Journal of the Operational Research Society

ISSN: 0160-5682 (Print) 1476-9360 (Online) Journal homepage: https://www.tandfonline.com/loi/tjor20

New insights on the single machine total tardiness problem

B C Tansel & I Sabuncuoglu

To cite this article: B C Tansel & I Sabuncuoglu (1997) New insights on the single machine total tardiness problem, Journal of the Operational Research Society, 48:1, 82-89, DOI: 10.1057/palgrave.jors.2600321

To link to this article: https://doi.org/10.1057/palgrave.jors.2600321

Published online: 20 Dec 2017.

Journal of the Operational Research Society (1997) 48, 82–89

New insights on the single machine total tardiness problem

B C Tansel and I Sabuncuoglu

Bilkent University, Turkey

Virtually all algorithmic studies on the single machine total tardiness problem use Emmons' theorems that establish precedence relations between job pairs. In this paper, we investigate these theorems with a geometric viewpoint. This approach provides a compact way of representing Emmons' theorems and promotes better insights into dominance properties. We use these insights to differentiate between certain classes of easy and hard instances.

Keywords: single machine scheduling; tardiness

Introduction

In this paper, we analyze the total tardiness scheduling problem and identify some easy and hard instances using a geometric viewpoint. A single machine is to process n jobs with known processing times and due dates. Ready times are zero. Tardiness is the positive lateness a job incurs if it is completed after its due date and the object is to sequence the jobs to minimize the total tardiness. In the weighted case, each job's tardiness is multiplied by a positive weight.

The weighted tardiness problem is NP-hard in the strong sense (Lenstra *et al*[1]) and the unweighted case is NP-hard in the ordinary sense (Du and Leung[2]). The first thorough study of the problem was done by Emmons[3] in 1969. Emmons proved three fundamental theorems that helped establish precedence relations among job pairs that must be satisfied in at least one optimal schedule. These are generalized to arbitrary nondecreasing cost functions by Rinnooy Kan *et al*[4]. As Emmons' theorems have become widely known, most of the research on optimizing algorithms have used Emmons' theorems to first establish job precedences followed by some form of implicit enumeration (for example Fisher[5], and Schrage and Baker[6]).

In general, dynamic programming (DP) algorithms seem to be more efficient than branch and bound algorithms (Baker and Schrage[7]). One drawback of DP is its $O(2^n)$ storage requirements which renders it impractical for large problems. In this context, Sen, Austin, and Ghandforoush[8] proposed a more efficient implicit enumeration technique which is not based on Emmons' theorems. They compared their branching algorithm with the DP approach of Schrage and Baker and found that the proposed method requires less storage space. Recently, Sen and Borah[9] proposed a new branching algorithm based on theorems some of which are corollaries to Rinnooy Kan *et al*[4]. Their algorithm is effective in reducing the problem size by providing a smaller set of candidate sequences.

A fundamentally different line of theory, decomposition, was first developed by Lawler[10]. Decomposition theory focuses on relations between jobs and positions in a sequence rather than job pairs. Lawler[10] gave a pseudo-polynomial dynamic programming algorithm that relies on a repeated use of his decomposition theorem. In later work, Potts and van Wassenhove[11,12] refined and strengthened the decomposition principle and developed a dynamic programming algorithm.

There are also a number of heuristics developed in the literature. Heuristics range from simple dispatching rules to more sophisticated algorithms (Wilkerson and Irwin[13], Fry *et al*[14], Potts and van Wassenhove[15], Holsenback and Russel[16] and Panwalkar, Smith, and Koulamas[17]).

The paper is organized as follows. Section 1 introduces a geometric viewpoint and interprets Emmons' theorems. The interpretation of Theorem 1 leads to a class of easy instances which we present in Section 2. Next, we identify a class of hard instances and present them in Section 3. We end the paper with concluding remarks in Section 4.

Geometric interpretation of Emmons' theorems

In 1969 Emmons proved a number of fundamental results that establish precedence relations between job pairs. These results have frequently been used in subsequent work to design approximate or exact algorithms. Despite the fundamental importance of these results, their full impact does not seem to be too well understood. We now examine Emmons' results with a geometric viewpoint.

Let n jobs be given with p_i, d_i denoting, respectively, the processing time and due date for job i. We assume the indexing is done so that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$ and that tied p_j are broken by assigning the smaller index to smaller due dates. That is, $i < j$ implies either $p_i < p_j$ or $p_i = p_j$ with

Correspondence: BC Tansel, Department of Industrial Engineering, Bilkent University, 06533 Bilkent, Ankara, Turkey.

$d_i \leqslant d_j$. All processing times are positive whereas due dates can be negative or positive. Define $\bar{p}$ to be the sum of all processing times. A schedule S is a permutation of job indices $1, \ldots, n$. Given a schedule S, let $C_j(S)$ be the completion time of job j defined by the sum of processing times of all jobs preceding job j in S plus the processing time of job j. The tardiness of job j is $T_j(S) = \max\{0, C_j(S) - d_j\}$ and the total tardiness is $T(S) = \sum_j T_j(S)$. Let $\mathscr{S}$ be the set of all schedules. There are $n!$ distinct schedules in $\mathscr{S}$ and the problem is to find a schedule S^* such that $T(S^*) \leqslant T(S)\ \forall S \in \mathscr{S}$. We call S^* an optimal schedule and denote by $\mathscr{S}^*$ the set of all optimal schedules.

To state Emmons' theorems, let $j \leftarrow k$ stand for the phrase 'there exists at least one optimal schedule in which job j precedes job k.' For a nonempty subset B_k of $J \equiv \{1, \ldots, n\}$, let $B_k \leftarrow k$ mean that there exists an optimal schedule in which every job j in B_k precedes job k. Similarly, for a nonempty subset A_k of J, let $k \leftarrow A_k$ mean that there exists an optimal schedule in which every job j in A_k succeeds job k. If $B_k = \emptyset$ ($A_k = \emptyset$), the notation $B_k \leftarrow k$ ($k \leftarrow A_k$) means that no job is known to precede (succeed) job k in any optimal schedule. Note that $B_k \leftarrow k$ and $k \leftarrow A_k$ need not imply B_k, A_k are disjoint. If B_k, A_k are disjoint we say B_k, A_k are *compatible*. We refer to B_k as the *before set of k* and to A_k as the *successor set of k*.

Emmons' theorems in their original form are as follows, for any two jobs j and k with $j < k$,

Theorem 1

If (1) $B_k \leftarrow k$ and (2) $d_j \leqslant \max\{\sum_{B_k} p_i + p_k, d_k\}$, then $j \leftarrow k$,

Theorem 2

If (1) $k \leftarrow A_k$ and $B_k \leftarrow k$, (2) $d_j > \max\{\sum_{B_k} p_i + p_k, d_k\}$, and (3) $d_j + p_j \geqslant \sum_{J-A_k} p_i$, then $k \leftarrow j$,

Theorem 3

If (1) $j \leftarrow A_j$ and (2) $d_k \geqslant \sum_{J-A_j} p_i$, then $j \leftarrow k$.

Theorems 1 and 3 specify when a shorter job j precedes a longer job k while Theorem 2 specifies when a longer job k precedes a shorter job j. Various special cases of these theorems have also been stated as corollaries in the same paper. Fisher[5] showed that the condition $j < k$ of Theorem 3 can be replaced by $j \neq k$ without violating the theorem. Fisher also relaxed condition (2) of Theorem 2 to $d_j > d_k$ and showed that the theorem still holds with the relaxed condition.

Define $E_k = \sum_{B_k} p_i + p_k$ and $L_k = \sum_{J-A_k} p_i$ with dependence on B_k and A_k implicitly understood. We call E_k the *earliest completion* time of job k when $B_k \leftarrow k$ holds and call L_k the *latest completion* time of job k when $k \leftarrow A_k$ holds. $B_k = \emptyset$ implies $E_k = p_k$ while $A_k = \emptyset$ implies $L_k = \bar{p}$. Further, $p_k \leqslant E_k \leqslant L_k \leqslant \bar{p}$ for all earliest and latest completion times induced by compatible predecessor and successor sets of job k.

To give a geometric interpretation, let the data set be $Q \equiv \{(p_1, d_1), \ldots, (p_n, d_n)\}$ and plot the n points of the data set in the (p, d)-plane. The cluster of points may look like anything. To identify certain patterns in the 'shape' of the cluster, define the *connecting curve of Q* to be the piecewise linear function obtained by connecting the ith point to the $i+lst$ point by a straight line segment for $i = 1, \ldots, n-1$. Denote the connecting curve of Q by $C(Q)$. Draw a 45° line through the origin and call this line *the projection line* or simply *the line*. For any point (p_j, d_j), depending on if the point is *above*, *on*, or *below* this line, we have, respectively, $p_j < d_j$, $p_j = d_j$, or $p_j > d_j$. For any point (α, β), we define the *projection* of (α, β) to be the point $P(\alpha, \beta) \equiv$
$(\alpha, \max\{\alpha, \beta\})$. If (α, β) is below the line, its projection is obtained by moving the point vertically up until it meets the projection line. Points above or on the projection line are projected onto themselves. Figure 1 illustrates these definitions.

With each job k we associate a rectangle. Given p_k, E_k, d_k, the *enclosing rectangle of job k* is the rectangle whose northeast corner is at $(p_k, \max\{E_k, d_k\})$. The other corners are at (0, 0), $(p_k, 0)$, and $(0, \max\{E_k, d_k\})$. Figure 2 shows the enclosing rectangle for points below or above the line. We say that a point (p_j, d_j) is in the enclosing rectangle of job k if the point is in the interior or on the boundary of the enclosing rectangle of job k.

The geometric version of Theorem 1 is as follows.

Theorem 1′

For $j < k$, if data point (p_j, d_j) is in the enclosing rectangle of job k, then $j \leftarrow k$.

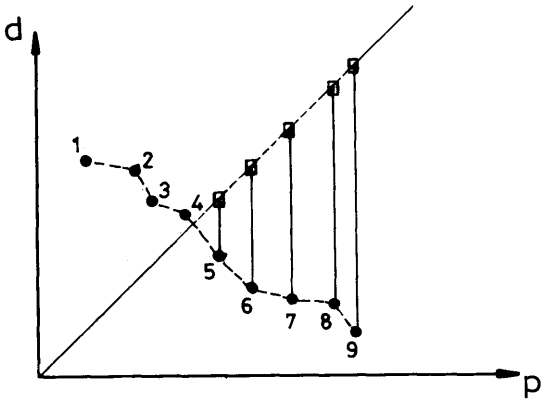

Figure 1 The connecting curve and the projection line.

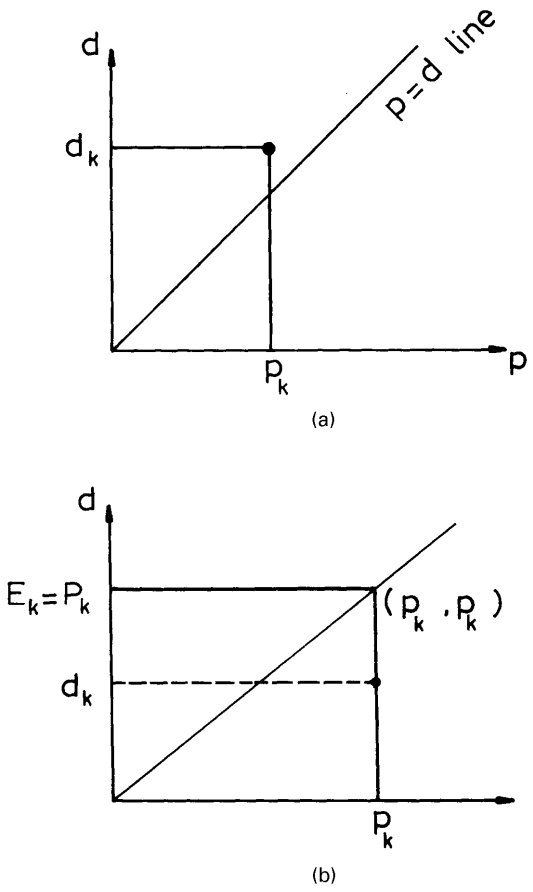

Figure 2 Enclosing rectangle of job *k*.

Proof

If (p_j, d_j) is in the enclosing rectangle of job k, each coordinate is no larger than the corresponding coordinate of the NE corner of the enclosing rectangle. This gives $p_j \leqslant p_k$ and $d_j \leqslant \max\{E_k, d_k\}$ which is equivalent to conditions (1) and (2) of Theorem 1. □

We can use Theorem 1′ repeatedly to establish precedence relations that may not be available from the initial data. Use the initial data to construct the enclosing rectangle for job k with $E_k = p_k$ initially. The NE corner is at $(p_k, \max(p_k, d_k))$. Let B_k be the set of indices $j < k$ for which (p_j, d_j) is in the enclosing rectangle of job k. Replace the old E_k by $E_k^{\text{new}} = \sum_{B_k} p_j + p_k$. Since $E_k^{\text{new}} \geqslant E_k^{\text{old}}$, this (possibly) expands the old rectangle of job k vertically to a new one with NE corner at $(p_k, \max\{E_k^{\text{new}}, d_k\})$. The new enclosing rectangle captures possibly more jobs j (with $j < k$) that were not captured before. This may make E_k even larger and cause another vertical expansion. The expansion procedure may be continued until no further expansion is possible. We then initiate the expansion procedure for another job k' and expand it as much as possible, then continue with another job and so on. When the procedure stops, all rectangles are expanded to their largest possible sizes. Either a total order is established that defines an optimal sequence or a partial order is established.

We now focus on a geometric interpretation of Theorem 2. This theorem specifically deals with the case when a shorter job j is outside the enclosing rectangle of a longer job k.

For each job j, define *job j's square* to be the square whose NE corner is positioned at $(p_j + d_j, p_j + d_j)$ and whose SW corner is at (0, 0). The geometric version of Theorem 2 is as follows.

Theorem 2′

For $j < k$, if job j's data point (p_j, d_j) is outside the enclosing rectangle of job k and (L_k, L_k) is inside the square of job j, then $k \leftarrow j$.

Proof

If (p_j, d_j) is outside the enclosing rectangle of job k, then $j < k$ implies $p_j \leqslant p_k$ so that d_j must be greater than the second coordinate of the NE corner of job k's enclosing rectangle. Since the NE corner is at $(E_k, \max\{E_k, d_k\})$, this gives $d_j > \max\{E_k, d_k\}$ implying that condition (2) of Theorem 2 is fulfilled. Since (L_k, L_k) is inside the square of job j, we have $L_k \leqslant p_j + d_j$. Hence, conditions (1), (2), (3) of Theorem 2 are equivalent to the stated conditions in Theorem 2′. □

Observe that the most favorable condition for Theorem 2′ to hold is when L_k is as small as possible. This occurs at the termination of the rectangular expansion procedure which uses Theorem 1′ repeatedly until no more expansion occurs in any of the enclosing rectangles. At this point, one switches to Theorem 2′ and checks unrelated pairs j, k to see if (L_k, L_k) is inside the square of job j. Whenever it is, j succeeds k in an optimal schedule. Whenever this occurs, Theorem 1′ can be used again for job j since job j's enclosing rectangle expands now due to the inclusion of job k in job j's before set. That is, E_j is replaced by $E_j + p_k$ which causes a vertical expansion of job j's rectangle (unless $E_j + p_k \leqslant d_j$). Note also that when j is found to succeed k on the basis of Theorem 2′, L_k gets smaller by the amount p_j. This slides the point (L_k, L_k) towards the origin along the projection line and this may cause the new point $(L_k - p_j, L_k - p_j)$ to be included in another square it was not included in earlier.

Finally, we may state a geometric version of Theorem 3 as follows.

Theorem 3′

For $j < k$, if the point (d_k, d_k) is no nearer to the origin than the point (L_j, L_j), then $j \leftarrow k$.

Proof

The stated condition is equivalent to conditions (1) and (2) of Theorem 3.

Again, the most favorable condition to use Theorem 3′ is when L_j attains its final (smallest) value at the end of repeated use of Theorems 1′ and 2′. We may assume without loss of generality that all due dates are strictly less than $\bar{p}$ as otherwise if $d_j \geqslant \bar{p}$ for some j, then job j will be early regardless of its position in the sequence so that it can be pushed to the end of the sequence without increasing total tardiness. All such jobs can be eliminated by a preprocessing of initial data. Under this assumption, Theorem 3′ cannot be initiated without the prior use of Theorems 1′
and 2′ since $L_j = \bar{p}\ \forall j$ at the beginning.

Easy problem instances

We now investigate certain data patterns that are easy to handle with this geometric viewpoint. One particularly easy case is what we call the SRD pattern, an acronym for *S*trong *R*ectangular *D*omination. We define the data set Q to be SRD if the connecting curve of Q is monotone nondecreasing. Figure 3 shows an SRD pattern with the connecting curve shown in dashed lines. If we draw a rectangle with NE corner positioned at (p_j, d_j), as shown in Figure 3, and call this rectangle the *jth rectangle*, we observe that data points 1 to $j-1$ are each contained in the jth rectangle for $j = 2, \ldots, n$. The enclosing rectangle defined by $(p_j, \max(E_j, d_j))$ is at least as large as the jth rectangle at (p_j, d_j), so Theorem 1′ is satisfied for all pairs j, k with $1 \leqslant j \leqslant k \leqslant n$. This gives a total order $1 \leftarrow 2$, $2 \leftarrow 3, \ldots, n-1 \leftarrow n$, which is the SPT sequence.

Note also that SPT and EDD sequences are identical for the SRD pattern. We have:

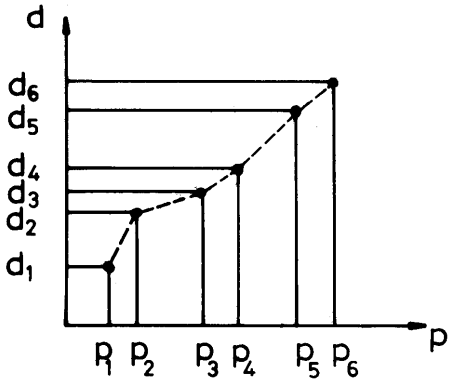

Figure 3 SRD pattern.

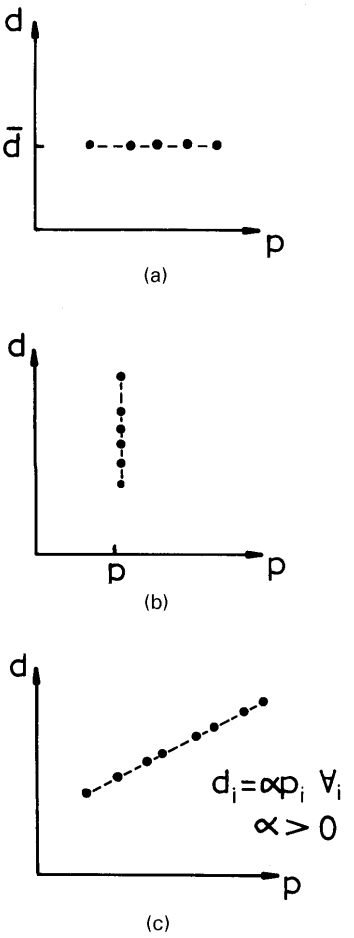

Figure 4 Special cases of SRD: (a) common due dates, (b) common processing times, and (c) due dates proportional to processing times.

SRD Rule: For all instances with SRD pattern, the SPT(EDD) sequence is an optimal schedule.

Some special cases of SRD are the cases with *common due dates* $(d_i = d\ \forall i)$, *common processing times* $(p_i = p\ \forall i)$, and *due dates proportional to processing times* ($d_i = \alpha p_i\ \forall i$ for some positive α). These are shown in Figure 4. In all these cases SPT(EDD) solves the problem.

A more general, nevertheless, equally easy case is obtained by relaxing the monotone nondecreasing requirement on the connecting curve. This relaxation is equivalent to allowing SPT sequence to be different from EDD sequence. Figure 5 shows what we call an ERD pattern (*E*xtended *R*ectangular *D*omination). We define the data set Q to be ERD if the transformed data set $P(Q) \equiv \{(p_i, \max\{p_i, d_i\}): i = 1, \ldots, n\}$ is SRD. The transformation $P(\cdot)$ moves each point (p_j, d_j) *below* the projection line vertically up to the corresponding point (p_j, p_j) *on* the projection line. Points on or above the projection

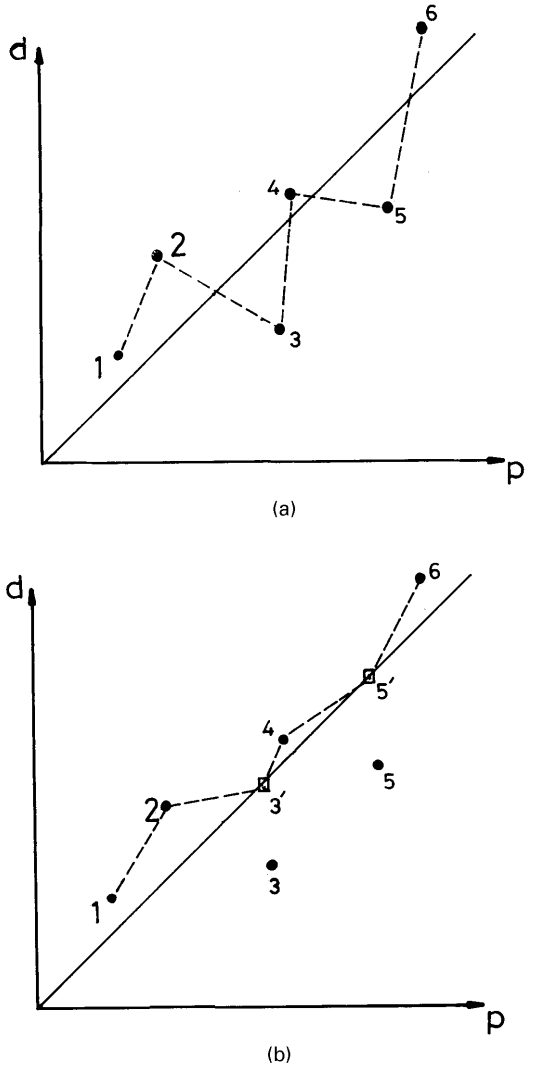

Figure 5 ERD pattern: (a) original data, and (b) transformed data.

line remain intact. Observe that if we place a rectangle at each transformed point, we get the enclosing rectangles. If the data is ERD, then the first $k-1$ original data points are contained in the kth enclosing rectangle. This again defines a total order $i \leftarrow i+1$ for $i = 1, \ldots, n-1$. Hence, we have

ERD Rule: For all problem instances with ERD pattern, the SPT sequence is an optimal schedule (the EDD sequence may or may not be identical to the SPT sequence for the ERD pattern).

For example, the data of Figure 5(a) has a connecting curve which is not monotone nondecreasing, so this data is not SRD. However, the transformed data, shown in Figure 5(b), has a monotone nondecreasing connecting curve. So it satisfies the ERD rule. This implies that the problem defined by the data of Figure 5(a) is optimally solved by the SPT sequence.

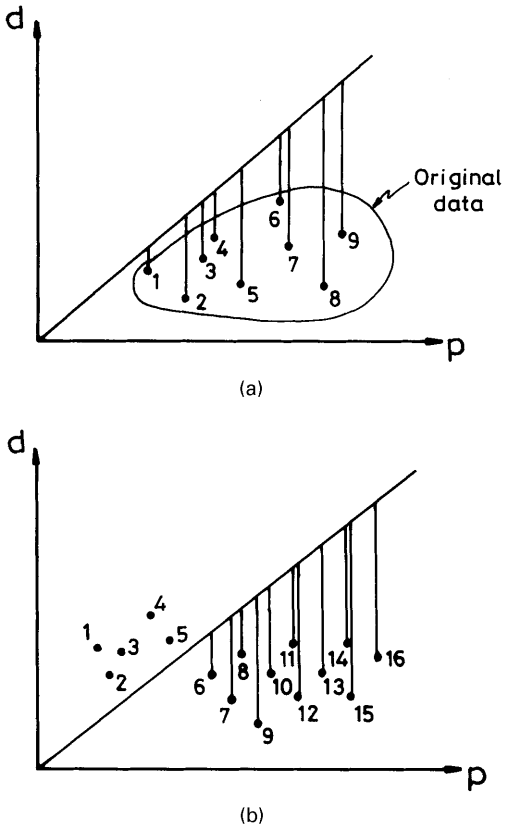

Figure 6 Transformed data: (a) all points below the line (total order), and (b) some points below the line (partial order).

It is clear that whenever a given data is SRD, it is also ERD. The converse is not true in general. Hence, the SRD pattern is a special case of the ERD pattern.

It is evident that the due dates of points below the line have no significance in the construction of their enclosing rectangles. What matters is their processing times. The larger their processing times, the more (original) data points they will capture in their enclosing rectangles. One special case occurs when all points are below the line. In this case, the transformed points define a total order which is again SPT (see Figure 6a).

This illustrates that SPT is optimal when all jobs are tardy in all schedules, observed earlier in Baker[18]. If not all points are below the line, as in Figure 6(b), a total order may or may not emerge. Even if it does not, an initial partial order is obtained by moving points below the line to corresponding points on the line. For example, the data of

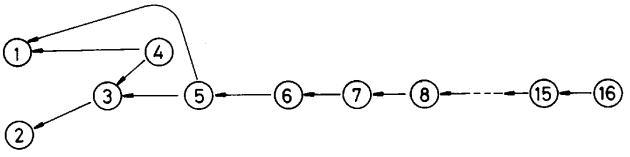

Figure 7 Partial order from data of Figure 6b.

Figure 6(b) gives the partial order shown by a directed graph in Figure 7.

Further evidence for the idea of moving points below the line to their new positions on the line (namely replacing d_i by p_i whenever $d_i < p_i$) can be found in the literature. For example, Holsenback and Russel[16] use modified due dates defined by $\max(d_i, p_i)$ to improve the algorithmic efficiency of their proposed heuristic. In a different context (the early/tardy problem), Rachamadugu[19] gives theorems in which negative slacks (defined by $d_i - (p_i + t)$ where t is a reference time point) are replaced by zeros. This is equivalent to changing d_i to $p_i + t$ whenever $d_i - (p_i + t) < 0$. ERD pattern addresses the same situation by moving points below the line to their new positions on the line.

Hard problem instances

The question of what makes a particular problem hard or easy is certainly not an easy one to answer. Algorithms that work quite well on a class of instances may fail to do so in other instances. In this respect, characterization of instances that appear to defy efficient solution methods is a worthwhile undertaking. In our context, we define an instance of the total tardiness problem to be a *hard instance* (*with respect to Emmons' theorems*) if the data set Q is such that any attempted use of Emmons' theorems meets with failure. Characterization of such data sets amounts to finding the conditions under which none of Emmons' theorems can be initiated. We now focus on these conditions.

Define the data set $Q = \{p_1, d_1), \ldots, (p_n, d_n)\}$ to be NRD (*N*o *R*ectangular *D*omination) if the connecting curve $C(P(Q))$ of the transformed data set $P(Q)$ is strictly decreasing. It is direct to show that $C(P(Q))$ is strictly decreasing iff $C(Q - \{p_n, d_n\})$ is strictly decreasing and $d_{n-1} > \max\{p_n, d_n\}$. That is, the only way a data set Q to be NRD is by having at most one point, namely the last point (p_n, d_n), to be below or on the line such that the connecting curve of the first $n-1$ original points and the last transformed point is strictly decreasing. Observe that if Q penetrates into the region below the line with more than one point, the projections of these points to the line will destroy the strictly decreasing property of $C(P(Q))$. The next theorem shows that the NRD data completely characterizes the failure of Theorem 1.

Theorem 4

Theorem 1′ cannot be initiated on the data set Q if Q is NRD.

Proof

(Necessity). Let the data set Q be NRD. Pick any pair j, k with $j < k$. The indexing convention implies either $p_j < p_k$ or $p_j = p_k$ with $d_j \leqslant d_k$. The latter case cannot occur because $j < k \leqslant n$ implies that job j's data point is above the line (see the paragraph preceding the theorem) so that $d_j > p_j = p_k$ contradicting the fact that $C(P(Q))$ is strictly decreasing (i.e. $\max\{p_j, d_j\} > \max\{p_k, d_k\}$). In the former case, the fact that $C(P(Q))$ is strictly decreasing and $p_j < p_k$ imply $\max\{p_j, d_j\} > \max\{p_k, d_k\}$. But $j < n$ implies job j's data point is above the line so that $d_j = \max\{p_j, d_j\}$. With $p_j < p_k$ and $d_j > \max\{p_k, d_k\}$, data point (p_j, d_j) is outside the enclosing rectangle of job k. Thus, Theorem 1′ cannot be used with the initial data.

(Sufficiency). Assume Theorem 1′ cannot be initiated. Then for any pair j, k with $j < k$, *it must be the case that* (p_j, d_j) is outside the enclosing rectangle of job k. This rectangle has its NE corner at $(p_k, \max\{p_k, d_k\})$ since the predecessor set of job k is null so that $E_k = p_k$. The fact that job j's data point is outside the enclosing rectangle of job k implies $p_j \leqslant p_k$ while $d_j > \max\{p_k, d_k\}$. It follows that for $j = 1, \ldots, n-1$, we have $d_j > \max\{p_{j+1}, d_{j+1}\}$ while $p_j \leqslant p_{j+1}$. This ensures that the connecting curve $C(P(Q))$ of the transformed data is strictly decreasing. Hence, the data set Q is NRD. □

We now focus on conditions which make both theorems 1′ and 2′ fail. The next theorem supplies the required conditions.

Theorem 5

Both Theorems 1′ and 2′ fail if the data set Q is NRD and $d_j + p_j < \bar{p}\ \forall j = 1, \ldots, n-1$.

Proof

To prove the sufficiency, assume Q is NRD and $d_j + p_j < \bar{p}\ \forall j$, $j \neq n$. Theorem 4 implies Theorem 1′ fails and the connecting curve of the transformed data is strictly decreasing. Thus, $d_j > \max\{p_k, d_k\}$ for all pairs j, k with $j < k$. Since Theorem 1′ cannot be used on its own, the predessor sets are null for all jobs before any attempted use of Theorem 2′. This implies $L_j = \bar{p}\ \forall j$. For $j < k$, the only way for $(L_k, L_k) = (\bar{p}, \bar{p})$ to be inside the square with NE corner at $(d_j + p_j, d_j + p_j)$ is by having $\bar{p} \leqslant d_j + p_j$. This is not possible due to the assumption of the theorem. Hence Theorem 2′ also fails.

To prove the necessity, assume the data is such that neither Theorem 1′ nor Theorem 2′ can be initiated. Hence all predecessor sets are null. Property 2 implies that Q is NRD so that for $j < k$, job j's data point (p_j, d_j) is outside the enclosing rectangle of job k. Hence, the only way Theorem 2′ cannot be initiated is by having $d_j + p_j < L_k = \bar{p}$. In particular, this is true for $k = n$ and $1 \leqslant j < n$ completing the proof. □

As was pointed out earlier at the end of section 2, the failure of Theorem 3′ with the initial data need not be

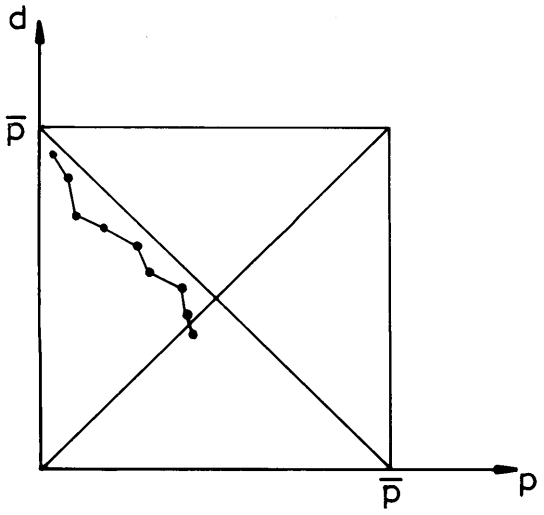

Figure 8 An illustrative hard instance.

considered (under the assumption $d_j < \bar{p}\ \forall j$) since this theorem cannot be initiated unless one of Theorems 1′ or 2′ have already been initiated. This leads to

Theorem 6

The data set Q defines a hard instance with respect to Emmons' theorems iff Q is NRD and $d_j + p_j < \bar{p}$ for $j = 1, \ldots, n-1$. □

Figure 8 illustrates a typical hard instance. Emmons noticed this particular hard pattern and demonstrated it with a four job example which he referred to as a 'perverse' situation. It appears that any exact or heuristic method must test itself on hard instances (characterized in Theorem 6) to substantiate any claims of success. To our knowledge, this has rarely been done in the computational literature most likely owing to the fact that the probability of randomly generating such an instance is extremely low. For example, a Monte Carlo simulation with 3, 5, 8 jobs and uniformly distributed data yields probabilities of hard instances in the neighborhood of 0.0016, 0.002, 0.00001, respectively. For more than 8 jobs, the probability is virtually zero (for example less than 0.000001 for $n = 10$).

We find it appropriate to caution the reader against a possible misunderstanding of the word 'hard' in the sense we are using it here. The fact that an instance is hard in the sense defined above does not imply that instances that do not obey the conditions of Theorem 6 are easy. It simply means that, if one wants to make judgments on how difficult a particular instance is by looking at the initial data alone, one can do so by simply testing the two conditions given in Theorem 6. Hence, 'hard' is used in the sense 'impossible to initiate the dominance rules proposed by Emmons' rather than 'hard in the sense that any other instance is solvable in polynomial time.'

Conclusion

The key result that emerges from our analysis in the paper is that it is possible to characterize certain classes of easy and hard instances by inspecting the geometry of data. The easy instances identified in the paper can be uniformly characterized by the monotone nondecreasing property of the connecting curve of the transformed data. This helps to unify various piecemeal results observed earlier in the literature. The hard instances identified in the paper are characterized by the monotone decreasing property of the connecting curve of the data with the additional restriction that all but one of the points are on or above the 45° line through the origin. The probability of occurrence of a hard instance in randomly generated problems is extremely low for more than 10 jobs. It appears that such an instance is also nontypical in the real world.

There are certainly other data instances which permit the initiation of Emmons' theorems and lead to a partial order which may or may not be easy to solve by a subsequent exact method. If there are many job pairs that are not related in the established partial order, the subsequent branch and bound tree will have many nodes (candidate sequences) that cannot be pruned immediately on the basis of established precedence relations. Hence, passing judgments (on the basis of the initial data alone) about the computational difficulty of instances that do not obey the conditions of Theorem 6 seems to be a rather difficult task. It may be appropriate to apply Emmons' theorems to see what the resulting partial order is to make meaningful predictions on the computational demands of the subsequent implicit enumeration.

Acknowledgement—We thank an anonymous referee for providing us with constructive comments which improved the initial draft.

References

1 Lenstra JK, Rinnooy Kan AHG and Brucker P (1977). Complexity of machine scheduling problems, *Annals of Disc Math* **1**: 343–362.
2 Du J and Leung JY-T (1990). Minimizing total tardiness on one machine is NP-hard, *Math Opns Res* **15**: 483–495.
3 Emmons H (1969). One-machine sequencing to minimize certain functions of job tardiness, *Opns Res* **17**: 701–715.
4 Rinnooy Kan AHG, Lageweg BJ and Lenstra JK (1975). Minimizing total costs in one machine scheduling, *Opns Res* **23**: 908–927.
5 Fisher ML (1976). A dual algorithm for the one machine scheduling problem, *Math Prog* ;**11**: 229–251.
6 Schrage LE and Baker KR (1978). Dynamic programming solution of sequencing problem with precedence constraints, *Opns Res* **26**: 444–449.
7 Baker KR and Schrage LE (1978). Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks, *Opns Res* **26**: 111–120.
8 Sen TT, Austin LM and Ghandforoush P (1983). An algorithm for the single machine sequencing problem to minimize total

tardiness, *IIE Transactions* **15**: 363–366.
9 Sen TT and Borah BN (1991). On the single machine scheduling problem with tardiness penalties, *J Opl Res Soc* **42**: 695–702.
10 Lawler EL (1977). A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness, *Annals of Disc Math* **1**: 331–342.
11 Potts CN and Van Wassenhove LN (1982). A decomposition algorithm for the single machine total tardiness problem, *Opns Res Letters* **11**: 177–181.
12 Potts CN and Van Wassenhove LN (1987). Dynamic programming and decomposition approaches for the single machine total tardiness problem, *Eur J Opl Res* **32**: 405–414.
13 Wilkerson LJ and Irwin JD (1971). An improved algorithm for scheduling independent tasks, *AIIE Transactions* **3**: 239–245.
14 Fry TD, Vincent L, Macleod K and Fernandez S (1989). A heuristic solution procedure to minimize mean tardiness on a single machine, *J Opl Res Soc* **40**: 293–297.
15 Potts CN and Van Wassenhove LN (1991). Single machine tardiness sequencing heuristics, *IIE Transactions* **23**: 346–354.
16 Holsenback JE and Russell RM (1992). A heuristic algorithm for sequencing on one machine to minimize total tardiness, *J Opl Res Soc* **43**: 53–62.
17 Panwalkar SS, Smith ML and Koulamas CP (1993). A heuristic for the single machine tardiness problem, *Eur J Opl Res* **70**: 304–310.
18 Baker KR (1974). *Introduction to Sequencing and Scheduling.* Wiley: New York.
19 Rachamadugu R (1995). Scheduling jobs with proportionate early/tardy penalties, *IIE Transactions* **27**: 679–682.

Received November 1994;
accepted May 1996 after one revisions.