



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

SCIENCE @ DIRECT®

Computers & Operations Research 31 (2004) 1727–1751

computers &  
operations  
research

[www.elsevier.com/locate/dsw](http://www.elsevier.com/locate/dsw)

# A branch-and-bound algorithm for the early/tardy machine scheduling problem with a common due-date and sequence-dependent setup time

Ghaith Rabadi<sup>a,\*</sup>, Mansooreh Mollaghasemi<sup>b</sup>, Georgios C. Anagnostopoulos<sup>c</sup>

<sup>a</sup>Department of Engineering Management & Systems Engineering, Old Dominion University, Norfolk, VA 23529, USA

<sup>b</sup>Department of Industrial Engineering & Management Systems, University of Central Florida, Orlando, FL 32816, USA

<sup>c</sup>Department of Electrical Engineering and Computer Engineering, Florida Institute of Technology, Melbourne, FL 32901-6975, USA

## Abstract

The single-machine early/tardy (E/T) scheduling problem is addressed in this research. The objective of this problem is to minimize the total amount of earliness and tardiness. Earliness and tardiness are weighted equally and the due date is common and large (unrestricted) for all jobs. Machine setup time is included and is considered sequence-dependent. When sequence-dependent setup times are included, the complexity of the problem increases significantly and the problem becomes NP-hard. In the literature, only mixed integer programming formulation is available to optimally solve the problem at hand. In this paper, a branch-and-bound algorithm (B&B) is developed to obtain optimal solutions for the problem. As it will be shown, the B&B solved problems three times larger than what has been reported in the literature.

© 2003 Elsevier Ltd. All rights reserved.

*Keywords:* Single-machine scheduling; Earliness/tardiness; Early/tardy; Sequence-dependent setup times; Common due date; Branch-and-bound

## 1. Introduction

The problem considered in this paper is scheduling a set of  $n$  jobs  $\{j_1, j_2, \dots, j_n\}$  on a single machine that is capable of processing only one job at a time without preemption. All jobs are available at time zero, and a job  $j$  requires processing time  $P_j$ . Machine setup time  $S_{ij}$  is included

\* Corresponding author. Tel.: +1-757-683-4918; fax: +1-757-683-5640.

E-mail address: [grabadi@odu.edu](mailto:grabadi@odu.edu) (G. Rabadi).

as sequence dependent. That is, the amount of machine setup required if job  $i$  precedes  $j$  may be different from when job  $j$  precedes  $i$ . The objective is to complete all the jobs as close as possible to a large, common due date  $d$ . To accomplish this objective, the summation of earliness and tardiness is minimized. The earliness of job  $j$  is defined as  $E_j = \max(0, d - C_j)$  and its tardiness as  $T_j = \max(0, C_j - d)$ , where  $C_j$  is the completion time of job  $j$ . Earliness and tardiness penalties for job  $j$  are weighted equally. The objective function is given by

$$\min Z = \sum_{j=1}^n (E_j + T_j) = \sum_{j=1}^n |d - C_j|. \quad (1)$$

This problem became more popular after the introduction of JIT manufacturing philosophy where jobs are desired to complete as close as possible to their due dates. The earliness criterion captures the holding or deterioration cost of the finished jobs, and the tardiness criterion accounts for the cost of customer compensation for missing the due date or the loss of goodwill. For example, a common due date is compatible with the notion of producing components to be assembled into a finished product. Another application would be meeting a committed shipping due date for an order that consists of a set of products.

The single-machine E/T problem was first introduced by Kanet [1]. Since then many researchers worked on various extensions of the problem. Baker and Scudder [2] published a comprehensive state-of-the-art review for different versions of the E/T problem. The literature reviewed in this section will be limited to exact algorithms developed to solve the single-machine E/T problem with a common due date. Special consideration will be given to the research that considered equal earliness and tardiness penalties, common due date, and sequence-dependent setup time.

Kanet [1] examined the E/T problem with equal penalties and unrestricted common due date. A problem is considered unrestricted when the due date is large enough not to constrain the scheduling process. He introduced a polynomial-time algorithm to solve the problem optimally. Hall [3] extended Kanet's work and developed an algorithm that finds a set of optimal solutions for the problem based on some optimality conditions. Hall and Posner [4] solved the weighted version of the problem with no setup times. Azizoglu and Webster [5] introduced a B&B to solve the problem with setup times; however, they assumed that setup times are not sequence dependent. Other researchers worked on the same problem but with a restricted (small) due date (see for example Bagchi et al. [6], Szwarc [7], Szwarc [8], Hall et al. [9], Alidaee and Dragan [10], and Mondal and Sen [11]). None of the previous papers considered sequence-dependent setup times.

In most of the E/T literature, it has been assumed that no setup time is required. In many realistic situations, however, setup times are needed and are sequence-dependent. In general, scheduling problems with sequence-dependent setup times are similar to the travelling salesman problem (TSP), which is NP-hard [12]. Coleman [13] presented a 0/1 mixed integer programming model (MIP) for the single-machine E/T problem with job-dependent penalties, distinct due dates, and sequence-dependent setup times. By solving Coleman's MIP, optimal solutions were found for problems with up to eight jobs. Coleman's work was one of the few papers that dealt with the E/T problem with sequence-dependent setup times, but for a small number of jobs. Chen [14] addressed the E/T problem with batch sequence-dependent setup times. He showed that the problem with unequal penalties is NP-hard even when there are only two batches of jobs and two due dates that are unrestrictively large. Allahverdi et al. [15] reviewed the scheduling literature that involved setup times. In their review, very few papers addressed the E/T problem with setup times, and no paper

tackled the problem addressed in this research. In this paper, the problem addressed is similar to that introduced by Kanet [1] (i.e., unrestricted common due date and equal E/T penalties) with the addition of the sequence-dependent setup times. The sequence-dependency changes the complexity of the problem from solvable in polynomial time to NP-hard.

This paper is organized as follows. In Section 2, optimality properties for the problem are discussed. Section 3 presents the notation used throughout the paper. The B&B and its implementation are discussed in Sections 4 and 5, respectively. In Sections 6, a computational experience to study the effectiveness of the B&B and its elements is presented. Finally, Section 7 summarizes the conclusions and future research.

## 2. Properties for the E/T problem

### 2.1. Properties for the problem without setup times

The single-machine E/T problem with unrestricted common due date and no setup times has an elegant structure based on the following optimality properties:

- (i) In an optimal schedule, there will be no idle time inserted.
- (ii) There is an optimal schedule in which one job completes exactly on the due date.
- (iii) In an optimal schedule, the  $b$ th job in the sequence completes on the due date, where

$$b = \begin{cases} \frac{n}{2} \text{ or } \frac{n}{2} + 1 & \text{if } n \text{ is even,} \\ \frac{n+1}{2} & \text{if } n \text{ is odd.} \end{cases} \quad (2)$$

- (iv) An optimal schedule is V-shaped. This means that non-tardy jobs are sequenced in LPT order (longest processing time first) and tardy jobs are sequenced in SPT order (shortest processing time first).

Kanet [1] proved the previous four properties based on which he introduced an algorithm that finds an optimal solution for the problem in polynomial time.

### 2.2. Properties for the problem with sequence-dependent setup times

When sequence-dependent setup times are introduced into the problem, not all of the previous properties hold. Property (i) still holds, simply because removing any gaps from the schedule will make the jobs' completion times closer to the due date, and thus, the objective function value will be reduced. Properties (ii) and (iii) also hold and their proofs are included in Appendix A. Property (iv), however, does not hold when sequence-dependent setup times are included. Trying an example of a small size problem with a known optimal solution can easily prove that the optimal solution is not V-shaped. In fact, if this property held, the problem would be solvable in polynomial time using Kanet's algorithm.

From a mathematical perspective, complexity, properties, and structure of the problem addressed in this research remain the same if the processing times and the setup times are combined in one

matrix. This makes instances of the problem unique, and more convenient to study and experiment with. Thus, we define this combined matrix as the adjusted processing time  $[AP]$  where

$$\text{for } j = 1, \dots, n: \quad AP_{ij} = S_{ij} + P_j \quad \text{for } i = 1, \dots, n. \tag{3}$$

A similar approach was used by Gendreau et al. [16]. Throughout this paper, the adjusted processing times will be used as defined by (3) instead of separate processing times and the setup times. Note that property (iv) does not hold for the adjusted processing times either.

### 3. Notation

In addition to the basic notation mentioned earlier (i.e.,  $n, P_j, C_j, d, S_{ij}, E_j, T_j, AP_{ij}$ ), the following notation will be used in the forthcoming sections:

- $A$  the ordered set of tardy jobs, i.e.,  $\{j \mid C_j > d\}$  ordered by  $C_j$
- $AP$  the adjusted processing time matrix
- $b$  the job that completes on the due date as given by (2)
- $B$  an ordered set of non-tardy jobs, i.e.,  $\{j \mid C_j \leq d\}$  ordered by  $C_j$
- $MAP_j$  the minimum adjusted processing time for job  $j$  where for  $j = 1, \dots, n$   

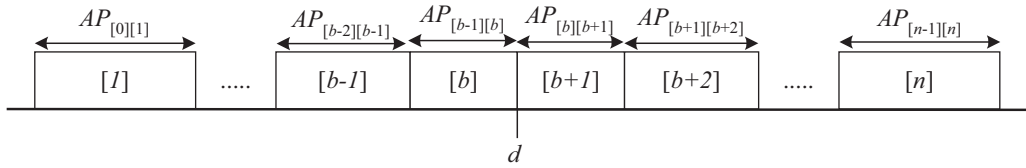
$$MAP_j = \min_{\substack{i=1, \dots, n \\ i \neq j}} \{AP_{ij}\}$$
- $MAP$  vector of minimum adjusted processing times for all jobs
- $S_P$  partial sequence: an ordered set of jobs already assigned in the sequence
- $N_P$  cardinality of  $S_P = |S_P|$
- $S_{PB}$  an ordered set of the non-tardy jobs within  $S_P$ ; where  $e = |S_{PB}|$
- $S_{PA}$  an ordered set of the tardy jobs within  $S_P$ ; where  $t = |S_{PA}|$
- $S_B$  an ordered set of the unscheduled non-tardy positions =  $B - S_{PB}$
- $S_A$  an ordered set of the unscheduled tardy positions =  $A - S_{PA}$
- $S_U$  set of unscheduled jobs =  $\{j \mid j \notin S_P\}$
- $E_{Actual}$  earliness cost for  $S_{PB}$  based on  $AP_{ij}$  (i.e., based on *actual*  $AP_{ij}$ )
- $E_{Min}$  earliness cost for  $S_B$  based on  $MAP_j$  (i.e., based on *minimum*  $AP_{ij}$ )
- $T_{Actual}$  tardiness cost for  $S_{PA}$  based on  $AP_{ij}$  (i.e., based on *actual*  $AP_{ij}$ )
- $T_{Min}$  tardiness cost for  $S_A$  based on  $MAP_j$  (i.e., based on *minimum*  $AP_{ij}$ )
- $TE$  total earliness for a job sequence
- $TT$  total tardiness for a job sequence

Using  $AP_{ij}$  notation, the objective function given in (1) can be restated as follows:

Objective  $Z = TE + TT$ ,

where

$$TE = \sum_{j=1}^b (j - 1)AP_{[j-1][j]} = 0AP_{[0][1]} + 1AP_{[1][2]} + 2AP_{[2][3]} + \dots + (b - 1)AP_{[b-1][b]}, \tag{4}$$



[j]: Job in position j  
 $AP_{[j][j+1]}$ : Adjusted Processing Time for the job in position j followed by the job in position [j+1]  
 [b]: The median position

Fig. 1. Reasoning for Eqs. (4) and (5).

$$\begin{aligned}
 TT &= \sum_{j=b}^{n-1} (n-j)AP_{[j][j+1]} \\
 &= 1AP_{[n-1][n]} + 2AP_{[n-2][n-1]} + \dots + (n-b-1)AP_{[n-b-1][n-b]} + (n-b)AP_{[b][b+1]}. \quad (5)
 \end{aligned}$$

Fig. 1 gives a pictorial explanation of Eqs. (4) and (5). When  $i = 0$ , i.e., for the first job in the sequence,  $AP_{[0][1]} = P_1$  assuming no setup time is required for the first job in the sequence (i.e.,  $S_{01} = 0$ ). Even if  $S_{01} \neq 0$ , the optimal sequence will not be affected as can be seen from (4) where  $AP_{[0][1]}$  is multiplied by a coefficient of zero.

Note that in both Eqs. (4) and (5),  $d$  is not needed; hence, the objective function value does not depend on it as long as  $d$  is large. An exact lower bound ( $\Delta$ ) on how large  $d$  must be to be considered unrestricted cannot be calculated beforehand due to sequence-dependency. Therefore, to know if  $d$  is unrestricted, an optimal solution has to be found first, the schedule has to be shifted to start from zero, and then  $\Delta$  is calculated where  $\Delta = \sum_{j=1}^b AP_{[j-1][j]}$ . If  $d \geq \Delta$ , then the obtained solution is optimal; otherwise, optimality is not guaranteed. If  $d < \Delta$ , the problem becomes restricted.

#### 4. The branch-and-bound algorithm

In any B&B, three major procedures are involved: *initialization*, *branching* and *bounding*. During *initialization*, fast heuristics are usually employed to find a good initial solution. This solution serves as an upper bound (*UB*) for the problem until a better solution is found. This helps in eliminating (or fathoming) any nodes that have a lower bound (*LB*) worse than that *UB*. *Branching* partitions the problem into smaller sub-problems. Each sub-problem represents a partial solution and is represented by a node. A search strategy must be associated with the branching scheme. This strategy decides which node to branch next. The *bounding* procedure is used to calculate a *LB* at each node considered for branching to help eliminating nodes. If the *LB* is worse (higher in the case of a minimization problem) than the best solution obtained so far, the node is eliminated because a better complete solution can never be reached in that case. In the following sections, the three procedures are customized for the E/T problem and described in more detail.

#### 4.1. Initialization (upper bound)

During this phase, an initial complete solution is found until a better solution is obtained. This initial solution is referred to as *UB*. Any node with a *LB* worse than the *UB* is eliminated. In this paper, the shortest adjusted processing time first (*SAPT*) heuristic developed by Rabadi [17] is used. In an optimal job sequence, jobs with shorter adjusted processing times tend to be scheduled closer to the median position, and those with longer adjusted processing times away from the median position. This is consistent with (4) and (5) where the adjusted processing times for jobs closer to the median of the schedule are multiplied by a higher coefficient. The *SAPT* heuristic is based on this concept and it consists of two phases: (I) schedule construction, and (II) schedule improvement. Phase I starts by selecting jobs  $i^*$  and  $j^*$  with the smallest entry in [*AP*] and placing them in positions  $b - 1$  and  $b$ , respectively. Another two jobs with the next smallest [*AP*] entry before  $i$  or after  $j$  are then scheduled either after job  $j$  or before job  $i$  depending on which location gives a lower objective function value. This selection process is repeated until all jobs are scheduled. It is important to maintain feasibility throughout the scheduling process by eliminating the selected [*AP*] entries and their corresponding rows and columns to avoid scheduling conflicts. This procedure is repeated  $n$  times for the  $n$  smallest entries in [*AP*]. The rationale behind this repetition is that the optimal schedule will most likely include in its median position a job corresponding to a small entry of [*AP*], but not necessarily the smallest one.

In Phase II, the *general pairwise interchange* (*GPI*), a neighborhood search method, is used to improve the schedule. In *GPI*, any two jobs (not just adjacent) may be swapped starting by swapping the job in the first position with the succeeding jobs one at a time until the  $n$ th position. Then, it continues by swapping the job in the second position with the succeeding jobs until the  $n$ th position, and so on until the last two jobs in the sequence are swapped for improvement. For  $n$  jobs, the neighborhood would consist of  $n(n - 1)/2$  sequences. If the swapping of any two jobs improves the schedule, they are left in their current position and *GPI* continues with the next job. For more detail on *GPI* and *SAPT*, see Rabadi [17].

#### 4.2. Branching and search strategy

Branching is the procedure used to develop the search tree. A general branching scheme for B&Bs starts at the top of the tree (level 0), where no jobs have been assigned to any position in the sequence. At level 1,  $n$  branches are created to  $n$  nodes. For each node in this level, a specific job is assigned to a certain position in the sequence. This means that for each node there are  $n - 1$  jobs, whose position in the sequence is not determined. For example, with  $n = 4$ , at level 0 there will be one node with no jobs assigned. This is denoted as (\*\*\*\*), where \* represents a wild card that can take any job number. At level 1 of the tree, 4 nodes are created. At each node, a different job will be assigned to a certain position in the sequence. The most common way of assigning jobs to positions in most B&B branching strategies is to assign jobs to the first or last position in the sequence. This is not necessarily the best job-assigning method for the E/T problem. A job is typically assigned to the position that contributes the most to the value of the *LB*. High *LB* values are desirable because of the increased chances for the node to be eliminated early in the search process. For the E/T problem, and based on Eqs. (4) and (5), jobs in the median positions will most likely contribute the most to the objective function value because they are multiplied by the largest coefficients,  $b - 1$

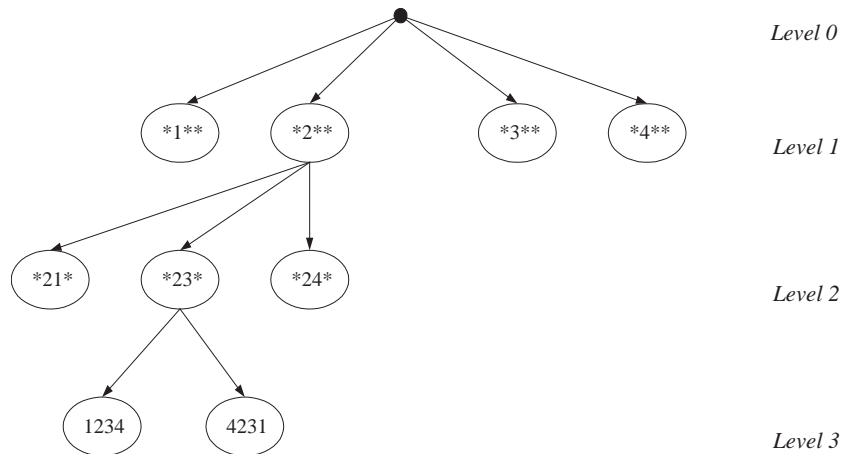


Fig. 2. The B&amp;B approach.

and  $n - b$ . Since the value of the  $LB$  will be based on the same equations, then it is more suitable to assign jobs starting from the median of the sequence going outwards. This way, high  $LB$  values will be attained at high levels of the search tree; thus, the B&B should be more efficient.

Property (iii) guarantees that in an optimal sequence one of the jobs completes on  $d$ , which coincides with the median position. The median position will be  $b = n/2$  if  $n$  is even and  $(n + 1)/2$  if  $n$  is odd. Therefore, for  $n = 4$ , the median position would be the second position. So, at level 1,  $S_p$  for the first node would be (\*1\*\*), for the second would be (\*2\*\*), and so on as shown in Fig. 2. Going from level 1 to level 2,  $(n - 1)$  arcs are spawned from the selected node at level 1. Therefore, there will be a maximum of  $n(n - 1)$  nodes at level 2. At each node in level 2, two jobs are assigned to the two positions in the middle of the sequence. In our example, if the second node is to be branched, the following nodes will be created: (\*21\*), (\*23\*) and (\*24\*) (see Fig. 2). The same rationale applies here as well, where jobs closer to the middle of the sequence contribute more to the  $LB$  value. This procedure is continued until the last level is reached. At that level, a complete solution will be obtained. For the example at hand, the last level will be level 3, and if the second node (\*23\*) is branched, two complete solutions will be generated: (1234) and (4231). The one with the minimum objective function value  $Z_{\min}$  is selected. If  $Z_{\min} < UB$ , then  $UB$  is set equal to  $Z_{\min}$ .

To avoid a full enumeration of the tree, a bounding procedure is applied to find a  $LB$  for each node. For a certain node  $i$  at level  $k$ , if  $LB_{ki} \geq UB$ , then this node is eliminated. Obtaining a  $LB$  at each node is not only useful in fathoming nodes, but also in guiding the search. In this B&B, at each level of the tree, the node with the smallest  $LB$  is followed since it is a reasonable indication for an optimal solution to reside under that sub-tree. The derivation of the  $LB$  is discussed in the following section.

#### 4.3. Bounding: lower bound derivation for the E/T problem

In this section, a lower bound algorithm for a given partial sequence  $S_p$  is presented. Kanet [1] introduced an algorithm to optimally solve the E/T problem without setup times assuming that

Table 1  
Example of processing and setup times ( $S_{i,j}$ )

$j$	1	2	3	4
$p_j$	50	60	90	70
	1	2	3	4
1	—	30	50	90
2	40	—	20	80
3	30	30	—	60
4	20	15	10	—

$d \geq \sum_{j=1}^n p_j$ . Kanet's algorithm can be presented in simple terms as given by Baker [18]:

*Step 1:* Assign the longest job to  $B$ .

*Step 2:* Find the next two longest jobs. Assign one to  $B$  and one to  $A$ .

*Step 3:* Repeat Step 2 until there are no jobs left, or until there is only one job left, in which case, assign this job to either  $A$  or  $B$ .

*Step 4:* Finally, order the jobs in  $B$  according to LPT and the jobs in  $A$  according to SPT.

When sequence-dependent setup times are included, the previous algorithm does not remain optimal. However, the solution provided by this algorithm can be utilized to obtain a  $LB$ . That is, the optimal solution for the problem with zero setup time can be used as a  $LB$  for the problem with setup times. This  $LB$  can be further improved by adding the minimum amount of setup time for each job to its processing time. The rationale of the proposed  $LB$  can be easily seen: the optimal solution's objective function value for the problem with zero setup time will always be smaller than that with setup times. This is because the setup time can only worsen the solution by increasing the deviations of the completion times from the due date. It is also easy to prove that adding the minimum amount of setup times to the processing time will improve the  $LB$ , because each job requires some amount of setup time. This amount would not be known beforehand, but it could not be less than the minimum amount of setup time regardless of the position of that job. Since it has been proven that the algorithm presented by Kanet [1] is optimal for the problem without setup time, then by the same token, Kanet's algorithm (including minimum amount of setup time) is a  $LB$  for the problem under investigation.

Consider the example given in Table 1. Job  $j_1$  requires 50 units of processing time, and setup time of 40, 30, or 20 depending on whether it is preceded by job  $j_2$ ,  $j_3$ , or  $j_4$  respectively. Regardless of what job precedes  $j_1$ , it will require at least 20 units of setup time. The  $LB$  algorithm utilizes this amount of minimum setup time to improve the quality of the  $LB$  value. For this example,  $[AP]$  is given in Table 2.

### LB Algorithm for the E/T problem

To calculate the lower bound  $LB1$  for a partial sequence  $S_p$ , execute the following:

*Step 1:* Find  $N_p$ .

*Step 2:* If  $N_p$  is even, then:

Table 2  
[AP] for the example in Table 1

<i>j</i>	1	2	3	4
1	—	90	140	160
2	90	—	110	150
3	80	90	—	130
4	70	75	100	—

Step 2.1: Partition  $S_P$  to  $S_{PB}$  and  $S_{PA}$  where:

$$S_B = S_A = \phi, \quad e = t = |S_{PB}| = |S_{PA}| = N_P/2.$$

Step 2.2: Find job  $j$  with the longest  $MAP$  in  $S_U$  (i.e.,  $\max_{j \in S_U} \{MAP_j\}$ ) and assign it to  $S_B$ . Remove  $j$  from  $S_U$ .

Step 2.3: Find the next two jobs  $i, k \in S_U$  with the longest  $MAP$ , and assign one to  $S_B$  and the other to  $S_A$ . Remove  $i, k$  from  $S_U$ .

Step 2.4: Repeat Step 2.3 until  $S_U = \phi$  or until  $|S_U| = 1$ , in which case assign the remaining job to  $S_A$ .

Step 3: If  $N_P$  is odd, then:

Step 3.1: Partition  $S_P$  to  $S_{PB}$  and  $S_{PA}$  where:

$$S_B = S_A = \phi; e = |S_{PB}| = (N_P + 1)/2; t = |S_{PA}| = (N_P - 1)/2$$

Step 3.2: Find job  $j$  with the longest  $MAP$  in  $S_U$  (i.e.,  $\max_{j \in S_U} \{MAP_j\}$ ) and assign it to  $S_B$ . Remove  $j$  from  $S_U$ .

Step 3.3: Find job  $j$  with the longest  $MAP$  in  $S_U$  and assign it to  $S_A$ . Remove  $j$  from  $S_U$ .

Step 3.4: Repeat Steps 3.2 and 3.3 until  $S_U = \phi$  or until  $|S_U| = 1$ , in which case assign the remaining job to  $S_A$ .

Step 4: Order the jobs in  $S_B$  according to (longest minimum adjusted processing (LMAP) time first), and the jobs in  $S_A$  according to (shortest minimum adjusted processing (SMAP) time first).

Step 5: Calculate the total earliness ( $E\_cost$ ) for the sequence:

Step 5.1: If ( $e > 1$ ) Then  $E_{Actual} = \sum_{j=b-e+2}^b (j-1)AP_{[j-1][j]}$ ; else  $E_{Actual} = 0$ .

Step 5.2: Insert the first job in  $S_{PB}$  in the last position in  $S_B$ .<sup>1</sup>

Step 5.3: Calculate  $E_{Min} = \sum_{j=1}^{b-e+1} (j-1)MAP_{[S_{B_j}]}$ .

Step 5.4:  $E\_cost = E_{Actual} + E_{Min}$ .

Step 6: Calculate the total tardiness ( $T\_cost$ ) for the sequence:

Step 6.1: If ( $t = 0$ ) Then  $T_{Actual} = T_{Min} = 0$ ; Go to Step 6.4.

Step 6.2:  $T_{Actual} = \sum_{j=b}^{b+t-1} (n-j)AP_{[j][j+1]}$ .

Step 6.3:  $T_{Min} = \sum_{j=1}^{b-t} (n-b-j)MAP_{[S_{A_j}]}$ .

Step 6.4:  $T\_cost = T_{Actual} + T_{Min}$ .

Step 7:  $LB1 = E\_cost + T\_cost$ .

Step 8: Return ( $LB1$ ).

<sup>1</sup> Although the first job in  $S_{PB}$  is scheduled already, the job that precedes it is unknown. Hence, it is inserted in  $S_B$  to include its  $MAP$  value in the  $LB$  and not its  $AP$  value.

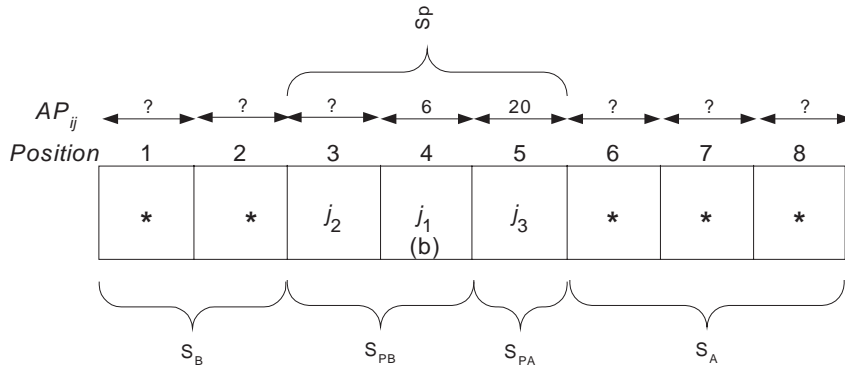


Fig. 3. Example of LB1 calculation of a partial sequence (\*\*213\*\*).

Table 3  
[AP] for an example with  $n = 8$

$j$	1	2	3	4	5	6	7	8
1	—	5	20	13	(7)	11	12	12
2	6	—	12	(4)	22	(8)	9	16
3	14	13	—	15	9	10	22	17
4	10	(3)	(7)	—	17	13	14	21
5	(5)	5	23	16	—	9	(2)	19
6	11	19	8	6	15	—	5	11
7	25	7	20	19	10	15	—	(10)
8	17	8	18	19	22	9	6	—

Table 4  
Minimum adjusted processing times  $MAP$  for an example with  $n = 8$

$J$	1	2	3	4	5	6	7	8
$MAP_j$	5	3	7	4	7	8	2	10

**Example of the LB algorithm**

To show how LB1 is calculated for a partial sequence such as (\*\*213\*\*) shown in Fig. 3, where  $n = 8$  and [AP] as given in Table 3, calculate  $MAP$  first by taking the minimum entry from each column in the [AP]. These values are in parentheses in Table 3, and are listed in Table 4. From the given data, the following is obtained:

$$b = n/2 = 4; S_p = \{j_1, j_2, j_3\}, |S_p| = 3; S_u = \{j_4, j_5, j_6, j_7, j_8\}.$$

LB1 algorithm is applied as follows:

Step 1:  $N_p = |S_p| = 3$ .

Step 2: If  $N_p$  is odd then go to Step 3.

Step 3:

Step 3.1:  $S_B = S_A = \phi; e = |S_{PB}| = (3 + 1)/2 = 2; t = |S_{PA}| = (3 - 1)/2 = 1.$

Step 3.2:  $\max_{j \in S_U} \{MAP_j\} = \max\{MAP_4, MAP_5, MAP_6, MAP_7, MAP_8\} = \max\{4, 7, 8, 2, 10\} = 10 \rightarrow j = j_8, S_B = \{j_8\}, S_U = \{j_4, j_5, j_6, j_7\}.$

Step 3.3:  $\max_{j \in S_U} \{MAP_j\} = \max\{MAP_4, MAP_5, MAP_6, MAP_7\} = \max\{4, 7, 8, 2\} = 8, j = j_6, S_A = \{j_6\}, S_U = \{j_4, j_5, j_7\}.$

Step 3.4: Repeat Steps 3.2 and 3.3:

$$S_B = \{j_8, j_5\}, S_U = \{j_4, j_7\},$$

$$S_A = \{j_6, j_4\}, S_U = \{j_7\}, |S_U| = 1 \rightarrow \text{assign } j_7 \text{ to } S_A \rightarrow S_A = \{j_6, j_4, j_7\}.$$

Step 4: Order  $S_B$  according to LMAP:  $S_B = \{j_8, j_5\}$ , order  $S_A$  according to SMAP:  $S_A = \{j_7, j_4, j_6\}.$

Step 5: Calculate  $E\_cost$ :

Step 5.1:  $e = |S_P| = 2$ , then  $E_{Actual} = \sum_{b=e+2}^b (j-1)AP_{[j-1][j]} = \sum_{4-2+2}^4 (j-1)AP_{[j-1][j]} = 3AP_{[3][4]} = 3(6) = 18.$

Step 5.2: Last job in  $S_B$  is  $j_2 \rightarrow S_B = \{j_8, j_5, j_2\}.$

Step 5.3:

$$\begin{aligned} E_{Min} &= \sum_{j=1}^{4-2+1} (j-1)MAP_{[S_{B_j}]} \\ &= 0MAP_{[S_{B_1}]} + 1MAP_{[S_{B_2}]} + 2MAP_{[S_{B_3}]} \\ &= 0 + 1(7) + 2(3) = 13. \end{aligned}$$

Step 5.4:  $E\_cost = E_{Actual} + E_{Min} = 31.$

Step 6: Calculate  $T\_cost$ :

Step 6.1:  $t = 1.$

Step 6.2:  $T_{Actual} = \sum_{j=4}^{4+1-1} (n-j)AP_{[j][j+1]} = (8-4)AP_{[4][5]} = 4(20) = 60.$

Step 6.3:

$$\begin{aligned} T_{Min} &= \sum_{j=1}^{4-1} (8-4-j)MAP_{[S_{A_j}]} \\ &= 3MAP_{[S_{A_1}]} + 2MAP_{[S_{A_2}]} + 1MAP_{[S_{A_3}]} \\ &= 3(8) + 2(4) + 1(2) = 34. \end{aligned}$$

Step 6.4:  $T\_cost = T_{Actual} + T_{Min} = 60 + 34 = 94.$

Step 7:  $LB = E\_cost + T\_cost = 31 + 94 = 125.$

Step 8: Return (LB1).

## 5. Implementation and verification

The B&B was implemented using C++ on a SUN Microsystems Enterprise 4000 workstation running SunOS 5.5. Later on, it was ported to a 450 MHz PC running Microsoft Windows 98.

To verify that the B&B produces optimal solutions, a set of problems were solved using 0/1 mixed integer programming (MIP). Coleman [13]'s MIP is slightly modified to optimally solve the problem as follows:

$$\min Z = \sum_{j=1}^n (E_j + T_j) \quad (6)$$

s.t.

$$C_j \geq P_j + S_{jj}, \quad (7)$$

$$C_j - T_j + E_j = d, \quad (8)$$

$$C_j - C_i + M(1 - Y_{ij}) \geq P_j + S_{ij}, \quad i = 1, \dots, n; \quad j = i + 1, \dots, n, \quad (9)$$

$$C_i - C_j + M(Y_{ij}) \geq P_i + S_{ji}, \quad i = 1, \dots, n; \quad j = i + 1, \dots, n \quad (10)$$

$$C_j, E_j, T_j \geq 0, \quad (11)$$

$$Y_{ij} \in \{0, 1\}, \quad (12)$$

where all variables are defined as before and  $Y_{ij}$  is 1 if job  $i$  precedes job  $j$ , and 0 otherwise and  $M$  the large positive number.

Note that for the previous MIP to be valid, the setup times have to satisfy the triangular inequality  $S_{ij} + S_{jk} \geq S_{ik} \forall 1 \leq i, j, k \leq n$ . Problems with up to 11 jobs and triangular setup times were generated and solved optimally. The B&B developed in this paper, however, does not rely on this triangular inequality assumption and it can find an optimal solution for arbitrary setup times (and thus, for arbitrary [AP]). The B&B was applied to the same set of problem instances solved using the MIP. The B&B reached the same optimal solutions obtained by the MIP.

## 6. Computational experience

### 6.1. Comparison between the B&B and the MIP

The E/T MIP presented in Section 5 was modeled using AMPL, and solved using CPLEX 6.0. CPLEX as well as other MIP solvers use built-in B&Bs that find LBs by linear programming relaxation. For more information on AMPL and CPLEX, please see Fourer et al. [19] and ILOG CPLEX [20]. Data were generated from uniform discrete distributions similar to those used by Balakrishnan et al. [21] since they used a similar model to solve the E/T problem in a parallel machine environment, where  $P_j \sim U[5, 200]$ ,  $S_{ij} \sim U[25, 75]$ , and  $d$  was large integer chosen arbitrarily. [AP] was not used in this part of the study because the MIP requires the setup times to be triangular, and not both processing times and setup times. Since the setup times were generated randomly, they needed to be corrected to satisfy the triangle inequality. To do that, the Floyd–Warshall algorithm (F–W) was utilized. The F–W finds the shortest path between all pairs of nodes in a directed graph [22]. Data for problems with number of jobs ranging from 6 to 11 jobs were

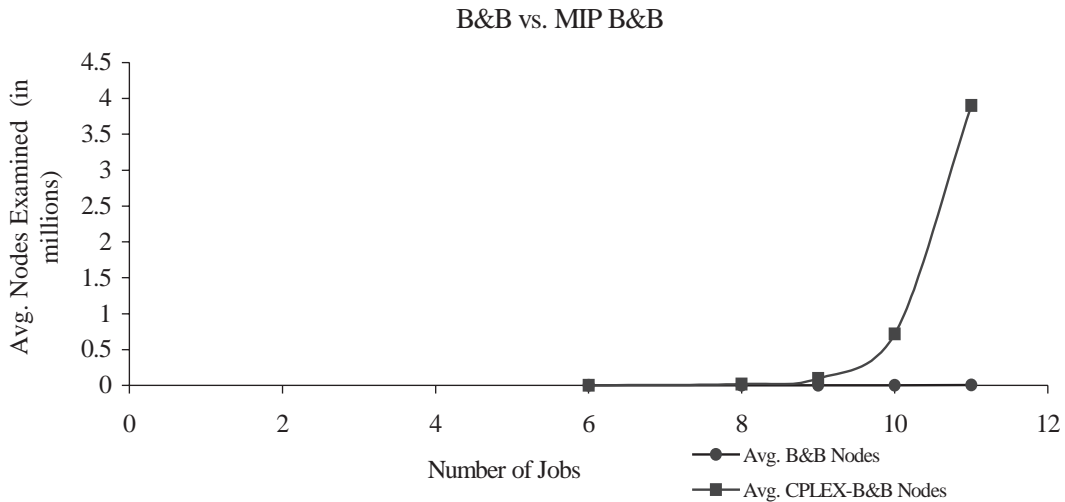


Fig. 4. Comparison between the performance of the CPLEX MIP solver and the B&B for the E/T problem.

generated. The time needed to solve some problems with 11 jobs using CPLEX was more than 1.5 hour, which made the MIP impractical for solving larger problems. For each problem size, 5 instances were generated and solved. The same set of problems was solved using the B&B developed in this research, which obtained the same optimal solutions as the MIP solver with CPU of one second or less per instance. Fig. 4 compares the performance of the solver's built-in B&B to the B&B developed in this research. The number of nodes examined was used to compare between both approaches. It is obvious that the B&B developed in this paper outperforms that used by the MIP solver. The significant difference between both is obvious and is due to the fact that the B&B developed here was tailored to the E/T problem, while the MIP used a generic B&B.

### 6.2. Effectiveness of the lower bound

There does not exist another lower bound for this problem to compare our lower bound  $LB1$  with. Therefore, we compare it with another lower bound ( $LB2$ ) that includes only the contribution of the actual  $AP_{ij}$  values in a partial sequence to the objective function. This can be accomplished simply by executing and adding the results of Steps 5.1 and 6.2 in the lower bound algorithm discussed in Section 4.3. Using the same example presented in Section 4.3:

$$LB2(**213**) = E_{\text{Actual}} + T_{\text{Actual}} = (4 - 1)(6) + (3)(20) = 78.$$

There are two reasons for selecting  $LB2$  to compare with. First, since it does not include the properties presented by Kanet's algorithm, we can measure how much the inclusion of Kanet's algorithm is benefiting the B&B. Second, one could argue that  $LB2$  is simpler but faster to calculate than  $LB1$ , and thus, it could be more effective.

The experiments were conducted on problem sizes that ranged from 10 to 20 jobs. For each problem size, 15 instances were solved with  $[AP]$  randomly generated from three uniform distributions

[10, 60], [10, 110], and [10, 160] for low, medium, and high ranges, respectively (a total of 495 instances). All instances were solved using *LB1* one time and *LB2* another. The maximum number of jobs was limited to 20 in this part of the study because under some settings, it became very time consuming to solve larger instances. The number of nodes branched was used as a measure of effectiveness. The more nodes branched, the less effective the lower bound is. Other B&B research papers, such as Liaw [23], have used the same measure of performance. Table 5 summarizes the average percentage reduction in nodes branched due to the use of *LB1* by calculating the average ratio of  $[Nodes(LB2) - Nodes(LB1)] \times 100 / Nodes(LB2)$ . Note that for all instances, SAPT was used as an *UB*, and the branching scheme was to branch at the median position. From the percentage of node reduction, it is obvious that *LB1* is very tight compared to *LB2*. In terms of CPU time, the last column of Table 5,  $CPU(LB2) - CPU(LB1)$  shows that the use of *LB1* is more efficient. One can conclude that although *LB2* is faster to compute, the quality of *LB1* increases the effectiveness of the B&B. It is worth mentioning also that *LB1* does not seem to be sensitive to the range factor in contrast to *LB2* where *LB2* became effective as  $[AP]$  range became larger. Since *LB2* depends solely on the actual  $AP_{ij}$  values, as  $[AP]$  range becomes larger,  $AP_{ij}$  have higher chances to be larger. This increases the chances for *LB2* values to become larger and thus more nodes to be eliminated early in the search process. Nevertheless, *LB1* remains superior to *LB2*.

### 6.3. Effectiveness of the upper bound

To test the effectiveness of SAPT, which served as an *UB* for the B&B, two measures were used. First, the percentage relative deviations of the SAPT solutions from the optimal solutions, and second the number of times the SAPT heuristic reached an optimal solution. The same set of 495 instances used in testing the lower bound was used to test the *UB*. The results are reported in Table 6. The minimum and maximum deviations from the optimal solutions are also reported. The average percentage deviations of the SAPT's solutions from the optimal for low, medium and high ranges were 3.54%, 5.54%, and 6.3%, respectively. The average deviation for all three ranges across all problem sizes was 5.13%, which is good considering the complexity of the problem. Another remark regarding the results is that as the range was increased, the percentage deviation also increased. As the  $[AP]$  range increases, the effect of sequence-dependency becomes more significant. Consider for example an extreme case of a very small  $[AP]$  range. In this case, the significance of the sequence-dependency diminishes because all entries will be almost the same, and so, regardless of the job sequence, the optimal solution would not differ much.

### 6.4. Effectiveness of the branching strategy

To test the effectiveness of the strategy of branching at the median position, it was compared to another branching strategy that branched at the beginning of the sequence (i.e., from the left end of the sequence). A partial sequence, at level 3 for an instance with 7 jobs for example would be (123\*\*\*\*) compared to (\*\*123\*\*) when branching at the median position. Note that a branching strategy from the right end of the sequence is the same as that from the left end of it because earliness and tardiness are weighted equally. The set of 450 instances used earlier were solved once using the median branching strategy and using the left branching strategy another. In both cases, the lower bound was fixed to *LB1* (as calculated in Section 4.3), and the *UB* was obtained via

Table 5  
Lower bound effectiveness test results

	<i>n</i>	Avg. nodes branched × 1000 (with <i>LB1</i> )	Avg. <i>CPU</i> (min)	Avg. nodes branched × 1000 (with <i>LB2</i> )	Avg. <i>CPU</i> (min)	Avg.% node reduction	Avg. <i>CPU</i> reduction (min)
Low range	10	0.209	0.00	5.633	0.00	96.29	0.00
	11	0.395	0.00	15.816	0.00	97.50	0.00
	12	0.879	0.00	52.422	0.03	98.32	0.03
	13	1.300	0.00	173.574	0.13	99.25	0.13
	14	2.491	0.00	433.450	0.36	99.43	0.36
	15	3.334	0.00	1048.128	0.96	99.68	0.95
	16	11.594	0.03	4171.211	4.02	99.72	3.99
	17	11.760	0.04	9913.149	10.45	99.88	10.42
	18	24.977	0.10	40474.649	45.20	99.94	45.10
	19	19.291	0.08	68574.920	83.75	99.97	83.67
	20	65.088	0.33	— <sup>a</sup>	— <sup>a</sup>		
Average			0.05		14.49	99.90	14.47
Med range	10	0.287	0.00	3.328	0.00	91.38	0.00
	11	0.493	0.00	7.607	0.00	93.52	0.00
	12	1.001	0.00	19.031	0.01	94.74	0.01
	13	1.421	0.00	58.165	0.04	97.56	0.04
	14	1.800	0.00	89.179	0.07	97.98	0.07
	15	2.552	0.00	226.273	0.20	98.87	0.20
	16	10.522	0.03	738.418	0.71	98.58	0.69
	17	7.772	0.02	1439.854	1.57	99.46	1.54
	18	15.814	0.06	3959.081	4.51	99.60	4.45
	19	21.725	0.09	9252.372	11.20	99.77	11.11
20	43.957	0.22	22722.650	29.23	99.81	29.01	
Average		9.758	0.04	3501.451	1.83	97.39	4.28
High range	10	0.273	0.00	2.427	0.00	88.77	0.00
	11	0.452	0.00	4.864	0.00	90.70	0.00
	12	1.018	0.00	14.144	0.00	92.81	0.00
	13	0.975	0.00	23.943	0.01	95.93	0.01
	14	2.372	0.00	68.677	0.05	96.55	0.05
	15	3.504	0.00	154.287	0.14	97.73	0.13
	16	5.204	0.01	269.255	0.27	98.07	0.25
	17	11.337	0.05	958.920	1.01	98.82	0.97
	18	18.912	0.08	1752.680	1.99	98.92	1.91
	19	26.612	0.12	3959.078	4.83	99.33	4.71
20	46.170	0.24	6966.720	9.12	99.34	8.89	
Average		10.621	0.05	1288.636	0.83	96.09	1.54

<sup>a</sup>The algorithm was stopped due to very long computation.

SAPT. From the summarized results in Table 7, it is clear from the percentage node reduction that branching at the median position gives a great advantage to the B&B as it eliminates many more nodes. CPU times are also reported.

Table 6  
Upper bound (SAPT) effectiveness test results

	<i>n</i>	Min % dev	Avg. % dev	Max % dev	No. opt sol
Low [ <i>AP</i> ] range U[10,60]	10	0.00	2.70	11.46	7
	11	0.00	3.32	6.78	2
	12	0.00	3.16	8.91	4
	13	0.00	3.97	9.14	2
	14	0.00	2.23	5.73	4
	15	1.09	4.36	9.42	0
	16	0.12	4.25	7.48	0
	17	0.00	3.47	8.03	1
	18	0.45	4.26	7.33	0
	19	0.00	2.49	6.66	1
	20	0.88	4.76	8.87	0
Average		0.23	3.54	8.16	
Med [ <i>AP</i> ] range U[10,110]	10	0.00	6.49	24.11	4
	11	0.00	4.11	12.43	2
	12	0.00	6.20	19.08	3
	13	0.00	5.97	14.43	1
	14	0.00	4.19	9.67	4
	15	0.00	6.78	15.38	2
	16	0.00	4.19	7.30	2
	17	0.00	5.11	10.83	1
	18	0.00	5.46	14.79	1
	19	3.58	7.65	16.10	0
	20	0.00	4.82	14.94	1
Average		0.33	5.54	14.46	
High [ <i>AP</i> ] range U[10,160]	10	0.00	3.44	10.09	6
	11	0.00	6.86	13.94	1
	12	0.00	6.08	14.74	3
	13	0.00	3.80	13.75	4
	14	0.00	6.39	14.31	1
	15	0.00	6.38	16.75	2
	16	0.00	8.15	18.88	1
	17	0.00	5.39	9.97	1
	18	0.55	8.42	17.33	0
	19	0.54	7.77	14.40	0
	20	0.00	6.63	13.10	1
Average		0.10	6.30	14.30	

### 6.5. Experimental design for the B&B

A full factorial experimental design was conducted to test the overall performance of the B&B. The factors considered in the experiments are the following:

- (1) *n*: The number of jobs is the most natural factor to use as it decides the problem size.

Table 7  
Branching strategy effectiveness test results

	<i>n</i>	Avg. nodes branched × 1000 (at median)	Avg. CPU (min)	Avg. nodes branched × 1000 (from left)	Avg. CPU (min)	% Node reduction	CPU reduction (min)
Low range	10	0.209	0.00	0.532	0.00	60.67	0.00
	11	0.395	0.00	1.748	0.00	77.41	0.00
	12	0.879	0.00	5.046	0.01	82.59	0.01
	13	1.300	0.00	8.938	0.02	85.45	0.02
	14	2.491	0.00	28.059	0.07	91.12	0.07
	15	3.334	0.00	31.701	0.09	89.48	0.09
	16	11.594	0.03	170.828	0.58	93.21	0.55
	17	11.760	0.04	245.289	0.96	95.21	0.93
	18	24.977	0.10	1029.421	4.53	97.57	4.44
	19	19.291	0.08	735.970	3.70	97.38	3.62
	20	65.088	0.33	4069.652	22.62	98.40	22.28
Average		12.847	0.05	575.198	2.96	88.04	2.91
Med range	10	0.287	0.00	0.898	0.00	68.07	0.00
	11	0.493	0.00	2.216	0.00	77.77	0.00
	12	1.001	0.00	5.448	0.01	81.62	0.01
	13	1.421	0.00	10.156	0.02	86.01	0.02
	14	1.800	0.00	15.653	0.04	88.50	0.04
	15	2.552	0.00	24.241	0.07	89.47	0.07
	16	10.522	0.03	168.496	0.59	93.76	0.56
	17	7.772	0.02	185.638	0.74	95.81	0.72
	18	15.814	0.06	361.143	1.62	95.62	1.56
	19	21.725	0.09	853.516	4.24	97.45	4.14
	20	43.957	0.22	2570.248	14.27	98.29	14.05
Average		9.758	0.04	381.605	1.96	88.40	1.92
High range	10	0.273	0.00	0.849	0.00	67.91	0.00
	11	0.452	0.00	1.878	0.00	75.92	0.00
	12	1.018	0.00	5.223	0.01	80.52	0.01
	13	0.975	0.00	7.750	0.01	87.41	0.01
	14	2.372	0.00	28.228	0.07	91.60	0.07
	15	3.504	0.00	34.244	0.10	89.77	0.10
	16	5.204	0.01	55.996	0.20	90.71	0.18
	17	11.337	0.05	199.828	0.82	94.33	0.77
	18	18.912	0.08	523.314	2.36	96.39	2.28
	19	26.612	0.12	1074.816	5.44	97.52	5.31
	20	46.170	0.24	2691.364	15.07	98.28	14.83
Average		10.621	0.05	420.317	2.19	88.21	2.14

(2) The  $[AP]_{range} = R = \max[AP] - \min[AP]$ : This factor was selected because it affects how long or short  $AP_{ij}$  values are. Based on the problem properties, the length of these times may affect the difficulty of the problem. This factor also helps in testing the B&B for different averages of adjusted processing times.

Table 8  
Factors and their levels in the B&B experiments

Factor		Value	Setting or level
Number of jobs,	$n$	10 Jobs	10
		15 Jobs	15
		20 Jobs	20
		25 Jobs	25
[ $AP$ ] range,	$R$	[10, 60]	Low (Low)
		[10, 110]	Medium (Med)
		[10, 160]	High (High)
Upper bound,	$UB$	UB exists	Yes
		UB exists	No

- (3) The existence of an  $UB$ : In any B&B, an initial solution (i.e.,  $UB$ ) is used to eliminate more nodes. It will be interesting to know if this factor significantly affects the performance of the algorithm. If this is the case, then using a good  $UB$  will significantly reduce the amount of computation. Thus, more attention should be given to improving the quality of this initial solution.

The number of nodes branched was again selected as a measure of performance. For best performance of the B&B and based on the tests performed earlier in this paper,  $LB1$  was used as lower bound,  $SAPT$  was used as an  $UB$  and branching was done at the median position. After running the B&B for a few sets of randomly generated data, it turned out that the algorithm solved instances with less than 10 jobs without difficulty. After the number of jobs was increased to more than 25 jobs, longer computational times were observed for most of the cases. All factors considered in this experiment along with their levels are presented in Table 8. The number of treatments for this experimental design is  $4 \times 3 \times 2 = 24$ . The number of randomly generated instances per factor-level combination was 15. The total number of replicates is  $15 \times 24 = 360$ . Table 9 summarizes the results of the experiment including CPU times.

### 6.6. Analysis of variance (ANOVA)

To identify the significant factors and determine whether there are any significant interactions among them, analysis of variance (ANOVA) was carried out. Results of this analysis are given in Table 10. Based on the  $p$ -value for the whole model, the model is significant since the  $p$ -value is very small. This means that at least one of the selected factors and/or their interactions have a significant effect on the variability when going from one level to another. Interactions higher than second order are considered as a part of the residual.

At significance level of 5% (i.e.,  $\alpha = 0.05$ ), the only factor is the number of jobs with a  $p$ -value  $< 0.0001$ . From this, one can conclude that regardless of what range of [ $AP$ ] is chosen, the B&B will, on average, behave the same. One can also conclude that even if the initial solution provided by the  $UB$  is optimal, the B&B will still take its time to investigate the search tree. The importance of  $n$  is quite intuitive to explain since the number of possible solutions (permutations) grows exponentially

Table 9  
Experiment results for the B&B

<i>n</i>	<i>R</i>	<i>UB</i>	Nodes branched × 1000			CPU (min)		
			Min.	Avg.	Max.	Min.	Avg.	Max.
10	Low	Yes	0.059	0.35793	0.771	0.0	0.0	0.0
		No	0.164	0.43933	0.88	0.0	0.0	0.0
	Med	Yes	0.034	0.36693	0.842	0.0	0.0	0.0
		No	0.077	0.43793	0.854	0.0	0.0	0.0
	High	Yes	0.061	0.39533	0.811	0.0	0.0	0.0
		No	0.132	0.44	0.848	0.0	0.0	0.0
15	Low	Yes	1.79	10.8862	54.341	0.0	0.03	0.12
		No	2.027	11.82307	54.346	0.0	0.03	0.12
	Med	Yes	1.424	5.80627	18.592	0.0	0.02	0.12
		No	1.424	6.79067	18.592	0.0	0.02	0.12
	High	Yes	1.314	5.91613	19.899	0.0	0.02	0.12
		No	1.325	7.0838	19.899	0.0	0.02	0.12
20	Low	Yes	7.566	80.83193	385.975	0.0	0.28	1.12
		No	7.869	89.2177	429.761	0.0	0.30	1.12
	Med	Yes	23.251	151.2383	552.816	0.08	0.54	2.12
		No	25.616	162.1006	639.764	0.08	0.55	2.12
	High	Yes	7.384	105.6769	494.258	0.00	0.39	1.12
		No	9.254	112.3564	498.907	0.00	0.40	1.12
25	Low	Yes	84.777	1908.47	8447.156	0.43	18.10	172.12
		No	84.808	1993.788	8717.485	0.47	18.83	173.12
	Med	Yes	197.427	2380.678	17236.83	0.92	11.20	78.12
		No	213.226	2405.977	17241.54	0.97	11.50	79.12
	High	Yes	187.12	2092.852	11483.99	0.95	9.93	51.12
		No	187.557	2109.016	11509.28	0.95	10.08	51.12

Table 10  
ANOVA for the B&B experiments

Source	<i>DF</i>	Sum of squares	Mean square	<i>F</i> ratio	<i>Prob</i> > <i>F</i> (or <i>p</i> -value)
Model	17	$2.957 \times 10^{14}$	$1.74 \times 10^{13}$	5.8146	<0.0001
Error	342	$1.023 \times 10^{15}$	$2.992 \times 10^{12}$		
C total	359	$1.318 \times 10^{15}$			
<i>n</i>	3	$2.91 \times 10^{14}$		32.3818	<0.0001
<i>R</i>	2	$1.08 \times 10^{12}$		0.1813	0.8343
<i>UB</i>	1	$5.78 \times 10^{11}$		0.1931	0.6606
<i>n</i> × <i>R</i>	6	$2.19 \times 10^{12}$		0.122	0.9937
<i>n</i> × <i>UB</i>	3	$1.19 \times 10^{12}$		0.132	0.941
<i>R</i> × <i>UB</i>	2	$5.76 \times 10^{10}$		0.0096	0.9904

when the number of jobs is increased. The insignificant effect of the  $[AP]$  Range and the  $UB$ , however, needs more insight. The search is based on a *depth-first* approach. In other words, the algorithm follows nodes with the smallest lower bound values until the bottom of the tree is reached. Therefore, it is expected that the algorithm would reach a solution with a good quality at an early stage of the search since a small lower bound implies a small objective function value. This means that, even if the  $UB$  is set to a very high value (i.e.,  $UB = \infty$ ), it is not going to make much difference in this case.

In case of the  $[AP]$  range, the results were more difficult to explain because an impact was expected when the adjusted setup times are sampled from wider ranges. Hence, to gain more intuition regarding this factor, more experiments were carried out. Data from integer ranges of  $[1, 2], [1, 4], [1, 6], \dots, [1, 30]$  for  $[AP]$  were generated and solved for  $n = 10, 15, 20$ , and  $25$ . The average number of nodes branched was recorded. No  $UB$  was used in this experiment to make sure that the results obtained were due to the influence of changing the range and nothing else. To normalize the result, the percentage of increase (or decrease) in nodes as the range was increased was calculated for range  $r$  by calculating  $[\text{nodes}(r) - \text{nodes}(r - 1) / \text{nodes}(r - 1)]100\%$ . When  $r = 1$ , this percentage is 0. From the results in Table 11, it is obvious that for very small ranges (up to  $r = 3$ ), the number of nodes branched is very small, and this number increase significantly for larger ranges, and then levels off. The reason for branching a small number of nodes in the case of very tight ranges lies in the way the lower bound is calculated.  $LB1$  is calculated based on the minimum entries in  $[AP]$ . When the range is very small, the values of the  $[AP]$  entries will be very close to each other; thus, they will be very close to the value of the minimum entry in the columns. As a result, the minimum entry in a column will in many cases be equal to the actual adjusted processing time, and therefore, the value of the lower bound will be very close or even equal to the value of the optimal solution. This way, once the B&B reaches a complete solution, it is very likely that this solution will be equal to many lower bound values in the upper levels of the tree. This results in eliminating many nodes very quickly. Using the same reasoning, one can explain why as the range increases, its influence on the number of nodes branched becomes insignificant.

## 7. Conclusions and future research

The contribution of this paper is that a B&B was developed to solve instances of the E/T problem with unrestricted common due date and sequence-dependent setup time up to 25 jobs within a reasonable time. In the E/T literature, problems with up to 8 jobs only have been optimally solved by solving a MIP. The lower bound, upper bound and a branching strategy developed for this B&B were tested and showed to be effective. The quality of the lower bound greatly affects the performance of the B&B, and better lower bounds means better overall performance of the B&B, although it may take longer to calculate the lower bound at each node. To test the overall performance of the B&B, the number of jobs  $n$ ,  $[AP]$  range and  $UB$  were used in an experimental design.  $n$  was the most significant.

Future research should consider the weighted version of this problem as well as extending the problem to other environments such as the parallel machine environment. Another extension of this work would be to address the restricted version of the problem and improving the quality of the lower bound.

Table 11  
[AP] range vs. nodes branched

<i>n</i>	Range	Avg. nodes branched	% Node increase	<i>n</i>	Range	Avg. nodes branched	% Node increase
10	1	9.0	0.0	20	1	19.0	0.0
	3	162.8	1708.9		3	194.3	922.6
	5	222.1	36.4		5	58084.6	29794.3
	7	254.2	14.4		7	37867.7	− 34.8
	9	252.7	− 0.6		9	32994.7	− 12.9
	11	267.9	6.0		11	26662.2	− 19.2
	13	253.3	− 5.4		13	40449.3	51.7
	15	239.4	− 5.5		15	30506.3	− 24.6
	17	253.7	6.0		17	45253.8	48.3
	19	244.4	− 3.7		19	47445.4	4.8
	21	296.2	21.2		21	42653.8	− 10.1
	23	319.6	7.9		23	47915.8	12.3
25	321.0	0.4	25	49501.8	3.3		
27	322.3	0.4	27	49997.2	1.0		
29	281.2	− 12.8	29	42057.3	− 15.9		
15	1	14.0	0.0	25	1	24.0	0.0
	3	224.3	1502.1		3	220.4	818.3
	5	3237.3	1343.3		5	276806.0	125492.6
	7	2349.2	− 27.4		7	730748.6	164.0
	9	4278.3	82.1		9	542704.1	− 25.7
	11	3531.7	− 17.5		11	342871.3	− 36.8
	13	3542.8	0.3		13	340556.8	− 0.7
	15	4736.2	33.7		15	460899.0	35.3
	17	4412.5	− 6.8		17	410904.3	− 10.8
	19	4627.7	4.9		19	575824.7	40.1
	21	4881.0	5.5		21	492993.0	− 14.4
	23	4940.1	1.2		23	579104.4	17.5
25	3688.1	− 25.3	25	733353.7	26.6		
27	3280.9	− 11.0	27	764837.2	4.3		
29	3515.3	7.1	29	600966.6	− 21.4		

**Appendix A.**

*A.1. Proof of property (ii) with sequence-dependent setup times*

This property can be proved by contradiction, similar to that presented by Baker [18] with the addition of sequence-dependent setup times:

Suppose that  $n_1 = |B|$  and  $n_2 = |A|$ . Suppose also that  $S$  is an optimal sequence and job  $j$  does not complete on  $d$  (see Fig. 5). That is  $C_j - P_j - S_{ij} < d < C_j$ .

Case 1:  $n_2 > n_1$ . Shift the sequence  $\Delta t$  time units earlier such that job  $j$  completes on  $d$ . So

$$\Delta t = C_j - d > 0.$$

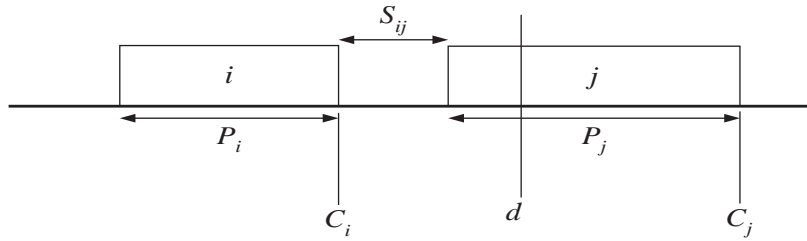


Fig. 5. Proof of property (ii) with sequence-dependent setup times.

The increase in the earliness cost =  $n_1 \Delta t$ . The decrease in tardiness cost =  $n_2 \Delta t$ .  $\epsilon$ , the net difference in cost =  $n_1 \Delta t - n_2 \Delta t = (n_1 - n_2) \Delta t < 0$ .

Note that there is no need to shift the sequence in the other direction since this will increase the cost (i.e.,  $\epsilon > 0$ ).

Case 2:  $n_1 \geq n_2$ . Shift the sequence  $\Delta t$  time units later such that the setup for job  $j$  starts on  $d$ . So

$$\Delta t = d - (C_j - P_j - S_{ij}) > 0.$$

The increase in the tardiness cost =  $n_2 \Delta t$ . The decrease in earliness cost =  $n_1 \Delta t$ .  $\epsilon$ , the net difference in cost =  $n_2 \Delta t - n_1 \Delta t = (n_2 - n_1) \Delta t < 0$ .

Again, there is no need to shift the sequence in the other direction since this will increase the cost (i.e.,  $\epsilon > 0$ ).

A conclusion can be made that a sequence as good as  $S$  can always be obtained by shifting the sequence such that one of the jobs completes on  $d$ .

A.2. Proof of property (iii) with sequence-dependent setup times

The following theorem will be proven in general, and property (iii) will be a direct result of it.

**Theorem.** An optimal value  $t^*$  that minimizes the function

$$\Phi(t) = \sum_{i=1}^n |t - t_i| \tag{A.1}$$

is the median of  $t_i$ 's for  $1 \leq i \leq n$ .

**Proof.** Without loss of generality, assume that  $t_1 < t_2 < \dots < t_n$ .

Let  $k$  be an integer such that  $1 \leq k \leq n$ .

$\Phi(t)$  can be expressed as the sum of two quantities:

$$\Phi(t) = \Phi_B(t) + \Phi_A(t),$$

where

$$\Phi_B(t) = \sum_{i=1}^{k-1} |t - t_i| \quad \text{and} \quad \Phi_A(t) = \sum_{i=k}^n |t - t_i| \tag{A.2}$$

for any  $t_{k-1} \leq t \leq t_k$ :

$$\Phi_B(t) = \sum_{i=1}^{k-1} |t - t_i| = \sum_{i=1}^{k-1} (t - t_i) \tag{A.3}$$

and

$$\Phi_A(t) = \sum_{i=k}^n |t - t_i| = \sum_{i=k}^n (t_i - t). \tag{A.4}$$

So

$$\Phi(t) = \sum_{i=1}^{k-1} (t - t_i) + \sum_{i=k}^n (t_i - t), \tag{A.5}$$

$$\Phi(t) = (k - 1)t - \sum_{i=1}^{k-1} t_i + \sum_{i=k}^n t_i - (n - k + 1)t,$$

$$\Phi(t) = \underbrace{[2(k - 1) - n]}_{S(k)} t + \underbrace{\sum_{i=k}^n t_i + \sum_{i=1}^{k-1} t_i}_{C(k)},$$

$$\Phi(t) = S(k)t + C(k) \quad \forall t_{k-1} \leq t \leq t_k. \tag{A.6}$$

By (A.1) and (A.6),  $\Phi$  is a continuous piecewise linear function. Since  $S(k)$  represents the slope of the function within the interval  $t_{k-1} \leq t \leq t_k$ . Therefore, in that interval:

If  $S(k) < 0 \rightarrow \Phi$  is decreasing.

If  $S(k) = 0 \rightarrow \Phi$  is constant.

If  $S(k) > 0 \rightarrow \Phi$  is increasing.

Case 1:  $n$  is odd.

For  $k < (n + 1)/2, S(k) < 0 \rightarrow \Phi$  is decreasing for any  $t < t_{(n+1)/2}$ .

For  $k > (n + 1)/2, S(k) > 0 \rightarrow \Phi$  is increasing for any  $t \geq t_{(n+1)/2}$ .

Therefore,

$$t_{\text{minimum}} = t^* = t_{(n+1)/2}.$$

Case 2:  $n$  is even.

For  $k \leq n/2, S(k) < 0 \rightarrow \Phi$  is decreasing for any  $t < t_{n/2}$ .

For  $k = n/2 + 1, S(k) = 0 \rightarrow \Phi$  is constant for any  $t_{n/2} \leq t < t_{n/2+1}$ .

For  $k > n/2 + 1, S(k) > 0 \rightarrow \Phi$  is increasing for any  $t_{n/2+1} < t$ .

Note that any  $t$ , such that

$$t_{n/2} \leq t \leq t_{n/2+1}$$

minimizes  $\Phi$ .

Since the values that  $t_i$  can take are discrete and  $n$  is even in this case, then  $t^* = t_{\text{minimum}}$  can take either of the following two values:

$$t^* = \begin{cases} t_{n/2}, \\ t_{(n/2)+1}. \end{cases}$$

In case of the E/T problem, the objective function is

$$\min Z = \sum_{j=1}^n |d - C_j|.$$

Directly from the previous theorem, the optimal value of  $d$  is  $d^*$  where:

$$d^* = \begin{cases} C_{n/2} \text{ or } C_{(n/2)+1} & \text{if } n \text{ is even,} \\ C_{(n+1)/2} & \text{if } n \text{ is odd.} \end{cases} \quad (\text{A.7})$$

That is, the  $b$ th job that completes on  $d$  is

$$b = \begin{cases} \frac{n}{2} \text{ or } \frac{n}{2} + 1 & \text{if } n \text{ is even,} \\ \frac{n+1}{2} & \text{if } n \text{ is odd.} \end{cases} \quad (\text{A.8})$$

## References

- [1] Kanet JJ. Minimizing the average deviation of job completion times about a common due date. *Naval Research Logistics* 1981;28:643–51.
- [2] Baker KR, Scudder GD. Sequencing with earliness and tardiness penalties: a review. *Operations Research* 1990;38(1):22–36.
- [3] Hall NG. Single- and multiple-processor models for minimizing completion time variance. *Naval Research Logistics Quarterly* 1986;33:49–54.
- [4] Hall NG, Posner ME. Earliness-tardiness scheduling problems, I: weighted deviation of completion times about a common due date. *Operations Research* 1991;39(5):836–46.
- [5] Azizoglu M, Webster S. Scheduling job families about an unrestricted common due date on a single machine. *International Journal of Production Research* 1997;35(5):1321–30.
- [6] Bagchi U, Sullivan R, Chang Y-L. Minimizing mean absolute deviation of completion times about a common due date. *Naval Research Logistics Quarterly* 1986;33:227–40.
- [7] Szwarc W. Single machine scheduling to minimize absolute deviation of completion times from a common due date. *Naval Research Logistics* 1989;36:663–73.
- [8] Szwarc W. The weighted common due date single machine scheduling problem revisited. *Computers and Operations Research* 1996;23(3):255–62.
- [9] Hall NG, Kubiak W, Sethi SP. Earliness-tardiness scheduling problems, II: deviation of completion times about a restrictive common due date. *Operations Research* 1991;39(5):847–56.
- [10] Alidaee B, Dragan I. A note on minimizing the weighted sum of tardy and early completion penalties in a single machine: a case of small common due date. *European Journal of Operational Research* 1997;96:559–63.
- [11] Mondal SA, Sen AK. Single machine weighted earliness-tardiness penalty problem with a common due date. *Computers and Operation Research* 2001;28(7):649–69.
- [12] French S. *Sequencing and scheduling: an introduction to the mathematics of the job-shop*. New York: Wiley, 1982.
- [13] Coleman BJ. A simple model for optimizing the single machine early/tardy problem with sequence-dependent setups. *Production and Operation Management* 1992;1:225–8.

- [14] Chen Z-L. Scheduling with batch setup times and earliness-tardiness penalties. *European Journal of Operational Research* 1997;96:518–37.
- [15] Allahverdi A, Gupta JND, Aldowaisan T. A review of scheduling research involving setup consideration. *OMEGA* 1999;27(2):219–39.
- [16] Gendreau M, Laporte L, Guimaraes EM. A divide and merge heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *European Journal of Operational Research* 2001;133:183–9.
- [17] Rabadi G. Minimizing the total earliness and tardiness for a single machine scheduling problem with a common due date and sequence dependent setup times. Dissertation, University of Central Florida, USA, 1999.
- [18] Baker KR. *Elements of sequencing and scheduling*. Hanover: Baker Press, 1997.
- [19] Fourer R, Gay DM, Kernighan BW. *AMPL: a modeling language for mathematical programming*. Danvers, MA: Boyd & Fraser, 1993.
- [20] ILOG CPLEX, *ILOG CPLEX 6.5 user's manual*. France: ILOG, 1999.
- [21] Balakrishnan N, Kanet JJ, Sridharan SV. Early/tardy scheduling with sequence dependent setup on uniform parallel machines. *Computers and Operations Research* 1999;26(2):127–41.
- [22] Papadimitriou CH, Steiglitz KS. *Combinatorial optimization*. Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [23] Liaw C-F. A branch-and-bound algorithm for the single-machine earliness and tardiness scheduling problem. *Computers and Operations Research* 1999;26(7):679–93.