Computers and Operations Research 109 (2019) 1–11

Contents lists available at ScienceDirect

Computers and Operations Research

journal homepage: www.elsevier.com/locate/cor

Branch-and-bound algorithms for minimizing total earliness and tardiness in a two-machine permutation flow shop with unforced idle allowed

Jeffrey Schaller[a,*], Jorge Valente[b]

[a] Department of Business Administration, Eastern Connecticut State University, 83 Windham St., Willimantic, CT 06226-2295, USA
[b] LIAAD-INESC TEC, Faculdade de Economia, Universidade do Porto, Rua Dr. Roberto Frias, 4200-464 Porto, Portugal

ARTICLE INFO

Article history:
Received 22 April 2018
Revised 27 February 2019
Accepted 22 April 2019
Available online 23 April 2019

Keywords:
Scheduling
Branch-and-Bound
Flow shop
Earliness and tardiness

ABSTRACT

The two-machine permutation flow shop scheduling problem with the objective of minimizing total earliness and tardiness is addressed. Unforced idle time can be used to complete jobs closer to their due dates. It is shown that unforced idle time only needs to be considered on the second machine. This result is then used to extend a lower bound and dominance conditions for the single-machine problem to the two-machine permutation flow shop problem. Two branch-and-bound algorithms are developed for the problem utilizing the lower bound and dominance conditions. The algorithms are tested using instances that represent a wide variety of conditions.

1. Introduction

The focus of our research in this paper is on the scheduling problem of minimizing total earliness and tardiness in a flow shop environment. When scheduling customer jobs to be completed using an organization's resources, a variety of objectives can be considered, including the efficient use of resources and providing good customer service.

An important customer service objective is meeting customer due dates. Typically, when a customer places an order, a date for the completion of the order or job is agreed upon. If the job is completed after the date it was due, the job is tardy, and the length of time between the job's completion time and its due date is its tardiness. On the contrary, if a job is completed before its due date, then the job is early, and the length of time between the due date and the job's completion time is its earliness.

Tardiness has long been a traditional measure of scheduling effectiveness. When jobs are tardy, this can cause sales to be lost or, even worse, customers might switch to another supplier. Producers must also be concerned with early completion of customer orders. Indeed, when jobs are completed early the product must be stored, which not only requires space, but also causes capital to be tied up in inventory. This is true whether the product is stored at the customer's site or the producer's site.

As industry has intensified its emphasis on improving supply chain management in recent decades, earliness has been increasingly included as a measure of scheduling effectiveness. Therefore, there has been increased research into scheduling problems that have the objective of minimizing total earliness and tardiness.

In this paper, we consider a flow shop environment. In a flow shop there are multiple resources of different types, which we refer to as machines, which are used to complete jobs. A key feature of a flow shop is that all the jobs use the machines in the same order as they progress through the shop.

Historically, most of the research on the flow shop environment has considered the so-called permutation flow shop. In a permutation flow shop the jobs are processed in the same order on each machine. There are two reasons why most of the research has focused on permutation flow shops. First, a permutation flow shop is conceptually simpler. Second, in practice it is often difficult to change the order of the jobs from machine to machine. We only consider permutation schedules in this paper.

One of the ways to reduce the earliness of a job is to delay its processing and therefore its completion time. On a single-machine this is referred to as inserted idle time. Methods for finding optimal schedules that consider inserted idle have been developed (Schaller, 2007) for the single-machine problem, but not for more advanced multi-stage production systems. For example, research for flow shops with minimizing total earliness and tardiness (see

* Corresponding author.
E-mail addresses: schallerj@ecsu.ctstateu.edu (J. Schaller), jvalente@fep.up.pt (J. Valente).

https://doi.org/10.1016/j.cor.2019.04.017
0305-0548/

Fernandez-Viagas et al., 2016) have focused only on heuristic methods, and do not consider delaying of job processing by using unforced idle time.

In this research our goal is to extend the methods used in branch-and-bound algorithms for the single-machine problem to the two-machine permutation flow shop problem. By doing this, we hope to develop methods that can optimally solve small-sized instances of the problem. Additionally, we hope to obtain insights that will prove useful for developing heuristic methods that can be applied to larger-sized instances.

The remainder of the paper is organized as follows. Section two provides a literature review of relevant research for the problem. Section three gives a formal description of the problem and explains how to insert unforced idle time, in a two-machine permutation flow shop, to minimize total earliness and tardiness given a sequence. In section four, we show how a lower bound for the single-machine problem can be extended to the two-machine permutation flow shop and propose two branch-and-bound algorithms.

In section five, we develop dominance conditions that are derived from the single-machine problem and integrate them into the branch-and-bound algorithms. In section six we describe how the computational tests were constructed, and then present the results of those tests. Finally, section seven concludes the paper.

2. Literature review

Many papers for scheduling problems with both earliness and tardiness costs have been published. The first survey for early/tardy scheduling that covers most of the early work was provided by Baker and Scudder (1990). Hoogeveen (2005) reviewed multicriteria problems that included earliness and tardiness in a more recent survey. The single-machine environment has the most papers with an early/tardy objective. Valente (2009) reviews more recent papers that address an early/tardy scheduling with no idle time on a single-machine.

Kanet and Sridharan (2000) reviewed the early papers that addressed scheduling problems that include inserted idle time. Three of these early papers focused on how to insert idle time to optimize the objective, given a sequence for the single-machine problem. Fry et al. (1987) first addressed this problem by using a linear programming formulation. Davis and Kanet (1993) and Yano and Kim (1991) used special characteristics of the problem to develop more efficient timetabling procedures.

Branch-and-bound procedures were developed by Davis and Kanet (1993), Kim and Yano (1994), and Schaller (2007) for finding an optimal sequence and schedule for the single-machine problem. In all three of these branch-and-bound procedures a partial sequence of jobs is represented by a node in the branch-and-bound-tree.

Davis and Kanet (1993) use their timetabling procedure to calculate the earliness and tardiness of jobs in a partial sequence, which provides a lower bound for the objective, but does not consider the yet to be sequenced jobs. Kim and Yano (1994) developed an improved lower bound by considering all jobs, i.e. both those in the partial sequence and the ones that are still to be sequenced.

Schaller (2007) integrated the timetabling algorithm into the lower bound calculation, and also considered all jobs (those in the partial sequence, as well as the remaining ones), and found this lower bound to be better than that of Kim and Yano. Dominance conditions for the single-machine problem with inserted idle time were also developed by Kim and Yano (1994), Szwarc (1993), and Schaller (2007).

Much less research has been conducted, for the earliness and tardiness objective, on production environments other than a single-machine. To the best of our knowledge, there are eight papers for the flow shop environment that include earliness and tardiness in the objective. None of these papers considers unforced idle time.

Moslehi et al. (2009) consider the objective of minimizing the sum of maximum earliness and tardiness in a two-machine flow shop and present an optimal procedure. Chandra et al. (2009) considered a permutation flow shop problem where all the jobs have the same due date. Several objectives, including earliness and tardiness minimization, were considered by Madhushini et al. (2009), and branch-and-bound procedures were developed for permutation flow shops.

Zegordi et al. (1995) present a simulated annealing algorithm with specialized knowledge for scheduling permutation flow shops to minimize the sum of weighted earliness and tardiness. A kanban flow shop was addressed by Rajendran (1999), and heuristics were developed for scheduling the kanban containers, for the objectives of minimizing the total weighted flowtime, weighted tardiness and weighted earliness.

Schaller and Valente (2013b) proposed a genetic algorithm and compared it with five other neighborhood search procedures and metaheuristics, for a permutation flow shop to minimize total earliness and tardiness. The same problem was considered by M'Hallah (2014), who developed a variable neighborhood search-inspired heuristic for the same problem.

Schaller and Valente (2013a) compare several metaheuristics for a permutation flow shop to minimize total earliness and tardiness with family setups; and found that a genetic algorithm worked best. Fernandez-Viagas et al. (2016) present a new constructive heuristic, as well as bounded local search procedures, for minimizing total earliness and tardiness in permutation flow shops.

To the best of our knowledge, the consideration of using unforced idle time in a permutation flow shop to reduce the sum of earliness and tardiness has not been addressed in previous research. In this paper, we show how results for the single-machine problem to minimize total earliness and tardiness can be extended to a two-machine permutation flow shop.

3. Problem description

In the problem considered there are n jobs that need to be processed in a flow shop with two machines. Let d_j be the due date of job j ($j = 1, \ldots, n$). The processing time and completion time of job j ($j = 1, \ldots, n$) on machine m ($m = 1, 2$) are represented by p_{jm} and C_{jm}, respectively. The earliness of job j, E_j, is defined as: $E_j = \max\{d_j - C_{j2}, 0\}$, for $j = 1,\ldots, n$. The tardiness of job j, T_j, is defined as: $T_j = \max\{C_{j2} - d_j, 0\}$, for $j = 1,\ldots, n$. Minimizing $Z = \sum_{j=1}^{n} E_j + T_j$ (total earliness and tardiness) is the objective.

This problem has a non-regular objective, and therefore unforced idle time can be inserted to possibly improve the objective, by increasing the completion time of jobs that would be early. Forced idle time is required whenever a machine becomes available but the next job to be processed on that machine is not yet ready for processing. However, to our knowledge, previous research has not considered unforced idle time for this problem.

In this paper, we consider unforced inserted idle time. We use [j] to denote the job sequenced in position j. Also, we use $FI_{[j]m}$ to denote the forced idle time, and $UI_{[j]m}$ to denote the unforced idle time, before the job in position j on machine m. Furthermore, we use $I_{[j]m}$ to denote the total idle time on machine m before the job in position j ($I_{[j]m} = FI_{[j]m} + UI_{[j]m}$). Completion times for the job in position j can then be calculated as $C_{[0]1} = C_{[0]2} = 0$, $C_{[j]1} = C_{[j-1]1} + UI_{[j]1} + p_{[j]1}$ and $C_{[j]2} = \max\{C_{[j]1}, C_{[j-1]2}\} + UI_{[j]2} + p_{[j]2}$.

Since only permutation schedules are considered in this research, a sequence of jobs that will be processed in the same

Table 1
Notation used in the paper.

Notation	Description
n	Number of jobs
j	Job index. j = 1, ..., n
m	Machine index. m = 1 or 2
d_j	Due date of job j
C_{jm}	Completion time of job j on machine m
E_j	Earliness of job j, $E_j = \max\{d_j - C_{j2}, 0\}$
T_j	Tardiness of job j, $T_j = \max\{C_{j2} - d_j, 0\}$
[j]	Job sequenced in position j
$FI_{[j]m}$	Forced idle time on machine m before the job sequenced in position j
$UI_{[j]m}$	Unforced idle time on machine m before the job sequenced in position j
$I_{[j]m}$	Total idle time on machine m before the job sequenced in position j ($I_{[j]m} = FI_{[j]m} + UI_{[j]m}$)

order on each machine is required to define a solution. Also, since unforced inserted idle time can be used to reduce the earliness of jobs, we also need to determine if and where to insert unforced idle time, for a given sequence of jobs, to determine a schedule, and hence a solution. Table 1 provides a summary of the notation introduced in this section.

For a given sequence of jobs, the completion time of job [j] on machine 2 will determine the earliness or tardiness of the job in position j of the sequence. As previously shown in this section, the completion time $C_{[j]2} = \max\{C_{[j]1}, C_{[j-1]2}\} + UI_{[j]2} + p_{[j]2}$, where $C_{[0]2} = 0$. Therefore, a lower bound on the start time of the job in position j on the second machine is given by $\max\{C_{[j]1}, C_{[j-1]2}\}$. Similarly, a lower bound on the completion time of the job in position j is $\max\{C_{[j]1}, C_{[j-1]2}\} + p_{[j]2}$ ($UI_{[j]2} = 0$).

A single-machine timetabling procedure for inserting idle time into a given sequence for minimizing total earliness and tardiness (Fry et al., 1987; Davis and Kanet, 1993; Kim and Yano, 1994) can be used with the constraint that the jobs cannot start before max $\{C_{[j]}, C_{[j-1]2}\}$ for [j] = 1,..., n. Inserting unforced idle time on machine 1 does not need to be considered, as doing so can only tighten the above constraint, and possibly increase a solution's objective value.

The problem can therefore be written as the mathematical program FS2ET.

FS2ET:

$$\textbf{Minimize}\, Z = \sum_{j=1}^{n} \max\left\{C_{[j]2} - d_{[j]}, 0\right\} + \max\left\{d_{[j]} - C_{[j]2}, 0\right\} \tag{1}$$

Subject to:

$$C_{[j]1} = \sum_{k=1}^{j} \left(p_{[k]1}\right) \qquad \text{for } j = 1, \ldots, n \tag{2}$$

$$C_{[j]2} \geq C_{[j-1]2} + UI_{[j]2} + p_{[j]2} \qquad \text{for } j = 1, \ldots, n \tag{3}$$

$$C_{[j]2} \geq C_{[j]1} + UI_{[j]2} + p_{[j]2} \qquad \text{for } j = 1, \ldots, n \tag{4}$$

$$C_{[0]2} = 0 \tag{5}$$

$$UI_{[j]2} \geq 0 \qquad \text{for all } j. \tag{6}$$

Eq. (1) is the sum of earliness and tardiness. Constraint sets 2, 3, 4 and 5 develop the completion times for each job on the second machine. These completion times are compared to the due dates to calculate the total earliness and tardiness in Eq. (1). Constraint set (6) requires nonnegative unforced idle times.

4. Branch-and-bound algorithms and bounds

4.1. Branch-and-bound algorithms

In this section, we propose two branch-and-bound algorithms that extend a procedure previously developed by Schaller (2007) for the single-machine problem. In both branch-and-bound algorithms, a node in the branch-and-bound tree represents a partial sequence of jobs. For each node in the branch-and-bound tree, both a lower bound and an upper bound on the optimal objective value is calculated, an upper bound is calculated, and some conditions that could help fathom the node are also examined.

An incumbent value, that represents the value of the total earliness and tardiness of the current best sequence, is compared to the lower bound found for a node. If the incumbent value is less than or equal to the lower bound, the node is fathomed. Each node's associated partial sequence is completed using a simple heuristic, and its associated objective value is calculated to obtain an upper bound. If the upper bound found is less than the incumbent value, then the incumbent value is updated, and the sequence is retained as the best sequence found so far.

If a complete sequence is found with an objective value that is less than the incumbent, then the incumbent is updated, and the complete sequence is retained as the best sequence found. An initial incumbent value and solution are obtained by sorting the jobs in earliest due date order (EDD), inserting idle time in the resulting sequence and calculating the solution's total earliness and tardiness.

The difference between the two algorithms is that in the first one, referred to as BBI, a node represents an initial partial sequence, while in the second one, denoted as BBP, a node instead corresponds to a post partial sequence. When a node represents an initial partial sequence, the sequence is built from the beginning, starting with the first job to be processed, and working toward the end of the sequence to the last job to be processed. On the contrary, when a node represents a post partial sequence, the sequence is built from the end, starting with the last job to be processed, and working toward the beginning of the sequence to the first job to be processed.

The reason for trying these two versions of the branch-and-bound algorithm is that in Schaller (2007)'s branch-and-bound algorithm for the single-machine problem, which was shown to be the most effective, a node represented a post partial sequence. Using a post partial sequence in the single machine problem was straightforward as a job's completion time before the consideration of inserting idle time is known (it is equal to the sum of the processing times of the not yet scheduled jobs, plus the job's processing time).

In the two-machine permutation flow shop problem we would know the job's completion time on the first machine. However, we

Table 2
Notation introduced in this section.

Notation	Description
σ	A partial sequence.
p	The number of jobs in the partial sequence σ'.
σ'	Set of jobs that have not yet been sequenced.
q	The number of jobs in the set σ' (q = n − p).
P (SPT_{jm})	The sum of the processing times of the j jobs with the shortest processing times on machine m, in the set σ'.
P (LPT_{jm})	The sum of the processing times of the j jobs with the longest processing times on machine m, in the set σ'.
$d_{EDD[j]}$	The jth job's due date when the jobs in σ' are sorted in earliest due date order.
$LBC_{[j]2}$	A lower bound for the completion time on the second machine, for the job in position j of a sequence, when no unforced idle time is used.
$I_{[j]2}$	The idle time on machine two before the job in position j.

would not know its completion time on the second machine. We can calculate a lower bound for a job's completion time on the second machine and use it in place of the actual completion time, but this causes the lower bound to be weaker. Using a node to represent an initial partial sequence in the branch-and-bound tree overcomes this problem, as we can calculate the completion times on both the first and second machines, for each job in the partial sequence, before any unforced idle time is considered.

4.2. Lower bounds

The lower bounds developed in this section are based on the lower bound developed by Schaller (2007) for a single machine environment, suitably modified to reflect scheduling in a two- machine flow shop. The lower bounds for the two branch-and-bound algorithms are quite similar in nature, though they necessarily differ somewhat due to the use of initial or post partial sequences.

We will first consider a node in BBP that represents p jobs in a post partial sequence σ. Let q = n − p and let σ' be the set consisting of these q jobs that have not yet been sequenced. When we complete the sequence, including the jobs in the set σ', then a timetabling algorithm will be used to solve FS2ET and obtain its sum of earliness and tardiness.

We could obtain a lower bound without considering the jobs in σ' by solving FS2ET for the jobs in σ, but this lower bound would be weak. The lower bound can be improved by considering the jobs that have not yet been sequenced, that is, the jobs in the set σ.' To develop the lower bound we use the following notation.

$P(SPT_{j1})$ = the sum of the processing times of the j jobs with the shortest processing times on the first machine, in the set σ'.
$P(SPT_{j2})$ = the sum of the processing times of the j jobs with the shortest processing times on the second machine, in the set σ'.
$P(LPT_{j2})$ = the sum of the processing times of the j jobs with the longest processing times on the second machine, in the set σ'.
$d_{EDD[j]}$ = the jth job's due date when the jobs in σ' are sorted in earliest due date (EDD) order ($d_{EDD[j]} \le d_{EDD[k]}$ if $j < k$).
$LBC_{[j]2}$ = a lower bound for the completion time on the second machine, for the job in position j of the sequence, when no unforced idle time is used.
$I_{[j]2}$ = the idle time immediately inserted before the job in position j.

Table 2 provides a summary of the notation introduced in this section.

Schaller (2007) proved that if the due dates in EDD order (d_{EDD}) are substituted for the actual due dates, and compared to the actual completion times, a lower bound on the total earliness and tardiness is obtained. That is, we have: $\sum_{j=1}^{n} (\max\{C_{[j]2} - d_{EDD[j]}, 0\}) + \max\{d_{EDD[j]} - C_{[j]2}, 0\}) \le \sum_{j=1}^{n} (\max\{C_{[j]2} - d_{[j]}, 0\} + \max\{d_{[j]} - C_{[j]2}, 0\})$. This result can be used as a substitute for (1) to provide a lower bound.

We also would not know the actual completion times ($C_{[j]2}$ for j = 1, …, n) of the jobs in a sequence. However, we can calculate lower and upper bounds on these completion times using the following equations. First the lower bound:

$$P(SPT_{j2}) + \sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \qquad \text{for } j = 1, \ldots, q$$

and

$$\sum_{k\in\sigma'} p_{k2} + \sum_{k=q+1}^{j} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \qquad \text{for } j = q+1, \ldots, n$$

And the upper bound:

$$P(LPT_{j2}) + \sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \qquad \text{for } j = 1, \ldots, q$$

and

$$\sum_{k\in\sigma'} p_{k2} + \sum_{k=q+1}^{j} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \qquad \text{for } j = q+1, \ldots, n$$

We also know that the completion time for the job in position j of a sequence must be greater than or equal to the lower bound on the completion time if no unforced idle time is used before the job in position j ($C_{[j]2} \ge LBC_{[j]2}$). These equations can be substituted for (2), (3) and (4) of the mathematical program FS2ET, and we can then formulate the following mathematical program FS2ETEDD1 to develop a lower bound, given a post partial sequence σ.

FS2ETEDD1:

Minimize Z_{LB}

$$= \sum_{j=1}^{q} \left(\max\left\{C_{[j]2} - d_{EDD[j]}, 0\right\} + \max\left\{d_{EDD[j]} - C_{[j]2}, 0\right\}\right) + \sum_{j=q+1}^{n} \left(\max\left\{C_{[j]2} - d_{[j]}, 0\right\} + \max\left\{d_{[j]} - C_{[j]2}, 0\right\}\right) \qquad (7)$$

Subject to:

$$P(SPT_{j2}) + \sum_{k=q+1}^{j} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \qquad \text{for } j = 1, \ldots, q \qquad (8)$$

$$\sum_{k\in\sigma'} p_{k2} + \sum_{k=q+1}^{j} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \le C_{[j]2} \qquad \text{for } j = q+1, \ldots, n \qquad (9)$$

$$P(LPT_{j2}) + \sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \qquad \text{for } j = 1, \ldots, q \qquad (10)$$

$$\sum_{k\in\sigma'} p_{k2} + \sum_{k=q+1}^{j} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \ge C_{[j]2} \qquad \text{for } j = q+1, \ldots, n \qquad (11)$$

$$C_{[j]2} \ge LBC_{[j]2} \qquad \text{for } j = 1, \ldots, n \qquad (12)$$

$$I_{[j]2}, \ge 0 \qquad \text{for} j = 1, \ldots n. \qquad (13)$$

Eq. (7) is the objective function in mathematical program FS2ETEDD1. The due dates of the jobs that have not yet been sequenced (σ') are sorted in EDD order. These due dates are used for the first q positions, and the actual due dates are used for the other positions. Completion times and idle times need to be determined for the solution of the mathematical program.

Constraint set 8 sets a lower bound for the completion times on the second machine for the first q jobs of a complete sequence. This constraint set requires the completion time of the job sequenced in the jth position of a complete sequence to be at least as great as the sum of the processing times of the jobs in the set σ' on the second machine, sorted in shortest processing time order, for the first j positions, plus any idle time that is used.

Constraint set 9 sets a lower bound for the completion times on the second machine, for the last p jobs of a complete sequence. This constraint set requires the completion time of the job sequenced in the jth position of a complete sequence, for positions $q+1$ to n, to be at least as great as the sum of the processing times of the jobs in the set σ' on the second machine, plus the sum of the processing times on the second machine of the jobs sequenced in positions $q+1$ to j, plus any idle time that is used.

Constraint set 10 sets an upper bound for the completion times on the second machine for the first q jobs of a complete sequence. This constraint set requires the completion time of the job sequenced in the jth position of a complete sequence to be no greater than the total processing time, on the second machine, of the j jobs in the set σ' with the longest processing times on the second machine, plus the total processing time on the second machine of the jobs sequenced in positions $q+1$ to j (if $j > q$), plus any idle time that is used.

Constraint set 11 sets an upper bound for the completion times on the second machine for the last p jobs of a complete sequence. This constraint set requires the completion time of the job sequenced in the jth position of a complete sequence, for positions $q+1$ to n, to be no greater than the sum of the processing times of the jobs in the set σ' on the second machine, plus the sum of the processing times, on the second machine, of the jobs sequenced in positions $q+1$ to, j plus any idle time that is used.

Constraint set 12 requires the completion time of the job sequenced in the jth position of a complete sequence to be at least as great as a lower bound on the completion time considering both machines. This lower bound will be defined below. Finally, constraint set (13) requires nonnegative inserted idle times.

The lower bounds on completion times in constraint sets 8 and 9 only consider the processing times on the second machine. By considering the processing times of jobs on the first machine, an additional lower bound for completion times on the second machine can be developed. We use the notation $LBC_{[j]2}$ to denote that this is a lower bound on the completion time on the second machine of the job in the jth position of a complete sequence. This lower bound does not consider any unforced idle time but may be tighter than the lower bound obtained by using constraint sets 8 and 9.

For positions 1 through q, $LBC_{[j]2}$ is defined as: $LBC_{[j]2} = \max\{P(SPT_{k1}) + P(SPT_{(j-k+1)2})\ k = 1, \ldots, j\}$ for positions 1 through q. For positions $q+1$ through n, it is instead calculated as:

$$LBC_{[j]2} = \max\left\{\sum_{k\in\sigma'} p_{k1} + \sum_{k=q+1}^{j} p_{[k]1}, LBC_{[j-1]2}\right\} + p_{[j]2}.$$

In this lower bound, and for the first q positions, we obtain a lower bound for the processing times on the first machine for the first k jobs. This represents a lower bound on the start time, on the second machine, of the job in position k. After the job starts, on the second machine, of the job in position k, we still need to process the jobs in positions k through j, on the second machine, to complete the job in position j. We would not know these processing times but use a lower bound on their sum. We then take the maximum lower bound for $k < j$.

If $j > q$, we know the sum of the processing times on the first machine for the first j jobs of a complete sequence. This represents a lower bound on the start, on the second machine, of the job in position k. To this start time, we then add the processing time on the second machine of the job in position k, to obtain a lower bound on its completion. A lower bound on the start, on the second machine, of the job in position k can also be obtained by using the lower bound on the completion time of the job sequenced immediately before job [k] ($LBC_{[j-1]2}$). To this, we then add the processing time, on the second machine, of the job in position k, to obtain a lower bound on its completion. We use the maximum of these two lower bounds for the jobs sequenced in positions $j > q$.

The objective (7) and constraint sets 8, 9, and 11 correspond to the mathematical model that provides a lower bound for the single machine problem developed by Schaller (2007). The timetabling algorithm developed by Schaller (2007) can then be used to solve the model with these constraint sets. A minor modification to the timetabling algorithm, in order to include constraint set 10, is used in this research to develop the lower bound.

Given an initial partial sequence σ with p jobs, in the BBI algorithm we can use the mathematical program FS2ETEDD2 to obtain a lower bound.

FS2ETEDD2:

Minimize Z_{LB}

$$= \sum_{j=1}^{p}\left(\max\left\{C_{[j]2} - d_{[j]}, 0\right\} + \max\left\{d_{[j]} - C_{[j]2}, 0\right\}\right)$$

$$+ \sum_{j=p+1}^{n}\left(\max\left\{C_{[j]2} - d_{EDD[j]}, 0\right\} + \max\left\{d_{EDD[j]} - C_{[j]2}, 0\right\}\right) \quad (14)$$

Subject to:

$$\sum_{k=1}^{j} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} = C_{[j]2} \qquad \text{for } j = 1, \ldots, p \quad (15)$$

$$C_{[j]2} \geq C_{[j--1]2} + I_{[j]2} + p_{[j]2} \qquad \text{for } j = 1, \ldots, p \quad (16)$$

$$C_{[j]2} \geq C_{[j]1} + I_{[j]2} + p_{[j]2} \qquad \text{for } j = 1, \ldots, p \quad (17)$$

$$P\left(SPT_{(j-p)2}\right) + \sum_{k=1}^{p} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \leq C_{[j]2} \qquad \text{for } j = p+1, \ldots, n \quad (18)$$

$$P\left(LPT_{(j-p)2}\right) + \sum_{k=1}^{p} p_{[k]2} + \sum_{k=1}^{j} I_{[k]2} \geq C_{[j]2} \qquad \text{for } j = p+1, \ldots, n \quad (19)$$

$$C_{[j]2} \geq LBC_{[j]2} \qquad \text{for } j = p+1, \ldots, n \quad (20)$$

$$I_{[j]2}, \geq 0 \qquad \text{for } j = 1, \ldots n. \quad (21)$$

The objective (14) is the same as (7). However, and since we have an initial partial sequence instead of a post partial sequence, the actual due dates are used for the jobs in the first p positions, while due dates sorted in earliest due date order are used for the jobs in the last n – p positions (that is, for the jobs in set σ').

Since we know the processing times on both machines of the first p jobs in a sequence, we can calculate the actual completion times when the amount of unforced idle time is determined. This is represented by constraints 15, 16, and 17.

We can obtain lower and upper bounds for the completion times for the positions that will be filled by the jobs in the set σ', and this is represented by constraint sets 18, 19, and 20. Constraint set (21) requires nonnegative inserted idle times. The timetabling algorithm used in FS2ETEDD1 was modified to account for the different constraints, and this algorithm is used to solve FS2ETEDD2 to obtain a lower bound.

4.3. Upper bound

For each node to be evaluated, in either of the branch-and-bound algorithms (BBI or BBP), if its lower bound is less than the incumbent value, then an upper bound is calculated. First, the partial sequence is completed. This is done by sorting the unscheduled jobs in earliest due date order (EDD). For the BBI algorithm these jobs are placed after the initial partial sequence, while for the BBP algorithm they are placed before the partial sequence.

A timetabling algorithm is then used to insert unforced idle time, by minimizing the objective value for the sequence. The total earliness and tardiness of this schedule is the upper bound. If the upper bound is less than the incumbent value, then the incumbent value is updated, and this sequence is retained as the best solution found.

5. Dominance conditions

This section presents conditions for jobs that are adjacent in a partial sequence that can eliminate further consideration of the partial sequence in the branch-and-bound tree. There are conditions that have been used for the single-machine early/tardy problem with inserted idle allowed that can be modified, to be used in the two-machine permutation flow shop problem.

The modifications to the single-machine conditions are needed because we need to consider processing on the first machine, as well as the second machine, in the two-machine permutation flow shop problem. Also, since the conditions are examined with respect to a partial sequence, we need to consider what will happen to the start and completion times on the second machine, as jobs are added to a partial sequence. In order to evaluate a partial sequence, we can use a timetabling algorithm to insert unforced idle time on the second machine for the jobs in the partial sequence and calculate the earliness and tardiness of the jobs in the partial sequence.

5.1. Initial versus post partial sequences

When we check the conditions, it is important to remember that if we are working with an initial partial sequence (BBI), then the completion times of the jobs (on the second machine) in this initial partial sequence could be pushed earlier as jobs are added to the partial sequence. If we are working with a post partial sequence (BBP), then the completion times of the jobs in the post partial sequence could be pushed to a later completion time as jobs are added to the partial sequence.

This can be demonstrated with the following five job example. The data for p_{j1}, p_{j2}, and d_j, $j = 1, ..., 5$, are shown in Table 3.

If we are considering the initial partial sequence 1 – 2 – 3, the completion times of these jobs without unforced idle are shown in Table 4A. Fig. 1 shows the Gantt chart for this initial partial sequence without unforced idle time.

As unforced idle time is used, the three jobs in the initial partial sequence will have their completion times increased, as their processing times are right-shifted because each of the jobs is early. In this example, eventually the start and completion times of the three jobs on the second machine will form a block, and there will be no idle time between these jobs.

Table 3
Data for the example.

	Job				
	1	2	3	4	5
p_{j1}	2	8	5	5	2
p_{j2}	2	4	6	10	10
d_j	39	37	38	36	39

Table 4
Completion times for the example initial partial sequence.

4A. Before unforced idle time is used.

	job		
Machine	1	2	3
C_{j1}	2	10	15
C_{j2}	4	14	21

4B. After unforced idle time is used.

	Job		
Machine	1	2	3
C_{j1}	2	10	15
C_{j2}	33	37	43

4C. After adding two more jobs.

	Job				
Machine	1	2	3	4	5
C_{j1}	2	10	15	20	22
C_{j2}	28	32	38	48	58

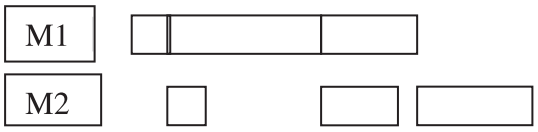

Fig. 1. Gantt chart for initial partial sequence, without unforced idle time.

The completion times of the jobs are shown in Table 4B, and the associated Gantt chart is shown in Fig. 2. At this point, increasing the idle time before the first job of the block by one unit will cause the objective to increase one unit. This is due to the fact, that one job is early, one is on time, and one job is tardy. Likewise, decreasing the idle time before the first job of the block by one unit will also cause the objective to increase one unit.

When jobs 4 and 5 are added to the partial sequence to create a complete sequence 1 – 2 – 3 – 4 – 5, then the completion times for the jobs will be as shown in Table 4C. The associated Gantt chart is shown in Fig. 3. Since the processing on the second machine for the two additional jobs occurs after the other jobs, these two additional jobs will be tardy if the completion times for the first three jobs shown in Table 4B were used.

Therefore, the processing of the first three jobs is left shifted, and idle time is decreased to decrease the objective value. With the completion times shown in Table 4C, two jobs are early, one job is on time, and two jobs are tardy, so adding or decreasing idle time before the block of jobs will increase the objective.

If we consider the post partial sequence 3 – 4 – 5, then the completion times on the second machine of these jobs could be right-shifted (increased) as additional jobs are added. Table 5 shows the completion times for the post partial sequence for the above example.

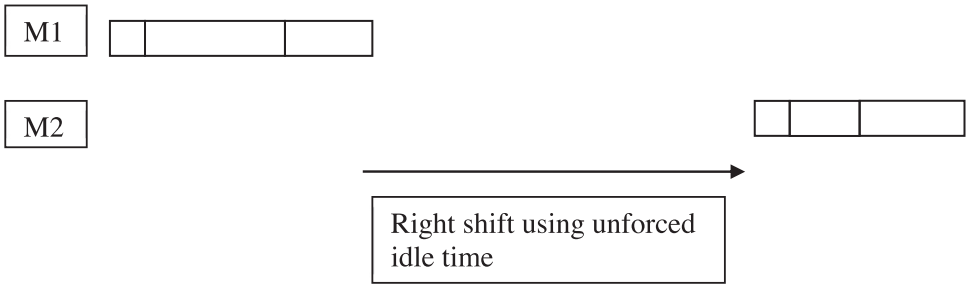

Fig. 2. Gantt chart for initial partial sequence, with unforced idle time.

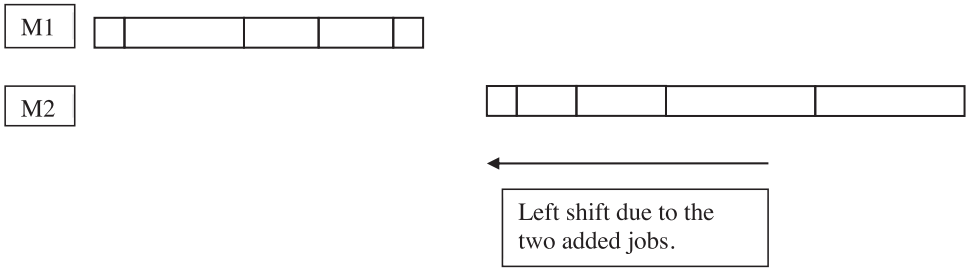

Fig. 3. Gantt chart with two additional jobs added.

Table 5
Completion times for the example post partial sequence.

5A. With unforced idle time.					
	Job				
Machine	1	2	3	4	5
C_{j1}			15	20	22
C_{j2}			26	36	46
5B. After adding two more jobs.					
	Job				
Machine	1	2	3	4	5
C_{j1}	2	10	15	20	22
C_{j2}	28	32	38	48	58

5.2. Rules for the single-machine problem

In this section, we provide a review of the rules used to eliminate nodes from further consideration in the single-machine early/tardy problem. In this paper, these rules will be modified so they can be used for the two-machine flow shop problem.

A rule for adjacent jobs was developed by Szwarc (1993). This rule determines the order for the two adjacent jobs based on whether they are scheduled before or after a calculated time. Szwarc's (1993) rule, restated by Schaller (2007) for the case where all weights for the costs are equal to 1, is provided below.

Szwarc's (1993) Rule: For each pair of jobs j and k, if they are to be adjacent to each other, job k should be before job j if they are started before t_{jk}, and j should be before job k if they are started after t_{jk}, where

$$t_{jk} = \{d_j - -0.5 * (p_j + p_k), \quad \text{if} d_j - -d_k > 0.5 * (p_j - -p_k) < 0,$$

$$d_j - -p_j - -0.5 * (p_j + p_k), \quad \text{if} d_j - -d_k \leq 0.5 * (p_j - -p_k) < 0,$$

$$0, \quad \text{if} p_j = p_k \& d_j - -d_k \leq 0\}.$$

Kim and Yano (1994) provided two rules for a pair of adjacent jobs. The two rules are derived from Szwarc's rule (1993) and are therefore corollaries of Szwarc's rule. The first rule determines which job is first if the jobs are early or on time, and the second rule determines which job is first if the jobs are tardy or on time.

Rule 1: If adjacent jobs j and k are started later or equal to max $\{d_j, d_k\}$, job j should be before job k when $p_k > p_j$, and job k should be before job j when $p_k < p_j$.

Rule 2: If adjacent jobs j and k are completed earlier than or at the same time as min $\{d_j - p_j, d_k - p_k\}$, job j should be before job k when $p_k < p_j$, and job k should be before job j when $p_k > p_j$.

Szwarc's rule is stronger than these rules, that is, Szwarc's rule eliminates the partial sequences that are eliminated by these rules. An additional corollary of Szwarc's rule, as noted by Schaller (2007), is that if there exist two jobs j and k such that $p_j = p_k$ and $d_j < d_k$, and the jobs are adjacent, then job j should be before job k. Schaller (2007) also shows that job j should be before job k, even when they are not adjacent, if $p_j = p_k$ and $d_j < d_k$.

5.3. Rules for Two-machine permutation flow shop scheduling

The rules reviewed in the previous section could be valuable in reducing the search space in a flow shop environment. One condition that is necessary for the rules of the previous section to apply in flow shops is that there is no idle time on the second machine between the two adjacent jobs to be examined.

Let jobs j and k be adjacent jobs in a sequence, with job j preceding job k. Also, let St_{km} be the start time of processing for job k on machine m. Then, for any of the rules in the preceding subsection to be of use, we must have $St_{k2} = C_{j2}$. Consider schedules S and S' which are the same, with the following exception. In both schedules jobs j and k are adjacent, but job j precedes job k in S, while job k precedes job j in S'.

Let job h be the first job scheduled after jobs j and k, if jobs j and k are not the last two jobs in schedules S and S'. If jobs j and k are the last jobs to be in the schedules, we define a fictitious job h with $p_{h1} = \infty$. Let C_{jm} (S), C_{km} (S), C_{jm} (S'), and C_{km} (S') represent the completion times of jobs j and k on machine m in schedules S and S'. Since jobs j and k are adjacent, and there will be no idle time between the two jobs on the first machine, then C_{k1} (S) = C_{j1} (S').

The completion time on the second machine, for the job in the latter of the two positions, could differ in schedules S and S'. If

the following two conditions (A and B) are both true, the rules for adjacent jobs in previous subsection can be used for the two-machine permutation flow shop problem.

A) Either C_{k2} (S) $\leq C_{j2}$ (S') or C_{k2} (S) $\leq C_{j1}$ (S') + p_{h1}, and
B) St_{k2} (S) = C_{j2} (S) and St_{j2} (S') = C_{k2} (S').

Proof. If schedule S has an earlier completion time for the two jobs than schedule S', that will not affect the completion times of the jobs sequenced after jobs j and k, unless the earlier completion time under schedule S is beneficial. The reason for this is unforced idle time could be used to increase the completion time of these jobs on the second machine, if that is needed to reduce earliness.

If the completion time on the second machine for the two jobs in schedule S is greater than that of schedule S', but less than the completion time of the two jobs on the first machine plus the processing time on the first machine of the next job to be scheduled after jobs j and k (job h), then again the completion times for the jobs sequenced after jobs j and k will also not be affected. If the completion time is later under schedule S than schedule S', however, that could increase the completion times on the second machine of the jobs sequenced after jobs j and k, and unforced idle time cannot be used to cause earlier completion times, so the later completion times could cause the objective to increase.

The completion times of the jobs that are scheduled before jobs j and k would also not be affected by whether schedule S or S' is chosen. Since there is no idle time between the two jobs on the second machine (the start time of the job sequenced second equals the completion time of the job sequenced first in both schedules), the comparison of the schedules involves a simple interchange between the two jobs on the second machine and is not affected by the completion times of the jobs on the first machine. Therefore, the local problem (adjacent job exchange) becomes the same as the single-machine problem, and the rules for comparing exchanges of adjacent jobs on a single-machine can be applied to the two-machine permutation flow shop problem. **//End of Proof**

We can now restate the rules for adjacent jobs on a single machine, so they can be applied to the two-machine permutation flow shop problem. Condition 1 is Kim and Yano's (1994) rule 1 modified to be used for the two-machine flow shop problem.

Condition 1: If jobs j and k are adjacent and cannot be started on the second machine earlier than max $\{d_j, d_k\}$, if (1) $p_{k2} > p_{j2}$, (2) St_{j2} (S') = C_{k2} (S') and St_{k2} (S) = C_{j2} (S), and (3) C_{k2} (S) $\leq C_{j2}$ (S') or C_{k2} (S) $\leq C_{j1}$ (S') + p_{h1}, then schedule S will be at least as good as schedule S' in terms of the objective, and job j should be before job k.

When evaluating partial sequences, condition 1 will be effective for post partial sequences, as in BBP. A timetabling algorithm can be used to optimize the partial sequence, and if a pair of adjacent jobs is found to meet condition 1, then it can be used to eliminate a node. The reason for this is that as jobs are added to partial sequences in BBP, the completion times on the second machine could become later, but not earlier, and therefore condition 1 will still be met.

However, this is not the case for initial partial sequences, as in BBI. When an initial partial sequence is being evaluated, it must be remembered that as jobs are added, the completion times on the second machine of the jobs in the partial sequence could be pushed earlier, and therefore cause the criteria of condition 1 to not be met.

If it is determined that criteria 2) and 3) of condition 1 will be met (this can be determined by scheduling the initial partial sequence without using any unforced idle time) then Swarc's rule (1994) can be used to determine maximum completion times, on the second machine, for jobs j and k as defined by condition 2.

These completion times can be used to strengthen the lower bound for an initial partial sequence. These maximum completion times could also cause other completion times for jobs sequenced earlier than jobs j and k to violate condition 3 (to be presented later).

Condition 2: If there are two adjacent jobs in an initial partial sequence, j and k, that cannot be started on the second machine earlier than max $\{d_j, d_k\}$, and job k precedes job j, then if (1) $p_{k2} > p_{j2}$, and (2) St_{j2} (S') = C_{k2} (S') and St_{k2} (S) = C_{j2} (S), and (3) C_{k2} (S) $\leq C_{j2}$ (S') or C_{k2} (S) $\leq C_{j1}$ (S') + p_{q1}, with the completion of the partial sequence then the completion time of job k in S' (C_{k2} (S')) must be less than or equal to $t_{jk} + p_{k2}$, and the completion time of job j in S' (C_{j2} (S')) must be less than or equal to $t_{jk} + p_{k2} + p_{j2}$.

Condition 3 is Kim and Yano's (1994) rule 2 modified to be used for the two-machine permutation flow shop problem.

Condition 3: If two adjacent jobs, j and k, are completed on the second machine before or equal to min $\{d_j - p_{j2}, d_k - p_{k2}\}$, if (1) $p_{k2} < p_{j2}$ and (2) C_{k2} (S) $\leq C_{j2}$ (S') or C_{k2} (S) $\leq C_{j1}$ (S') + p_{h1}, then schedule S will be at least as good as S', and job j should be before job k.

Note that St_{j2} (S') = C_{k2} (S') and St_{k2} (S) = C_{j2} (S) has been dropped from this condition. This is because the job sequenced first of the two jobs in either schedule will be early, and unforced idle time will be used to increase its completion time to equal the start time of the job sequenced second of the two jobs. Therefore, these criteria will be met.

Condition 3 will be effective for initial partial sequences, as in BBI. A timetabling algorithm can be used to optimize the partial sequence, and if a pair of adjacent jobs is found to meet condition 3, then it can be used to eliminate a node. The reason for this is that as jobs are added to partial sequences in BBI, the completion times on the second machine could become earlier, but not later, and therefore condition 3 will still be met.

However, this is not case for post partial sequences, as in BBP. When a post partial sequence is being evaluated, it must be remembered that as jobs are added, the completion times on the second machine of the jobs in the partial sequence could be pushed later, and therefore cause the criteria of condition 3 to not be met. If it is determined that criteria (2) of condition 3 will be met, then Swarc's rule (1994) can be used to determine minimum completion times on the second machine for jobs j and k, as defined by condition 4.

These completion times can be used to strengthen the lower bound for a post partial sequence. These minimum completion times could also cause other completion times for jobs sequenced later than jobs j and k to violate condition 1.

Condition 4: If there are two adjacent jobs in a post partial sequence, j and k, and job k precedes job j, then if (1) $p_{k2} < p_{j2}$, and (2) C_{k2} (S) $\leq C_{j2}$ (S') or C_{k2} (S) $\leq C_{j1}$ (S') + p_{h1}, then the completion time of job k in S' (C_{k2} (S')) must be greater than or equal to $t_{jk} + p_{k2}$, and the completion time of job j in S' (C_{j2} (S')) must be greater than or equal to $t_{jk} + p_{k2} + p_{j2}$.

Condition 5 is the modified version of Schaller's (2007) rule for jobs that have equal processing times, and are adjacent to each other, for use with the two-machine permutation flow shop problem. In this condition, the two jobs must have equal processing times on the second machine.

Condition 5. If jobs j and k are adjacent, (1) $p_{k2} = p_{j2}$, (2) St_{j2} (S') = C_{k2} (S'), and (3) $d_j < d_k$, then schedule S will be at least as good as S', and job j should be before job k.

The next condition, condition 6, is not derived from the single-machine early/tardy problem, but from the two-machine permutation flow shop problem with an objective of minimizing total tardiness (Sen et al. 1989). For this condition, let set B be the set of jobs sequenced before jobs j and k.

Condition 6. If jobs j and k are adjacent, (1) $p_{j2} \leq p_{k2}$, (2) $p_{j1} \leq p_{k1}$, (3) $p_{j1} \leq p_{j2}$, and (4) $\sum_{l \in B} p_{l1} + p_{j1} + p_{j2} \geq d_j$, then schedule S will be at least as good as S', and job j should be before job k.

Proof. The completion times of the jobs sequenced before jobs j and k are not be affected by which schedule, S or S', is selected. Requirements (1), (2) and (3) ensure $C_{k2}(S) \leq C_{j2}(S')$, so additional unforced idle time can be used on the second machine for schedule S for the jobs sequenced after jobs j and k. This allows that the completion times for schedule S, of the jobs sequenced after jobs j and k, could equal those for schedule S'. So, the total earliness and tardiness of the jobs sequenced after jobs j and k under schedule S is less than or equal to that of schedule S'.

Criterion 4) ensures that job j will not be early in either schedule. Therefore, to prove this condition we only need to prove that $T_j(S) + E_k(S) + T_k(S) \leq T_j(S') + E_k(S') + T_k(S')$. If $d_k \geq C_{j2}(S')$, then $T_k(S) = T_k(S') = 0$, $T_j(S) \leq T_j(S')$, and $E_k(S') < E_k(S)$, so we have $T_j(S) + E_k(S) + T_k(S) \leq T_j(S') + E_k(S') + T_k(S')$.

If $d_k \leq d_j$, then neither job j or k will be early, and $E_k(S) = E_k(S') = 0$. By (1) and (2), we have $C_j(S) \leq C_k(S')$, and $C_{k2}(S) \leq C_{j2}(S')$. Therefore, $T_j(S) + E_k(S) + T_k(S) \leq T_j(S') + E_k(S') + T_k(S')$. If $d_j < d_k < C_{j2}(S')$, then $T_j(S') - T_j(S) \geq E_k(S) + T_k(S)$. Thus, we have $T_j(S) + E_k(S) + T_k(S) \leq T_j(S') + E_k(S') + T_k(S')$. **//End of Proof**

Condition seven is based on Schaller (2007)'s single-machine condition for jobs that have equal processing times, but these jobs are not necessarily adjacent to each other. For this condition let set B be the set of jobs sequenced between jobs j and k. Consider a schedule S in which job j precedes job k and is sequenced immediately before the set of jobs B, and job k is sequenced immediately after the set of jobs B. Let S' be a schedule that has the same sequence as schedule S, with the exception that the positions of jobs j and k are exchanged, so job k immediately precedes the set of jobs B, and job j is sequenced immediately after the set of jobs B.

Condition 7. If for two jobs j and k, (1) $p_{k2} = p_{j2}$, (2) $p_{j1} \leq p_{k1}$, and (3) $d_j < d_k$, then job j precedes job k in an optimal schedule.

Proof. The completion times of the jobs sequenced before jobs j and k will not be affected by the choice of which job, j or k, is sequenced earlier. Since $p_{j1} \leq p_{k1}$, the jobs in set B will be completed on the first machine earlier or at the same time (if $p_{j1} = p_{k1}$) if job j precedes job k. Therefore, unforced idle time can be used, if necessary, to set the completion times on the second machine, of the jobs in set B when j is sequenced before job k to be equal to those when job k is sequenced before job j.

Since the completion time on the first machine will be the same for the job sequenced immediately after the set B (either job j or k) and $p_{j2} = p_{k2}$, the completion times of the jobs sequenced after the job that is sequenced second among jobs j and k will not be affected by the choice of which schedule S or S' is chosen. Therefore, to prove this theorem it only needs to be proved that $E_j(S) + T_j(S) + E_k(S) + T_k(S) \leq E_j(S') + T_j(S') + E_k(S') + T_k(S')$.

Note that we have $C_{j1}(S) + p_{j2} \leq C_{k1}(S') + p_{k2}$ and $C_{k1}(S) + p_{k2} = C_{j1}(S') + p_{j2}$. Let $Z_j(S) = E_j(S) + T_j(S)$, $Z_k(S) = E_k(S) + T_k(S)$, $Z_j(S') = E_j(S') + T_j(S')$, and $Z_k(S') = E_k(S') + T_k(S')$. Also let $A = d_k - d_j$ and $D = C_{k2}(S) - C_{k2}(S')$. There are two cases to consider: (1) $C_{k1}(S) + p_{k2} - d_k \geq 0$ and (2) $C_{k1}(S) + p_{k2} - d_k < 0$.

Case (1) $C_{k1}(S) + p_{k2} - d_k \geq 0$. Having $C_{k1}(S) + p_{k2} - d_k \geq 0$ implies that $C_{k2}(S) = C_{j2}(S')$ and $Z_j(S') = Z_k(S) + A$. Also, we have $C_{k2}(S') \geq C_{k1}(S') + p_{k2} \geq C_{j1}(S) + p_{j2}$. Since unforced idle time can be used to set $C_{j2}(S) = C_{k2}(S')$, if $d_k \geq C_{k2}(S')$ then $Z_k(S') \geq Z_j(S) - A$. Since $C_{j2}(S) \leq C_{k2}(S')$, if $d_k < C_{k2}(S')$ then $Z_k(S') \geq Z_j(S) - A$. Therefore, for this case, $Z_j(S) + Z_k(S) \leq Z_j(S') + Z_k(S')$.

Case 2) $C_{k1}(S) + p_{k2} - d_k < 0$. If $d_k < C_{k2}(S)$, then $C_{j2}(S') = C_{k2}(S)$ and $Z_j(S') = Z_k(S) + A$. Since $Z_k(S') \geq Z_j(S) - A$, we have $Z_j(S) + Z_k(S) \leq Z_j(S') + Z_k(S')$. If $d_k \geq C_{k2}(S)$, then $Z_k(S') = Z_k(S) + D$. Also, $C_{j2}(S') \leq C_{k2}(S)$ and $d_j \leq d_k$. Since unforced idle time can be used to set $C_{j2}(S) = C_{k2}(S')$, if necessary, then $Z_j(S) \leq Z_j(S') + D$. Therefore, we have $Z_j(S) + Z_k(S) \leq Z_j(S') + Z_k(S')$ for this case. **//End of Proof**

Two additional procedures are created by incorporating the dominance conditions. The BBID algorithm uses a node to represent an initial partial sequence and incorporates the dominance conditions. The BBPD algorithm uses a node to represent a post partial sequence and incorporates the dominance conditions.

6. Computational test

The proposed algorithms are tested on instances generated randomly with several levels of the number of jobs, and under various conditions of due date range and tightness.

6.1. Data

The procedures described in sections four and five were tested on problems that consisted of several levels of the number of jobs, and for nine due date range and tightness distributions. A set of 10 instances is generated for each combination of number of jobs and due date range and tightness parameters. Nine levels of number of jobs (n) are tested: $n = 8$, 10, 12, 14, 16, 18, 20, 25, and 30.

A uniform distribution was used to generate the processing times of the jobs on each machine. These times were generated over the integers 1 and 10. To randomly generate the due dates for the jobs, a uniform distribution was also used over the integers MS $(1 - r - R/2)$ and MS $(1 - r + R/2)$, where MS is the makespan estimated for the instance using the makespan lower bound proposed in Taillard (1993), and R and r referred to as the due date range and tardiness factors are parameters.

Three levels of due date range (R) were tested: $R = 0.2$, 0.6 and 1.0. We also considered three levels of due date tightness (r): $r = 0.0$, 0.2 and 0.4. These levels of R and r result in nine sets of due date parameters for each level of n.

The Turbo Pascal programming language was used to code the procedures, which then tested on a Dell Inspiron 1525 GHz Lap Top computer. Each procedure was performed for a maximum of 300 seconds for an instance. If a procedure was unable to prove an optimal solution for an instance within the time limit, it was terminated. For each procedure and for each combination of n, R and r, we recorded the average seconds used per instance, as well as the number of instances solved within 300 seconds.

6.2. Results of the test

Tables 6 and 7 show the results of the test for each level of number of jobs. Table 6 shows the average amount of time required per instance, while Table 7 shows how many instances were solved within the time limit. It is shown that for each of the algorithms the time required to solve instances increases with an increase in the number of jobs.

The results also show that the procedures that use a node to represent a post partial sequence (BBP and BBPD) instead of an initial partial sequence (BBI and BBID) generally require less time to solve instances than their counterpart algorithms. These algorithms were able to solve more of the instances with 16 or more jobs than their counterparts.

Including the dominance conditions into the algorithms had a positive effect. The BBID and BBPD algorithms required less time than the BBI and BBP procedures, respectively, for all levels of jobs.

Table 6
Average seconds by number of jobs.

	Number of Jobs								
Procedure	8	10	12	14	16	18	20	25	30
BBI	0.061	0.629	6.77	27.83	97.84	172.36	265.21	299.47	300.00
BBP	0.060	0.516	2.63	21.40	94.69	177.67	259.23	297.85	298.49
BBID	0.047	0.238	1.22	8.15	26.52	73.97	157.42	287.91	288.73
BBPD	0.050	0.264	0.97	4.08	15.67	67.56	129.56	265.34	289.67

Table 7
Number of problems solved by number of jobs.

	Number of Jobs								
Procedure	8	10	12	14	16	18	20	25	30
BBI	90	90	90	90	73	54	17	2	0
BBP	90	90	90	90	78	51	20	1	1
BBID	90	90	90	90	88	77	61	9	5
BBPD	90	90	90	90	90	82	71	18	6

Table 8
Average seconds by r and R for n = 16.

		Procedure			
r	R	BBI	BBP	BBID	BBPD
0.00	0.20	129.86	127.57	16.32	8.40
0.00	0.60	91.53	107.39	10.10	9.19
0.00	1.00	54.11	8.42	10.93	2.20
0.20	0.20	103.23	129.76	13.32	26.07
0.20	0.60	71.13	99.69	14.71	14.87
0.20	1.00	12.37	14.08	3.92	2.81
0.40	0.20	204.54	121.39	91.06	31.44
0.40	0.60	166.06	174.60	53.47	33.64
0.40	1.00	47.71	69.27	26.04	12.41

Table 9
Number of problems solved by r and R for n = 16.

		Procedure			
r	R	BBI	BBP	BBID	BBPD
0.00	0.20	9	7	10	10
0.00	0.60	8	8	10	10
0.00	1.00	9	10	10	10
0.20	0.20	8	8	10	10
0.20	0.60	10	8	10	10
0.20	1.00	10	10	10	10
0.40	0.20	4	10	9	10
0.40	0.60	6	7	9	10
0.40	1.00	9	10	10	10

The two algorithms with the dominance conditions included also solved more instances with 16 or more jobs. As the number of jobs increased the difference in the number of instances solved by the algorithms with the dominance conditions, compared to their counterparts without the dominance conditions increased.

Tables 8 and 9 show the results of the test for each combination of due date tightness and range parameters (r and R) when n = 16. Table 8 shows the average amount of time required per instance, while Table 9 shows the number of instances solved within the time limit.

The results show that as due dates become tighter, the algorithms generally required more time to solve instances. Three of the four algorithms were unable to solve all the instances with the tightest due dates tested (r = 0.40). As the due date range increases, the algorithms generally required less time to solve instances.

When the due date range was 1.00, three of the four algorithms were able to solve all the instances within the time limit and the other algorithm was able to solve 28 of the 30 instances. The BBPD algorithm required the least amount of time for seven of the nine combinations, and was the only algorithm that was able to solve all the instances within the time limit.

7. Conclusion

In this research, the two-machine permutation flow shop problem with the objective of minimizing total earliness and tardiness is addressed. Using unforced idle time to reduce the earliness of some jobs is considered. It is shown that unforced idle time is only needed on the second machine.

Approaches for the single-machine early/tardy problem with inserted idle time allowed are extended to the two-machine permutation flow shop. These include lower bounding procedures and conditions to reduce the search. Four branch-and-bound procedures were developed and tested. The tests show that the branch-and-bound procedure that uses a node to represent a post partial sequence and includes dominance conditions generally worked the best.

Since the approaches use properties of the two-machine permutation flow shop problem, and the results of the tests show the amount of time required to solve problems to optimality increases quickly, it is unlikely that extending the methods to more than two machines will allow for quickly solving to optimality problems other than those with a small number of jobs. Therefore, future research on heuristic methods that can generate good solutions for larger sized instances would be beneficial.

References

Baker, K.R., Scudder, G.D., 1990. Sequencing with earliness and tardiness penalties: a review. Oper. Res. 38, 22–36.
Chandra, C., Mehta, P., Tirupati, D., 2009. Permutation flow shop scheduling with earliness and tardiness penalties. Int. J. Prod. Res. 47, 5591–5610.
Davis, J.S., Kanet, J.J., 1993. Single-machine scheduling with early and tardy completion costs. Naval Res. Logist. 40, 85–101.
Fernandez-Viagas, V., Dios, M., Framinan, J.M., 2016. Efficient constructive and composite heuristics for the permutation flowshop to minimize total earliness and tardiness. Comput. Oper. Res. 70, 38–68.
Fry, T.D., Armstrong, R.D., Blackstone, J.H., 1987. Minimizing weighted absolute deviation in single machine scheduling. IIE Trans. 9, 445–450.
Hoogeveen, H., 2005. Multicriteria scheduling. Eur. J. Oper. Res. 167, 592–623.
Kanet, J.J., Sridharan, V., 2000. Scheduling with inserted idle time: problem taxonomy and literature review. Oper. Res. 48, 99–110.
Kim, Y.D., Yano, C.A., 1994. Minimizing mean tardiness and earliness in single–machine scheduling problems with unequal due dates. Naval Res. Logist. 41, 913–933.
Madhushini, N., Rajendran, C., Deepa, Y., 2009. Branch-and-bound algorithms for scheduling in permutation flowshops to minimize the sum of weighted flowtime/sum of weighted tardiness/sum of weighted flowtime and weighted tardiness/sum of weighted flowtime, weighted tardiness and weighted earliness of jobs. J. Oper. Res. Soc. 40, 991–1004.
Moslehi, G., Mirzaee, M., Vasei, M., Azaron, A., 2009. Two-machine flow shop scheduling to minimize the sum of maximum earliness and tardiness. Int. J. Prod. Econ. 122, 763–773.
M'Hallah, R., 2014. An iterated local search variable neighborhood descent hybrid heuristic for the total earliness tardiness permutation flow shop. Int. J. Prod. Res. 52 (13), 3802–3819.
Rajendran, C., 1999. Formulations and heuristics for scheduling in a Kanban flowshop to minimize the sum of weighted flowtime, weighted tardiness and weighted earliness of containers. Int. J. Prod. Res. 37, 1137–1158.
Schaller, J.E., 2007. A comparison of lower bounds for the single-machine early tardy problem. Comput. Oper. Res. 34, 2279–2292.

Schaller, J.E., Valente, J.M., 2013. An evaluation of heuristics for scheduling a non-delay permutation flow shop with family setups to minimize total earliness and tardiness. J. Oper. Res. Soc. 64 (6), 805.
Schaller, J.E., Valente, J.M., 2013. A comparison of metaheuristic procedures to schedule jobs in a permutation flow shop to minimise total earliness and tardiness. Int. J. Prod. Res. 51 (3), 772.
Sen, T., Dileepan, P., Gupta, J.N.D., 1989. The two-machine flowshop scheduling problem with total tardiness. Comput. Oper. Res. 16, 333–340.
Szwarc, W., 1993. Adjacent ordering in single-machine scheduling with earliness and tardiness penalties. Naval Res. Logist. 40 (2), 229–244.
Yano, C.A., Kim, Y-D., 1991. Algorithms for a class of single machine weighted tardiness and earliness problems. Eur. J. Oper. Res. 52, 161–178.
Taillard, E., 1993. Benchmarks for basic scheduling problems. Eur. J. Oper. Res. 64, 278–285.
Valente, J.M.S., 2009. Beam search heuristics for the single machine scheduling problem with linear earliness and quadratic tardiness costs. Asia-Pacific J. Oper. Res. 26, 319–339.
Zegordi, S.H., Itoh, K., Enkawa, T., 1995. A knowledgeable simulated annealing scheme for the early/tardy flow shop scheduling problem. Int. J. Prod. Res. 33, 1449–1466.