathrm{t} + p_i - d_i) + 0 + p_i + (\mathrm{t} + p_i + p_j) - r + (\mathrm{t}+ p_i + p_j - d_j) + 0 + p_j] = 4\mathrm{t} + 5p_i + 3p_j - d_i - d_j - 2r \ldots (\mathrm{a})$$
$$*\,\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r+ (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 2\mathrm{t} + 3p_i + 2p_j + d_j - d_i - 2r \ldots (\mathrm{b})$$

$(\boldsymbol{when V}_j = \mathbf{0})$

$$*\,\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + (\mathrm{t} + p_j + d_j) + (\mathrm{t} + p_j+ p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 3\mathrm{t} + 3p_i + 3p_j - d_i - 2r \ldots (\mathrm{c})$$
$(\boldsymbol{when V}_j = \boldsymbol{C}_j - \boldsymbol{d}_j)$

1- $\boldsymbol{\Delta}_{ij} = (\boldsymbol{a}) - (\boldsymbol{b}) = 2\boldsymbol{t} + 2\boldsymbol{p}_i + \boldsymbol{p}_j - \mathbf{2}\boldsymbol{d}_j > \mathbf{0}$, then job j precedes i

2- $\boldsymbol{\Delta}_{ij} = (\boldsymbol{a}) - (\boldsymbol{c}) = \boldsymbol{t} + 2\boldsymbol{p}_i - \boldsymbol{d}_j > \mathbf{0}$, then job j precedes i

□

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

Proof . When a job is always late, $E_i = 0$; if the job is always early, $T_j = 0$, and $V_j = 0$ (i.e. j, is early) Since the job i, is always tardy then $E_i = 0$ and if job j is always early, then $T_j = 0$, and $V_j = 0$ (i.e. j, is early).
(*) when $V_j = 0$

$$* \quad \mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + (\mathrm{t} + p_i - d_i) + 0 + p_i + (\mathrm{t} + p_i + p_j) - r + 0+ (d_j - \mathrm{t} - p_i - p_j) + 0 + 0] = 2\mathrm{t} + 3p_i - d_i + d_j - 2r \ldots .(\mathrm{a})$$
$$\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r+ (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 2\mathrm{t} + 3p_i + 2p_j + d_j - d_i - 2r \ldots ..(\mathrm{b})$$

$\Delta_{ij} = (\boldsymbol{a}) - (\boldsymbol{b}) = -2\boldsymbol{p}_j < \mathbf{0}_h$ then must job i precede j.
(**) when $\boldsymbol{V}_j = \boldsymbol{C}_j - \boldsymbol{d}_j$

$$*\,\mathcal{F}_{ijij} = [(\mathrm{t} + p_j) - r + 0 + (\mathrm{t} + p_j - d_j) + 0 + p_i + (\mathrm{t} + p_j + p_i) - r + 0 + (d_i - \mathrm{t} - p_j - p_i) + 0 + \mathrm{t} + p_i + p_j - d_j [= 3\mathrm{t} + 4p_i + p_j - d_i- 2r \ldots ..\ (\mathrm{c})$$
$$* \quad \mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) - r + 0 + (\mathrm{t} + p_j - d_j) + (\mathrm{t}- p_j - p_i) - r\,(\mathrm{t} + p_j + p_i - d_i) + 0 + p_i\,[= 3\mathrm{t} + 3p_i - d_i - 2r \ldots \ldots (\mathrm{e})$$

$\Delta_{ij} = (c) - (e) = \boldsymbol{p}_i - 2p_j \leq \mathbf{0}$, then job i precede j □

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

Proof . Since the job j, is always tardy then Ej = 0 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 $V_i = 0$).

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + 0 + (\mathrm{t} + p_i + p_j) - r + (\mathrm{t}+ p_i + p_j - d_j) + 0 + p_j] = 2\mathrm{t} + 2p_i + 3p_j - d_j + d_i - 2r \ldots\ldots \text{ (a).}$$

• $\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + (\mathrm{t} + p_j - d_j) + 0 + p_j + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t}+ p_j + p_i - d_i) + 0 + p_i] = 4\mathrm{t} + 3p_i + 5p_j - d_j - d_i - 2r \ldots \text{ (b).}$

$\Delta_{ij} = (a) - (b) = -2t - p_i - 2p_j + 2d_i \le 0$, then job i precede j.
• (when $V_i = C_i - d_i$).

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + (\mathrm{t} + p_i - d_i) + (\mathrm{t} + p_i + p_j) - r + (\mathrm{t} + p_i + p_j - d_j) + 0 + p_j] = 3\mathrm{t} + 3p_i + 3p_j - d_j - 2r \ldots \text{(c)}$$

• $\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + (\mathrm{t} + p_j - d_j) + 0 + p_j + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t}+ p_j + p_i - d_i) + 0 + p_i] = 4\mathrm{t} + 3p_i + 5p_j - d_j - d_i - 2r \ldots\ldots \text{ (e).}$

$\Delta_{ij} = (c) - (e) = -t - 2p_j + d_i - 1 \le 0$, 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. if they aren't scheduled first, jobs I and j will be late.).

Proof . • (when $V_i = 0$)

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + 0 + (\mathrm{t} + p_i + p_j) - r + (\mathrm{t} + p_i+ p_j - d_j) + 0 + p_j] = 2\mathrm{t} + 2p_i + 3p_j - d_j + d_i - 2r \ldots\ldots \text{(a)}$$

• (when $V_j = 0$)

$$\mathcal{F}'{}_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j+ p_i - d_i) + 0 + p_i] = 2\mathrm{t} + 3p_i + 2p_j + d_j - d_i + 1 - 2r \ldots \text{(b)}$$

$\boldsymbol{\Delta}_{ij} = (\boldsymbol{a}) - (\boldsymbol{b}) = -\boldsymbol{p}_i + \boldsymbol{p}_j + 2\boldsymbol{d}_i - 2\boldsymbol{d}_j > \mathbf{0}$, then job j precede i.
• (when $V_i = C_i - d_i$)

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + (\mathrm{t} + p_i - d_i) + (\mathrm{t} + p_i + p_j) - r+ (\mathrm{t} + p_i + p_j - d_j) + 0 + p_j] = 3\mathrm{t} + 3p_i + 3p_j - d_j - 2r \ldots \text{(c)}$$

• (when $V_j = C_j - d_j$)

$$\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + (\mathrm{t} + p_j - d_j) + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 3\mathrm{t} + 3p_i + 3p_j - d_i - 2r \ldots \text{(e)}$$

$\boldsymbol{\Delta}_{ij} = (\boldsymbol{c}) - (\boldsymbol{e}) = -\boldsymbol{d}_j + \boldsymbol{d}_i \le \mathbf{0}$, then job i precede j.
• (when $V_i = 0$)

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + 0 + (\mathrm{t} + p_i + p_j) - r + (\mathrm{t} + p_i +p_j - d_j) + 0 + p_j] = 2t + 2p_i + 3p_j - d_j + d_i - 2r \ldots\ldots (a')$$

• (when$V_j = C_j - d_j$)

$$\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + (\mathrm{t} + p_j - d_j) + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 3\mathrm{t} + 3p_i + 3p_j - d_i - 2r \ldots\ldots (\mathrm{e}')$$

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

• (when $V_i = C_i - d_i$)

$$\mathcal{F}_{ij} = [(t + p_i) - r + 0 + (d_i - t - p_i) + 0 + (t + p_i - d_i) + (t + p_i + p_j) - r + (t + p_i + p_j - d_j) + 0 + p_j] = 3t + 3p_i + 3p_j - d_j - 2r \ldots (\mathrm{c}')$$

• (when $V_j = 0$)

$$\mathcal{F'}_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 2\mathrm{t} + 3p_i + 2p_j + d_j - d_i - 2r \ldots (\mathrm{b}')$$

$\Delta_{ij} = (c') - (b') = t + p_j + d_i - 2d_j \leq 0$, then job i precede j □

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

Proof . • (when $V_i, V_j = 0$)

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + 0 + (\mathrm{t} + p_i + p_j) - r + 0 + (d_j - \mathrm{t} - p_i - p_j) + 0 + 0] = d_i + d_j - 2r \ldots (\mathrm{a})$$

• (when $V_j = 0$)

$$\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 2\mathrm{t} + 3p_i + 2p_j + d_j - d_i - 2r \ldots (\mathrm{b})$$

$\Delta_{ij} = (\boldsymbol{a}) - (\boldsymbol{b}) = -2\mathrm{t} - 3p_i - 2p_j + 2d_i \leq \mathbf{0}$, then job i precede j.

• (when $V_i = C_i - d_i$, and $V_j = C_j - d_j$)

$$\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + (\mathrm{t} + p_i - d_i) + (\mathrm{t} + p_i + p_j) - r + 0 + (d_j - \mathrm{t} - p_i - p_j) + 0 + \mathrm{t} + p_i + p_j - d_j] = 2\mathrm{t} + 2p_i + p_j - 2r_i(\mathrm{c})_i$$

• (when $V_j = C_j - d_j$)

$$\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + (\mathrm{t} + p_j - d_j) + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d_i) + 0 + p_i] = 3\mathrm{t} + 3p_i + 3p_j - d_i - 2r \ldots\ldots (\mathrm{e})$$

$\Delta_{ij} = (\mathrm{c}) - (\mathrm{e}) = -\boldsymbol{t} - \boldsymbol{p}_i - 2p_j + \boldsymbol{d}_i - \mathbf{1} \leq \mathbf{0}$, then job i precede j □

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

Proof . When $V_i, V_j = 0$

• $\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + 0 + (\mathrm{t} + p_i + p_j) - r + 0 + (d_j - \mathrm{t} - p_i - p_j) + 0 + 0] = d_i + d_j - 2r_i ...(\mathrm{a})$

- $\mathcal{F}'_{ji} = [(\mathrm{t}+p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t}+p_j+p_i) - r + 0 + (d_i - \mathrm{t} - p_j - p_i) + 0 + 0] = d_j + d_i - 2r_i$...(b)

$\Delta_{ij} = (a) - (b) = 0$, then scheduling i or j first is irrelevant.
When $V_i = C_i - d_i, V_j = C_j - d_j$.

- $\mathcal{F}_{ij} = [(\mathrm{t}+p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + (\mathrm{t}+p_i-d_i) + (\mathrm{t}+p_i+p_j) - r + 0 + (d_j - \mathrm{t} - p_i - p_j) + 0 + (\mathrm{t}+p_i+p_j-d_j)] = 2\mathrm{t} + 2p_i + p_j - 2r_{-i-(\mathrm{c})}$

- $\mathcal{F}'_{ji} = [(\mathrm{t}+p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + (\mathrm{t}+p_j-d_j) + (\mathrm{t}+p_j+p_i) - r + 0 + (d_i - \mathrm{t} - p_j - p_i) + 0 + (\mathrm{t}+p_j+p_i-d_i)] = 2\mathrm{t} + p_i + 2p_j - 2r \ldots$ (e)

$\Delta_{ij} = (c) - (e) = p_i - p_j \leq 0$, then job i comes before j.
When $V_i = C_i - d_i, \quad V_j = 0$

- $\mathcal{F}_{ij} = [(\mathrm{t}+p_i) - r + 0 + (d_i - \mathrm{t} - p_i) + 0 + (\mathrm{t}+p_i-d_i) + (\mathrm{t}+p_i+p_j) - r + 0 + (d_j - \mathrm{t} - p_i - p_j) + 0 + 0] = \mathrm{t} + p_i + d_j - 2r \ldots$ (n)

- $\mathcal{F}'_{ji} = [(\mathrm{t}+p_j) - r + 0 + (d_j - \mathrm{t} - p_j) + 0 + 0 + [(\mathrm{t}+p_j+p_i) - r + 0 + (d_i - \mathrm{t} - p_j - p_i) + 0 + (\mathrm{t}+p_j+p_i-d_i) = \mathrm{t} + p_j + p_i + d_j - 2r_{\ldots}$(m)

$\Delta_{ij} = (n) - (m) = -p_j \leq 0$, then job i comes before j □

Second: If $\mathbf{r} > t$, then we demonstrate that the theorem is correct in the same way as we did earlier (integral).

Theorem 5.1. *Jobs I and (j) are neighboring jobs in the problem (P) if $r_i = r$, and $d_i = d$, for all ($i \in \mathrm{N}$), and if $p_i \leq p_j$ and $d_j = d$, then job i should come before job (j) in at least one optimum sequence for the problem (P). Proof. By re-using the same methods as previously, we can demonstrate the validity of this theorem in the following situations:*
First: $\mathbf{r} \leq \mathbf{t}$:
Case 1: *If $d \leq t + p_i \leq t + p_j$(i. e.both the jobs i, j are always tardy).*

Proof . To display that S dominates S', it suffices to display that $(\Delta_{ij} = \mathcal{F}_{ij}(S) - \mathcal{F}_{ji}(S') \leq 0)$. Since the jobs i and j are both tardy, then $E_i = E_j = 0, V_i = p_i, V_j = p_j$, Now, let

- $\mathcal{F}_{ij}(S) = [(\mathrm{t}+p_i) - r + (\mathrm{t}+p_i-d) + 0 + p_i + (\mathrm{t}+p_i+p_j) - r + (\mathrm{t} + p_i + p_j - d) + 0 + p_j] = 4\mathrm{t} + 5p_i + 3p_j - 2d - 2r \ldots$ (a)
- $\mathcal{F}'_{ji} = [(\mathrm{t}+p_j) - r + (\mathrm{t}+p_j-d) + 0 + p_j + (\mathrm{t}+p_j+p_i) - r + (\mathrm{t}+p_j + p_i - d) + 0 + p_i] = 4\mathrm{t} + 3p_i + 5p_j - 2d - 2r \ldots$ (b)

$\boldsymbol{\Delta}_{ij} = (\boldsymbol{a}) - (\boldsymbol{b}) = 2\boldsymbol{p}_i - 2\boldsymbol{p}_j \leq \mathbf{0}$, then job i comes before 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 . Because job I is always late, $E_i = 0$; similarly, if job j is always on time, $T\ _j = 0$, and $V_j = \{0, C_j - d_j\}$ (i.e. j, is early or partial early).
(when $V_j = 0$).

- $\mathcal{F}_{ij} = [(\mathrm{t}+p_i) - r + (\mathrm{t}+p_i-d) + 0 + p_i + (\mathrm{t}+p_i+p_j) - r + 0 + (d - \mathrm{t} - p_i - p_j) + 0 + 0] = 2\mathrm{t} + 3p_i - 2r \ldots (a)$

- $\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d) + 0 + p_i] = 2t + 3p_i + 2p_j - 2r \dots (b)$

$\Delta_{ij} = (a) - (b) = -2p_j < 0$. then job i precede j.
(when $V_j = C_j - d$)=

- $\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + (\mathrm{t} + p_i - d) + 0 + p_i + (\mathrm{t} + p_i + p_j) + 0 + (d - \mathrm{t} - p_i - p_j) - r + 0 + \mathrm{t} + p_i + p_j - d] = 3\mathrm{t} + 4p_i + p_j - d - 2r \dots (c)$

- $\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d - \mathrm{t} - p_j) + 0 + (\mathrm{t} + p_j - d) + (\mathrm{t} + p_j + p_i) - r + (\mathrm{t} + p_j + p_i - d) + 0 + p_i] = 3\mathrm{t} + 3p_i + 3p_j - d - 2r \dots (e)$

$\Delta_{ij} = (\boldsymbol{c}) - (\boldsymbol{e}) = \boldsymbol{p}_i - 2\boldsymbol{p}_j \leq \mathbf{0}$, then job i precede j. □

Case 3. If $\boldsymbol{t} + \boldsymbol{p}_i + \boldsymbol{p}_j \leq \boldsymbol{d}(\boldsymbol{i} \cdot \boldsymbol{e}.\boldsymbol{both}$ the jobs $\boldsymbol{i}, \boldsymbol{j}$ are always early).

Proof . When $V_i, V_j = 0$

- $\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d - \mathrm{t} - p_i) + 0 + 0 + (\mathrm{t} + p_i + p_j) - r + 0 + (d - \mathrm{t} - p_i - p_j) + 0 + 0] = 2d - 2r \dots (\mathrm{a})$

- $\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d - \mathrm{t} - p_j) + 0 + 0 + (\mathrm{t} + p_j + p_i) - r + 0 + (d - \mathrm{t} - p_j - p_i) + 0 + 0] = 2d - 2r \dots (\mathrm{b})$

$\Delta_{ij} = (a) - (b) = 0$, then scheduling i or j first is irrelevant.

When $V_i = C_i - d, V_j = C_j - d$,

- $\mathcal{F}_{ij} = [(\mathrm{t} + p_i) - r + 0 + (d - \mathrm{t} - p_i) + 0 + (\mathrm{t} + p_i - d) + (\mathrm{t} + p_i + p_j) - r + 0 + (d - \mathrm{t} - p_i - p_j) + 0 + (\mathrm{t} + p_i + p_j - d)] = 2\mathrm{t} + 2p_i + p_j - 2r_{\mathrm{i}} \dots (c)$.

- $\mathcal{F}'_{ji} = [(\mathrm{t} + p_j) - r + 0 + (d - \mathrm{t} - p_j) + 0 + (\mathrm{t} + p_j - d) + (\mathrm{t} + p_j + p_i) - r + 0 + (d - \mathrm{t} - p_j - p_i) + 0 + (\mathrm{t} + p_j + p_i - d)] = 2\mathrm{t} + p_i + 2p_j - 2r \dots (\mathrm{e})$

$\Delta_{ij} = (c) - (e) = p_i - p_j \leq 0$, then job i comes before j.

Second: If $r > t$, then we demonstrate that the theorem is correct in the same way as we did earlier (integral). □

6. Conclusions and suggestions for future works

The researchers, in the current investigation, provided certain special cases (CS) for the problem, as well as some dominoes' rules for it (p). In further investigations, we recommend that we use certain weights with each function of the presented issue and that we investigate the CS's for each of those functions. In addition, we must investigate the solution of issue (p), which is NP-hard, by employing certain precise methods, such as the BAB method, as well as some heuristic and metaheuristic approaches.

References

[1] T.S. Abdul Razaq, S.K. Al Saidy and M.K. Al Zuwaini, *Single machine scheduling to minimize total weighted late work with release date*, J. Al-Qadisyah Pure Sci. 13(4) (2008) 91–112.

[2] R.H. Ahmadi and B. Uttarayan, *Lower bounds for single-machine scheduling problems*, Naval Res. Logistics 37(6) (1990) 967–979.

[3] S.P. Ali and M. Bijari, *Minimizing maximum earliness and tardiness on a single machine using a novel heuristic approach*, Proc. Int. Conf. Indust. Engin. Operat. Manag. (2012).

[4] M.K. Al-Zuwaini and M.K. Mohanned, *One machine scheduling problem with release dates and tow criteria*, J. Thi-Qar Univ. 6(2) (2011) 1–14.

[5] J. Błażewicz, *Scheduling preemptible tasks on parallel processors with information loss*, Rech. Tech. Sci. Inf. 3 (1984) 415–420.

[6] C. Briand and O. Samia, *Minimizing the number of tardy jobs for the single machine scheduling problem: MIP-based lower and upper bounds*, RAIRO-Operat. Res. 47(1) (2013) 33–46.

[7] R. Chandra, *On n/1/F dynamic deterministic problems*, Naval Res. Logistics Quart. 26(3) (1979) 537–544

[8] W.Y. Chen and G.N. Sheen, *Single-machine scheduling with multiple performance measures: Minimizing job-dependent earliness and tardiness subject to the number of tardy jobs*, Int. J. Production Econom. 109(1-2) (2007) 214–229.

[9] C. Chu, *A branch-and-bound algorithm to minimize total flow time with unequal release dates*, Naval Res. Logistics 39(6) (1992) 859–875.

[10] P.S. Dauzère, *Minimizing late jobs in the general one machine scheduling problem*, European J. Operat. Res. 81(1) (1995) 134–142.

[11] M.I. Dessouky and S.D. Jitender, *Sequencing jobs with unequal ready times to minimize mean flow time*, SIAM J. Comput. 10(1) (1981) 192–202.

[12] R.L. Graham, E.L. Lawler, J.K. Lenstra and A.R. Kan, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Ann. Discrete Math. 5 (1979) 287–326.

[13] R.B. Kethley and A. Bahram, *Single machine scheduling to minimize total weighted late work: a comparison of scheduling rules and search algorithms*, Comput. Indust. Engin. 43(3) (2002) 509–528.

[14] J.K. Lenstra, A.R.Kan and P. Brucker, *Complexity of machine scheduling problems*, Ann. Discrete Math. 1 (1977) 343–362.

[15] M. Mahnam and M. Ghasem, *A branch-and-bound algorithm for minimizing the sum of maximum earliness and tardiness with unequal release times*, Engin. Optim. 41(6) (2009) 521–536.

[16] J.M. Moore, *One machine sequencing algorithm for minimizing the number of late jobs*, Manag. Sci. 15(1) (1968) 102–109.

[17] C.N. Potts and L.N. Van Wassenhove, *Approximation algorithms for scheduling a single machine to minimize total late work*, Operat. Rese. Lett. 11(5) (1992) 261–266.

[18] C.N. Potts and L.N. Van Wassenhove, *Single machine scheduling to minimize total late work*, Operat. Res. 40(3) (1991) 586–595.

[19] A.H.G. Rinnooy Kan, *Machine Sequencing Problem: Classification, complexity and computations*, 1976.

[20] L. Schrage, *Letter to the editor a proof of the optimality of the shortest remaining processing time discipline*, Operat. Res. 16(3) (1968) 687–690.

[21] J.B. Sidney, *Optimal single-machine scheduling with earliness and tardiness penalties*, Operat. Res. 25(1) (1977) 62–69.