A New Lower Bounding Scheme for the Total Weighted Tardiness Problem

M. Selim Akturktt
Bilkent University, Turkey, akturk@bilkent.edu.tr

Mehmet Bayram Yildirim
Bilkent University, Turkey, bayram.yildirim@wichita.edu

A NEW LOWER BOUNDING SCHEME FOR THE TOTAL WEIGHTED TARDINESS PROBLEM

M. Selim Akturktt and M. Bayram Yildirim§
Department ofIndustrial Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey

Scope and Purpose--In enumerative methods, such as branch and bound algorithms, reducing the number of alternatives for finding the optimal solution is a key issue. This is usually done both by developing efficient upper and lower bounding schemes and by utilizing dominance properties, which usually rely heavily on the Emmons' dominance rules, to restrict the search space. We propose a new dominance rule which takes its background from adjacent pairwise interchange method for the single machine total weighted tardiness problem with job dependent penalties. 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 are found as a function of start time of this pair.

Abstract-We propose a new dominance rule that provides a sufficient condition for local optimality for the $1||\Sigma w_i T_i$ problem. We prove that if any sequence violates the proposed dominance rule, then switching the violating jobs either lowers the total weighted tardiness or leaves it unchanged. Therefore, it can be used in reducing the number of alternatives for finding the optimal solution in any exact approach. We introduce an algorithm based on the dominance rule, which is compared to a number of competing approaches for a set of randomly generated problems. We also test the impact of the dominance rule on different lower bounding schemes. Our computational results over 30,000 problems indicate that the amount of improvement is statistically significant for both upper and lower bounding schemes.

I. INTRODUCTION

As firms struggle to survive in an increasingly competitive environment, a greater emphasis needs to be placed on coordinating the priorities of the firms throughout the functional areas. 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 as stated by Jensen *et al.* [6], such as the firm's length of relationship with the customer, how frequently they provide business to the firm, how much of the firm's capacity they fill with orders and the potential of a customer to provide orders in the future. In many applications, meeting due dates and avoiding delay penalties are the most important goals of scheduling. The costs of tardy deliveries, such as customer bad will, lost future sales and rush shipping costs, vary significantly over customers and orders, and the implied strategic weight should be reflected in job priority. The vast majority of the job shop scheduling literature is replete with rules that do not consider job tardiness penalty or customer importance information. The firm's strategic priorities thus require the information pertaining to customer importance be incorporated into its shop floor control 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 propose a new dominance rule for the single machine total weighted tardiness problem with job dependent penalties, and implement in upper and lower bounding schemes.

Lawler [7] shows that the total weighted tardiness problem, $1||\Sigma w_i T_i$, is strongly NP-hard and gives a pseudo polynomial algorithm for the total tardiness problem, $1||\Sigma T_i$. Various enumerative solution methods have been proposed for both the weighted and unweighted cases. Emmons [3] derives several dominance rules that restrict the search for an optimal solution to the $1||\Sigma T_i$ problem. Emmons' rules are

used in both branch and bound (B&B) and dynamic programming algorithms (Fisher [4] and Potts and Van Wassenhove [8,9]). Rinnooy Kan *et al.* [11] extended these results to the weighted tardiness problem. Rachamadugu [10] identifies a condition characterizing adjacent jobs in an optimal sequence for $1||\Sigma w_i T_i$. Chambers *et al.* [2] develop new heuristic dominance rules and a flexible decomposition heuristic. The exact approaches used in solving the weighted tardiness problem are tested by Abdul-Razaq *et al.* [1] and they use Emmons' dominance rules to form a precedence graph for finding upper and lower bounds. They show that the most promising lower bound both in quality and time consumption is the linear lower bound method by Potts and Van Wassenhove [8], which is obtained from Lagrangian relaxation of machine capacity constraints. Hoogeveen and Van de Velde [5] reformulate the problem by using slack variables and show that better Lagrangian lower bounds can be obtained.

Szwarc [12] 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. Szwarc and Liu [13] present a two-stage decomposition mechanism to $1||\Sigma w_i T_i$ problem when tardiness penalties are proportional to the processing times. The importance of a customer can depend on a variety of factors as stated above but it is important for manufacturing to reflect these priorities in the scheduling decisions. Therefore, we present a new dominance rule for the most general case of total weighted tardiness problem. The proposed rule covers and extends the Emmons' results and generalizations of Rinnooy Kan *et al.* by considering the time dependent orderings between each pair of jobs.

Since the implicit enumerative algorithms may require considerable computer resources both in terms of computation times and memory, several heuristics and dispatching rules have been proposed. Vepsalainen and Morton [14] develop and test efficient dispatching rules for the weighted tardiness problem with specified due dates and delay penalties. The proposed dominance rule provides a sufficient condition for local optimality, and it generates schedules that cannot be improved by adjacent job interchanges. 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 prove 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 weighted tardiness problem is NP-hard and the lower bounds in the literature are either weak or not practical to use due to extensive computational requirements. As a result, the exact solution for a 50 job problem is a barrier that could not be passed. The linear lower bound of Potts and Van Wassenhove is rather a weak lower bound but it is found to be the most promising one by Abdul-Razaq *et al.* which contradicts the often heard conjecture that one should restrict the search tree as much as possible by using the sharpest possible bounds. The linear lower bound calculations are based on an initial sequence. We show that having a better upper bound value, which is close to optimal solution, improves the lower bound value obtained from the linear lower bound method.

The remainder of this paper is organized as follows. In the Section 2, 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 in Section 4. The lower bounding scheme is described in Section 5. Computational analysis is reported in Section 6. Finally, some concluding remarks are provided in Section 7.

2. PROBLEM DEFINITION AND NOTATION

The single machine total weighted tardiness problem, $1||\Sigma w_i T_i$, may be stated as follows. Each of n 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 has 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 $i<j$. For a particular schedule the tardiness of job i, T_i, is either zero (in case it is completed before its due date d_i) or otherwise it equals the difference of its completion time and its due date. The problem can be formally stated as: find a schedule S that minimizes $f(S)=\Sigma_{i=1}^{n} w_i T_i$. To introduce the dominance rule, consider schedules $S_1=Q_1ijQ_2$ and $S_2=Q_1jiQ_2$, where Q_1 and Q_2 are two disjoint subsequences of the remaining $n-2$ jobs. Let $t=\Sigma_{k\in Q_1} p_k$ be the completion time of Q_1.

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.

$$f_{ij}(t)=\begin{cases}0 & t\le d_i-(p_i+p_j),\\ (t+p_i+p_j-d_i)\cdot w_i & d_i-(p_i+p_j)\le t\le d_i-p_i,\\ p_j\cdot w_i & d_i-p_i\le t,\end{cases}$$

$$g_{ij}(t)=\begin{cases}0 & t\le d_j-(p_i+p_j),\\ (t+p_i+p_j-d_j)\cdot w_j & d_j-(p_i+p_j)\le t\le d_j-p_j,\\ p_i\cdot w_j & d_j-p_j\le t.\end{cases}$$

Then, $\Delta_{ij}(t)=f_{ij}(t)-g_{ij}(t)$.

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, j should precede i at time t;
- if $\Delta_{ij}(t)>0$ then, i should precede j at time t;
- if $\Delta_{ij}(t)=0$, 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 $t\le$*breakpoint*, i precedes j (or j precedes i) and then j precedes i (or i precedes j).
- An ordering relation R between two jobs is *transitive* whenever iRj and jRk implies iRk.
- i *globally* precedes j, $i\Rightarrow j$, (j *globally* precedes i, $j\Rightarrow i$) if it implies the existence of an optimal sequence in which job i (job j) precedes job j (job i) is guaranteed and the transitivity property holds such that if $i\Rightarrow j$ and $j\Rightarrow k$ then $i\Rightarrow k$.
- i *unconditionally* precedes j, ($i\rightarrow j$) the ordering does not change, i.e. i always precedes j when they are adjacent, but the transitivity property might not hold. Thus it does not imply that an optimal sequence exists in which i precedes j.
- i *conditionally* precedes j, ($i<j$) 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

Potts and Van Wassenhove [8] have verified the effectiveness of adjacent pairwise interchange (API) method as a pruning device in their B&B algorithm for the $1||\Sigma w_iT_i$ problem. The total weighted tardiness function is not convex so that API method can lead only to local improvements. However, the API method preceded by a good heuristic has a reasonable chance to lead to an optimal solution. Furthermore, 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.

When all of the possible cases are studied, it can be seen that there are at most three possible breakpoints for the $1||\Sigma w_iT_i$ problem as shown below:

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

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

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

These breakpoints may be determined by looking at the points where the piecewise linear and continuous functions $f_{ij}(t)$ and $g_{ij}(t)$ intersect. Assuming the EDD indexing convention in the sequel, the following six cases are exhaustive.

Case 1. $d_i<d_j$, $p_iw_j<p_jw_i$, $p_i(w_j-w_i)>(d_j-d_i)w_i$.

Case 2. $d_i=d_j$, $w_i<w_j$, $p_iw_j<p_jw_i$.

Case 3. $d_i<d_j$, $p_iw_j\ge p_jw_i$, $p_j(w_j-w_i)>(d_j-d_i)w_j$.

Case 4. $d_i\le d_j$, $p_iw_j\le p_jw_i$, $p_i(w_j-w_i)\le(d_j-d_i)w_i$.

Case 5. $d_i=d_j$, $p_iw_j\ge p_jw_i$.

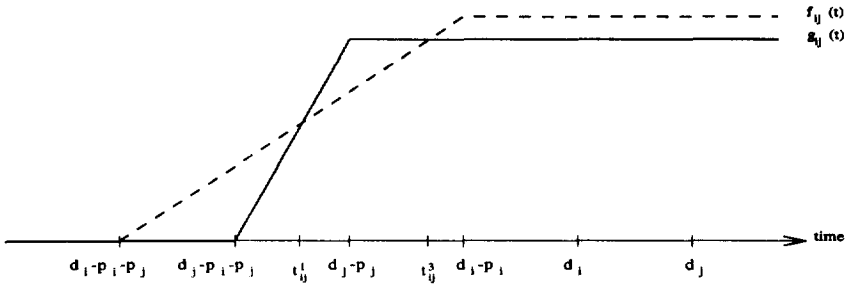

Fig. 1. Case 1.

Case 6. $d_i<d_j$, $p_iw_j>p_jw_i$, $p_j(w_j-w_i)\leq(d_j-d_i)w_j$.

In order to explain the new dominance rule, we will investigate how the interchange function $\Delta_{ij}(t)$ changes for each case. As a result, three breakpoints are found and transitivity properties are shown for certain instances.

3.1. Case 1

In this case, we have the two breakpoints t^1_{ij} and t^3_{ij} as it can be seen from Fig. 1. The following proposition can be used to specify the order of jobs at time t for this case.

Proposition 1. *If $d_i<d_j$, $p_iw_j<p_jw_i$ and $p_i(w_j-w_i)>(d_j-d_i)w_i$ then there are two breakpoints t^1_{ij} and t^3_{ij}, and for $t\leq t^1_{ij}$, $i<j$, for $t^1_{ij}\leq t\leq t^3_{ij}$, $j<i$ and for $t\geq t^3_{ij}$, $i<j$.*

Proof. If $(t\leq d_i-(p_i+p_j))$ then no tardiness occurs, so it is indifferent to schedule either i or j first. If $(d_i-(p_i+p_j)<t<d_j-(p_i+p_j))$, then 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 $i<j$. If $(d_j-(p_i+p_j)\leq t\leq d_j-p_j)$, then either i or j is tardy if not scheduled first. Here, $\Delta_{ij}(t)=(w_i-w_j)t+(p_i+p_j)(w_i-w_j)-w_id_i+w_jd_j$. The breakpoint $t^1_{ij}=[\{(w_jd_i-w_jd_j)/(w_i-w_j)\}-(p_i+p_j)]$ is defined in this region. If the processing of this pair starts up to $t\leq t^1_{ij}$ then $i<j$ and $j<i$ if the processing begins after t^1_{ij}. If $(d_j-p_j\leq t<d_i-p_i)$, then j is always tardy but i is not tardy if scheduled first. Here, $\Delta_{ij}(t)=(t+p_i+p_j-d_i)w_i-p_iw_j$. There is an another breakpoint $t^3_{ij}=d_i-p_j-p_i(1-w_j/w_i)$, and $j<i$ for $t\leq t^3_{ij}$, $i<j$ afterwards. If $(d_i-p_i\leq t)$ both jobs are tardy. Here, $\Delta_{ij}(t)=p_jw_i-p_iw_j$. *If $\Delta_{ij}(t)\geq 0$ then $i<j$, otherwise $j<i$. Moreover, from (3), we know that if $t^3_{ij}<d_i-p_i$ then $p_iw_i<p_jw_i$, which means $\Delta_{ij}(t)>0$, so $i<j$.* ■

3.2. Case 2

The second case can be considered as a special case of the first one such that $d_i=d_j=d$ as depicted in Fig. 2. In this case, $t^1_{ij}=d-(p_i+p_j)$, therefore we are indifferent to schedule either job i or job j first up to t^1_{ij}. Since $d_i=d_j$, $p_i\leq p_j$ by the EDD ordering convention. Therefore, there can be a breakpoint if $w_i<w_j$ as stated below.

Proposition 2. *If $d_i=d_j$, $w_i<w_j$ and $p_iw_j<p_jw_i$ then there is the breakpoint t^3_{ij}. If the processing of this pair starts up to $t\leq t^3_{ij}$ then $j<i$ and $i<j$ if the processing begins after t^3_{ij}.*

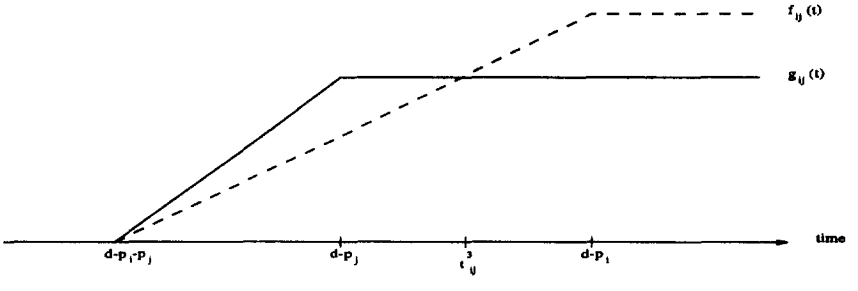

Fig. 2. Case 2.

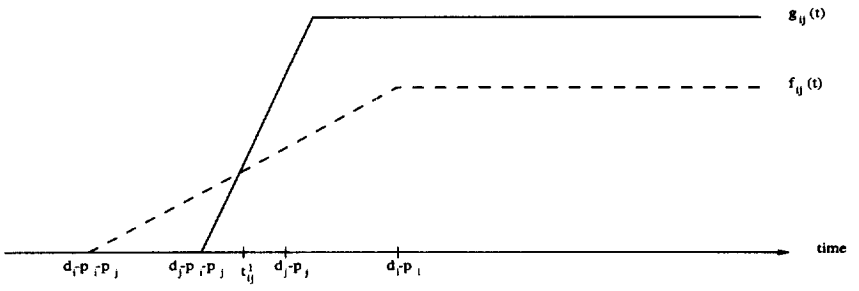

Fig. 3. Case 3.

3.3. Case 3

This case is similar to the first case except, $p_i w_j \geq p_j w_i$, hence there is only one breakpoint t_{ij}^1 as shown in Fig. 3.

Proposition 3. *If $d_i < d_j$, $p_i w_j \geq p_j w_i$ and $p_j(w_j - w_i) > (d_j - d_i)w_j$ then there is the breakpoint t_{ij}^1, and $i \prec j$ for $t \leq t_{ij}^1$, and $j \prec i$ afterwards.*

Proof. In this case we have to make sure that the nonconstant segments of $f_{ij}(t)$ and $g_{ij}(t)$ intersect, and the breakpoint t_{ij}^1 satisfies $t_{ij}^1 = [(w_i d_i - w_j d_j)/(w_i - w_j)] - (p_i + p_j) < d_i - p_i$. It leads to the condition of $p_j(w_j - w_i) > (d_j - d_i)w_j$, since $\Delta_{ij}(t) = p_j w_i - p_i w_j \leq 0$ for $t \geq d_i - p_i$. ■

3.4. Case 4

In this case, if we can show that $i \prec j$ for every t then $i \rightarrow j$, that means $\Delta_{ij}(t) \geq 0 \ \forall t$, i.e. $f_{ij}(t) \geq g_{ij}(t) \ \forall t$, as shown in Fig. 4.

Proposition 4. *If $d_i \leq d_j$, $p_i w_j \leq p_j w_i$ and $p_i(w_j - w_i) \leq (d_j - d_i)w_i$ then job i unconditionally precedes job j, i.e. $(i \rightarrow j)$.*

Proof. As defined earlier $\Delta_{ij}(t) = f_{ij}(t) - g_{ij}(t)$. If we let $t = d_j - (p_i + p_j)$then $\Delta_{ij}(d_j - p_i - p_j) = (d_j - d_i)w_i \geq 0$, since $d_j \geq d_i$ and $g_{ij}(d_j - p_i - p_j) = 0$, so $i \prec j$ at time $d_j - (p_i + p_j)$. Let $t = d_j - p_j$ then $\Delta_{ij}(d_j - p_j) = (d_j + p_i - d_i)w_i - p_i w_j = (d_j - d_i)w_i - p_i(w_j - w_i) \geq 0$, since $p_i(w_j - w_i) \leq (d_j - d_i)w_i$, so $i \prec j$ at time $d_j - p_j$. If we let $t = d_i - p_i$ then $\Delta_{ij}(d_i - p_i) = p_j w_i - p_i w_j \geq 0$, since $p_i w_j \leq p_j w_i$, consequently $i \prec j$ again at time $d_i - p_i$. Therefore, the result follows and $i \rightarrow j$. ■

3.5. Case 5

In the fifth case, $d_i = d_j = d$ as shown in Fig. 5, consequently $p_i \leq p_j$ by the EDD ordering convention, that means $w_j \geq w_i$ in order to satisfy the $p_i w_j \geq p_j w_i$ condition. As discussed in the second case, $t_{ij}^1 = d - (p_i + p_j)$ when $d_i = d_j$, hence we are indifferent to schedule either job i or job j first up to t_{ij}^1. If we can show that $\Delta_{ij}(t) \leq 0 \ \forall t$, that means $j \prec i$ for every t, then $j \rightarrow i$ as stated below.

Proposition 5. *If $d_i = d_j$ and $p_i w_j \geq p_j w_i$ then job j unconditionally precedes job i, i.e. $(j \rightarrow i)$.*

Proof. Let $t = d - p_j$ then $\Delta_{ij}(d - p_j) = p_i w_i - p_i w_j = p_i(w_i - w_j) \leq 0$, since $w_j \geq w_i$. If we let $t = d - p_i$ then $\Delta_{ij}(d - p_i) = p_j w_i - p_i w_j \leq 0$. Therefore, $\Delta_{ij}(t) \leq 0 \ \forall t$ and $j \rightarrow i$. ■

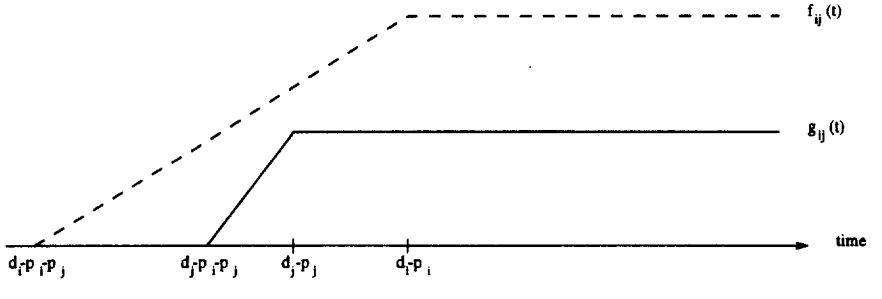

Fig. 4. Case 4.

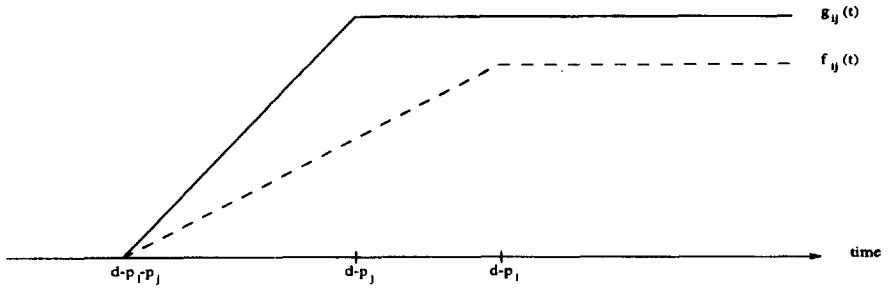

Fig. 5. Case 5.

3.6. Case 6

The sixth case is similar to Case 3 except $d_i-p_i<d_j-p_j$ as shown in Fig. 6, although the relative positions of d_i-p_i and $d_j-(p_i+p_j)$ might change such that if $d_j-d_i\leq p_j$ then $d_j-(p_i+p_j)\leq d_i-p_i$, otherwise $d_i-p_i>d_j-(p_i+p_j)$.

Proposition 6. *If $d_i<d_j$, $p_iw_j>p_jw_i$ and $p_j(w_j-w_i)\leq(d_j-d_i)w_j$ then there is the breakpoint t_{ij}^2, and $i<j$ for $t\leq t_{ij}^2$, and $j<i$ afterwards.*

Proof. In order to have the breakpoint t_{ij}^2 we have to make sure that the nonconstant segment of $g_{ij}(t)$ intersects with the constant segment of $f_{ij}(t)$. This is the case if the breakpoint t_{ij}^2 with $(t_{ij}^2+p_i+p_j-d_j)=p_jw_i$ satisfies $d_i-p_i\leq t_{ij}^2<d_j-p_j$. It leads to the conditions of $p_iw_j>p_jw_i$ for $t_{ij}^2<d_j-p_j$ and $p_j(w_j-w_i)\leq(d_j-d_i)w_j$ for $t_{ij}^2\geq d_i-p_i$. If $(d_j-p_j\leq t)$ then $j<i$ since $\Delta_{ij}(t)=p_jw_i-p_iw_j<0$. ■

After analyzing 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 there are at most two breakpoints, which are t_{ij}^1 and t_{ij}^3 in Case 1. Furthermore, there is exactly one breakpoint in Cases 2, 3 and 6, which are t_{ij}^1, t_{ij}^3 and t_{ij}^2, respectively. We do not have any breakpoint in Cases 4 and 5. As a result, we can state the following general rule, which provides a sufficient condition for local optimality, by using all of the properties of the dominance rule.

General rule

IF $d_i=d_j$
THEN IF $p_iw_j\geq p_jw_i$
  THEN $j\rightarrow i$
ELSE IF $w_i\geq w_j$
  THEN $i\rightarrow j$
ELSE $j<i$ for $t\leq t_{ij}^3$

  $i<j$ for $t\geq t_{ij}^3$

ELSE IF $p_i(w_j-w_i)>(d_j-d_i)w_i$
  THEN $i<j$ for $t\leq t_{ij}^1$
    IF $p_iw_j\geq p_jw_i\wedge p_j(w_j-w_i)>(d_j-d_i)w_j$

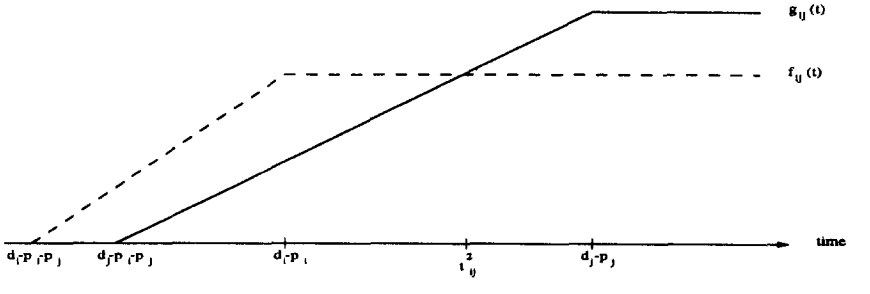

Fig. 6. Case 6.

THEN $j<i$ for $t\geq t_{ij}^1$
ELSE $j<i$ for $t_{ij}^1\leq t\leq t_{ij}^3$

$i<j$ for $t\geq t_{ij}^3$

ELSE IF $p_iw_j\leq p_jw_i$
THEN $i\rightarrow j$
ELSE $i<j$ for $t\leq t_{ij}^2$

$j<i$ for $t\geq t_{ij}^2$

Let Z denote the set of all jobs, C the set of pairs (i,j) for which $\Delta_{ij}(t)$ has at least one breakpoint t_{ij}, $i,j\in C$, the largest of these breakpoints, and $t_C=\max\{t_{ij}\mid (i,j)\in C\}$. The following lemma can be used quite effectively to find an optimal sequence for the remaining jobs on hand after a time point t_C.

Lemma 1. *If $t>t_C$ then the weighted shortest processing time (WSPT) rule gives an optimal sequence for the remaining unscheduled jobs.*

Proof. The t_C is the last breakpoint for any pair of jobs i,j on the time scale. Furthermore, it is a well-known result that the WSPT rule gives an optimal sequence for the $1||\Sigma w_iT_i$ problem when either all due dates are zero or all jobs are tardy, i.e. $t>\max_{i\in Z}\{d_i-p_i\}$. The problem reduces to 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 $t_C<\max_{i\in Z}\{d_i-p_i\}$, so we enlarge the region for which the total weighted tardiness problem can be solved optimally by the WSPT rule. We already show that for every job pair (i,j), one of these conditions must hold either there is a breakpoint or unconditional ordering $(i\rightarrow j)$ or globally precedence $(i\Rightarrow j)$. The WSPT rule holds for both $i\rightarrow j$ and $i\Rightarrow j$. If there is a breakpoint then for $t>t_{ij}$ the job having higher w_i/p_i is scheduled first, so WSPT again holds. For $t>t_C$, consider a job i which conflicts with the WSPT rule, then we can have a better schedule by making adjacent job interchanges which either lowers the total weighted tardiness value or leaves it unchanged. If we do same thing for all of the remaining jobs, we get the WSPT sequence. ■

4. TRANSITIVITY

The transitivity property is very crucial for reducing the number of sequences that have to be considered in an implicit enumeration technique. Szwarc [12] shows that there is a transitivity property for the $1||\Sigma T_i$ problem. The transitivity property does not hold for the $1||\Sigma w_iT_i$ problem even for the assumption that the weights are proportional to processing times as shown by Szwarc and Liu [13]. Therefore, we will present two possible cases in which transitivity property holds for the proposed dominance rule. Let $\Delta_{iQj}(t)$ is the cost of interchanging two jobs i and j whose processing starts at time t, and they are not necessarily adjacent and are separated by the subsequence Q. Notice that when $Q=\varnothing$, $\Delta_{iQj}(t)$ reduces to $\Delta_{ij}(t)$. Furthermore, $T_{iQj}(t)$ and $T_Q(t)$ are the total weighted tardiness of all jobs in sets $\{iQj\}$ and $\{Q\}$ respectively if their processing starts at time t,

$$\Delta_{iQj}(t)=T_{jQi}(t)-T_{iQj}(t)$$

$$=w_j\max(0,t+p_j-d_j)+T_Q(t+p_j)+w_i\max\left(0,t+p_j+\sum_{k\in Q}p_k+p_i-d_i\right)$$

$$-w_i\max(0,t+p_i-d_i)-T_Q(t+p_i)-w_j\max\left(0,t+p_i+\sum_{k\in Q}p_k+p_j-d_j\right).$$

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. $i\Rightarrow j$.*

Proof. It has been already proved by Rinnooy Kan *et al.* [11]. ■

Lemma 3. *If $d_i\leq d_j$, $p_i>p_j$ and $w_i<w_j$ then job j globally precedes job i $(j\Rightarrow i)$ for $t>t_{ij}$.*

Proof. The breakpoint t_{ij} can be either t_{ij}^1 or t_{ij}^2 from Cases 3 or 6, respectively, since the breakpoint t_{ij}^3 can only occur when $p_iw_j<p_jw_i$ as shown in Cases 1 and 2. We have three possibilities to examine such as either both jobs can be tardy, that means there is not any breakpoint, or t_{ij}^1 is the only breakpoint from Case 3 or t_{ij}^2 is the only breakpoint from Case 6. We have already shown that both of these breakpoints

cannot occur at the same time for these cases. For proving j globally precedes i when both of the jobs are tardy ($j \Rightarrow i$), we have to show that inserting a set of jobs Q between j and i does not alter the relative sequence of j and i (i.e. $j<i$) at a given time $t>d_j-p_j$. When both jobs are tardy, $\Delta_{iQj}(t)$ reduces to

$$\Delta_{iQj}(t)=(w_i-w_j)\sum_{k\in Q}p_k+T_Q(t+p_j)-T_Q(t+p_i)+w_ip_j-w_jp_i.$$

Since $w_i-w_j<0$, $p_j-p_i<0 \Rightarrow w_ip_j-w_jp_i<0$ and $T_Q(t+p_j)\leq T_Q(t+p_i)$, therefore, $\Delta_{iQj}(t)<0$ which means $j \Rightarrow i$. We can extend this result to $t>t_{ij}^2$. Now, we are in the region where i is always tardy but j is not tardy if scheduled first. In this region,

$$\Delta_{iQj}(t)=w_i\left(\sum_{k\in Q}p_k+p_j\right)+T_Q(t+p_j)-T_Q(t+p_i)-w_j\left(t+p_i+\sum_{k\in Q}p_k+p_j-d_j\right)$$
$$=T_Q(t+p_j)-T_Q(t+p_i)+\sum_{k\in Q}p_k(w_i-w_j)+w_ip_j-w_j(t+p_i+p_j-d_j).$$

Given

$t>t_{ij}^2$ for jobs i and j, suppose that job i is scheduled before job j. If i and j are adjacent jobs then we can make an interchange which either lowers the total weighted tardiness value or leaves it unchanged to have a better schedule. Now, let us look at the case where they are not adjacent. Since $t>t_{ij}^2=d_j-p_i-p_j(1-w_i/w_j) \Rightarrow t=d_j-p_i-p_j(1-w_i/w_j)+\epsilon^1$ and $\epsilon^1\geq 0$. $\Delta_{iQj}(t)$ can be written as follows:

$$\Delta_{iQj}(t)=T_Q(t+p_j)-T_Q(t+p_i)+\sum_{k\in Q}p_k(w_i-w_j)+w_ip_j-w_j(d_j-p_i-p_j(1-w_i/w_j)+\epsilon^1+p_i+p_j-d_j)$$
$$=T_Q(t+p_j)-T_Q(t+p_i)+\sum_{k\in Q}p_k(w_i-w_j)-w_j\epsilon^1\leq 0$$

Using the similar arguments as stated above, we can improve the current schedule by replacing the positions of jobs i and j, i.e. $j \Rightarrow i$. A similar proof can be done for the $t>t_{ij}^1$ case where i or j will not be tardy if scheduled first but the other one will be tardy as follows:

$$\Delta_{iQj}(t)=T_Q(t+p_j)+w_i\left(t+p_i+\sum_{k\in Q}p_k+p_j-d_i\right)-T_Q(t+p_i)-w_j\left(t+p_i+\sum_{k\in Q}p_k+p_j-d_j\right)>0$$
$$=T_Q(t+p_j)-T_Q(t+p_i)+\sum_{k\in Q}p_k(w_i-w_j)+(w_i-w_j)(p_i+p_j)+t(w_i-w_j)-w_id_i+w_jd_j.$$

Given

$$t>t_{ij}^1,\ t=[(w_id_i-w_jd_j)/(w_i-w_j)]-(p_i+p_j)+\epsilon^2 \text{ and } \epsilon^2\geq 0.$$
$$\Rightarrow \Delta_{iQj}(t)=T_Q(t+p_j)-T_Q(t+p_i)+\sum_{k\in Q}p_k(w_i-w_j)+\epsilon^2(w_i-w_j)\leq 0,$$

so

$$j \Rightarrow i. \blacksquare$$

We have proved that the dominance properties provide a sufficient condition for local optimality. 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 by making necessary interchanges. The proposed heuristic takes into account all of the global, unconditional and conditional precedence relationships. Let seq[i] denote index of the job in the ith position in the given sequence. The algorithm can be summarized as follows:

Initialization:

Sort the jobs in EDD order.

Calculate the breakpoint matrix using the general rule.

Forward procedure:

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

Lemma 4. *The algorithm has a time complexity of* $O(n^3)$.

Proof. We first obtain an EDD order that takes a total of $O(n \log n)$ time. In order to calculate the breakpoint matrix that has $n(n-1)/2$ entries, we check for global dominances, and the breakpoints t_{ij}^1, t_{ij}^2 and t_{ij}^3 in the worst case for every job pair (i,j), which require a total of $O(n^2)$ time. In the forward procedure, we first check for the global dominances so that total of $n(n-1)/2$ comparisons are done which take a total of $O(n^2)$ time. Furthermore, we have to make $\sum_{i=1}^{n-1} i = n(n-1)/2$ comparisons to determine the job in position 1, i.e. seq [1]. Similarly, for the job in position 2, we have to make $\sum_{i=1}^{n-2} i = (n-1)(n-2)/2$ comparisons. In order to determine the job for the kth position, we have to make $\sum_{i=1}^{n-k} i = (n-k)(n-k+1)/2$ comparisons which make a total of

$$\sum_{i=1}^{n-1} \frac{i(i-1)}{2} = \frac{1}{2} \sum_{i=1}^{n-1} [i^2 - i] = \frac{1}{2} \left[\frac{(n-1)n(2n-1)}{6} - \frac{n(n-1)}{2} \right]$$

comparisons or a total of $O(n^3)$ time. Therefore, the algorithm takes a total of $\max[O(n \log n), O(n^2), O(n^3)] = O(n^3)$ time.

5. LOWER BOUNDING SCHEME

We now present two lower bounding approaches for the $1\|\Sigma w_i T_i$ problem both by exploiting the results discussed in the previous sections and by relaxing some of the constraints of the original problem.

5.1. Lower bound 1

The Emmons' dominance rules play a major rule in the enumerative algorithms in the literature. Assume that these rules have already been applied to get a sequence for each job h, and let B_h and A_h be the set of jobs which precede and succeed job h respectively in at least one optimal sequence. Let Z denote a set of all jobs, J be a set of scheduled jobs and U be a set of unscheduled jobs, i.e. $Z=J \cup U$. We will first present the three conditions of Rinnooy Kan *et al.* generalizations based on the Emmons' theorem and then show how the global dominance property can be extended by using the proposed dominance rule. The proof of the last condition is already given in Lemma 3.

Dominance Theorem.

(1) *There exists an optimal sequence in which job i is sequenced before job j, i.e.* $i \Rightarrow j$, *if one of the following three conditions is satisfied.*

(*a*) $p_i \le p_j$, $w_i \ge w_j$ *and* $d_i \le \max\{d_j, \sum_{h \in B_j} p_h + p_j\}$;

(*b*) $w_i \ge w_j$, $d_i \le d_j$ *and* $d_j \ge \sum_{h \in S - A_i} p_h - p_j$;

(*c*) $d_j \ge \sum_{h \in S - A_i} p_h$.

(2) *Given* $d_i \le d_j$, $p_i > p_j$ *and* $w_i < w_j$, *j globally precedes i* ($j \Rightarrow i$) *for* $t > t_{ij}$.

Whenever jobs i and j satisfy the above conditions, an arc (i,j) is add to the precedence graph with any other arcs that are implied by transitivity. The lower bound below is designed to see the impact of the new dominance property on the graph generated by the Emmons' dominance theorem. When the number of global dominances found increases, the following lower bounding scheme gives a tighter lower bound value. We propose a method in which requirement that the machine can process at most one job is relaxed

Table 1. A ten job example for comparison of dominance rules

Job	1	2	3	4	5	6	7	8	9	10
d_i	11	26	26	27	28	31	32	32	32	42
p_i	7	10	10	1	6	3	5	7	9	2
w_i	5	8	1	9	7	9	9	1	7	10

in conjunction with the global dominance relationships. The revised completion time of each job, $\tilde{C}_i$, and a lower bound on the total weighted tardiness are found as follows:

Lower Bound 1 (LB_1):

Step 1. Find the global dominance set, GD_i, for each unscheduled job i:

$$GD_i = \{j \mid j \Rightarrow i \text{ and } j \in U\}.$$

Step 2. For every $i \in U$, calculate the earliest starting time, es_i, as follows:

$$es_i = \min_{j \in GD_i} \{es_j\} + \sum_{j \in GD_i} p_j$$

Step 3. Calculate the revised completion times, $\tilde{C}_i = es_i + p_i$ for every $i \in U$.

Step 4.

$$LB_1 = \sum_{i=1}^{n} w_i T_i = \sum_{i \in J} w_i \max\{0,(C_i - d_i)\} + \sum_{i \in U} w_i \max\{0,(\tilde{C}_i - d_i)\}.$$

By the example given in Table 1, we try to figure out the effect of last dominance rule on the number of global dominances generated and the corresponding lower bound values. In Fig. 7, the dashed lines represent the global dominances found by Rinnooy Kan *et al.* generalizations and the bold lines represent the additional global dominances added by the proposed dominance rule. The number of edges generated by the first three conditions is 17 and the lower bound value is 1. When the additional global dominance rule is considered, the number of edges increases by 13, which makes the total of 30. Consequently, the lower bound value increases to 40. Furthermore, the proposed dominance rule provides the breakpoint values as a function of time as shown in Table 2. In the matrix of breakpoints, the following notation is used: the numbers in cells correspond to the breakpoints, the global precedences ($\Rightarrow$) and unconditional precedences ($\rightarrow$). A more detailed discussion on the computational results can be found in Section 6.

5.2. Linear lower bound

The linear lower bound is originally obtained by Potts and Van Wassenhove [8] by using the Lagrangian Relaxation approach with subproblems that are total weighted completion time problems.

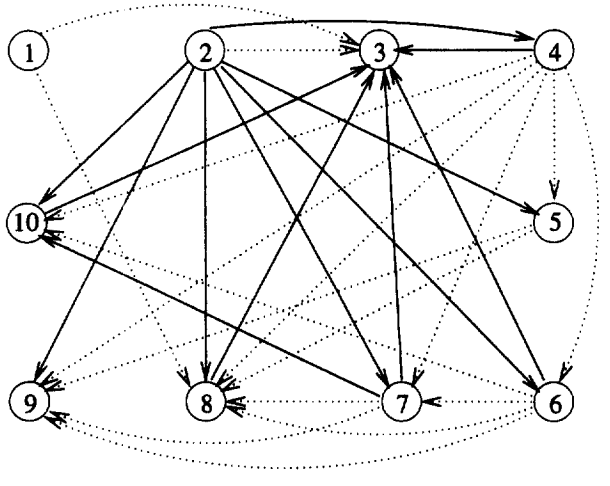

Fig. 7. Precedence graphs generated by dominance rule.

Table 2. Breakpoint matrix

Jobs	2	3	4	5	6	7	8	9	10
1	15.25	⇒	19.56	19.29	22.67	22.78	⇒	22.43	34.00
2		⇒	⇒	⇒	⇒	⇒	⇒	⇒	⇒
3			⇐	12.33	⇐	⇐	⇐	14.00	⇐
4				⇒	⇒	⇒	⇒	⇒	⇒
5					24.33	24.89	⇒	⇒	35.40
6						⇒	⇒	⇒	⇒
7							⇒	⇒	⇒
8								→	33.20
9									32.40

Abdul-Razaq *et al.* [1] show that it may also be derived by reducing the total weighted tardiness criterion to a linear function, i.e. total weighted completion time problem. For the job $i(i=1,\ldots,n)$, we have

$$w_iT_i=w_i\max\{C_i-d_i,0\}\geq u_i\max\{C_i-d_i,0\}\geq u_i(C_i-d_i),$$

where $w_i\geq u_i\geq 0$ and C_i is the completion time of job i. Let $u=(u_1,\ldots,u_n)$ be a vector of linear weights, i.e. weights for the linear function C_i-d_i, chosen so that $0\leq u_i\leq w_i$. Then a lower bound is given by the following linear function

$$\mathrm{LB}_{\mathrm{Lin}}(u)=\sum_{i=1}^{n}u_i(C_i-d_i)\leq\sum_{i=1}^{n}w_i\max\{C_i-d_i,0\}.$$

This shows that the solution of total weighted completion time problem provides a lower bound on the total weighted tardiness problem. Given u, the total weighted completion problem can be solved optimally by the WSPT rule in which the jobs are sequenced in non increasing order of u_i/p_i. To obtain the linear lower bound, an initial sequence is required to determine job completion times C_i. Then the vector of linear weights u is chosen to maximize $\mathrm{LB}_{\mathrm{Lin}}(u)$, subject to $u_i\leq w_i$ for each job i. Abdul-Razaq *et al.* [1] compare several lower bounding approaches and their computational results indicate that the linear lower bound is superior to others in the literature due to its quick computability and low memory requirements. We will test the impact of an initial sequence on the linear lower bound value and try to demonstrate having a better, i.e. near the optimal, upper bound value will improve the lower bound value. In Section 6, we will show how the proposed dominance rule can be used to improve the weighted tardiness criterion to obtain a better initial sequence.

6. COMPUTATIONAL RESULTS

We tested each lower bounding scheme on a set of randomly generated problems on a Sun Ultra Sparc 1 workstation using Sun Pascal. The lower bounding scheme was tested on problems with 50, 100 and 150 jobs that were generated as follows. For each job i, an integer processing time p_i and an 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 $[P(1-\mathrm{TF}-\mathrm{RDD}/2), P(1-\mathrm{TF}+\mathrm{RDD}/2)]$ was generated for each job i, where P is the total processing time, $\Sigma_{i=1}^{n}p_i$. As summarized in Table 3, a total of 300 example sets were considered and 100 replications were taken for each combination resulting in 30,000 randomly generated runs.

First, we have tested LB_1 which uses the global dominance information for calculating the lower bound. We have calculated the average lower bound values (LB-Value), average improvement (Improv), average number of global dominances generated (Glob.) and the average real time used in centiseconds (Time). Although the real time depended on the utilization of 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

Table 3. Experimental design

Factors	No. of levels	Settings
Number of jobs	3	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

Table 4. The effect of the new dominance rule

	$n=50$			$n=100$			$n=150$		
Criteria	>	=	<	>	=	<	>	=	<
LB value	5340	4660	0	5824	4176	0	6160	3840	0
No. of arcs	7362	2638	0	7344	2656	0	7297	2703	0

time. The improvement in the lower bound for each run is found as follows: Improv$=[(F(S^D)-F(S^h))/F(S^h)]\times 100$, if $F(S^h)\neq 0$, and zero otherwise, where $F(S^h)$ is the lower bound value obtained by the Rinnooy Kan *et al.* generalizations and $F(S^D)$ is the lower bound value given by the proposed dominance rule. We also performed a paired t-test for the difference between lower bound values generated by these rules for each run.

In Tables 4–6, we give the statistics about the effect of Emmons' rules (EDR), including the Rinnooy Kan *et al.* generalizations, and the proposed dominance rule (PDR). (>) represents the number of cases in which PDR gives better results than EDR, where as (=) represents number of cases in which PDR gives results as well as EDR and (<) represents cases for which EDR gives better results. Both the lower bound value and number of arcs (global dominances) generated by PDR dominate EDR as indicated in Table 4. In Table 5, the results are averaged over 10,000 runs for 50, 100 and 150 job cases. The large t-test values and the average percentage improvement indicate that there is a significant improvement in the lower bound value. In Table 6, we present the effect of RDD and TF on the number of global dominances generated by PDR and EDR for 50 job case. For TF=0.1, almost all of the relations can be defined by global dominances, so both Emmons' and the proposed dominance rule can solve the problem optimally. While for TF≥0.3, the number of global dominances decreases substantially, but still PDR dominates EDR.

In order to find an initial sequence for the linear lower bound, we have selected a number of heuristics and their priority indexes are summarized in Table 7. The EDD, WSPT, SPT and LPT are examples of static dispatching rules, where as ATC and COVERT are dynamic ones. Vepsalainen and Morton [14] have shown that the ATC rule is superior to other sequencing heuristics and close to the optimal for the $1||\Sigma w_iT_i$ problem. Furthermore, we have already shown that if any sequence violates the dominance rule, then the proposed algorithm discussed in Section 4 either lowers the weighted tardiness or leaves it unchanged. First, we use one of the dispatching rules to find an initial sequence, later we apply the algorithm to get the sequence denoted as Heuristic+DR. For each heuristic, we have calculated the average lower bound value before and after implementing the algorithm along with the average improvement, (improv), as summarized in Table 8. ATC, COVERT, and WSPT seem to perform better

Table 5. Comparison of Emmons' rule with the proposed rule

n	LB-Value		Improv (%)	Time		Glob.		t-test
	Emmons	Domin.		Emmons	Domin.	Emmons	Domin.	
50	23591.9	36988.9	85.1	4.06	5.98	694.8	743.7	31.53
100	88391.1	140194.6	96.19	18.7	27.5	2827.5	3021.3	32.09
150	189130.8	296505.1	102.7	49.8	73.1	6411.2	6830.7	31.84

Table 6. Number of global dominances for $n=50$

		Tardiness factor (TF)				
RDD value		0.1	0.3	0.5	0.7	0.9
0.1	Emmons	1041.64	601.51	433.55	368.44	348.50
	Dominance	1041.67	602.70	444.88	410.28	515.45
0.3	Emmons	1225.00	744.08	477.04	378.14	345.06
	Dominance	1225.00	744.59	485.31	420.44	536.59
0.5	Emmons	1225.00	977.19	552.82	394.00	347.12
	Dominance	1225.00	977.29	558.68	442.82	531.73
0.7	Emmons	1225.00	1202.86	661.79	410.44	351.97
	Dominance	1225.00	1202.87	667.38	484.06	527.88
0.9	Emmons	1225.00	1220.95	811.29	440.43	361.40
	Dominance	1225.00	1220.95	818.62	530.32	530.38

Table 7. Dispatching rules used in lower bounding scheme

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

than other heuristics when the dominance rule is applied to get the local optimal sequence. We have tested each heuristic over 10,000 runs for 50, 100 and 150 job cases in Table 9. As discussed above, (>) represents number of runs in which the sequence obtained from Heuristic+DR gives a higher linear lower bound value than the sequence obtained from the heuristic, where as (=) represents number of runs in which Heuristic+DR performs as well as heuristic, and (<) represents number of runs in which Heuristic+DR performs worse. For example, the EDD+DR combination performed 4598 times better (>) than the EDD rule. The large t-test values on the average improvement indicate that the proposed dominance rule provides a significant improvement on all rules and the amount of improvement is notable at 99.5% confidence level for all heuristics.

In Table 10, we summarize the average percent gap, real time consumed in centiseconds and number of times the lower bound value of a heuristic outperforms others over 10000 runs. Notice that more than one heuristic can have the "best" value for a certain run if there is a tie. The average gap between heuristic and its lower bound, (gap) for each run is found as follows: $\text{gap}=[(F(S^{\text{UB}})-F(S^{\text{LB}}))/F(S^{\text{UB}})]\times 100$, if $F(S^{\text{UB}})\neq 0$, and zero otherwise, where $F(S^{\text{UB}})$ is the total weighted tardiness value obtained by the heuristic and $F(S^{\text{LB}})$ is the corresponding lower bound value. It can be seen that Heuristic+DR always performs better than the heuristic alone. Furthermore, the average real time consumed for each dispatching rule is very small, i.e. the maximum time used is 1.2 centiseconds for $n=150$. The average time consumed by LB_1 is higher than other methods. The time for LB_1 includes both the calculations for the breakpoint matrix and testing the dominance rules while the linear lower bound assumes that an initial sequence is given. In summary, we can reach to the following conclusions from these results. The lower bound value of LB_1 is not as good as the linear lower bound method, but it can be used in B&B algorithms because it contains the global dominance information which will restrict the search space

Table 8. Linear lower bound: computational results for $n=50$

	Upper bound			Lower bound		
Heuristic	Before	After	Improv (%)	Before	After	Improv (%)
ATC	119721.1	118569.8	6.26	103248.1	103295.6	0.10
COVERT	127375.3	125861.9	3.96	97996.1	98215.6	0.58
WSPT	153656.5	134254.7	49.41	100286.7	103020.8	1.09
EDD	275662.4	128032.7	39.55	26174.3	93948.9	103.65
SPT	228100.5	213260.4	18.10	18803.5	21049.4	7.45
LPT	535667.8	152872.4	81.05	19031.5	76026.4	147.64

Table 9. Comparison of linear lower bounds

	$n=50$				[$n=100$]				$n=150$			
Heuristic	>	=	<	t-test	>	=	<	t-test	>	=	<	t-test
ATC+DR	2490	7499	11	23.16	2531	7405	64	24.10	2555	7315	130	24.35
COVERT+DR	447	9406	147	3.56	661	9044	295	4.24	858	8834	308	4.21
WSPT+DR	2462	7532	6	22.58	2426	7566	8	22.44	2405	7584	11	21.95
EDD+DR	4598	5176	226	31.73	4463	5202	335	31.16	4368	5346	286	30.55
SPT+DR	3516	6221	263	24.26	3904	5857	259	23.76	3995	5763	242	29.53
LPT+DR	4772	5008	220	32.68	4742	5005	253	31.87	4582	5221	197	31.39

Table 10. Overall computational results

	$n=50$			$n=100$			$n=150$		
Lower bound	$overli\neq\overline{Gap}(\%)$	$overli\neq\overline{Time}$	Best	$overli\neq\overline{Gap}$ (%)	$overline\overline{Time}$	Best	$overli\neq\overline{Gap}$ (%)	$overline\overline{Time}$	Best
ATC	43.44	0.12	6326	42.20	0.44	5419	43.13	0.89	5450
ATC+DR	39.94	0.12	7849	40.63	0.43	6845	42.58	0.91	6814
COVERT	43.28	0.12	6992	45.15	0.45	6468	46.63	0.87	6494
COVERT+DR	42.36	0.12	7164	44.68	0.53	6658	46.37	0.89	6724
WSPT	69.90	0.16	7049	70.76	0.40	7159	71.36	1.20	7260
WSPT+DR	64.48	0.15	7230	67.20	0.44	7197	68.36	1.12	7311
EDD	71.04	0.12	4908	72.33	0.38	4957	72.68	0.83	5192
EDD+DR	44.03	0.12	6600	45.84	0.52	6106	47.11	0.87	6176
SPT	94.03	0.12	4899	95.34	0.46	4898	95.95	0.85	5134
SPT+DR	92.18	0.12	4920	94.20	0.49	4959	95.21	0.85	5191
LPT	97.32	0.11	4873	98.14	0.41	4961	98.36	0.82	5179
LPT+DR	70.42	0.12	5213	76.29	0.46	5195	79.71	0.82	5394

quite substantially. The linear lower bound gives the tightest lower bound value and either the ATC+DR or WSPT+DR can be used as an initial sequence.

7. CONCLUSION

In this paper, we prove that there are certain time points, called as breakpoints, in which the ordering changes for adjacent jobs, such that the arrangement of these jobs in an optimal schedule depends on their start times. Based on these results, we have developed a new dominance rule for the $1||\Sigma w_iT_i$ problem which provides a sufficient condition for local optimality. Therefore, a sequence generated by the proposed rule cannot be improved by adjacent job interchanges. We also enlarge the region for which the $1||\Sigma w_iT_i$ problem can be solved optimally by the WSPT rule. The proposed dominance rule covers and extends the Emmons' results, including Rinnooy Kan *et al.* generalizations, by considering the time dependent orderings between each pair of jobs, so that tighter upper and lower bounds are found as a function of start time of this pair. Furthermore, it can be used as a good pruning device for any exact algorithm. We have implemented the proposed rule in two different lower bounding schemes. Our computational experiments over 30,000 randomly generated problems indicate that the amount of improvement is statistically significant for both methods. For further research, we will look at how the proposed dominance properties can be incorporated in a B&B solution methodology in conjunction with a branching strategy.

Acknowledgements—The authors would like to thank an anonymous referee for pointing out a more elegant and intuitive proof to the derivations of breakpoints than we had originally included.

REFERENCES

1. Abdul-Razaq, T. S., Potts, C. N. and Van Wassenhove, L. N., A survey of algorithms for the single-machine total weighted tardiness scheduling problem. *Discrete Applied Mathematics*, 1990, **26**, 235–253.
2. Chambers, R. J., Carraway, R. L., Lowe, T. J. and Morin, T. L., Dominance and decomposition heuristics for single machine scheduling. *Operations Research*, 1991, **39**, 639–647.
3. Emmons, H., One machine sequencing to minimize certain functions of job tardiness. *Operations Research*, 1969, **17**, 701–715.
4. Fisher, M. L., A dual algorithm for the one-machine scheduling problem. *Mathematical Programming*, 1976, **11**, 229–251.
5. Hoogeveen, J. A. and Van de Velde, S. L., Stronger Lagrangian bounds by use of slack variables: Applications to machine scheduling problems. *Mathematical Programming*, 1995, **70**, 173–190.
6. Jensen, J. B., Philipoom, P. R. and Malhotra, M. K., Evaluation of scheduling rules with commensurate customer priorities in job shops. *Journal of Operations Management*, 1995, **13**, 213–228.
7. Lawler, E. L., A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness. *Annals of Discrete Mathematics*, 1977, **1**, 331–342.
8. Potts, C. N. and Van Wassenhove, L. N., A branch and bound algorithm for total weighted tardiness problem. *Operations Research*, 1985, **33**, 363–377.
9. Potts, C. N. and Van Wassenhove, L. N., Dynamic programming and decomposition approaches for the single machine total tardiness problem. *European Journal of Operational Research*, 1987, **32**, 405–414.
10. Rachamadugu, R. M. V., A note on weighted tardiness problem. *Operations Research*, 1987, **35**, 450–452.
11. Rinnooy Kan, A. H. G., Lageweg, B. J. and Lenstra, J. K., Minimizing total costs in one-machine scheduling. *Operations Research*, 1975, **23**, 908–927.
12. Szwarc, W., Adjacent orderings in single machine scheduling with earliness and tardiness penalties. *Naval Research Logistics*, 1993, **40**, 229–243.
13. Szwarc, W. and Liu, J. J., Weighted tardiness single machine scheduling with proportional weights. *Management Science*, 1993, **39**, 626–632.
14. Vepsalainen, A. P. J. and Morton, T. E., Priority rules for job shops with weighted tardiness costs. *Management Science*, 1987, **33**, 1035–1047.