A new dominance rule for the total weighted tardiness problem

M. SELIM AKTURK and M. BAYRAM YILDIRIM

Abstract. We present a new dominance rule for the single machine total weighted tardiness problem with job dependent penalties. The proposed dominance rule provides a sufficient condition for local optimality. We show that if any sequence violates the dominance rule, then switching the violating jobs either lowers the total weighted tardiness or leaves it unchanged. We also develop a new algorithm based on the dominance rule, which is compared to a number of competing heuristics for a set of randomly generated problems. Our computational results of over 40 000 problems indicate that the proposed algorithm dominates the competing heuristics in all runs.

1. Introduction

The buyer–vendor relationship plays an important role in business. Usually, buyers desire reliable time delivery for meeting their schedules, so the primary objective becomes reducing the amount by which the individual completion times exceed the promised times, i.e. due dates. Jensen *et al.* (1995) emphasize the importance of the marketing/manufacturing interface. They state the fact that firms have a variety of customers, some of which are more important than others. The importance of a customer can depend on a variety of factors, but it is important for manufacturing to reflect these priorities in their scheduling decisions. In addition, in the presence of job tardiness penalties, it may not be enough to measure the shop floor performance by employing unweighted performance measures alone which treat each job in the shop as equally important. In this paper, we prove a new dominance rule as a function of the start time of a job for the single machine total weighted tardiness problem with job dependent penalties.

Lawler (1977) shows that the total weighted tardiness problem, $1 \parallel \sum w_i T_i$, is strongly NP-hard and also gives a

Authors: M. Selim Akturk and M. Bayram Yildirim, Department of Industrial Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey.

M. SELIM AKTURK is Assistant Professor of Industrial Engineering at Bilkent University, Turkey. He holds a Ph.D. in Industrial Engineering from Lehigh University, USA, and B.S.I.E. and M.S.I.E. from Middle East Technical University, Turkey. His current research interests include production scheduling, cellular manufacturing systems and advanced manufacturing technologies. His papers have been published in *Computers & Operations Research*, *European Journal of Operational Research*, *International Journal of Advanced Manufacturing Technology*, *International Journal of Computer Integrated Manufacturing*, *International Journal of Industrial Engineering*, *International Journal of Production Research*, and *Production Planning & Control*. Dr Akturk is a senior member of IIE and member of INFORMS.

M. BAYRAM YILDIRIM is a research assistant in the Department of Industrial Engineering at Bilkent University, Turkey. He holds a B.S.I.E. from Bosphorus University, Turkey, and an M.S.I.E. from Bilkent University. His current research interests include production scheduling and applied optimization.

pseudopolynomial algorithm for the total tardiness problem, $1\|\sum T_i$. Various enumerative solution methods have been proposed for both the weighted and unweighted cases. Emmons (1969) derives several dominance rules that restrict the search for an optimal solution to the $1\|\sum T_i$ problem. Rinnooy Kan *et al.* (1975) extended these results to the weighted tardiness problem. Rachamadugu (1987) identifies a condition characterizing adjacent jobs in an optimal sequence for $1\|\sum w_i T_i$. The exact approaches used in solving the weighted tardiness problem are tested by Abdul-razaq *et al.* (1990), and they use Emmons' dominance rules to form a precedence graph for finding upper and lower bounds.

Szwarc (1993) proves the existence of a special ordering for the single machine earliness–tardiness (E/T) problem with job independent penalties, where the arrangement of two adjacent jobs in an optimal schedule depends on their start time, although the presented results may no longer be valid if the penalties of the E/T model are job dependent, as stated by the author. Szwarc and Liu (1993) present a two-stage decomposition mechanism to the $1\|\sum w_i T_i$ problem when tardiness penalties are proportional to the processing times. As stated by Jensen *et al.* (1995), the importance of a customer can depend on a variety of factors, e.g. the firm's length of relationship with the customer, how frequently they provide business to the firm, and the potential of a customer to provide orders in the future. Therefore, we present a new dominance rule for the most general case of the total weighted tardiness problem. The proposed rule provides sufficient condition for local optimality, and it generates schedules that cannot be improved by adjacent job interchanges.

The implicit enumerative algorithms for the total weighted tardiness problem, e.g. the branch and bound algorithm proposed by Potts and Van Wassenhove (1985), guarantee the optimality, but they require considerable computer resources both in terms of computation times and memory requirements. It is important to note that the number of local minimums is very high because of the nature of the scheduling problems. Currently, even a 50-job case, where the jobs are available at time zero with known weights, due dates and processing times, cannot be solved optimally in a reasonable amount of computation time. Therefore, several heuristics and dispatching rules have been proposed to generate good, but not necessarily optimal, solutions. Two types of methods are widely used for the scheduling problems, which are construction and interchange methods.

Construction techniques use dispatching rules to build a solution by fixing a job in a position at each step. They are fast and highly efficient, but the quality of the solution is not very good. The dispatching rule might be a static one, i.e. time independent like the earliest due date (EDD) rule, or a dynamic one, i.e. time dependent like the apparent tardiness cost (ATC) rule. Vepsalainen and Morton (1987) propose the ATC rule and test efficient dispatching rules for the weighted tardiness problem with specified due dates and delay penalties. Caskey and Storch (1996) tested the ATC rule along with the other dispatching rules in job and flow shops, and showed its effectiveness in minimizing the average tardiness. A more detailed discussion on the heuristics and dispatching rules can be found in Morton and Pentico (1993), and Pinedo (1995).

The interchange methods require an initial sequence. If the change yields a better solution, it is kept, otherwise it is discarded. In descent methods, the change still might be kept if there is no improvement, i.e. the objective value remains the same. The interchanges are continued until a solution that cannot be improved is obtained which is a local minimum. When randomization is applied to an interchange method, the improvements might lead to the global minimum, but this requires a considerable computational effort compared to the dispatching rules and heuristics. Potts and Van Wassenhove (1991) compare several heuristics for the $1\|\sum w_i T_i$ problem ranging from simple dispatching rules to more sophisticated algorithms exploiting problem structure. Their computational results indicate that the pairwise interchange methods perform very well for this problem.

In this study, we also propose an algorithm to demonstrate how the proposed dominance rule can be used to improve a sequence given by a dispatching rule. We show that if any sequence violates the proposed dominance rule, then switching the violating jobs either lowers the total weighted tardiness or leaves it unchanged. The remainder of this paper is organized as follows. In the following section, we discuss the underlying assumptions and give a list of definitions used throughout the paper. We discuss the proposed dominance rule in section 3 along with the transitivity properties. The proposed algorithm is described in section 4. Computational analysis is reported in section 5. Finally, some concluding remarks are provided in section 6.

2. Problem definition and notation

The single machine total weighted tardiness problem, $1\|\sum w_i T_i$, may be stated as follows. A set of jobs (numbered $1,\ldots,n$) is to be processed without interruption on a single machine that can handle only one job at a time. All jobs become available for processing at time zero. Job i has an integer processing time p_i, a due date d_i and a positive weight w_i. For convenience, the jobs are arranged in an EDD indexing convention such that

$d_i < d_j$ or $d_i = d_j$ then $p_i < p_j$, or $d_i = d_j$ and $p_i = p_j$ then $w_i \geq w_j$ for all i and j, such that $i < j$. Furthermore, a weighted tardiness penalty is incurred for each time unit if job i is completed after its due date d_i. The problem can be formally stated as: find a schedule s that minimizes $f(s) = \sum_{i=1}^{n} w_i T_i$. To introduce the dominance rule, consider schedules $s_1 = Q_1 i j Q_2$ and $s_2 = Q_1 j i Q_2$ where Q_1 and Q_2 are two disjoint subsequences of the remaining $n - 2$ jobs. Let $t = \sum_{k \in Q_1} p_k$ be the completion time of Q_1. Define $T_i(t)$ as the total weighted tardiness of job i if scheduled at time t, and let $T_{ij}(t)$ be the total weighted tardiness of jobs i and j if i precedes j and their processing starts at time t. Then, $T_i(t) = w_i \max(0, t + p_i - d_i)$ and $T_{ij}(t) = w_i \max(0, t + p_i - d_i) + w_j \max(0, t + p_i + p_j - d_j)$.

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

$$\Delta_{ij}(t) = T_{ji}(t) - T_{ij}(t) = w_j \max(0, t + p_j - d_j)$$
$$+ w_i \max(0, t + p_i + p_j - d_i)$$
$$- w_i \max(0, t + p_i - d_i)$$
$$- w_j \max(0, t + p_i + p_j - d_j)$$

Note that this cost $\Delta_{ij}(t)$ does not depend on how the jobs are arranged in Q_1 and Q_2, but depends on start time t of the pair, and:

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

Throughout the paper, we also use the following definitions. A 'breakpoint' is a critical start time for each pair of adjacent jobs after which the ordering changes direction such that if $t \leq$ breakpoint then i precedes j (or j precedes i), else j precedes i (or i precedes j). i 'globally' precedes j, $(i \Rightarrow j)$ if it implies the existence of an optimal sequence in which job i precedes job j is guaranteed. i 'unconditionally' precedes j, $(i \rightarrow j)$ if the ordering does not change, i.e. i always precedes j when they are adjacent, but it does not imply that an optimal sequence exists in which i precedes j. i 'conditionally' precedes j, $(i \prec j)$ if there is at least one breakpoint between the pair of jobs, then the order of jobs depends on the start time of this pair and changes in two sides of that breakpoint.

3. Dominance rule

The proposed dominance rule is based on the adjacent pairwise interchange method, which can be used to improve the total weighted tardiness criterion of a given sequence. We will show that if any sequence violates the proposed dominance rule, then switching the violating jobs either lowers the total weighted tardiness or leaves it unchanged. The proposed rule provides a sufficient condition for local optimality, and it generates schedules that cannot be improved by adjacent job interchanges.

After giving the intuition behind the adjacent job interchange, there are 16 cases of p_i versus p_j, w_i versus w_j, $d_j - d_i$ versus p_j and $d_i - p_i$ versus $d_j - p_j$ to be considered, as shown below.

Case	$p_i \leq p_j$	$w_i < w_j$	$d_j - d_i \leq p_j$	$d_i - p_i \leq d_j - p_j$	Section	Ordering
1	Y	Y	Y	Y	(3.1)	Theorem 1
2	Y	Y	Y	N	(3.2)	Theorem 2
3	Y	Y	N	Y	(3.3)	Lemma 1
4	Y	Y	N	N		Inconsistent
5	Y	N	Y	Y	(3.4)	$i \Rightarrow j$
6	Y	N	Y	N	(3.4)	$i \Rightarrow j$
7	Y	N	N	Y	(3.4)	$i \Rightarrow j$
8	Y	N	N	N		Inconsistent
9	N	Y	Y	Y	(3.5)	Theorem 1
10	N	Y	Y	N		Inconsistent
11	N	Y	N	Y	(3.5)	Lemma 3
12	N	Y	N	N		Inconsistent
13	N	N	Y	Y	(3.5)	Lemma 4
14	N	N	Y	N		Inconsistent
15	N	N	N	Y	(3.5)	Lemma 1
16	N	N	N	N		Inconsistent

Cases 4 and 8 are ruled out, since $d_i \leq d_j$, $p_i \leq p_j$ and $d_j - d_i > p_j$, then inequalities $d_i - p_i < d_j - p_j$ and $d_i - p_i > d_j - p_j$ are inconsistent. Cases 10, 12, 14 and 16 are also ruled out, since $d_i \leq d_j$ and $p_i > p_j$, then inequalities $d_i - p_i < d_j - p_j$ and $d_i - p_i > d_j - p_j$ are inconsistent. When we analyse the interchange function $\Delta_{ij}(t)$ for each possible case, it can be seen that there are at most three possible breakpoints as shown below.

$$t_{ij}^1 = \frac{w_i d_i - w_j d_i}{w_i - w_j} - (p_i + p_j) \tag{1}$$

$$t_{ij}^2 = d_j - p_i - p_j(1 - w_i/w_j) \tag{2}$$

$$t_{ij}^3 = d_i - p_j - p_i(1 - w_j/w_i) \tag{3}$$

The breakpoints will be valid if they are in their specified intervals as follows:

- t_{ij}^1 will be valid if $d_j - (p_i + p_j) \leq t_{ij}^1 \leq d_i - p_i$;
- t_{ij}^2 will be valid if $\max\{d_i - p_i, d_j - (p_i + p_j)\} \leq t_{ij}^2 < d_j - p_j$;
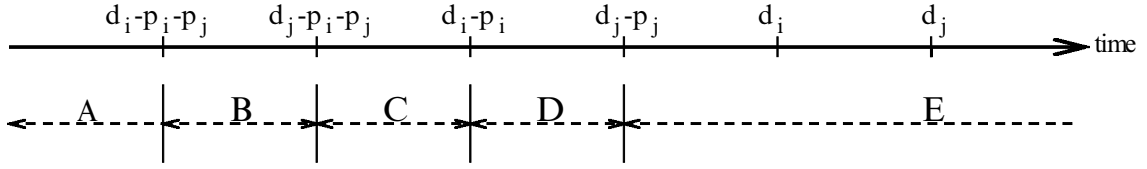- t_{ij}^3 will be valid if $d_j - p_j \leq t_{ij}^3 < d_i - p_i$.

Figure 1. $d_i \leq d_j$, $p_i \leq p_j$, $d_j - d_i \leq p_j$, $d_i - p_i \leq d_j - p_j$.

3.1. $d_i \leq d_j$, $p_i \leq p_j$, $w_i < w_j$, $d_j - d_i \leq p_j$, $d_i - p_i \leq d_j - p_j$

We will first investigate how $\Delta_{ij}(t)$ changes in this case. As can be seen from figure 1, there are five regions to examine:

(1) In region A $[t \leq d_i - (p_i + p_j)]$ no tardiness occurs, so it is indifferent to schedule either i or j first.

(2) In region B $[d_i - (p_i + p_j) < t < d_j - (p_i + p_j)]$, i is tardy if not scheduled first. Here, $\Delta_{ij}(t) = w_i(t + p_i + p_j - d_i)$. Since $d_i - (p_i + p_j) < t$, $\Delta_{ij}(t) > 0$, so $i \prec j$.

(3) In region C $[d_j - (p_i + p_j) \leq t \leq d_i - p_i]$, either i or j is tardy if not scheduled first. Here, $\Delta_{ij}(t) = (w_i - w_j)t + (p_i + p_j)(w_i - w_j) - w_i d_i + w_j d_j$. The breakpoint $t_{ij}^1 = (w_i d_i - w_j d_j / w_i - w_j - (p_i + p_j))$ is defined in this region. If the processing of this pair starts up to $t \leq t_{ij}^1$ then $i \prec j$, and $j \prec i$ if the processing begins after t_{ij}^1.

(4) In region D $(d_i - p_i < t < d_j - p_j)$, i is always tardy but j is not tardy if scheduled first. Here, $\Delta_{ij}(t) = w_i p_j - w_j t - w_j(p_i + p_j - d_j)$. There is a new breakpoint $t_{ij}^2 = d_j - p_i - p_j(1 - w_i/w_j)$. Similar to above, $i \prec j$ for $t \leq t_{ij}^2$, and $j \prec i$ afterwards.

(5) In region E $(d_j - p_j \leq t)$ both jobs are tardy. Here, $\Delta_{ij}(t) = w_i p_j - w_j p_i$. If $\Delta_{ij}(t) \geq 0$ then $i \prec j$, otherwise $j \prec i$.

It seems that there is more than one breakpoint, but actually, as we are going to show below, at most one breakpoint can be valid, i.e. it is in the specified region of either C or D for t_{ij}^1 and t_{ij}^2, respectively, and up to that breakpoint $i \prec j$ and then $j \prec i$. If there is no valid breakpoint then i unconditionally precedes j, i.e. $i \rightarrow j$.

Proposition 1. If there is a valid breakpoint then it is $\min\{t_{ij}^1, t_{ij}^2\}$.

Proof: The cost functions $T_{ij}(t)$ and $T_{ji}(t)$ are continuous piecewise linear functions, while the breakpoints occur at the times where a job becomes tardy. The gradient of $T_{ij}(t)$ is equal to w_j from $d_j - p_i - p_j$ to $d_i - p_i$, after which it becomes equal to $w_i + w_j$. The gradient of $T_{ji}(t)$ is equal to w_i from $d_i - p_i - p_j$ to $d_j - p_j$, after which it becomes equal to $w_i + w_j$. Hence, if the functions inter-

sect then they will not intersect again in this case, as the gradient of $T_{ij}(t)$ is at least as large as the gradient of $T_{ji}(t)$ after this intersection point. $\square$

The following proposition is useful when both of the breakpoints are invalid.

Proposition 2. If t_{ij}^1 and t_{ij}^2 are both invalid then $i \rightarrow j$.

Proof: If we can show that $i \prec j$ in all regions, then $i \rightarrow j$. We know that if both of the breakpoints are invalid then $t_{ij}^1 > t_{ij}^2 > d_j - p_j$. In region B, we know that $\Delta_{ij}(t) > 0$. So $i \prec j$. In region C, $t_{ij}^1 > d_i - p_i$ since t_{ij}^1 is invalid, hence $d_j = d_i + p_j(1 - w_i/w_j) - \varepsilon^1$ for

$$\varepsilon^1 > 0 \Rightarrow \Delta_{ij}(t) = (w_i - w_j)t + (p_i + p_j)(w_i - w_j) - w_i d_i$$
$$+ w_j(d_i + p_j(1 - w_i/w_j) - \varepsilon^1) \Rightarrow \Delta_{ij}(t) = (w_i - w_j)$$
$$\times (t - (d_i - p_i)) + w_i \varepsilon^1.$$

Since $(w_i - w_j) < 0$, $(t - (d_i - p_i)) \leq 0$ and $\varepsilon^1 > 0$ then $\Delta_{ij}(t) > 0$. So $i \prec j$. In region D, since t_{ij}^1 and t_{ij}^2 are both invalid, $t_{ij}^2 > d_j - p_j$, hence $p_j = p_i(w_j/w_i) + \varepsilon^2$ for $\varepsilon^2 > 0$.

$$\Rightarrow \Delta_{ij}(t) = w_i(p_i(w_j/w_i) + \varepsilon^2) - w_j t - w_j(p_i + p_j - d_j)$$
$$\Rightarrow \Delta_{ij}(t) = w_i \varepsilon^2 - w_j(t - (d_j - p_j)).$$

Since $(t - (d_j - p_j)) < 0$, then $\Delta_{ij}(t) > 0$. So $i \prec j$. Finally, in region E, $t_{ij}^2 > d_j - p_j \Rightarrow \Delta_{ij}(t) > 0$. So $i \prec j$. The result follows, if both t_{ij}^1 and t_{ij}^2 are invalid, then $i \rightarrow j$. $\square$

Theorem 1. If $d_i \leq d_j$, $w_i < w_j$, $d_j - d_i \leq p_j$ and $d_i - p_i \leq d_j - p_j$, then there can be at most one valid breakpoint, such that $i \prec j$ for $t \leq \min\{t_{ij}^1, t_{ij}^2\}$ and $j \prec i$ afterwards, and if they are both invalid then $i \rightarrow j$.

Proof: Follows from Propositions 1 and 2. $\square$

3.2. $d_i \leq d_j$, $p_i \leq p_j$, $w_i < w_j$, $d_j - d_i \leq p_j$, $d_i - p_i > d_j - p_j$

We will start again by investigating how $\Delta_{ij}(t)$ changes in the second case. As can be seen from figure 2, there are five regions to examine, but regions A, B and E are identical to the previous case in Section 3.1, so only regions C and D are examined below.
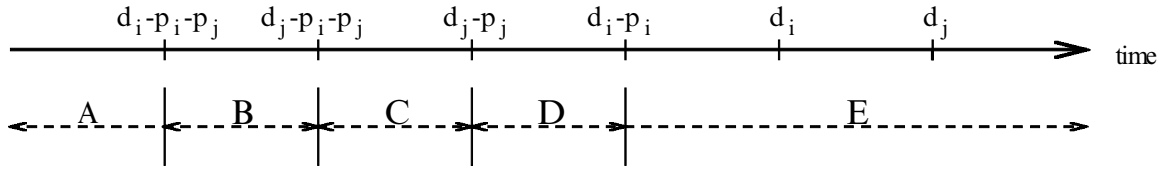
Figure 2. $d_i \leq d_j$, $p_i \leq p_j$, $d_j - d_i \leq p_j$ and $d_i - p_i > d_j - p_j$.
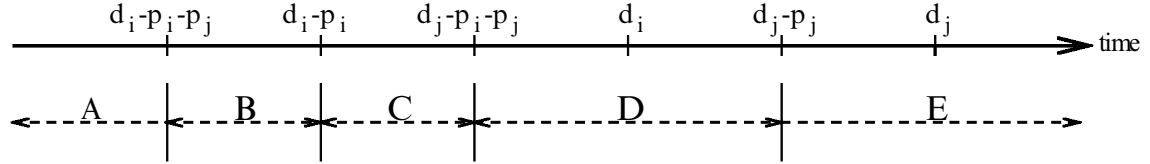
Figure 3. $d_i \leq d_j$, $p_i \leq p_j$, $d_j - d_i > p_j$ and $d_i - p_i \leq d_j - p_j$.

(1) In region C $\left(d_j - (p_i + p_j) \leq t < d_j - p_j\right)$, $\Delta_{ij}(t) = (w_i - w_j)t + (p_i + p_j)(w_i - w_j) - w_i d_i + w_j d_j$. Similar to the previous case, we have the breakpoint t_{ij}^1 in this region. Therefore, $i \prec j$ for $t \leq t_{ij}^1$, and $j \prec i$ afterwards.

(2) In region D $\left(d_j - p_j \leq t < d_i - p_i\right)$, $\Delta_{ij}(t) = -w_j p_i + w_i(t + p_i + p_j - d_i)$. There is a new breakpoint $t_{ij}^3 = d_i - p_j - p_i(1 - w_j/w_i)$. If it is a valid breakpoint, then $j \prec i$ for $t \leq t_{ij}^3$, and $i \prec j$ afterwards.

As shown above, there can be two breakpoints for this case. The following proposition shows how $\Delta_{ij}(t)$ changes if both of the breakpoints are valid.

Proposition 3. If both breakpoints are valid, then $i \prec j$ for $t \leq t_{ij}^1$, $j \prec i$ for $t_{ij}^1 < t \leq t_{ij}^3$, and $i \prec j$ for $t_{ij}^3 < t$.

Proof: The only region that should be examined is region E, since $\Delta_{ij}(t)$ remains the same in other regions. If $t_{ij}^3 < d_i - p_i$, then from equation (3) we know that $p_i w_j < p_j w_i$. Hence, $\Delta_{ij}(t) = w_i p_j - w_j p_i > 0$, so $i \prec j$. □
Furthermore, if t_{ij}^3 is valid, then t_{ij}^1 is also valid. On the other hand, t_{ij}^3 could be invalid, i.e. $t_{ij}^3 > d_i - p_i$, when t_{ij}^1 is valid. Therefore, Proposition 4 demonstrates how $\Delta_{ij}(t)$ changes if both of the breakpoints are invalid, or t_{ij}^1 is the only valid breakpoint. Due to space limitations, we will not include the following proofs but they can be obtained from the first author.

Proposition 4. If t_{ij}^1 is valid and t_{ij}^3 is invalid, i.e. $t_{ij}^3 > d_i - p_i$, then $i \prec j$ for $t \leq t_{ij}^1$ and $j \prec i$ afterwards. If t_{ij}^1 is invalid, then $i \rightarrow j$.

Theorem 2. If $d_i \leq d_j$, $p_i \leq p_j$, $w_i < w_j$, $d_j - d_i \leq p_j$ and $d_i - p_i > d_j - p_j$, then there can be at most two valid breakpoints, which are t_{ij}^1 and t_{ij}^3. If both are valid, then $i \prec j$ for $t \leq t_{ij}^1$, $j \prec i$ for $t_{ij}^1 < t \leq t_{ij}^3$, and $i \prec j$ afterwards.

If t_{ij}^1 is the only valid breakpoint, then $i \prec j$ for $t \leq t_{ij}^1$, and $j \prec i$ afterwards. If there is no valid breakpoint, then always $i \rightarrow j$.

Proof: Follows from Propositions 3 and 4.   □

3.3. $d_i \leq d_j$, $p_i \leq p_j$, $w_i < w_j$, $d_j - d_i > p_j$, $d_i - p_i \leq d_j - p_j$

First, let us investigate how $\Delta_{ij}(t)$ changes. As can be seen from figure 3, there are five regions to examine, but regions A and E are similar to regions A and E of the case in section 3.1. Furthermore, there might be only one valid breakpoint, t_{ij}^2, in region D as stated below, since $i \prec j$ in regions B and C.

Lemma 1. If $d_i \leq d_j$, $d_j - d_i > p_j$ and $d_i - p_i \leq d_j - p_j$. then the only breakpoint that can be valid is t_{ij}^2. If it is valid, then $i \prec j$ for $t \leq t_{ij}^2$, and $j \prec i$ afterwards. If there is no valid breakpoint then always $i \rightarrow j$.

3.4. $d_i \leq d_j$, $p_i \leq p_j$, $w_i \geq w_j$

In cases 5–7, it has already been proved by Rinnooy Kan *et al.* (1975) based on the Emmons' theorem that there is a global dominance property between the pair of jobs i and j as stated below.

Lemma 2. If $d_i \leq d_j$, $p_i \leq p_j$ and $w_i \geq w_j$ then job i globally precedes job j, i.e. $i \Rightarrow j$.

3.5. $d_i \leq d_j$, $p_i > p_j$, $d_i - p_i \leq d_j - p_j$

Cases 9, 11, 13 and 15 are similar to the previous cases discussed earlier. For example, in case 9, there can be at most one valid breakpoint, as shown in section 3.1, and

up to that breakpoint $i \prec j$ then $j \prec i$. If there is no valid breakpoint, then $i \to j$. Therefore, Theorem 1 is also applicable for case 9. On the other hand, case 11 is similar to case 3 (figure 3), although the range of breakpoint $t_{ij}^2 = d_j - p_i - p_j(1 - w_i/w_j)$ is between $d_j - (p_i + p_j) < t_{ij}^2 < d_j - p_j$, therefore t_{ij}^2 is always valid.

Lemma 3. If $d_i \leq d_j$, $p_i > p_j$, $w_i < w_j$, $d_j - d_i < p_j$ and $d_i - p_i \leq d_j - p_j$, then the breakpoint t_{ij}^2 is always valid, hence $i \prec j$ for $t \leq t_{ij}^2$, and $j \prec i$ afterwards.

Case 13 is also similar to case 1 (figure 1), therefore in regions C and D, t_{ij}^1 and t_{ij}^2 could be valid breakpoints, respectively. But $w_i \geq w_j$, hence t_{ij}^1 is always invalid.

Lemma 4. If $d_i \leq d_j$, $p_i > p_j$, $w_i \geq w_j$, $d_j - d_i \leq p_j$ and $d_i - p_i \leq d_j - p_j$, then only one breakpoint can be valid which is t_{ij}^2. If it is valid, then for $t < t_{ij}^2$, $i \prec j$ and then $j \prec i$.

Case 15 is similar to case 3, and t_{ij}^2 can be a valid breakpoint for this case as discussed earlier. Therefore, Lemma 1 given in section 3.3 is also applicable here.

After analysing all of the possible cases, we prove that there are certain time points, called breakpoints, in which the ordering might change for adjacent jobs. We find three such breakpoints and show that at most one breakpoint can be valid, except in case 2, for which both t_{ij}^1 and t_{ij}^2 can be valid at the same time. For cases 1 and 9, if there is a valid breakpoint then it is min $\{t_{ij}^1, t_{ij}^2\}$. For cases 3, 13 and 15, only one breakpoint can be valid, that is t_{ij}^2. On the other hand, t_{ij}^2 is always valid for case 11. In cases 5–7, there is a global dominance, $i \Rightarrow j$, as shown by Rinnooy Kan *et al.* (1975). Finally, if there is no valid breakpoint, then always $i \to j$.

3.6. *Transitivity*

The transitivity property is very crucial for reducing the number of sequences that have to be considered in an implicit enumeration technique. Szwarc (1993) shows that there is a transitivity property for the $1 \parallel \sum T_i$ problem. The transitivity property does not hold for the $1 \parallel \sum w_i T_i$ problem even for the assumption that the weights are proportional to the processing times, as shown by Szwarc and Liu (1993). Let J denote the set of all jobs, V be the set of pairs (i, j) for which there is a valid breakpoint t_{ij}^{valid} between i and j, and t_l be the last valid breakpoint for any pair of jobs i, j on the time scale, such that $t_l = \max_{(i,j) \in V} \{t_{ij}^{\text{valid}}\}$. Therefore, we will show that the transitivity property holds for the proposed dominance rule when $t \geq t_l$, which can be used quite effec-

tively to find an optimal sequence for the remaining jobs on hand after a time point t_l.

Lemma 5. If $t \geq t_l$, then the weighted shortest processing time (WSPT) rule gives an optimal sequence for the remaining unscheduled jobs.

Proof: We have already shown that for every job pair (i, j), one of these conditions must hold, either there is a valid breakpoint or unconditional ordering $(i \to j)$ or globally precedence $(i \Rightarrow j)$. The WSPT rule holds for both $i \to j$ and $i \Rightarrow j$. If there is a valid breakpoint, then for $t > t_{ij}$ the job having higher w_i/p_i is scheduled first, so WSPT again holds. For $t > t_l$, consider a job i which conflicts with the WSPT rule, then we can have a better schedule by making adjacent job interchanges which either lower the total weighted tardiness value or leave it unchanged. If we do the same thing for all of the remaining jobs, we get the WSPT sequence. □

It is a well-known result that the WSPT rule gives an optimal sequence for the $1 \parallel \sum w_i T_i$ problem when either all due dates are zero or all jobs are tardy, i.e. $t > \max_{i \in J} \{d_i - p_i\}$. The problem reduces to a total weighted completion time problem which is known to be solved optimally by the WSPT rule, in which jobs are sequenced in non increasing order of w_i/p_i. We know that $t_l \leq \max_{i \in J} \{d_i - p_i\}$, so we enlarge the region for which the total weighted tardiness problem can be solved optimally by the WSPT rule.

4. Algorithm

The $1 \parallel \sum w_i T_i$ problem is strongly NP-hard as stated earlier, hence it is important to develop a heuristic that provides a reasonably good schedule with reasonable computational effort. The ATC rule is a composite dispatching rule that combines the WSPT rule and the minimum slack rule. Under the ATC rule, jobs are scheduled one at a time; i.e. every time the machine becomes free, a ranking index is computed for each remaining job i. The job with the highest ranking index is then selected to be processed next. The ranking index is a function of the time t at which the machine became free, as well as the p_i, w_i and d_i of the remaining jobs. The index is defined as:

$$\pi_i(t) = \frac{w_i}{p_i} \cdot \exp\left(-\max\left(0, d_i - t - p_i\right)/(k \cdot \bar{p})\right)$$

where we set the look-ahead parameter k to 2, as suggested in Morton and Pentico (1993), and $\bar{p}$ is the average processing time of the remaining unscheduled jobs. Vepsalainen and Morton (1987) have shown that the
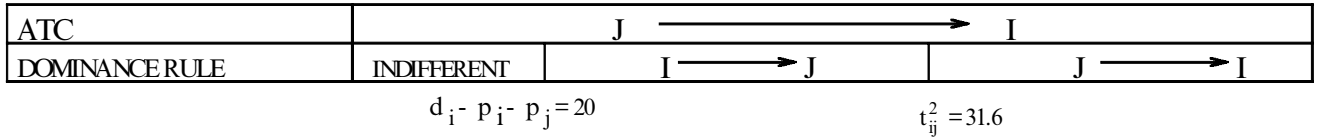
ATC		J $\xrightarrow{\hspace{2cm}}$ I	
DOMINANCE RULE	INDIFFERENT	I $\xrightarrow{\hspace{1cm}}$ J	J $\xrightarrow{\hspace{1cm}}$ I

$$d_i - p_i - p_j = 20 \qquad\qquad t_{ij}^2 = 31.6$$

Figure 4. Dominance properties.

Table 1. Two-job example.

Jobs	d_i	p_i	w_i
i	30	6	4
j	40	4	10

ATC rule is superior to other sequencing heuristics and close to optimal for the $1 \parallel \sum w_i T_i$ problem. In the following example, we will demonstrate how the proposed dominance rule can be used to improve the weighted tardiness criterion even for an efficient rule like ATC. Consider the two-job problem given in table 1. Since $\bar{p} = 5$, the following ranking indexes can be calculated for each job:

$$\pi_i(t) = \tfrac{4}{6}\exp\left(-(\max(0, 30 - t - 6))/(2*5)\right) < \pi_j(t)$$
$$= \tfrac{10}{4}\exp\left(-(\max(0, 40 - t - 4))/(2*5)\right)$$

Therefore, we can easily show that $j \Rightarrow i$ for all t under the ATC rule. But the dominance rule indicates that t_{ij}^2 is a valid breakpoint for this pair, and $i \prec j$ for $20 < t \leq 31.6$, as shown in figure 4. For example, if we set $t = 25$ then $T_{ij}(25) = 4$ and $T_{ji}(25) = 20$, therefore i should precede j if their processing starts in the time interval $[20, 31.6]$.

Now we will present an algorithm based upon the dominance rule that can be used to improve the total weighted tardiness criterion of any sequence s by making necessary interchanges. We have already shown that if any sequence violates the proposed dominance rule, then switching the violating jobs either lowers the total weighted tardiness or leaves it unchanged. The proposed heuristic takes into account all of the global, unconditional and conditional precedence relationships. Let $seq[i]$ denote the index of the job in the i-th position in a given sequence s. The algorithm can be summarized as follows:

For $i = 1$ to $n - 1$ do
     For $j = i + 1$ to n do
         If i globally precedes j (or $j \Rightarrow i$)
         and $seq[i] > seq[j]$ (or $seq[i] <$
         $seq[j]$) then change the order-
         ings of i and j.
Set $k = 1$ and $t = 0$.
While $k \leq n - 1$ do begin

Set $i = seq[k]$ and $j = seq[k + 1]$
     If $i < j$ then
         If t_{ij}^3 is valid, $d_j - p_i - p_j < t$ and $t_{ij}^1 < t \leq t_{ij}^3$
         then
            $t = t - p_{seq[k-1]}$, change the orderings of i
            and j and $k = k - 1$
         else if either t_{ij}^1 or t_{ij}^2 is valid and $t > t_{ij}^{\text{valid}}$ then
            $t = t - p_{seq[k-1]}$, change the orderings of i
            and j and $k = k - 1$
         else $t = t + p_i$ and $k = k + 1$.
     If $i > j$ then
         If t_{ji}^3 is valid, $d_j - p_i - p_j < t$ and either
         $t < t_{ji}^1$ or $t > t_{ji}^3$ then
            $t = t - p_{seq[k-1]}$, change the orderings of i and j
            and $k = k - 1$
         else if either t_{ji}^1 or t_{ji}^2 is valid and $t \leq t_{ji}^{\text{valid}}$ then
            $t = t - p_{seq[k-1]}$, change the orderings of i and j
            and $k = k - 1$
         else $t = t + p_i$ and $k = k + 1$
end.

Let us consider the following 20-job example to explain the proposed algorithm. The jobs are initially scheduled by the ATC rule, which is given in figure 5, along with the sequence, s, due date, d_i, processing time, p_i, weight, w_i, starting time, t, and weighted tardiness, WT, of each job i. The final schedule after implementing the proposed algorithm on the schedule given by the ATC rule is also given in figure 5. In the matrix of breakpoints, the following notation is used: the numbers in cells correspond to the valid breakpoints, the global precedences ($\Rightarrow$) and unconditional precedences ($\rightarrow$).

The algorithm works as follows: the global precedences are tested first, then the unconditional and conditional precedence relations are examined. Up to $t = 27$, the sequence generated by the ATC rule does not conflict with the dominance rule. But job 6 in $seq[5]$ violates the dominance rule when compared to job 2 in $seq[6]$ at time $t = 27$. From figure 5, $t_{2,6}$ is 35, which is greater than $t = 27$, that means $2 \prec 6$ at time $t = 27$, so an interchange should be made. As a result, t is set to $27 - p_{seq[4]} = 17$ and $k = k - 1 = 4$. So we can check the dominance rule between the jobs at $seq[k]$ and $seq[k + 1]$, i.e. jobs 1 and 2. A similar interchange is made at $t = 43$ between jobs 7 and 9, and we proceed on. Notice that, after all necessary interchanges performed on the sequence generated by the ATC rule,

ATC RULE

S	JOBS	di	Pi	Wi	t	WT
1	8	53	1	6	0	0
2	3	44	9	10	1	0
3	4	47	7	10	10	0
4	1	39	10	6	17	0
5	6	47	8	10	27	0
6	2	43	8	5	35	0
7	9	57	6	4	43	0
8	7	51	5	2	49	6
9	10	60	3	2	54	0
10	11	69	3	4	57	0
11	12	70	10	7	60	0
12	15	105	2	9	70	0
13	14	96	8	6	72	0
14	13	88	10	3	80	6
15	16	117	1	3	90	0
16	19	138	2	9	91	0
17	17	121	9	10	93	0
18	20	138	2	3	102	0
19	18	127	9	6	104	0
20	5	47	7	1	113	73
		Total Weighted Tardiness				85
		Number Of Tardy Jobs				3

DOMINANCE RULE

S	JOBS	di	Pi	Wi	t	WT
1	8	53	1	6	0	0
2	3	44	9	10	1	0
3	4	47	7	10	10	0
4	1	39	10	6	17	0
5	2	43	8	5	27	0
6	6	47	8	10	35	0
7	7	51	5	2	43	0
8	9	57	6	4	48	0
9	10	60	3	2	54	0
10	11	69	3	4	57	0
11	12	70	10	7	60	0
12	5	47	7	1	70	30
13	13	88	10	3	77	0
14	14	96	8	6	87	0
15	15	105	2	9	95	0
16	16	117	1	3	97	0
17	19	138	2	9	98	0
18	17	121	9	10	100	0
19	20	138	2	3	109	0
20	18	127	9	6	111	0
		Total Weighted Tardiness				30
		Number Of Tardy Jobs				1

BREAKEVEN POINTS

JOBS	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	34.60	30.40	34.20	→	33.80	→	43.00	50.00	56.00	60.50	58.57	→	86.00	94.33	108.00	107.40	117.00	127.33	130.00
2		28.00	35.50	→	35.00	→	44.83	50.50	56.50	61.75	59.14	⇒	86.67	96.11	109.67	108.50	117.50	129.11	131.33
3			38.00	→	38.00	→	44.67	→	→	64.50	⇒	⇒	→	96.22	110.33	→	→	129.22	133.67
4				⇒	⇒	→	46.67	→	→	→	⇒	⇒	⇒	98.22	112.33	⇒	⇒	131.22	135.67
5					→	41.50	45.17	45.50	51.50	59.75	54.43	74.33	82.33	96.22	109.33	105.90	112.50	129.22	129.67
6						→	45.67	→	→	65.50	⇒	→	97.22	111.33	⇒	⇒	130.22	134.67	
7							47.33	49.00	55.00	62.50	57.86	→	85.67	98.44	111.67	108.80	116.00	131.44	132.33
8								⇒	⇒	→	⇒	⇒	→	→	→	→	→	→	→
9									→	63.00	59.71	⇒	87.33	97.89	111.33	109.60	→	130.89	132.67
10										64.50	59.86	→	87.67	100.44	113.67	110.80	→	133.44	134.33
11											58.33	⇒	→	100.89	114.33	→	→	133.89	135.67
12												→	87.33	94.56	108.33	108.30	→	127.56	130.67
13													82.00	93.67	107.00	104.70	112.50	126.67	128.00
14														96.33	110.00	109.40	⇒	129.33	132.00
15															→	→	⇒	→	→
16																→	→	135.67	⇒
17																	→	129.22	133.67
18																		128.33	131.00
19																			⇒

Figure 5. A problem of $n = 20$.

Table 2. Experimental design.

Factors	Number of levels	Settings
Number of jobs	4	50, 100, 300, 500
Processing time variability	2	[1,10], [1, 100]
Weight variability	2	[1,10], [1, 100]
Relative range of due dates	5	0.1, 0.3, 0.5, 0.7, 0.9
Average tardiness factor	5	0.1, 0.3, 0.5, 0.7, 0.9

the total weighted tardiness dropped from 85 to 30, giving an improvement of $(85 - 30)/85 = 68\%$. For this example, the optimum solution is also equal to 30.

5. Computational results

We tested the proposed algorithm on a set of randomly generated problems on a Sun Ultra Sparc 1 workstation using Sun Pascal. The algorithm along with a number of heuristics were tested on problems with 50, 100, 300 and 500 jobs that were generated as follows. For each job i, an integer processing time p_i and integer weight w_i were generated from two uniform distributions [1, 10] and [1, 100] to create low or high variation, respectively. The relative range of due dates, RDD and average tardiness factor, TF were selected from the set $\{0.1, 0.3, 0.5, 0.7, 0.9\}$. An integer due date d_i from the uniform distribution $[P(1 - TF - RDD/2), P(1 - TF + RDD/2)]$ was generated for each job i, where P is the total processing time, $\sum_{i=1}^{n} p_i$. As summarized in table 2, a total of 400 example sets was considered and 100 replications were taken for each combination resulting in 40 000 randomly generated runs.

In order to show the efficiency of the proposed algorithm, a number of heuristics was implemented on the same problem sets. These dispatching rules and their priority indexes are summarized in table 3. The EDD, LPT, SPT, WSPT and WPD are examples of static dispatching rules, whereas ATC and COVERT are dynamic ones. The proposed algorithm can be implemented in two ways, i.e. a forward or backward procedure. In a forward procedure, we start from the first job of the given sequence and proceed on, as outlined in section 4. In a backward procedure, the only difference is that we start from the last job of the given sequence and proceed backwards towards the first job. Vepsalainen and Morton (1987) have shown that the ATC rule is superior to the other rules, therefore we tested both the forward and backward procedures on the ATC rule, as denoted by ATC(I) and ATC(II), respectively.

The results, which are averaged over 10 000 runs for each heuristic, are tabulated in tables 4–7 for 50-, 100-, 300- and 500-job cases. For each heuristic, the average weighted tardiness before and after implementing the proposed algorithm along with the average improvement, (improv), the average real time in centiseconds used for the heuristic and algorithm, and the average number of interchanges, (interch), are summarized. Finally, we performed a paired t-test for each run, and t-test values are reported in the last column. Although the real time depended on the utilization of the system when the measurements were taken, it was a good indicator for the computational requirements, since the cpu times were so small that we could not measure them accurately. In general, the actual cpu time is considerably smaller than the real time. The average improvement for each run is found as follows: $\text{Improv} = F(s^h) - F(s^{DR})/F(s^h) \times 100$, if $F(s^h) \neq 0$, and zero otherwise, where $F(s^h)$ is the total

Table 3. Dispatching rules.

Rule	Definition	Rank and priority index
ATC	Apparent tardiness cost	$\max\left(\dfrac{w_i}{p_i}\exp\left(-\dfrac{\max(0, d_i - t - p_i)}{k\bar{p}}\right)\right)$
COVERT	Weighted cost over time	$\max\left(\dfrac{w_i}{p_i}\max\left[0, 1 - \dfrac{\max(0, d_i - t - p_i)}{\left(\frac{w_i}{p_i}\right)k p_i}\right]\right)$
WPD	Weighted processing due date	$\max\left(\dfrac{w_i}{p_i d_i}\right)$
EDD	Earliest due date	$\min(d_i)$
WSPT	Weighted shortest processing time	$\max\left(\dfrac{w_i}{p_i}\right)$
SPT	Shortest processing time	$\min(p_i)$
LPT	Longest processing time	$\max(p_i)$

Table 4. Computational results for $n = 50$.

$N = 50$ Heuristic	Average tardiness			Real time			T-test value
	Before	After	Improv	Before	After	Interch	
ATC(I)	119 721.16	118 570.82	6.38	0.42	1.08	6.70	41.93
ATC(II)	119 721.16	118 570.96	6.38	0.42	1.08	6.69	41.92
COVERT	127 375.29	125 862.04	4.02	0.20	1.13	2.48	26.43
WPD	192 907.64	134 280.01	35.32	0.10	1.12	37.58	31.14
EDD	275 662.42	128 034.52	39.65	0.08	1.35	179.29	44.85
WSPT	153 793.84	135 811.00	24.09	0.12	1.12	27.21	45.32
SPT	225 345.92	213 221.45	10.91	0.17	1.11	20.53	52.77
LPT	538 416.80	153 283.34	81.04	0.12	1.23	64.57	61.53

Table 5. Computational results for $n = 100$.

$N = 100$ Heuristic	Average tardiness			Real time			T-test value
	Before	After	Improv	Before	After	Interch	
ATC(I)	468 101.05	465 699.16	2.96	1.62	4.42	13.71	44.42
ATC(II)	468 101.05	465 700.18	2.96	1.62	4.42	13.70	44.41
COVERT	493 596.90	490 617.53	2.04	0.84	4.41	4.91	22.93
WPD	773 667.22	527 598.81	33.38	0.26	4.51	126.42	31.54
EDD	1 096 719.34	495 660.43	41.07	0.04	5.35	728.29	45.14
WSPT	602 841.83	546 157.72	18.41	0.25	4.43	69.44	45.16
SPT	883 531.62	851 058.31	8.52	0.24	4.43	58.37	58.23
LPT	2 179 428.23	608 737.47	81.78	0.24	4.77	222.14	62.38

Table 6. Computational results for $n = 300$.

$N = 300$ Heuristic	Average tardiness			Real time			T-test value
	Before	After	Improv	Before	After	Interch	
ATC(I)	4 231 923.39	4 223 672.96	0.58	14.66	40.47	44.18	24.49
ATC(II)	4 231 923.39	4 223 673.24	0.58	14.66	40.47	44.09	24.49
COVERT	4 318 815.23	4 310 217.00	0.67	7.03	40.35	15.45	10.83
WPD	6 867 607.28	4 729 276.97	27.86	0.62	41.53	880.80	15.39
EDD	9 620 873.07	4 237 060.61	43.04	0.14	50.56	6504.16	22.59
WSPT	5 337 069.51	5 084 352.83	11.01	0.61	40.65	248.49	22.26
SPT	8 009 572.24	7 862 767.57	6.29	0.61	40.82	366.91	36.40
LPT	19 476 678.41	5 926 122.98	80.68	0.61	42.57	1394.11	31.64

Table 7. Computational results for $n = 500$.

$N = 500$ Heuristic	Average tardiness			Real time			T-test value
	Before	After	Improv	Before	After	Interch	
ATC(I)	11 907 061.07	11 893 486.77	0.39	40.31	111.54	74.72	24.85
ATC(II)	11 907 061.07	11 893 488.52	0.39	40.31	111.56	74.68	24.85
COVERT	12 089 858.58	12 075 066.33	0.48	18.93	111.51	24.70	10.51
WPD	19 234 993.63	13 439 006.41	25.64	1.17	114.37	2187.09	15.57
EDD	26 798 871.59	12 104 873.48	43.09	0.13	138.04	17 781.58	22.67
WSPT	15 029 863.93	14 580 342.57	7.87	1.15	111.9	436.73	23.22
SPT	21 986 707.78	21 656 919.94	5.72	1.17	112.67	960.38	38.97
LPT	54 596 748.72	17 400 919.52	79.91	1.20	116.24	3074.19	31.82

Table 8. The effect of the new dominance rule on the ATC rule.

Criterion	$n = 50$		$n = 100$		$n = 300$		$n = 500$	
	<	=	<	=	<	=	<	=
$\sum w_i T_i$	7043	2957	7150	2850	7400	2600	7390	2610

Table 9. Comparison of the ATC rule with the proposed rule for $n = 100$.

RDD value	Criterion	Tardiness factor				
		0.1	0.3	0.5	0.7	0.9
0.1	<	231	219	318	380	399
	=	169	181	82	20	1
0.3	<	54	322	398	400	400
	=	346	78	2	0	0
0.5	<	0	322	400	400	400
	=	400	78	0	0	0
0.7	<	0	97	400	400	400
	=	400	303	0	0	0
0.9	<	0	10	400	400	400
	=	400	390	0	0	0

weighted tardiness value obtained by the heuristic and $F(s^{DR})$ is the total weighted tardiness obtained by the algorithm, which takes the sequence generated by the heuristic as an input. Since there is no significant difference between ATC(I) and ATC(II), we only implement the forward procedure for the other rules.

The results of our large scale computational experiments reported in tables 4–7 are consistent with those found by Vepsalainen and Morton (1987). Among the competing rules, the ATC rule performs better than others, and the weighted COVERT is overall second. The EDD and SPT rules perform poorly since they do not take into account the individual job weights. Furthermore, the large t-test values on the average improvement indicate that the proposed algorithm provides a significant improvement on all rules, and the amount of improvement is notable at 99.5% confidence level for all heuristics. When we analyse the individual heuristics, the ATC rule is the best-known dispatching rule, but we can still perform 6.7 pairwise interchanges on average and improve the results by 6.38% for the 50-jobs case. On the other hand, the average number of interchanges increases to 179.29 for the EDD rule with a 39.65% improvement. The number of interchanges for EDD is substantially higher than the others because the dominance rule is constructed upon the EDD indexing convention. Although the computation time requirements for the proposed algorithm are relatively higher compared to the other dispatching rules, they are mostly used for evaluating the breakpoints and testing their

validity, which is done only once for each problem. This can be seen from the $n = 500$ case, the average number of interchanges for the EDD rule is substantially higher than others but the computation time only differs by 26.5 centiseconds from the minimum. In a complete enumeration method, e.g. a branch and bound (B&B) algorithm, the matrix of breakpoints will be calculated only once.

The proposed algorithm dominates the competing rules because the total weighted tardiness value is always less than or equal to those obtained from the heuristics in each run as shown above. In tables 8 and 9, we compare the results of individual runs with the ATC rule, which is the best one among the competing rules. In table 8, (<) represents the number of runs in which the proposed algorithm gives better results than the ATC rule, whereas (=) represents the number of runs in which they give the same total weighted tardiness value out of 10 000 randomly generated runs for each job case. These values indicate that there is a significant improvement in the upper bound value. In table 9, we compare these two rules in more detail for each RDD and TF combination for the the 100-job case as an example. When due dates are loose, e.g. for RDD = 0.5 or 0.7 or 0.9 and TF = 0.1 combinations, that means 1200 runs, the total weighted tardiness value was equal to zero for each run, hence there was no room for improvement. On the other hand, the problem becomes more difficult when due dates are tight, for which we decrease the upper bound value in each run. Therefore, we can easily claim that the

proposed algorithm provides a reasonably good schedule with reasonable computational effort.

6. Concluding remarks

In this paper, we develop a new dominance rule for the $1 \parallel \sum w_i T_i$ problem which provides a sufficient condition for local optimality. Therefore, a sequence generated by the proposed algorithm, that is based on the dominance rule, cannot be improved by adjacent job interchanges. We also enlarge the region for which the $1 \parallel \sum w_i T_i$ problem can be solved optimally by the WSPT rule. The proposed algorithm is implemented on a set of heuristics including the ATC rule that is shown to be close to the optimal solution by Vepsalainen and Morton (1987). Our computational experiments over 40 000 randomly generated problems indicate that the amount of improvement is statistically significant for all heuristics, and the proposed algorithm dominates the competing rules in all runs. Furthermore, Abdul-razaq *et al.* (1990) tested several B&B algorithms by using the Emmons' dominance rules in a node elimination procedure. The proposed dominance rule covers and extends the Emmons' results by considering the time-dependent orderings between each pair of jobs so that tighter upper and lower bounds can be found as a function of start time of this pair. This dominance rule can also be used for the infinite horizon total weighted tardiness problem. When a job is available, the breakpoint matrix should be updated and the proposed algorithm can be applied to the new sequence generated. For further research, we will look at how the presented results can be incorporated in a B&B solution method in conjunction with a branching condition and lower bounding scheme.

References

ABDUL-RAZAQ, T. S., POTTS, C. N., and VAN WASSENHOVE, L. N., 1990, A survey of algorithms for the single-machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, **26**, 235–253.

CASKEY, K., and STORCH, R. L., 1996, Heterogeneous dispatching rules in job and flow shops. *Production Planning & Control*, **7**, 351–361.

EMMONS, H., 1969, One machine sequencing to minimize certain functions of job tardiness. *Operations Research*, **17**, 701–715.

JENSEN, J. B., PHILIPOOM, P. R., and MALHOTRA, M. K., 1995, Evaluation of scheduling rules with commensurate customer priorities in job shops. *Journal of Operations Management*, **13**, 213–228.

LAWLER, E. L., 1977, A 'pseudopolynomial' algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, **1**, 331–342.

MORTON, T. E., and PENTICO, D. W., 1993, *Heuristic Scheduling Systems with Applications to Production Systems and Project Management* (New York: John Wiley).

PINEDO, M, 1995, *Scheduling: Theory, Algorithms and Systems* (New Jersey: Prentice Hall).

POTTS, C. N., and VAN WASSENHOVE, L. N., 1985, A branch and bound algorithm for total weighted tardiness problem. *Operations Research*, **33**, 363–377.

POTTS, C. N., and VAN WASSENHOVE, L. N., 1991, Single machine tardiness sequencing heuristics. *IIE Transactions*, **23**, 346–354.

RACHAMADUGU, R. M. V., 1987, A note on weighted tardiness problem. *Operations Research*, **35**, 450–452.

RINNOOY KAN, A. H. G., LAGEWEG, B. J., and LENSTRA, J. K., 1975, Minimizing total costs in one-machine scheduling. *Operations Research*, **23**, 908–927.

SZWARC, W., 1993, Adjacent orderings in single machine scheduling with earliness and tardiness penalties. *Naval Research Logistics*, **40**, 229–243.

SZWARC, W., and LIU, J. J., 1993, Weighted tardiness single machine scheduling with proportional weights. *Management Science*, **39**, 626–632.

VEPSALAINEN, A. P. J., and MORTON, T. E., 1987, Priority rules for job shops with weighted tardiness costs. *Management Science*, **33**, 1035–1047.