

2016


Heuristic and Exact Algorithms for the Two-Machine Just in Time Job Shop Scheduling Problem

Mohammed Al Salem

Leonardo Bedoya-Valencia

Ghaith Rabadi
Old Dominion University

Follow this and additional works at: https://digitalcommons.odu.edu/emse_fac_pubs

 Part of the [Dynamic Systems Commons](#), and the [Operations Research, Systems Engineering and Industrial Engineering Commons](#)

Repository Citation

Salem, Mohammed Al; Bedoya-Valencia, Leonardo; and Rabadi, Ghaith, "Heuristic and Exact Algorithms for the Two-Machine Just in Time Job Shop Scheduling Problem" (2016). *Engineering Management & Systems Engineering Faculty Publications*. 6.
https://digitalcommons.odu.edu/emse_fac_pubs/6

Original Publication Citation

Al-Salem, M., Bedoya-Valencia, L., & Rabadi, G. (2016). Heuristic and exact algorithms for the two-machine Just in Time job shop scheduling problem. *Mathematical Problems in Engineering*, 11. doi: 10.1155/2016/6591632

Research Article

Heuristic and Exact Algorithms for the Two-Machine Just in Time Job Shop Scheduling Problem

Mohammed Al-Salem,¹ Leonardo Bedoya-Valencia,² and Ghaith Rabadi³

¹Qatar University, Doha, Qatar

²Colorado State University-Pueblo, Pueblo, CO, USA

³Old Dominion University, Norfolk, VA, USA

Correspondence should be addressed to Mohammed Al-Salem; alsalem@qu.edu.qa

Received 23 April 2016; Revised 26 September 2016; Accepted 18 October 2016

Academic Editor: Thomas Hanne

Copyright © 2016 Mohammed Al-Salem et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The problem addressed in this paper is the two-machine job shop scheduling problem when the objective is to minimize the total earliness and tardiness from a common due date (CDD) for a set of jobs when their weights equal 1 (unweighted problem). This objective became very significant after the introduction of the Just in Time manufacturing approach. A procedure to determine whether the CDD is restricted or unrestricted is developed and a semirestricted CDD is defined. Algorithms are introduced to find the optimal solution when the CDD is unrestricted and semirestricted. When the CDD is restricted, which is a much harder problem, a heuristic algorithm is proposed to find approximate solutions. Through computational experiments, the heuristic algorithms' performance is evaluated with problems up to 500 jobs.

1. Introduction and Literature Review

Many objectives in scheduling are linked to due dates, which focus on meeting costumers' delivery dates. When a CDD is considered, a set of components must be assembled into a finished product, for example, or several jobs must be shipped together to a certain customer. In Just in Time (JIT) manufacturing and production systems, it is intended to reduce both earliness and tardiness by constructing a job schedule in which the jobs are finished as close as possible to their due date. Minimizing tardiness reduces the cost of missing due dates and increases customers' satisfaction, while minimizing earliness reduces the holding and inventory cost.

Most of the published literature on minimizing earliness/tardiness (E/T) addressed the single machine E/T problem (e.g., Kanet [1], Sundararaghavan and Ahmed [2], Szwarc [3], Bagchi et al. [4], Hall and Posner [5], Hall et al. [6], Hariri and Potts [7], Rabadi et al. [8], Mason et al. [9], Rabadi et al. [10], Atan and Selim Akturk [11], Hepdogan et al. [12], Baptiste et al. [13], and Cheng et al. [14]). Baker and Scudder [15] published a comprehensive state-of-the-art

review for different variants of the E/T problem including the problem with a CDD. Also, many papers addressed the same objective in multimachine scheduling environment such as parallel machines (e.g., Emmons [16], Cheng and Chen [17], Alvarez-Valdes et al. [18], Kayvanfar et al. [19], Li et al. [20], Kubiak et al. [21]) and flow shops (e.g., Sarper [22], Sung and Min [23], Mosheiov [24], Gupta et al. [25], Lauff and Werner [26], Chandra et al. [27], Behnamian et al. [28], and M'Hallah [29]). Much less has been published on scheduling problems for job shops (e.g., Lauff and Werner [30], Baptiste et al. [31], Yang et al. [32], and Wang and Li [33]). Gordon et al. [34] have reviewed more recent literature of the E/T problem with CDD. Lauff and Werner [30, 35] reviewed multistage systems involving earliness and tardiness problems with CDD as well.

Kanet [1] developed an algorithm to find an optimal solution for the single machine E/T problem with an unrestricted CDD. In his work, some properties of optimal solutions were proved and used to construct a polynomial algorithm. Sundararaghavan and Ahmed [2] developed a heuristic algorithm for the same problem but with a restricted CDD (i.e., when the CDD is small enough to constrain the

schedule) where they used some of the properties defined for the unrestricted case by Kanet. Bagchi et al. [4] extended Kanet's work and developed an exact algorithm to generate alternate optimal solutions. They also developed an implicit enumeration procedure for the restricted case in a single machine environment. Raghavachari [36] extended the V-shape property of optimal schedules established by Kanet [1] to any CDD.

Hoogeveen and van de Velde [37] developed a dynamic programming (DP) algorithm to solve the single machine scheduling problem with a small CDD and a positive weight for each job. They found out that the problem with equal processing times for all jobs and the problem with equal weight to processing time rates are polynomially solvable. Also, Hall and Posner [5] developed a DP algorithm for the single machine problem considering an unrestricted CDD and different weights for the jobs. Hall et al. [6] constructed an exact algorithm based on DP to find optimal solutions for the unweighted version of the single machine E/T problem with restricted CDD. Rabadi et al. [8] developed a branch-and-bound procedure to find optimal solutions for single machine problems with an unrestricted CDD and considering sequence-dependent setup times.

In multimachine environments, Emmons [16] developed an algorithm that is able to solve the identical parallel machine scheduling problem when all jobs have a CDD and earliness and tardiness have different cost rates. Cheng and Chen [17] studied the problem of assigning a common due date and sequencing a set of simultaneously available jobs on several identical parallel machines.

Sarper [22] developed a mixed integer linear programming formulation for the two-machine Flow Shop Scheduling Problems (FSSP) with unweighted earliness and tardiness cost over a CDD. Sung and Min [23] studied the two-machine FSSP with batch processing machines and a CDD. Mosheiov [24] studied the unit processing time on an m -machine flow shop E/T problem over nonrestricted and restricted CDD. Gupta et al. [25] defined some properties of optimal schedules for the two-machine E/T FSSP, developed lower and upper bounds, derived dominance criteria, and proposed an enumerative algorithm for finding an optimal schedule. Finally, based on some structural properties of the problem, Lauff and Werner [26] developed heuristic algorithms, both constructive and enumerative, to solve the two-machine FSSP with a given CDD considering asymmetric linear and quadratic penalty functions. Wang and Li [33] presented hybrid heuristic that combines variable neighborhood search with mathematical programming to minimize the sum of earliness and tardiness for the job shop scheduling problem with multiple due dates. Baptiste et al. [31] defined an integer programming model for the JIT/JSSP problem and proposed methods based on two Lagrangian relaxations of the model to derive lower and upper bounds. Yang et al. [32] introduced an enhanced genetic algorithm to solve the job shop scheduling problem of minimizing the weighted tardiness and earliness of jobs in the presence of due dates and deadlines. So far, there is no reported research on the job shop scheduling problem (JSSP) considering E/T over a CDD, and therefore, the two-machine JSSP is addressed in this paper.

In the next section, a formulation of the problem with its three different cases is given. In section three, optimality conditions are introduced for two out of the three cases. Section 4 presents a dynamic programming algorithm to optimally solve the semirestricted case and a heuristic algorithm to solve the restricted case. Then, in Section 5, computational experiments for the three cases are analyzed. Finally, Section 6 discusses the results and Section 7 presents the conclusions and further research.

2. Formulation of the Two-Machine Job Shop Scheduling Problem with a Common Due Date

This article addresses a nonpreemptive, no recirculation (jobs do not revisit the same machine) E/T JSSP with two machines, n jobs, and an integer CDD when all jobs are available at time $t = 0$. Although JIT entails more detail and concepts, the E/T problem seems to mathematically capture the scheduling essence of it.

Let C_j , E_j , and T_j represent the completion time, earliness, and tardiness of job j , respectively; E_j and T_j can be defined as

$$\begin{aligned} E_j &= \max(0, \text{CDD} - C_j) = |\text{CDD} - C_j|, \\ T_j &= \max(0, C_j - \text{CDD}) = |C_j - \text{CDD}|. \end{aligned} \quad (1)$$

Associated with each job there is an earliness penalty $\alpha_j > 0$ and a tardiness penalty $\beta_j > 0$ per time unit, which in this research are all equal to 1 (unweighted problem). The basic earliness and tardiness (E/T) objective function for a schedule S can be written as $f(S)$ as follows:

$$\min f(S) = \sum_{j=1}^n (\alpha_j E_j + \beta_j T_j). \quad (2)$$

The CDD is restricted when it is small enough to restrict the scheduling decision and hence impacts the optimal sequence. The restricted version of the problem is much harder than the unrestricted version [30]. For the single machine E/T problem over a CDD, it would be desirable to construct a schedule in which half of the jobs are before the CDD [37]. If the CDD were too tight, then not enough jobs would be scheduled before the CDD, because they cannot start before time zero. For the single machine problem, the unrestricted case can be solved by using a polynomial algorithm [1]. In the JSSP, if the CDD is large enough, the problem can be solved in polynomial time by solving m ($m = 2$ in this paper) single machine problems [30].

Formally, there is a quantitative procedure to decide if a CDD is restricted or unrestricted for the single machine problem. In this paper, such procedure will be extended to the E/T JSSP over a CDD with two machines.

Initially, for the E/T single machine problem, Kanet [1] assumed that $\text{CDD} \geq \sum_{j=1}^n p_j$, where p_j is the processing time for job j so that the problem can be solved optimally by using his algorithm SCHED. Later, Bagchi et al. [4] showed

that Kanet's algorithm is able to reach optimal solutions under the weaker assumption that the CDD $\geq \Delta$, where

$$\Delta = \begin{cases} p_1 + p_3 + \cdots + p_n, & \text{if } n \text{ is odd,} \\ p_2 + p_4 + \cdots + p_n, & \text{if } n \text{ is even,} \end{cases} \quad (3)$$

$$p_1 \leq p_3 \leq \cdots \leq p_n.$$

For a CDD in a two-machine JSSP to be unrestricted, two conditions must hold. First, the remaining time to process the last operations on each machine must be enough to apply the SCHED algorithm [1] as if each machine is an unrestricted single machine problem. Second, the completion time of each job's first operation on its corresponding machine has to be less than or equal to the starting time of its subsequent last operation. This starting time is given by the SCHED algorithm. The second condition is equivalent to finding a schedule for each single machine problem where no jobs are tardy.

In order to minimize the deviation over the CDD for all the jobs, the first operations of each job should have priority on each machine in order to allow the subsequent operations to be processed. The set of first operations and the set containing the last operations to be scheduled on each machine before the CDD compete for the time available within the interval from $t = 0$ to $t = \text{CDD}$. If the CDD is loose enough, say if $\text{CDD} \geq \text{PT}$, where

$$\text{PT} = \sum_{i=1}^2 \sum_{j=1}^n p_{ij}, \quad (4)$$

where p_{ij} is the processing time for the operation of job j to be performed on machine i with $i = 1, 2$, then, the SCHED algorithm can be applied to the two machines and an optimal solution will be obtained. The closer the CDD to $t = 0$ the tighter (i.e., more restricted) the schedule. This fact can be used to define whether the CDD is restricted or not. Let

M_1 = set with jobs to be finished on machine 1,
 M_2 = set with jobs to be finished on machine 2,
 $|M_1| = n_1$ = number of jobs to be finished on machine 1,
 $|M_2| = n_2$ = number of jobs to be finished on machine 2.
 Also, let

$$\Delta_1 = \begin{cases} p_{11} + p_{13} + \cdots + p_{1n_1}, & \text{if } n_1 \text{ is odd,} \\ p_{12} + p_{14} + \cdots + p_{1n_1}, & \text{if } n_1 \text{ is even,} \end{cases} \quad (5)$$

$$\Delta_2 = \begin{cases} p_{21} + p_{23} + \cdots + p_{2n_2}, & \text{if } n_2 \text{ is odd,} \\ p_{22} + p_{24} + \cdots + p_{2n_2}, & \text{if } n_2 \text{ is even,} \end{cases}$$

where

$$p_{11} \leq p_{12} \leq \cdots \leq p_{1n_1}, \quad (6)$$

$$p_{21} \leq p_{22} \leq \cdots \leq p_{2n_2}.$$

Finally, let

$$F_1 = \sum_{j \in M_1^C} p_{1j},$$

$$F_2 = \sum_{j \in M_2^C} p_{2j}, \quad (7)$$

where M_1^C is the complement of M_1 and M_2^C is the complement of M_2 .

Definition 1. A CDD is unrestricted if $\text{CDD} \geq \max\{F_1 + \Delta_1, F_2 + \Delta_2\}$ and the number of tardy jobs in sets M_1^C and M_2^C are equal to zero.

Discussion. If $\max\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1$, let $B_1 = \{i = 1, 1 \leq j \leq n, j \in M_1 \mid C_{1j} \leq \text{CDD}\}$, where C_{1j} is the completion time of the jobs with their last operation on machine 1 and to be completed before or on CDD. Similarly, let $A_1 = \{i = 1, 1 \leq j \leq n, j \in M_1 \mid C_{1j} > \text{CDD}\}$, where C_{1j} is the completion time of the jobs with their last operation on machine 1 and to be completed after CDD and Δ_1 is the summation of processing times of the jobs in B_1 . The optimal schedule for the machine 1 can be obtained by applying Kanet's SCHED algorithm to the jobs in sets A_1 and B_1 . The starting times of the jobs in sets A_1 and B_1 provide the due dates for their first operations to be processed on machine 2 (i.e., jobs in set M_2^C). The jobs whose first operation must be performed on machine 1, jobs in M_1^C , are processed before the jobs in B_1 without interfering with the optimal schedule since $\text{CDD} \geq F_1 + \Delta_1$.

In order to find out if the number of tardy jobs in M_1^C is equal to zero, the earliest due date (EDD) rule needs to be applied to jobs in M_1^C . If an EDD sequence yields either zero or one tardy job, then it minimizes the number of tardy jobs [38].

Let $B_2 = \{i = 2, 1 \leq j \leq n, j \in M_2 \mid C_{2j} \leq \text{CDD}\}$, where C_{2j} is the completion time of the jobs with their last operation on machine 2 to be completed before or on the CDD and Δ_2 is the summation of processing times of the jobs in B_2 . Since $F_2 + \Delta_2 \leq F_1 + \Delta_1$, jobs to be finished on machine 2 can be optimally scheduled by applying Kanet's SCHED algorithm to the jobs in sets A_2 and B_2 , where A_2 is defined similarly to A_1 . In the same way, the starting times of the jobs in sets A_2 and B_2 provide the due dates for their first operations to be processed on machine 1 (i.e., jobs in set M_1^C). The jobs whose first operation must be performed on machine 2, jobs in M_2^C , are performed before the jobs in B_2 without interfering with the optimal schedule. Also, it is possible to find whether the number of tardy jobs in M_2^C is equal to zero by applying the EDD rule to jobs in M_2^C .

The same reasoning can be applied if $\max\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2$. Tables 1 and 2 show the processing times and the operation-machine assignment for a seven-job example. Figure 1 illustrates the optimal solution for this example when the CDD is unrestricted and equal to 30. In this case $\max\{F_1 + \Delta_1, F_2 + \Delta_2\} = \max\{8 + 13, 20 + 10\} = F_2 + \Delta_2 = 30$. Note that if the CDD > 30 , the problem is still unrestricted.

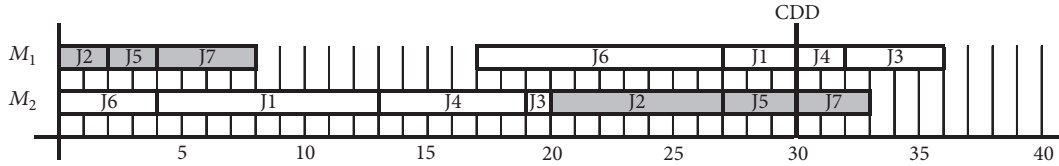


FIGURE 1: Example optimal schedule for the unrestricted version.

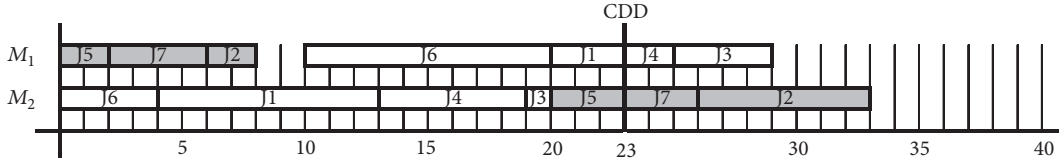


FIGURE 2: Example optimal schedule for the semirestricted version.

TABLE 1: Processing times.

Job	Processing time for first operation	Processing time for second operation
1	9	3
2	2	7
3	1	4
4	6	2
5	2	3
6	4	10
7	4	3

TABLE 2: Operation-machine assignment.

Job	First operation at machine	Second operation at machine
1	2	1
2	1	2
3	2	1
4	2	1
5	1	2
6	2	1
7	1	2

Definition 2. A CDD is semirestricted if $CDD \geq \min \{F_1 + \Delta_1, F_2 + \Delta_2\}$ and $CDD < \max \{F_1 + \Delta_1, F_2 + \Delta_2\}$ and the number of tardy jobs in the sets M_1^C and M_2^C are equal to zero.

Discussion. If $\max \{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2$, then $\min \{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1$. B_1 and B_2 are as in Definition 1. Hence, Δ_1 and Δ_2 are the summation of processing times of the jobs in B_1 and B_2 , respectively. Given $CDD \geq F_1 + \Delta_1$, in an optimal schedule jobs in M_1^C are performed before the jobs in B_1 without interference with the jobs in B_1 . The optimal schedule for machine 1 can be obtained by applying Kanet’s SCHED algorithm to the jobs in B_1 and A_1 . The starting times of the jobs in sets A_1 and B_1 provide the due dates for their first operations to be processed on machine 2 (i.e., jobs in the set M_2^C), which are processed before the jobs in B_2 . But given that $CDD < F_2 + \Delta_2$, jobs in M_2 cannot be

optimally scheduled by using the SCHED algorithm; instead this problem needs to be treated as a single machine problem with a restricted CDD, which can be optimally solved by using the dynamic programming (DP) procedures proposed by Hall et al. [6]. The starting times of the jobs in M_2 given by the optimal solution provide the due dates for their first operations to be processed on machine 1 (i.e., jobs in the set M_1^C). The jobs whose first operation must be performed on machine 1, jobs in M_1^C , are processed before the jobs in B_1 without interfering with the optimal schedule since $CDD \geq F_1 + \Delta_1$. Similar to the unrestricted case, jobs in M_1^C and M_2^C need to be sequenced by using the EDD rule to check if the number of tardy jobs is equal to zero. If both sequences yield zero tardy jobs then the problem is semirestricted. Following the same numeric example, Figure 2 describes the case when the CDD is equal to 23 which is greater than $\min \{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1 = 21$ and is less than $\max \{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2 = 30$.

In this sense, a CDD is semirestricted when the optimal schedule of either one of the machines can be obtained by using Kanet’s SCHED algorithm. The optimal schedule for the other machine has different features, and so the SCHED algorithm will not be able to find it. Instead, the DP procedures developed by Hall et al. [6] need to be used to find the optimal schedule. This procedure is extended to the problem studied here in the next section.

Definition 3. A CDD is restricted if neither Definition 1’s conditions nor Definition 2’s conditions hold.

Discussion. If $CDD < \min \{F_1 + \Delta_1, F_2 + \Delta_2\}$, let $\min \{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1$, and $B_1 = \{i = 1, 1 \leq j \leq n, j \in M_1 \mid C_{1j} \leq CDD\}$, where C_{1j} is the completion time of the jobs with their last operation on machine 1. Hence, Δ_1 is the summation of processing times of the jobs in B_1 . Since $CDD < F_1 + \Delta_1$, there is no way to optimally schedule jobs in M_1 without modifying the starting times of the jobs in M_1^C (the first operations of jobs in M_2). Hence, a tradeoff between jobs in M_1^C and jobs in B_1 must be made. The jobs in M_1^C (those with their first operation to be performed on machine 1) interfere with the optimal schedule on this machine. On the other hand, if at

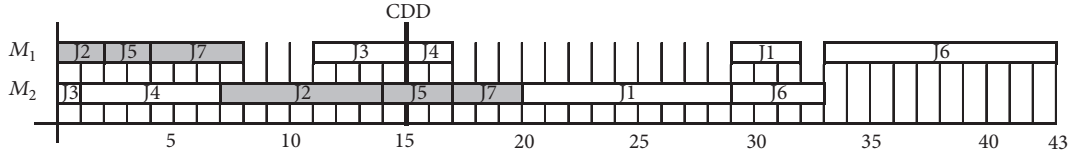


FIGURE 3: Example final scheduled for the restricted version.

least one of the jobs in M_1^C is delayed, this delay interferes with the optimal schedule on machine 2. The same reasoning can be applied if $\min\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_2 + \Delta_2$. Following the same numeric example, Figure 3 illustrates the case when the $CDD = 15$ which is less than $\min\{F_1 + \Delta_1, F_2 + \Delta_2\} = F_1 + \Delta_1 = 21$.

Additionally, if there is at least one tardy job in the cases given in Definitions 1 or 2, then the CDD is also considered restricted since precedence constraints for at least one job (i.e., the tardy job) do not hold.

Next, optimality conditions for the unrestricted and semirestricted case are presented. Also, two properties of the optimal solution for the restricted case are proved and used to construct approximate solutions.

3. Optimality Conditions

Optimal solutions for the two-machine E/T JSSP with restricted CDD are difficult to characterize. In this paper, optimal solutions will be defined when the CDD is unrestricted and semirestricted. Approximate solutions, obtained by a heuristic procedure, will be defined when the CDD is restricted.

3.1. Unrestricted CDD. If the CDD is unrestricted as described in Definition 1, the optimal solution can be found by using Kanet's SCHED procedure on each machine. The properties of the optimal schedule as defined in Kanet [1] will be extended for our use.

Property 1. There is no idle time between jobs in sets M_1 and M_2 .

Property 2. The jobs in both B_1 and B_2 are sequenced by longest processing time first (LPT).

Property 3. The last jobs in B_1 and B_2 are completed at time $t = CDD$.

Property 4. Let A_1 and A_2 represent an ordered set of jobs pertaining, respectively, to M_1 and M_2 to be scheduled without inserted idle time such that the first job in both A_1 and A_2 starts at time $t = CDD$. In an optimal schedule, jobs in both A_1 and A_2 are sequenced by shortest processing time (SPT) first.

Property 5. If n_1 is even then $|B_1| = |A_1|$. If n_1 is odd then $|B_1| = |A_1| + 1$. If n_2 is even then $|B_2| = |A_2|$. If n_2 is odd then $|B_2| = |A_2| + 1$.

Property 6. There is a one-to-one mapping of the jobs in both A_1 and A_2 onto the jobs in B_1 and B_2 such that $k_1 \in A_1$ and $k_2 \in A_2$ and $j_1 \in B_1$ and $j_2 \in B_2 \Rightarrow p_{1k_1} \leq p_{1j_1}$ and $p_{2k_2} \leq p_{2j_2}$.

The proofs of these properties and the proof that SCHED yields optimal solutions can be easily extended from Kanet [1] and Definition 1. It is important to note that SCHED runs in polynomial time as discussed by Kanet [1]. Figure 1 shows the optimal schedule of the same numerical example when the CDD is unrestricted.

3.2. Semirestricted CDD. If the CDD is semirestricted as described in Definition 2, the optimal solution can be found by using Kanet's SCHED procedure on the machine with $\min\{F_1 + \Delta_1, F_2 + \Delta_2\}$. The optimal solution in this machine preserves the properties given for the unrestricted case. For the other machine, some properties must be defined in order to characterize the optimal solution. The properties of the optimal schedule as defined by Hall et al. [6] will be extended for our use.

Let t_1^* or t_2^* denote the starting times in an optimal schedule of the first job processed on either M_1 or M_2 , respectively, corresponding to the machine where $\max\{F_1 + \Delta_1, F_2 + \Delta_2\}$ holds.

Also, define $E_{1(2)} = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} < CDD\}$, $E'_{1(2)} = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} \leq CDD\}$, $T_{1(2)} = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} - p_{1(2)j} \geq CDD\}$, and $T'_{1(2)} = \{i = 1(2), 1 \leq j \leq n \mid C_{1(2)j} > CDD\}$.

Property 1. There exists at least one of the following:

An optimal schedule with either $t_1^* = F_1$ or $t_2^* = F_2$.

An optimal schedule with $C_{1(2)a} = CDD$, where a is a job with its last operation on machine 1(2) starting before CDD and completing at CDD or later.

Property 2. In an optimal schedule, the jobs in $E'_{1(2)}$ are in LPT order, and the jobs in $T_{1(2)}$ are in SPT order.

Property 3. Each optimal schedule is weakly V-shaped. A weakly V-shaped schedule means a job does not necessarily end at the CDD.

Definition 4. On machine 1(2) a schedule is early V-Shaped (EVS) if $p_{1(2)a} \leq p_{1(2)tmin}$.

Definition 5. On machine 1(2) a schedule is tardy V-Shaped (TVS) if $p_{1(2)a} \leq p_{1(2)emin}$, where $emin = \min\{p_{1(2)j} \in E'_{1(2)}\}$, and $tmin = \min\{p_{1(2)j} \in T_{1(2)}\}$.

Property 4. If $t_1^* = F_1$, then $|E_1| \geq |T_1| - 1$ or if $t_2^* = F_2$, then $|E_2| \geq |T_2| - 1$.

Property 5. Consider $|E_{1(2)}| \leq |T_{1(2)}| + 1$.

Property 6. If $C_{1(2)a} = \text{CDD}$, then $\sum_{j \in E'_{1(2)}} p_{1(2)j} \leq \sum_{j \in T_{1(2)}} p_{1(2)j} + 2p_{1(2)s}$, where job s is the first job scheduled on machine 1(2).

The proofs of these properties can be extended from Hall et al. [6] and Definition 2.

Based on these properties, the DP procedure can be introduced to find the optimal solution examining all EVS and TVS schedules if $t_{1(2)}^* = F_{1(2)}$ by using EVS and TVS procedures described by Hall et al. [6]. If $t_{1(2)}^* > F_{1(2)}$, then it is possible to assume that $C_{1(2)a} = \text{CDD}$ and the *Nosplit* procedure described by Hall and Posner [5] can be extended to find an optimal schedule in which a job is completed at CDD. By jointly using these procedures, an optimal solution can be found for the two-machine JSSP with a semirestricted CDD. More detail on these procedures will be presented in the next section. It is important to note that, in the general case, the dynamic programming approach proposed by Hall et al. [6] for the single machine restricted case runs in pseudopolynomial time. Figure 2 shows the optimal schedule of the same previous numerical example when the CDD is semirestricted.

3.3. Restricted CDD. To characterize the optimal solution when the CDD is restricted is difficult. Hence, two properties are defined in order to develop a heuristic algorithm to obtain approximate solutions for the two-machine JSSP.

Define $I_1 =$ the set of jobs to be finished on machine 1 and to be scheduled around the restricted CDD. $I_2 =$ the set of jobs to be finished on machine 2 and to be scheduled around the restricted CDD.

Clearly, $I_1 \subseteq M_1$ and $I_2 \subseteq M_2$.

Property 1. Jobs in I_1 and I_2 are scheduled without idle time.

Proof. By contradiction and similar to the approach by Baker [38], assume that there exists an optimal schedule S with an idle interval of length t between consecutive jobs a and b , with b following a ; $a, b \in I_1 (I_2)$, and the predecessors of a and b are already scheduled at machine 2(1). If job a is early ($C_{1(2)a} < \text{CDD}$), then the total penalty cost can be reduced by shifting job a (and any jobs that precedes it) later by an amount Δt , where $\Delta t \leq \min(t, \text{CDD} - C_{1(2)a})$ without affecting the feasibility of S . Denoting the values after the shift with primes, it follows that $T'_{1(2)k} = T_{1(2)k}$ and $T'_{1(2)k} \leq T_{1(2)k}$ strictly for at least one job. Similarly, if job b is tardy ($C_{1(2)b} > \text{CDD}$), then the total penalty cost can be reduced by shifting job b (and any jobs that follows it) earlier by an amount Δt , where $\Delta t \leq \min(t, C_{1(2)b} - \text{CDD})$ without affecting the feasibility of S . Hence, it follows that $E'_{1(2)k} = E_{1(2)k}$ and $T'_{1(2)k} \leq T_{1(2)k}$ strictly for at least one job. Since any schedule must have either job a early or job b tardy, then schedule S can be improved, and therefore, it cannot be optimal. \square

Property 2. The optimal schedule for the jobs in I_1 and I_2 is weakly V-shaped, where a schedule S is weakly V-shaped if all jobs completed before the CDD are ordered according to LPT and all jobs that begin their processing after the CDD ordered according to SPT.

Proof. By contradiction and similar to the approach in Baker [38], assume S denotes an optimal schedule in which some adjacent pair of early jobs in $I_{1(2)}$ is not in LPT order. Then a pairwise interchange of these two jobs will reduce the total earliness penalty and leave the tardiness penalty unchanged on machine 1(2) without affecting the feasibility of S . Similarly, if S is an optimal schedule containing an adjacent pair of jobs that starts late in $I_{1(2)}$ and that violates the SPT order, then an adjacent pairwise interchange will reduce the total tardiness penalty and leave the total earliness penalty unchanged on machine 1(2). In either case, S cannot be an optimal schedule. \square

Once the jobs to be included in I_1 and I_2 have been defined, there are still two questions to be answered. First, in an optimal schedule, is there some job that must be completed exactly at $t = \text{CDD}$? It was shown by Hall et al. [6] for the single machine case with a restricted CDD that this is not necessarily true. Second, in an optimal schedule, which jobs are early and which ones are tardy? Figure 3 shows an approximate schedule of the numerical example when the CDD is restricted.

In order to find an approximate solution for the restricted version of the problem addressed in this paper, *Restricted Heuristic*, a heuristic procedure based on Properties 1 and 2 is proposed. By iteratively eliminating the job with the longest processing time in either machine 1 or machine 2, sets I_1 and I_2 are defined. This procedure tries to reduce a restricted problem to the semirestricted version in order to apply the SCHED, EVS, TVS, and Nosplit procedures. Given the jobs removed are going to be tardy anyway, the SPT rule is applied in order to minimize their tardiness. Each time a remaining tardy job is scheduled, an improvement procedure tries to look for early slots of time in the current schedule in order to decrease the tardiness cost. *Restricted Heuristic* reduces the problem to a semirestricted one in order to apply the dynamic programming approach proposed by Hall et al. [6] for the single machine restricted case; hence, it runs in pseudopolynomial time. It is important to note that *Restricted Heuristic* does not guarantee an optimal solution.

4. Dynamic Programming Algorithm for the Two-Machine JSSP

The algorithm JSSPET presented here uses a DP algorithm to find optimal solutions for the two-machine E/T JSSP when the CDD is semirestricted. This algorithm partitions the solution space into schedules with either $t_1^* = F_1$ or $t_2^* = F_2$ and those with either $t_1^* > F_1$ or $t_2^* > F_2$. In the first case, jobs either in M_1 or M_2 are scheduled in the interval $[F_1, F_1 + P_1]$ or $[F_2, F_2 + P_2]$, where $P_1 = \sum_{j \in M_1} p_{1j}$ and $P_2 = \sum_{j \in M_2} p_{2j}$. Based on Property 3, any optimal schedule is either EVS or TVS, and so EVS (TVS) procedure discussed next will find optimal

EVS (TVS) schedules which is completed at either $F_1 + P_1$ or $F_2 + P_2$. In the second case, suppose that either $t_1^* > F_1$ or $t_2^* > F_2$ and based on Property 1, it is possible to assume that either $C_{1a} = \text{CDD}$ or $C_{2a} = \text{CDD}$ so that *Nosplit* procedure discussed later will find an optimal schedule in which a job is completed at CDD. Finally, the lower cost offered by the three procedures is an optimal schedule. All three procedures make use of Properties 2 and 3 (V-shaped structure).

Procedures EVS and TVS consider jobs in nonincreasing order. From Property 3, job n_1 (to be finished on machine 1) either starts at F_1 or ends at $F_1 + P_1$, and job n_2 (to be finished on machine 2) either starts at F_2 or ends $F_2 + P_2$. The total processing time of previously scheduled jobs which finish before CDD in *EVS* and after CDD in *TVS* procedures

is stored. *Nosplit* procedure considers jobs in nondecreasing order.

4.1. Procedure EVS. Let $f_k(a_1)$ = the minimum cost to schedule jobs $n_1, n_1 - 1, \dots, n_1 - k + 1$ (similarly $n_2, n_2 - 1, \dots, n_2 - k + 1$) provided that the latest job scheduled which finished at or before CDD finishes at time $\text{CDD} - a_1, a_1 \geq 0$ (similarly $\text{CDD} - a_2, a_2 \geq 0$), and the earliest job scheduled which finishes after CDD starts at time $\sum_{j=1}^{n_1-k} p_{1j} - a_1 + \text{CDD}$ (similarly $\sum_{j=1}^{n_2-k} p_{2j} - a_2 + \text{CDD}$). That is, the latest job finishes at $F_1 + P_1$ (similarly $F_2 + P_2$).

Recurrence relation is as follows:

$$f_{k+1}(a_1) = \begin{cases} \min \left\{ a_1 + f_k(a_1 + p_{1n_1-k}), \sum_{j=1}^{n_1-k} p_{1j} - a_1 + f_k(a_1) \right\}, & \text{if } \sum_{j=1}^{n_1-k} p_{1j} > a_1, a_1 + p_{1n_1-k} \leq \text{CDD}, \\ a_1 + f_k(a_1 + p_{1n_1-k}), & \text{if } \sum_{j=1}^{n_1-k} p_{1j} \leq a_1, a_1 + p_{1n_1-k} \leq \text{CDD}, \\ +\infty, & \text{otherwise.} \end{cases} \quad (8)$$

Boundary condition is as follows:

$$\begin{aligned} f_0(a_1) &= 0, \quad \text{for } a_1 = \text{CDD}, \\ f_0(a_1) &= +\infty, \quad \text{for } a_1 \neq \text{CDD}. \end{aligned} \quad (9)$$

Minimum cost schedule is defined by

$$z(\sigma_{\text{EVS}}^*) = \min_{0 \leq a_1 \leq \text{CDD}} f_n(a_1). \quad (10)$$

4.2. Procedure TVS. Let $g_k(m_1)$ = the minimum cost to schedule jobs $n_1, n_1 - 1, \dots, n_1 - k + 1$ (similarly $n_2, n_2 - 1, \dots, n_2 - k + 1$) provided that the earliest job scheduled which starts at or after CDD starts at time $\text{CDD} + m_1, m_1 \geq 0$ (similarly $\text{CDD} - m_2, m_2 \geq 0$), and the latest job scheduled which starts before CDD finishes at time $\text{CDD} + m_1 - \sum_{j=1}^{n_1-k} p_{1j}$ (similarly $\text{CDD} + m_2 - \sum_{j=1}^{n_2-k} p_{2j}$). That is, the earliest such a job starts is at F_1 (similarly F_2).

Recurrence relation is as follows:

$$\begin{aligned} &g_{k+1}(m_1) \\ &= \begin{cases} \min \left\{ \left| \sum_{j=1}^{n_1-k-1} p_{1j} - m_1 \right| + g_k(m_1), m_1 + p_{1n_1-k} + g_k(m_1 + p_{1n_1-k}) \right\}, & \text{if } \sum_{j=1}^{n_1-k} p_{1j} > m_1, m_1 + p_{1n_1-k} \leq P_1 - \text{CDD}, \\ m_1 + p_{1n_1-k} + g_k(m_1 + p_{1n_1-k}), & \text{if } \sum_{j=1}^{n_1-k} p_{1j} \leq m_1, m_1 + p_{1n_1-k} \leq P_1 - \text{CDD}, \\ +\infty, & \text{otherwise.} \end{cases} \end{aligned} \quad (11)$$

Boundary condition is as follows:

$$\begin{aligned} g_0(m_1) &= 0, \quad \text{for } m_1 = P_1 - \text{CDD}, \\ g_0(m_1) &= +\infty, \quad \text{for } m_1 \neq P_1 - \text{CDD}. \end{aligned} \quad (12)$$

Minimum cost schedule is defined by

$$z(\sigma_{\text{TVS}}^*) = \min_{0 \leq m_1 \leq P_1 - \text{CDD}} g_n(m_1). \quad (13)$$

4.3. Procedure Nosplit. Let $h_k(e_1)$ = the minimum cost to schedule jobs $1, 2, \dots, k$ for either M_1 or M_2 without the CDD

splitting any job, given that the total processing time of jobs scheduled early (or on time) is either e_1 or e_2 .

$$f_{k+1}(a_1) = \begin{cases} \min \left\{ e_1 - p_{1k+1} + h_k(e_1 + p_{1k+1}), \sum_{j=1}^{k+1} p_{1j} - e_1 + h_k(e_1) \right\}, & \text{if } e_1 \geq p_{1k+1}, \\ \sum_{j=1}^{k+1} p_{1j} - e_1 + h_k(e_1), & \text{otherwise.} \end{cases} \quad (14)$$

Boundary condition is as follows:

$$\begin{aligned} h_0(e_1) &= 0, & \text{if } e_1 &= 0, \\ h_0(e_1) &= +\infty, & \text{if } e_1 &\neq 0. \end{aligned} \quad (15)$$

Minimum cost schedule is defined by

$$z(\sigma_{\text{Nosplit}}^*) = \min_{0 \leq e_1 \leq \text{CDD}} h_n(e_1). \quad (16)$$

In all of the three procedures, the first alternative in the recurrence relation represents the cost of scheduling the next job as early as possible and the second one, similarly, as late as possible. Since in the TVS procedure an early job may finish after the CDD, there is a need for the absolute value in the first equation on its recurrence relation.

4.4. JSSPET Algorithm. In order to solve the two-machine E/T JSSP over a CDD, the three cases of the due date must be considered. The JSSPET algorithm decides the type of CDD (restricted, semirestricted, or restricted) and then applies the appropriate procedure. When the CDD is unrestricted, the JSSPET uses the SCHED procedure [1] to find the optimal solution for both machines. When the CDD is semirestricted, JSSPET jointly uses the SCHED procedure to find the optimal solution for one of the machines (the one with $\min\{F_1 + \Delta_1, F_2 + \Delta_2\}$), and the EVS, TVS, and Nosplit procedures to find the optimal for the other machine. Finally, when the CDD is restricted, JSSPET uses *Restricted Heuristic* (explained earlier) to find approximate solutions. This heuristic procedure reduces a restricted problem to the semirestricted version by iteratively removing one job at the time. Once the problem is reduced to its semirestricted version, the SCHED, EVS, TVS, and Nosplit procedures are applied. Since the jobs removed are going to be tardy anyway, the SPT rule is applied in order to minimize their tardiness. Each time a removed job is scheduled, an improvement procedure runs in search for early time slots in the current schedule to decrease the tardiness cost. The pseudocode for JSSPET is given next.

Algorithm JSSPET

Calculate $F_1, F_2, \Delta_1, \Delta_2$
 Apply SCHED procedure to M_1
 Apply SCHED procedure to M_2

Recurrence relation is as follows:

Calculate T_1 = Number of tardy jobs in M_1^C
 Calculate T_2 = Number of tardy jobs in M_2^C
 If $\text{CDD} \geq \max\{F_1 + \Delta_1, F_2 + \Delta_2\}$ and $T_1 = 0$ and $T_2 = 0$ then

Optimal schedule is given by SCHED procedure on both machines

Schedule first operations on machine 1 and machine 2 by using EDD rule.

Stop.

Else

Apply SCHED to machine where $\min\{F_1 + \Delta_1, F_2 + \Delta_2\}$ holds

Apply EVS to the other machine

Apply TVS to the other machine

Apply Nosplit to the other machine

Solution for the other machine is $\min\{\text{EVS}, \text{TVS}, \text{Nosplit}\}$

Calculate T_1 = Number of tardy jobs on M_1^C

Calculate T_2 = Number of tardy jobs on M_2^C

If $\text{CDD} < \max\{F_1 + \Delta_1, F_2 + \Delta_2\}$ and $\text{CDD} \geq \min\{F_1 + \Delta_1, F_2 + \Delta_2\}$ and $T_1 = 0$ and $T_2 = 0$ then

Optimal schedule is given by SCHED and $\min\{\text{EVS}, \text{TVS}, \text{Nosplit}\}$

Schedule first operations on machine 1 and machine 2 by using EDD rule.

Stop.

Else

Apply Restricted Heuristic

End If

End If

End Algorithm

5. Computational Experiments

Sets of problems with 2 machines, 5, 6, 7, 8, 10, 20, 50, 100, and 500 jobs, with 30 problem instances per problem size, were generated. The processing times were generated from a

TABLE 3: Computational times for the unrestricted and semirestricted cases.

Jobs	Unrestricted			Semirestricted		
	Average (sec.)	Deviation (sec.)	Maximum (sec.)	Average (sec.)	Deviation (sec.)	Maximum (sec.)
$n = 5$	0.004	0.007	0.016	0.6230	1.1485	2.6698
$n = 6$	0.004	0.007	0.016	0.7674	1.2943	2.8778
$n = 7$	0.006	0.008	0.016	1.1335	1.5152	3.0914
$n = 8$	0.006	0.008	0.016	1.2123	1.6205	3.3062
$n = 10$	0.006	0.008	0.016	1.4929	1.8597	3.7322
$n = 20$	0.014	0.005	0.016	4.9724	1.9837	5.7373
$n = 50$	0.029	0.007	0.047	19.9488	5.5026	32.9513
$n = 100$	0.056	0.008	0.063	67.1799	9.2983	74.6444
$n = 500$	0.294	0.014	0.359	1299.8607	63.9390	1590.2551

discrete uniform distribution $U(1, 100)$ and the jobs routes were obtained from another discrete uniform distribution $U(1, m)$ where m is the number of machines (two in our case). Similar to most random numbers generators in use today, the processing times and jobs routes were generated by using random numbers coming from a linear congruential generator [39].

The CDD for the unrestricted case is given by

$$\text{CDD} = \max(F_1 + \Delta_1, F_2 + \Delta_2). \quad (17)$$

For the semirestricted case, the CDD is chosen to be in the middle of the interval between $\min(F_1 + \Delta_1, F_2 + \Delta_2)$ and $\max(F_1 + \Delta_1, F_2 + \Delta_2)$ as given by

$$\begin{aligned} \text{CDD} &= \min(F_1 + \Delta_1, F_2 + \Delta_2) + 0.5 \\ & * [\max(F_1 + \Delta_1, F_2 + \Delta_2) \\ & - \min(F_1 + \Delta_1, F_2 + \Delta_2)]. \end{aligned} \quad (18)$$

Finally, for the restricted case, the CDD is given by

$$\text{CDD} = \lfloor h * \min(F_1 + \Delta_1, F_2 + \Delta_2) \rfloor, \quad (19)$$

where h is the tightness factor, $h \in [0, 1)$, and it takes four possible values, $h = 0.7, 0.8, 0.9, 0.95$. $\lfloor x \rfloor$ is the largest integer less than or equal to x .

Considering the four cases of the restricted version, the single case of both the unrestricted and the semirestricted version, a total of $9 * 30 * 6 = 1620$ problem instances were solved.

6. Results

The results in Table 3 show the average, standard deviation, and maximum computational solution times for each set of instances. The times are in seconds and exclude input and output time. Computational solution times increase approximately linearly with n for the unrestricted case and in proportion to n^2 for the semirestricted case. These results confirm that *JSSPET* algorithm finds optimal solutions for both the unrestricted and the semirestricted cases for large

random instances of the problem within no more than 20 minutes. Such result is made possible by the new optimality conditions extended from the single machine problem provided in this paper and which enable us to prove the optimality of the dynamic programming procedure.

For restricted problems with 5, 6, 7, and 8 jobs, finding an optimal solution is not guaranteed, but when compared with optimal solutions obtained through a MILP formulation, it turned out the *JSSPET* algorithm found optimal solutions for about 30% to 40% of all the instances by applying the *Restricted Heuristic*.

For restricted problems with 10 or more jobs, the *JSSPET* algorithm is evaluated based on how far its solutions are from a lower bound (LB). The LB used in this case is the optimal solution for the same instances but with a semirestricted CDD. Recall that a problem instance's solution with a restricted CDD will always be larger than the same instance with unrestricted or semirestricted CDD. The closer the tightness factor h to one, the smaller the deviation from the LB (the semirestricted version). Also, as the number of jobs increases, the deviation from the LB decreases. Since the behavior of the optimal objective function value for the restricted version of the problem is unknown, using the solution of the semirestricted version as a LB for the restricted case tends to underestimate the performance of the *JSSPET* algorithm. Therefore, this LB needs to be used carefully and a better one needs to be found.

7. Conclusions

The earliness and tardiness problem is an important problem in machine scheduling involving nonregular measures of performance. In this work, a dynamic programming (DP) algorithm to deal with the two-machine job shop scheduling problem (JSSP) and a common due date (CDD) were presented. The CDD can be classified as unrestricted, restricted, or semirestricted depending on how large it is. Lauff and Werner [30] conjectured that the definition of the restrictedness of the CDD for a multimachine early/tardy job shop scheduling problem is a NP-hard problem. In this research, a pseudopolynomial procedure to define the class of restrictedness was presented for the two-machine *E/T* JSSP.

Additionally, some properties for this problem as well as optimality conditions for the unrestricted and semirestricted case were extended from the single machine problem. Optimal solutions for the unrestricted and semirestricted case were obtained for problems with up to 500 jobs by using the SCHED algorithm and a dynamic programming algorithm, respectively. Two properties for the restricted case were proved and used to come up with a heuristic algorithm for the two machines problem with restricted CDD.

Finally, through experiments, it was shown that the JSSPET algorithm could find optimal solutions for the unrestricted and semirestricted versions of the problem. Also, the proposed algorithm works well as a heuristic for the restricted case with large problems. However, there is room to improve the performance of the JSSPET algorithm through the development of optimality conditions for the restricted case and by introducing better lower bounds to better evaluate the performance of the heuristic.

Future research involves extending this work to job shop scheduling problems with more than two machines.

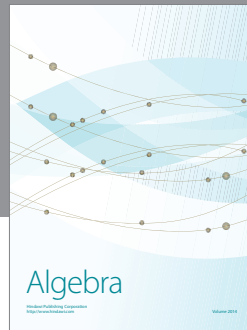
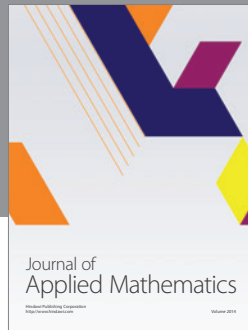
Competing Interests

The authors declare that they have no competing interests.

References

- [1] J. J. Kanet, "Minimizing the average deviation of job completion times about a common due date," *Naval Research Logistics Quarterly*, vol. 28, no. 4, pp. 643–651, 1981.
- [2] P. S. Sundararaghavan and M. U. Ahmed, "Minimizing the sum of absolute lateness in single-machine and multimachine scheduling," *Naval Research Logistics Quarterly*, vol. 31, no. 2, pp. 325–333, 1984.
- [3] W. Szwarc, "Single-machine scheduling to minimize absolute deviation of completion times from a common due date," *Naval Research Logistics*, vol. 36, no. 5, pp. 663–673, 1989.
- [4] U. Bagchi, R. S. Sullivan, and Y.-L. Chang, "Minimizing mean absolute deviation of completion times about a common due date," *Naval Research Logistics Quarterly*, vol. 33, no. 2, pp. 227–240, 1986.
- [5] N. G. Hall and M. E. Posner, "Earliness-tardiness scheduling problems. I. Weighted deviation of completion times about a common due date," *Operations Research*, vol. 39, no. 5, pp. 836–846, 1991.
- [6] N. G. Hall, W. Kubiak, and S. P. Sethi, "Earliness-tardiness scheduling problems, II. Deviation of completion times about a restrictive common due date," *Operations Research*, vol. 39, no. 5, pp. 847–856, 1991.
- [7] A. M. A. Hariri and C. N. Potts, "Single machine scheduling with deadlines to minimize the weighted number of tardy jobs," *Management Science*, vol. 40, no. 12, pp. 1712–1719, 1994.
- [8] G. Rabadi, M. Mollaghasemi, and G. C. Anagnostopoulos, "A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time," *Computers and Operations Research*, vol. 31, no. 10, pp. 1727–1751, 2004.
- [9] S. J. Mason, S. Jin, and J. Jampani, "A moving block heuristic for minimizing earliness and tardiness on a single machine with unrestrictive common due dates," *Journal of Manufacturing Systems*, vol. 24, no. 4, pp. 328–338, 2005.
- [10] G. Rabadi, G. C. Anagnostopoulos, and M. Mollaghasemi, "A heuristic algorithm for the just-in-time single machine scheduling problem with setups: a comparison with simulated annealing," *The International Journal of Advanced Manufacturing Technology*, vol. 32, no. 3-4, pp. 326–335, 2007.
- [11] M. O. Atan and M. Selim Akturk, "Single CNC machine scheduling with controllable processing times and multiple due dates," *International Journal of Production Research*, vol. 46, no. 21, pp. 6087–6111, 2008.
- [12] S. Hepdogan, R. Moraga, G. W. Depuy, and G. E. Whitehouse, "A Meta-RaPS for the early/tardy single machine scheduling problem," *International Journal of Production Research*, vol. 47, no. 7, pp. 1717–1732, 2009.
- [13] P. Baptiste, F. Della Croce, A. Grosso, and V. T'kindt, "Sequencing a single machine with due dates and deadlines: an ILP-based approach to solve very large instances," *Journal of Scheduling*, vol. 13, no. 1, pp. 39–47, 2010.
- [14] M. Cheng, P. R. Tadikamalla, J. Shang, and B. Zhang, "Single machine scheduling problems with exponentially time-dependent learning effects," *Journal of Manufacturing Systems*, vol. 34, pp. 60–65, 2015.
- [15] K. R. Baker and G. D. Scudder, "Sequencing with earliness and tardiness penalties: a review," *Operations Research*, vol. 38, no. 1, pp. 22–36, 1990.
- [16] H. Emmons, "Scheduling to a common due date on parallel uniform processors," *Naval Research Logistics*, vol. 34, no. 6, pp. 803–810, 1987.
- [17] T. C. E. Cheng and Z.-L. Chen, "Parallel-machine scheduling problems with earliness and tardiness penalties," *Journal of the Operational Research Society*, vol. 45, no. 6, pp. 685–695, 1994.
- [18] R. Alvarez-Valdes, J. M. Tamarit, and F. Villa, "Minimizing weighted earliness-tardiness on parallel machines using hybrid metaheuristics," *Computers and Operations Research*, vol. 54, pp. 1–11, 2016.
- [19] V. Kayvanfar, G. M. Komaki, A. Aalaei, and M. Zandieh, "Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times," *Computers & Operations Research*, vol. 41, no. 1, pp. 31–43, 2014.
- [20] Z. Li, H. Chen, R. Xu, and X. Li, "Earliness-tardiness minimization on scheduling a batch processing machine with non-identical job sizes," *Computers and Industrial Engineering*, vol. 87, pp. 590–599, 2015.
- [21] W. Kubiak, S. Lou, and S. Sethi, "Equivalence of mean flow time problems and mean absolute deviation problems," *Operations Research Letters*, vol. 9, no. 6, pp. 371–374, 1990.
- [22] H. Sarper, "Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem," *Applied Mathematical Modelling*, vol. 19, no. 3, pp. 153–161, 1995.
- [23] C. S. Sung and J. I. Min, "Scheduling in a two-machine flow-shop with batch processing machine(s) for earliness/tardiness measure under a common due date," *European Journal of Operational Research*, vol. 131, no. 1, pp. 95–106, 2001.
- [24] G. Mosheiov, "Scheduling unit processing time jobs on an m-machine flow-shop," *Journal of the Operational Research Society*, vol. 54, no. 4, pp. 437–441, 2003.
- [25] J. N. Gupta, V. Lauff, and F. Werner, "Two-machine flow shop scheduling with nonregular criteria," *Journal of Mathematical Modelling and Algorithms*, vol. 3, no. 2, pp. 123–151, 2004.

- [26] V. Lauff and F. Werner, "Heuristics for Two-Machine Flow Shop Problems with Earliness and Tardiness Penalties," Working Paper, pp. 1–24, 2003, <http://www.math.uni-magdeburg.de/~werner/preprints/p01-01.pdf>.
- [27] P. Chandra, P. Mehta, and D. Tirupati, "Permutation flow shop scheduling with earliness and tardiness penalties," *International Journal of Production Research*, vol. 47, no. 20, pp. 5591–5610, 2009.
- [28] J. Behnamian, S. M. T. Fatemi Ghomi, and M. Zandieh, "Development of a hybrid metaheuristic to minimise earliness and tardiness in a hybrid flowshop with sequence-dependent setup times," *International Journal of Production Research*, vol. 48, no. 5, pp. 1415–1438, 2010.
- [29] R. M'Hallah, "Minimizing total earliness and tardiness on a permutation flow shop using VNS and MIP," *Computers & Industrial Engineering*, vol. 75, no. 1, pp. 142–156, 2014.
- [30] V. Lauff and F. Werner, "Scheduling with common due date, earliness and tardiness penalties for multimachine problems: a survey," *Mathematical and Computer Modelling*, vol. 40, no. 5-6, pp. 637–655, 2004.
- [31] P. Baptiste, M. Flamini, and F. Sourd, "Lagrangian bounds for just-in-time job-shop scheduling," *Computers and Operations Research*, vol. 35, no. 3, pp. 906–915, 2008.
- [32] H.-A. Yang, Q.-F. Sun, C. Saygin, and S.-D. Sun, "Job shop scheduling based on earliness and tardiness penalties with due dates and deadlines: an enhanced genetic algorithm," *The International Journal of Advanced Manufacturing Technology*, vol. 61, no. 5–8, pp. 657–666, 2012.
- [33] S. Wang and Y. Li, "Variable neighbourhood search and mathematical programming for just-in-time job-shop scheduling problem," *Mathematical Problems in Engineering*, vol. 2014, Article ID 431325, 9 pages, 2014.
- [34] V. Gordon, J.-M. Proth, and C. Chu, "A survey of the state-of-the-art of common due date assignment and scheduling research," *European Journal of Operational Research*, vol. 139, no. 1, pp. 1–25, 2002.
- [35] V. Lauff and F. Werner, "On the complexity and some properties of multi-stage scheduling problems with earliness and tardiness penalties," *Computers and Operations Research*, vol. 31, no. 3, pp. 317–345, 2004.
- [36] M. Raghavachari, "A V-shape property of optimal schedule of jobs about a common due date," *European Journal of Operational Research*, vol. 23, no. 3, pp. 401–402, 1986.
- [37] J. A. Hoogeveen and S. L. van de Velde, "Scheduling around a small common due date," *European Journal of Operational Research*, vol. 55, no. 2, pp. 237–242, 1991.
- [38] K. R. Baker, *Elements of Sequencing and Scheduling*, Baker Press, Hanover, NH, USA, 1997.
- [39] W. D. Kelton and A. M. Law, *Simulation Modeling and Analysis*, McGraw Hill, Boston, Mass, USA, 2000.



Hindawi

Submit your manuscripts at
<http://www.hindawi.com>

