IEMS Vol. 3, No. 2, pp. 100-115, December 2004.

Branch and Bound Approach for Single-Machine Sequencing with Early/Tardy Penalties and Sequence-Dependent Setup Cost

Chananes Akjiratikarl
Industrial Engineering Program
Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12121, THAILAND
Tel: +662- 986-9009 ext. 2107, Fax: +662-986-9112, E-mail: pisal@siit.tu.ac.th

Pisal Yenradee†
Industrial Engineering Program
Sirindhorn International Institute of Technology, Thammasat University, Pathum Thani 12121, THAILAND
Tel: +662- 986-9009 ext. 2107, Fax: +662-986-9112, E-mail: pisal@siit.tu.ac.th

Abstract. The network representation and branch and bound algorithm with efficient lower and upper bounding procedures are developed to determine a global optimal production schedule on a machine that minimizes sequence-dependent setup cost and earliness/tardiness penalties. Lower bounds are obtained based on heuristic and Lagrangian relaxation. Priority dispatching rule with local improvement procedure is used to derive an initial upper bound. Two dominance criteria are incorporated in a branch and bound procedure to reduce the search space and enhance computational efficiency. The computational results indicate that the proposed procedure could optimally solve the problem with up to 40 jobs in a reasonable time using a personal computer.

Keywords: Single-machine scheduling, Branch and bound, Lagrangian relaxation, Heuristic, Early/tardy, Sequence-dependent setup cost

1. INTRODUCTION

Scheduling has been applied in a wide range of manufacturing activities. An efficient production schedule can result in substantial cost reduction and increased customer satisfaction. In this paper, a multi-item single-machine scheduling problem to minimize total cost is presented. The total cost is the sum of earliness/tardiness penalties of jobs and sequence-dependent setup cost. A single machine scheduling method is useful not only for scheduling jobs on a single machine but also for generating a master production schedule of flow shop and job shop where there is a single bottleneck work center.

The earliness/tardiness problem arises in the Just-In-Time (JIT) scheduling environment in which jobs are forced to be completed as close to their due dates as possible. Jobs that are completed early must be held in finished goods inventory and an earliness penalty (inventory holding cost) may be incurred, while jobs that are completed after their due dates may cause a tardiness penalty due to customer dissatisfaction, and compensation or contract penalty. These penalties could be different among jobs based on the values, priority of jobs, and customers.

In multi-item single-machine production systems, setup cost is incurred when production is switched from one job to another. The setup that depends only on the job to be processed is called sequence-independent setup and the setup that depends on the job to be processed and its immediately preceding job is called sequence-dependent setup. The setup costs, which occur due to the changing of mold, retooling, rearrangement of workstations, or the extent of cleaning required between process runs, are usually dependent of the degree of similarity between consecutive jobs, for example, sizes, shapes, and colors.

A number of studies involve single-machine scheduling problems. Table 1 presents a list of different objective functions that appeared in the literature.

1.1 Related Works on the Setup (Changeover) Cost

Glassey (1968) proposed an algorithm to determine the job sequence that minimizes the total number of

† : Corresponding Author

Table 1. List of related papers in literature.

Objective Functi	Condition(s)	Setup time (ST_{ji})	Setup Cost (SC_{ji})	Earliness Cost(h_i)	Tardiness Cost(w_i)	Authors
Min no. of chageovers	tardiness is not allowed	0	1	0	M	Glassey (1968)
Min seq-dep changeover Cost	tardiness is not allowed	0	Seq-dep	0	M	Driscoll and Emmoms (1977)
Min seq-dep changeover Cost	$SC_{ji} = 1$ if $j<i$ $SC_{ji} = 0$ if $j>=i$ tardiness is not allowed	0	$SC_{ji} = 1$ if $j<i$ $SC_{ji} = 0$ if $j>=i$	0	M	Hu *et al.* (1987)
Min weighted tardiness		0	0	0	R	Potts and Wassenhove (1984)
Min weighted earliness-tardiness penalties		0	0	R	R	Abdul-Razaq and Potts (1988), Ow and Morton (1989), Li (1997), Liaw (1999), Ibaraki and Nakamura (1994)
Min total earliness-tardiness penalties	$h_i = ap_i,$ $wi = bp_i$	0	0	ap_i	bp_i	Yano and Kim (1991)
Min weighted earliness-tardiness penalties with seq-dep setup time		Seq-dep	0	R	R	Coleman (1992)
Min weighted earliness-tardiness penalties with common due date	commom due date	0	0	R	R	Mondal and Sen (2001), Azizoglu and Webster (1997)
Min total earliness-tardiness penalties with common due date and seq-dep setup time	commom due date	Seq-dep	0	$h_i = h_j$ for all i,j	$w_i = w_j$ for all i,j	Rabadi *et al.* (2004)
Min total earliness-tardiness penalties with inserted idle time	idle time is allowed	0	0	$h_i = h_j$ for all i,j	$w_i = w_j$ for all i,j	Chang (1999) , Ventura (2003)
Min total earliness-square tardiness penalties with Inserted idle time	idle time is allowed	0	0	$h_i = h_j$ for all i,j	$w_i = w_j$ for all i,j	Shaller (2004)
Min weighted earliness-tardiness penalties with inserted idle time	idle time is allowed	0	0	R	R	Chen and Lin (2002)
Min weighted earliness-tardiness penalties with seq-dep setup cost and time	idle time is allowed	Seq-dep	Seq-dep	R	R	Sourd (in press)

R = Real data
M = Big positive number
p_i = Processing time of job i
a, b = positive real numbers t^2

changeovers of the machine when tardiness is not allowed. The problem was modeled as a shortest path problem. Driscoll and Emmons (1977) developed a similar work with the objective of minimizing sequence-dependent changeover penalty. They reported the use of forward-time and backward-time dynamic programming with the application of monotonicity property to find an optimal solution. A more specific problem of setup cost was considered by Hu *et al.* (1987). In this paper, the setup cost when switching from producing item *j* to item *i* is one dollar if *j* is less than *i*, and zero if *j* is greater than or equal to *i*. The setup time of these studies is assumed to be negligible. The review of scheduling literature considering setup can be found in Allahverdi *et al.* (1999).

1.2 Related Works on the Earliness and Tardiness

The scheduling problems with earliness and tardiness penalties have received much interest due to the growing adoption of the JIT manufacturing philosophy. A special case known as a common due date problem was studied by a number of researchers. Mondal and Sen (2001) investigated a graph search space algorithm with depth-first branch and bound scheme to the weighted earliness and tardiness problem with a restricted (small) due date. Azizoglu and Webster (1997) introduced a branch and bound algorithm and a beam search procedure to solve the problem with the sequence-dependent family setup time where the due date is common and unrestricted (large). A recent paper by Rabadi *et al.* (2004) considered the same problem with sequence-dependent setup time but earliness and tardiness are weighted equally. The branch and bound algorithm was developed to solve the problem with up to 25 jobs.

The scheduling problem with non-identical due dates was investigated by many researchers. Potts and Wassenhove (1985) presented a branch and bound algorithm with Lagrangian relaxation and multiplier adjustment procedure for the total weighted tardiness problem. For a weighted earliness and tardiness case, Abdul-Razaq and Potts (1988) proposed a branch and bound technique with the use of a relaxed dynamic programming procedure by mapping state-space onto a smaller state-space and performing recursion to obtain a good lower bound. The lower bound is further improved through the use of state-space modifier. The problem can handle up to 20 jobs with reasonable computational time. Ow and Morton (1989) presented a series of heuristics, dispatch priority rules and a filtered beam search technique, to provide near optimal solution with relatively small search tree when the number of jobs is less than 30. Ibaraki and Nakamura (1994) used a Successive Sublimation Dynamic Programming method for the same problem by performing several types of dynamic programming recursions on a number of generated states. The method succeeds in effectively solving problems with up to 35 jobs. Li (1997) developed a branch and bound algorithm based on decomposition of the problem into two sub-problems. The lower bound of each sub-problem is obtained using Lagrangian relaxation with multiplier adjustment procedure. The algorithm can be used to solve the problems with up to 50 jobs. Liaw (1999) proposed a similar technique in which the lower bounds are computed using Lagrangian relaxation with multiplier adjustment and single pass procedure. The upper bound is obtained using two-phase heuristic procedures. Simple dominance rules are also derived to help eliminating nodes in the branch and bound search tree. The computational experiments show that the algorithm performs very well on problems with up to 50 jobs. Coleman (1992) proposed a simple mixed integer-programming model for a weighted earliness and tardiness problem with sequence-dependent setup time. The model can handle the problem up to 8 jobs. Yano and Kim (1991) discussed a problem where the earliness and tardiness weights are proportional to the processing times of the jobs. Dominance criteria are derived which eliminate many possible sequences from consideration in the branch and bound procedure.

Many authors considered the problem where the idle time can be inserted into the schedule. Chang (1999) applied branch and bound algorithm with the lower bound based on overlap elimination procedure for the un-weighted earliness and tardiness problem. Ventura and Radhakrishnan (2003) formulated the same problem as 0-1 linear program. They implement Lagrangian relaxation procedure that utilizes the sub-gradient algorithm to obtain near optimal solutions. Shaller (2004) presented a timetabling algorithm that inserts idle time into a schedule in order to minimize the sum of job earliness and the square of job tardiness and establish a lower bound for a branch and bound method. The weighted earliness and tardiness case was considered by Chen and Lin (2002). They utilized a dominance rule to develop a relationship matrix and combined this matrix with a branching strategy to solve the problem. The most recent paper by Sourd (in press) also used branch and bound procedure for the same problem but with sequence-dependent setup time and cost between group of products. The time-indexed formulation with different relaxations was proposed to obtain the lower bound for the problem. The computational result has shown that the algorithm is limited to problems with no more than 20 jobs. Those interested in early/tardy problems are referred to Baker and Scudder (1990) for comprehensive survey of the previous literature.

Since the problem of Sourd (in press) is general and complex, the representation of the schedule and the algorithm are complicated. This results in a long computational time. Hence, it is only applicable to small-sized problems (up to 20 jobs). This paper aims to develop a simpler algorithm and schedule representation, which also guarantees the global optimal solution for larger problems. However, some assumptions are required. The problem under consideration in this paper is based on assumptions that the idle time cannot be inserted into the schedule and setup time is sequence-independent. In this case, the setup time can be added directly to the processing time. The insertion of idle time in the schedule can reduce the earliness of the jobs. However, this may not be appropriate for some situations, where the idle cost is higher than the earliness penalties, the capacity of machine is less than the demand, or the machine under consideration is a bottleneck resource, etc.

However, the problem under consideration in this paper is a general case of many problems presented in the literature (Abdul-Razaq and Potts (1988), Azizoglu and Webster (1997), Driscoll and Emmons (1977), Glassey (1968), Hu *et al.* (1987), Ibaraki and Nakamura (1994), Li (1997), Liaw (1999), Mondal and Sen (2001), Ow and Morton (1989), Potts and Wassenhove (1985), Yano and Kim (1991)). It can be easily adapted to deal with these problems by modifying some input parameters.

This paper is organized as follows: the next section describes the scheduling problem under consideration; sections 3 and 4 describe the derivation of lower bounds and upper bound, respectively; section 5 explains the dominance criteria; the branch and bound algorithm and an illustrative example are described in sections 6 and 7; computational performance is analyzed and discussed in section 8; finally, conclusions and recommendations for further studies are presented in section 9.

2. PROBLEM DESCRIPTIONS

X is a set of N independent jobs $\{x_1, x_2, ..., x_N\}$ which are to be scheduled non-preemptively on a machine that can handle at most one job at a time. Once the current job is finished, the next job is started immediately without an idle time. The sequence-dependent setup cost is described by a matrix **SC** = $[SC_{ji}]$, where SC_{ji} is the cost of switching from job j to i. The setup cost of the first job, i.e. SC_{0i}, is assumed to be zero. The setup time is sequence-independent and is added to the processing time. Each job is available at time zero and its distinct due date d_i is known. The processing requirement p_i is expressed in days of production, which can be obtained from a demand of job I, Q_i, divided by its production rate Rp_i. The earliness and tardiness penalties per period are represented by h_i and w_i, respectively. If the completion time C_i is before the due date, the earliness of job i is determined from $E_i = max\ (0,\ d_i - C_i)$. If the job is completed after the due date, the tardiness of job i is then determined from $T_i = max\ (0,\ C_i - d_i)$.

The above-mentioned problem can be found in many types of industries that adopt the Just-In-Time (JIT) production philosophy. Under JIT philosophy, jobs should be finished right at the due date since both early and tardy completions have penalties. The tardy penalty cost is normally higher than the early penalty cost. Both penalty costs may be dependent of the jobs. The JIT philosophy will also cause frequent setups. As a result, the total setup costs are high and should be considered explicitly in the objective function. For many production environments, the setup cost of a machine depends on the production sequence. The magnitude of setup cost often depends on the similarity of the process technology requirements of two consecutive jobs. The sequence-dependent setup problems can be found in the production of different colors of paint, strengths of detergent, and blends of fuel (see Morton and Pentico, 1993).

3. LOWER BOUND DERIVATION

In this section, procedures for deriving lower bounds are presented. The problem is decomposed into three sub-problems: weighted early, weighted tardy, and sequence-dependent setup. Based on the sequence of jobs, y_{ji} = 1 if job j precedes job i; otherwise $y_{ji} = 0$.

The early/tardy problem with sequence-dependent setup cost can be written as (P):

$$\text{(P)} \qquad V = \text{Min} \sum_{i=1}^{N} \left(h_i E_i + w_i T_i + y_{ji} SC_{ji} \right) \tag{1}$$

subject to

$$E_i \geq 0 \tag{2}$$

$$E_i \geq d_i - C_i \tag{3}$$

$$T_i \geq 0 \tag{4}$$

$$T_i \geq C_i - d_i \tag{5}$$

The problem P can be decomposed into three sub-problems P1, P2, and P3 as follows:

Sub-problem P1, weighted earliness sub-problem

$$\text{(P1)} \qquad V_1 = \text{Min} \sum_{i=1}^{N} h_i E_i \tag{6}$$

subject to

$$E_i \geq 0 \tag{7}$$

$$E_i \geq d_i - C_i \tag{8}$$

Sub-problem P2, weighted tardiness sub-problem

$$\text{(P2)} \qquad V_2 = \text{Min} \sum_{i=1}^{N} w_i T \tag{9}$$

subject to

$$T_i \geq 0 \tag{10}$$

$$T_i \geq C_i - d_i \tag{11}$$

Sub-problem P3, sequence-dependent setup sub-problem

$$\text{(P3)} \qquad V_3 = \text{Min} \sum_{i=1}^{N} y_{ji} SC_{ji} \tag{12}$$

The lower bounds of P1, P2, and P3 can be used to obtain the lower bound for P. The lower bounds of weighted early and weighted tardy sub-problems are computed based on Lagrangian relaxation, and a multiplier adjustment method is used to compute the value of Lagrange multipliers. The lower bound of sequence-dependent setup sub-problem is obtained using a minimum setup heuristic. The lower bounding procedures developed by Li (1997) are used to obtain the lower bounds for P1 and P2. The heuristic used to obtain the lower bound for P3 is developed in this paper. The lower bound for the problem is the sum of the three lower bounds. The proof of this statement is adapted from Li (1997) with the addition of sequence-dependent setup sub-problem P3.

Theorem 1. *If V_1, V_2, V_3, and V are the minimum objective functions of* P1, P2, P3, *and* P, *respectively, then $V_1 + V_2 + V_3 \leq V$.*

Proof. Let α be an optimal schedule and $V = C_1 + C_2 + C_3$, where $C_1 = \sum_{i=1}^{N} h_i E_i$, the total early cost of sequence α. $C_2 = \sum_{i=1}^{N} w_i T$, the total tardy cost of sequence α, $C_3 = \sum_{i=1}^{N} y_{ji} SC_{ji}$, the total setup cost of sequence α. Therefore, $V_1 \leq C_1$, $V_2 \leq C_2$, and $V_3 \leq C_3$. Since $V = C_1 + C_2 + C_3$, then $V_1 + V_2 + V_3 \leq V$.

Lemma 1. *If LB1, LB2, LB3 are lower bounds for* P1, P2, *and* P3, *respectively, then LB1 + LB2 + LB3 is a lower bound for* P.

Proof. Since *LB*1, *LB*2, *LB*3 are lower bounds for P1, P2, and P3, respectively, then $LB1 \leq V_1$, $LB2 \leq V_2$, $LB3 \leq V_3$. Therefore, $LB1 + LB2 + LB3 \leq V_1 + V_2 + V_3$. From Theorem 1, $V_1 + V_2 + V_3 \leq V$, then $LB1 + LB2 + LB3 \leq V$, a lower bound for P.

3.1. Lower Bound for Sub-Problem P1

A Lagrangian relaxation of constraint 8 yields the Lagrangian problem LR1:

$$\text{(LR1)} \qquad L_1(\lambda) = \text{Min} \sum_{i=1}^{N} \left[(h_i - \lambda_i) E_i + \lambda_i (d_i - C_i) \right] \tag{13}$$

subject to

$$E_i \geq 0 \tag{14}$$

where, $\lambda = (\lambda_1, \lambda_2, ..., \lambda_N)$ is the vector of corresponding multipliers. In LR1, the objective function can be expanded as follows:

$$L_1(\lambda) = \text{Min} \sum_{i=1}^{N} (h_i - \lambda_i) E_i + \sum_{i=1}^{N} \lambda_i d_i - \text{Max} \sum_{i=1}^{N} \lambda_i C_i \tag{15}$$

According to Li (1997), for any choice of nonnegative multipliers, $L_1(\lambda)$ provides a lower bound for problem P1. The first term can be minimized by setting all $E_i = 0$ if $h_i - \lambda_i \geq 0$. The second term is a constant. The third term can be maximized using the weighted longest processing time rule (WLPT), that is, sequencing jobs in non-decreasing order of λ_i / p_i. Assume that the jobs generated by the heuristic presented in section 3.1.1 are renumbered so that the sequence is (1, 2,…, N). Also, let C_i* be the completion time of jobs when they are sequenced by the heuristic. Then, a lower bound $L_1(\lambda)$ can be determined from the Lagrangian dual sub-problem D1.

$$\text{(D1)} \qquad L_1 = \text{Min} \sum_{i=1}^{N} \lambda_i \left(d_i - C_i^* \right) \tag{16}$$

subject to

$$\frac{\lambda_i}{p_i} \leq \frac{\lambda_{i+1}}{p_{i+1}} \qquad i = 1, ..., N - 1 \tag{17}$$

$$0 \leq \lambda_i \leq h_i \qquad i = 1, ..., N - 1 \tag{18}$$

The multiplier adjustment method developed by Potts and Wassenhove (1985) requires the following heuristic to sequence the jobs. Then, Li (1997) used this method to solve the problem. The multipliers obtained from the heuristic can solve the Lagrangian problem so that the resulting lower bound is as large as possible.

Heuristic for solving Lagrangian dual sub-problem D1

```
Step1. Sequence jobs by WLPT rule. Set $E_i = d_i - C_i^*$,
       $i = 1, \ldots, N$ and $V_i = \sum_{k=i}^{N} p_k E_k$, $i = 1, ..., N$
Step 2. Set $V_{N+1} = 0$, $S_1 = \{N + 1\}$, and $k = N$
Step 3. For $k = N$ to 1, when $k < 1$ go to step 4
        Let $m$ be the smallest integer in set $S_1$
        If $V_k > V_m$, then include $k$ in set $S_1$,
        i.e., $S_1 = S_1 \cup \{k\}$
        Set $k = k - 1$
Step 4. Set $k = 1$, and $S_1 = S_1 - \{N + 1\}$
Step 5. For $k = 1$ to $N$, when $k > N$ terminate
        If $k = 1$ and $k \notin S_1$, then $\lambda_k = 0$
        If $k > 1$ and $k \notin S_1$, then $\lambda_k = \lambda_{k-1}(p_k/p_{k-1})$
        If $k \in S_1$, then $\lambda_k = h_k$
        $k = k + 1$
```

The Lagrangian multiplier (λ_i) and C_i* can be obtained from the above heuristic. Then L_1 can be calculated and used as a lower bound $LB1$ for sub-problem P1.

3.2. Lower Bound for Sub-Problem P2

A Lagrangian relaxation of constraint 11 yields the Lagrangian problem LR2:

$$\text{(LR2)} \quad L_2(\lambda) = \text{Min}\sum_{i=1}^{N}\left[(w_i - \lambda_i)T_i + \lambda_i(C_i - d_i)\right] \tag{19}$$

subject to

$$T_i \geq 0 \tag{20}$$

where, $\lambda = (\lambda_1, \lambda_2, ..., \lambda_N)$ is the vector of corresponding multipliers. Similar to LR1, LR2 can be solved using the weighted shortest processing time rule (WSPT) by setting $T_i = 0$ if $w_i - \lambda_i \geq 0$. Then, the lower bound $L_2(\lambda)$ can be obtained from D2.

$$\text{(D2)} \quad L_2 = \text{Min}\sum_{i=1}^{N}\lambda_i\left(C_i^* - d_i\right) \tag{21}$$

subject to

$$\frac{\lambda_i}{p_i} \geq \frac{\lambda_{i+1}}{p_{i+1}} \qquad i = 1, ..., N-1 \tag{22}$$

$$w_i \geq \lambda_i \geq 0 \qquad i = 1, ..., N-1 \tag{23}$$

Similarly, the multiplier adjustment method is used to solve the problem.

Heuristic for solving Lagrangian dual sub-problem D2.

```
Step 1. Sequence jobs by the WSPT rule. Set $T_i = C_i^* - d_i$,
        $i = 1, \dots, N$ and $V_i = \sum_{k=1}^{i} p_k T_k,\ i = 1, ..., N$
Step 2. Set $V_0 = 0$, $S_2 = \{0\}$, and $k = 1$
Step 3. For $k = 1$ to $N$, when $k > N$ go to step 4
        Let $m$ be the largest integer in set $S_2$
        If $V_k > V_m$, then include $k$ in set $S_2$,
        i.e., $S_2 = S_2 \cup \{k\}$
        Set $k = k + 1$
Step 4. Set $k = N$, and $S_2 = S_2 - \{0\}$
Step 5. For $k = N$ to 1, when $k < 1$ terminate
        If $k = N$ and $k \notin S_2$, then $\lambda_k = 0$
        If $k < N$ and $k \notin S_2$, then $\lambda_k = \lambda_{k+1}(p_k/p_{k+1})$
        If $k \in S_2$, then $\lambda_k = w_k$
        $k = k - 1$
```

The Lagrangian multiplier (λ_i) and C_i* can be obtained from the above heuristic. Then L_2 can be calculated and used as a lower bound $LB2$ for sub-problem P2.

3.3. Lower Bound for Sub-Problem P3

A Lower bound of sub-problem P3 can be obtained using minimum setup heuristic developed in this paper.

Minimum setup heuristic

Let S_l be an ordered set of scheduled jobs of each node at level l excluding the last job in the schedule; at level 0, $S_o = \varnothing$. S_l' is a set of unscheduled jobs of that node at level l. $N - l$ is a number of elements in set S_l', where N is the total number of jobs.

```
Step 1. For each node, determine set $S_l$ and $S_l'$.
Step 2. For $i = 1$ to $N - l$, Calculate the minimum sequence-
        dependent setup cost ($MSC_{x_i}$) of each element
        $x_i$.
```

$$MSC_{x_i} = \underset{\substack{x_j \neq x_i \\ x_j \notin S_l \\ x_i \in S_l'}}{\text{Min}}\,(SC_{x_j x_i}) \tag{24}$$

where, $SC_{x_j x_i}$ is the sequence-dependent setup cost from job x_j to x_i.

Step 3. Calculate the minimum sequence-dependent setup cost of the node (MSC_l)

$$MSC_l = \sum_{i=1}^{N-l} MSC_{x_i} \tag{25}$$

Then, the MSC_l is used as a lower bound LB3 for sub-problem P3.

4. UPPER BOUND DERIVATION

In this section, two-phase heuristic procedures are used to determine an initial feasible solution for the problem. The priority dispatching rule developed by Ow and Morton (1989) is used in the first phase to establish an initial sequence. In the second phase, the local improvement procedure developed by Liaw (1999) is modified to take into account the sequence-dependent setup cost. The modified procedure is used to adapt the sequence in the first phase to achieve a better schedule and a tighter upper bound. The improvement is obtained by an insertion procedure followed by a swap procedure.

4.1 Priority Dispatching Rule

Step 1. Calculate the slack of each job i at time t, $s_i = d_i - t - p_i$.

Step 2. Calculate average processing time ($\bar{p}$) of all remaining jobs.

Step 3. Calculate priority index $I_i(t)$ at time t of each remaining job.

$$I_i(t) = \begin{cases} \dfrac{w_i}{p_i} & \text{if} \quad s_i \le 0 \\ \dfrac{w_i}{p_i} \exp\left[-\dfrac{(h_i + w_i)s_i}{h_i \bar{p}} \right] & \text{if} \quad 0 < s_i \le \left(\dfrac{w_i}{w_i + h_i} \right) k\bar{p} \\ h_i^{-2} \left[\dfrac{w_i}{p_i} - \dfrac{(h_i + w_i)s_i}{k p_i \bar{p}} \right]^3 & \text{if} \quad \left(\dfrac{w_i}{h_i + w_i} \right) k\bar{p} < s_i < k\bar{p} \\ -\dfrac{h_i}{p_i} & \text{otherwise} \end{cases} \tag{26}$$

where, k is a look-ahead parameter.

Step 4. Choose the job with the highest index to process next in the sequence.

Step 5. Repeat steps 1 to 4 until all jobs are included in the schedule.

4.2 Local Improvement Procedure

4.2.1 Insertion Procedure

Step 1. At iteration j, $j = 1, 2, \ldots, N$, select job A as the job in position j.

Step 2. Select job B among [N/3] jobs nearest to job A. The candidate for job B is restricted since it seldom occurs that the best job B candidate is far away from job A selected.

Step 3. Calculate the objective value (total cost of earliness- tardiness penalties and setup cost) after inserting job A before job B and compare with each one of the [N/3] cases.

Step 4. Select job B such that the resulting schedule has the minimum objective value.

Step 5. Insert job A before job B.

Step 6. Repeat steps 1 to 5 until $j = N$

4.2.2 Swap Procedure

Steps 1-2. They are the same as those of the insertion procedure.

Step 3. Compute the objective value after interchanging job A and job B and compare with each one of the [N/3] cases.

Step 4. Select job B such that the resulting schedule has the minimum objective value.

Step 5. Swap jobs A and B

Step 6. Repeat steps 1 to 5 until $j = N$.

5. DOMINANCE CRITERIA

In this section, the dominance criteria specially formulated for early/tardy problem with sequence-dependent setup cost are presented. The criteria are an extension of precedence relations developed by Ow and Morton (1989) and Liaw (1999) so that they can incorporate sequence-dependent setup cost. Dominance criteria consist of adjacency condition and non-adjacency condition, in which any adjacent and non-adjacent job must satisfy these conditions, respectively. Any node which does not satisfy these conditions will be removed from a branch and bound search tree.

5.1 Adjacency Condition

Suppose that jobs i and j are adjacent pairs of jobs before the last position and jobs i and j lie between jobs x and y as shown in figure 1. The total sequence-dependent setup cost, TSC_{ij} and TSC_{ji}, can be calculated using formulas (27) and (28)

.

$$TSC_{ij} = SC_{xi} + SC_{ij} + SC_{jy} \tag{27}$$

$$TSC_{ji} = SC_{xj} + SC_{ji} + SC_{iy} \tag{28}$$

When job i immediately precedes job j, Ω_{ij} and Ω_{ji} are defined as:

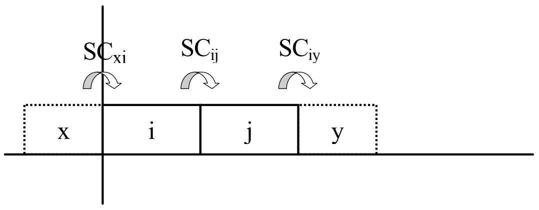

Figure 1. Illustration of adjacency condition of jobs i and j.

$$\Omega_{xy} = \begin{cases} 0 & \text{if} \quad s_x \le 0 \\ s_x & \text{if} \quad 0 < s_x < p_y \\ p_y & \text{if} \quad s_x \ge p_y \end{cases} \tag{29}$$

where $s_x = d_x - t - p_x$ is the slack of job x.
t = the sum of processing times of all preceding jobs.

The following condition must hold for all adjacent pairs of jobs:

$$AJC_{ij} = w_i p_j - \Omega_{ij}(w_i + h_i) + TSC_{ji} \tag{30}$$

$$AJC_{ji} = w_j p_i - \Omega_{ji}(w_j + h_j) + TSC_{ij} \tag{31}$$

$$AJC_{ij} \ge AJC_{ji} \tag{32}$$

See the proof of adjacency condition in Appendix A.

There are many nodes that are generated from the same parent node. Thus, calculating the adjacency condition using formulas (27) to (32) for all nodes is time consuming. It is possible to reduce the computation time by calculating the adjacency condition only for the first node and using formula (33) for other nodes generated from the same parent node.

$$AJC_{ij} + \Delta SC_{jy} \geq AJC_{ji} + \Delta SC_{iy} \quad (33)$$

where ΔSC_{jy} = SC_{jy} of the node under consideration – SC_{jy} of the first node
ΔSC_{iy} = SC_{iy} of the node under consideration – SC_{iy} of the first node.

Proof: Consider any node generated from the same parent node. Jobs i and j are the same; hence, $w_ip_j - \Omega_{ij}(w_i + h_i)$ is equal to $w_jp_i - \Omega_{ji}(w_j + h_j)$. The value of TSC_{ij} and TSC_{ji} are varied according to the last job. The only differences are SC_{jy} and SC_{iy}. Consequently, the adjacency condition can be computed by adding only the different values (ΔSC_{jy} and ΔSC_{iy}) to the same formula.

The adjacency condition can be applied to the node at level 3 and higher. The reason can be shown by the following example. Suppose the node in level 2 is considered, e.g., node (1, 2). Since the setup cost is sequence-dependent, from formulas (27) and (28), SC_{jy} and SC_{iy}, cannot be determined. But if the node is branched further to level 3, e.g., node (1, 2, 3), in this case SC_{jy} is SC_{23} and SC_{iy} is SC_{13}.

5.2 Non-Adjacency Condition

Suppose jobs i and j are non-adjacent pairs of jobs before the last position and they have the same processing

time. Job i lies between jobs u and v and job j lies between jobs w and z as shown in figure 2. The total sequence-dependent setup cost, TSC_{ij} and TSC_{ji}, can be calculated using formulas (34) and (35).

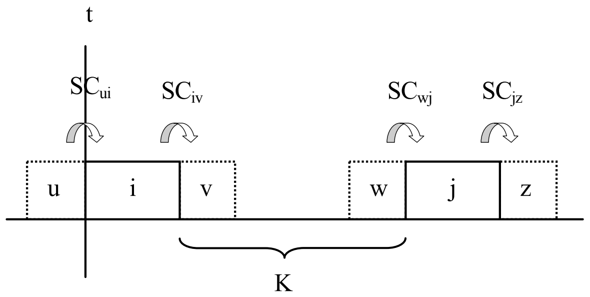

Figure 2. Illustration of non-adjacency condition of jobs i and j.

$$TSC_{ij} = SC_{ui} + SC_{iv} + SC_{wj} + SC_{jz} \quad (34)$$

$$TSC_{ji} = SC_{uj} + SC_{jv} + SC_{wi} + SC_{iz} \quad (35)$$

Δ_{ij} and Δ_{ji} are defined as:

$$\Delta_{xy} = \begin{cases} 0 & if \quad s_x \leq 0 \\ s_x & if \quad 0 < s_x < p_y + K \\ p_y + K & if \quad s_x \geq p_y + K \end{cases} \quad (36)$$

where $s_x = d_x - t - p_x$ is the slack of job x
K = the sum of processing times of jobs between jobs i and j

All non-adjacent pairs of jobs in the optimal schedule must satisfy the following condition:

if $p_i = p_j$, then

$$NJC_{ij} = w_i(p_j + K) - \Delta_{ij}(w_i + h_i) + TSC_{ji} \quad (37)$$

$$NJC_{ji} = w_j(p_i + K) - \Delta_{ji}(w_j + h_j) + TSC_{ij} \quad (38)$$

$$NJC_{ij} \geq NJC_{ji} \quad (39)$$

See the proof of non-adjacency condition in Appendix B.

The non-adjacency condition can be applied to the node at level 4 and higher. In other words, at least four jobs must be in the sequence. For example, non-adjacent jobs of the node at level 4 are jobs in positions 1 and 3. The last job in position 4 is used to determine the sequence-dependent setup cost.

6. BRANCH AND BOUND ALGORITHM

A network representation and branch and bound algorithm can be used to determine an optimal solution for the problem. A node represents a sub-problem or the sequence of jobs that are already scheduled. An arc leading to the node represents the associated cost of the node.

Initialization step

The branch and bound algorithm is started by determining an initial feasible solution to the problem. According to the two-phase heuristic procedures presented in section 4, the priority-dispatching rule is applied to establish the initial sequence. The local improvement procedures are used to modify the schedule obtained in the first phase in order to get a better sequence. The objective function value obtained from the two-phase heuristic is an initial upper bound of the objective function value.

Steps for each iteration

Step 1: Branching

The branching strategy is called the depth-first branching rule. Select the active (unfathomed) node at the highest level in the search tree for branching. If there is more than one node at the highest level, select the one with the lowest associated cost. The selected node is called the parent node. Create branches from the parent node to the nodes representing jobs that have not been scheduled.

Step 2: Bounding

For each newly created node, the associated cost is calculated. The associated cost is the sum of the associated cost of its parent node and the cost due to the newly assigned job of that node.

Step 3: Fathoming

For each generated node, three computational tests are performed in the fathoming step. First, a node is fathomed if its associated cost is greater than or equal to the current upper bound of the problem (denoted by F_1). Second, the dominance criteria described in section 5 are applied to further reduce the number of nodes (denoted by F_2). Dominance criteria consist of adjacency and non-adjacency conditions. All pairs of jobs in the optimal schedule must satisfy these conditions. Note that the adjacency condition can be applied to the nodes at level 3 and higher, while the non-adjacency condition can be applied to the nodes at level 4 and higher. If the node is not eliminated by the above two tests, the lower bounds of each of three sub-problems are computed according to the procedures presented in section 3. For fast elimination of nodes, the lower bounds of three sub-problems can be calculated in any sequence according to the characteristics of the problem under consideration. Suppose that the sequence of lower bound is *LB*1, *LB*2, and *MSC*. When the *LB*1 is determined, it is summed to the associated cost of the node and compared with the upper bound. If it is greater than or equal to the current upper bound, the node is fathomed (denoted by F_3). If the node is still not fathomed, the *LB*2 will be computed. The value of *LB*2 + *LB*1 + associated cost of the node will be again compared to the upper bound. If the node is still unfathomed, the *MSC* will be determined. Finally, the sum of lower bounds of three sub-problems (*MSC* + *LB*2 +*LB*1) plus the associated cost of that node is compared to the upper bound.

The remaining unfathomed nodes are called active nodes. When a complete sequence is found (all jobs are scheduled on the node), the node is fathomed (denoted by F_4), and its objective function value will be compared with the current upper bound. If the current upper bound is higher, replace the current upper bound by the objective function value of the complete sequence.

Optimality test

If some active nodes are still available, repeat the iteration steps; otherwise, stop. After the algorithm is terminated, the node in which its objective function value is equal to the current upper bound represents the optimal solution and its objective function value is the optimal total costs.

7. ILLUSTRATIVE EXAMPLE

The branch and bound algorithm can be illustrated using the following example. There are 4 jobs to be processed on a single machine. The data of sequence-dependent setup cost, processing time, due date, and earliness and tardiness penalties are shown in Table 2 and 3.

Table 2. Setup cost matrix.

	1	2	3	4
1	0	15	33	3
2	20	0	18	28
3	35	8	0	10
4	2	14	5	0

Table 3. Job information.

Job	p_i	d_i	h_i	w_i
1	3	8	2	9
2	2	6	7	7
3	4	10	3	8
4	3	9	8	6

The priority-dispatching rule generates the sequence of {1, 2, 4, 3} with the total cost of 89. The local improvement procedure modifies the sequence to be {2, 1, 4, 3} with a better total cost of 86, which is used as an initial upper bound for the problem.

Figure 3 shows a complete network representation of the illustrative example. In iteration 1, nodes 1, 2, 3, and 4 are constructed, indicating that each of the four jobs could be processed in the first order of the sequence. The associated cost of each node is calculated and shown on an arc leading to that node. Then, the associated cost is compared to the upper bound. The node will be fathomed if the associated cost is higher than or equal to the upper bound.

In order to further eliminate these nodes, the lower bounds of the three sub-problems are computed. Each time the lower bound is obtained, it is summed up with the associated cost and compared to the upper bound. If it is greater than or equal to upper bound, the node is fathomed. In this case, node {4} is fathomed.

Then, select job 1 as the branching node because job 1 has the lowest objective function value among the three nodes. Job 1 is now assigned to the first order of the sequence. Then, branch from job 1 for jobs 2, 3, and 4, and place each of jobs 2, 3, and 4 in the second order of the sequence. The associated cost and lower bounds of each node are calculated. Node {1,3} has the associated cost plus lower bounds (*LB*1+*LB*2) greater than the upper bound. Thus it is fathomed (making it an inactive node).

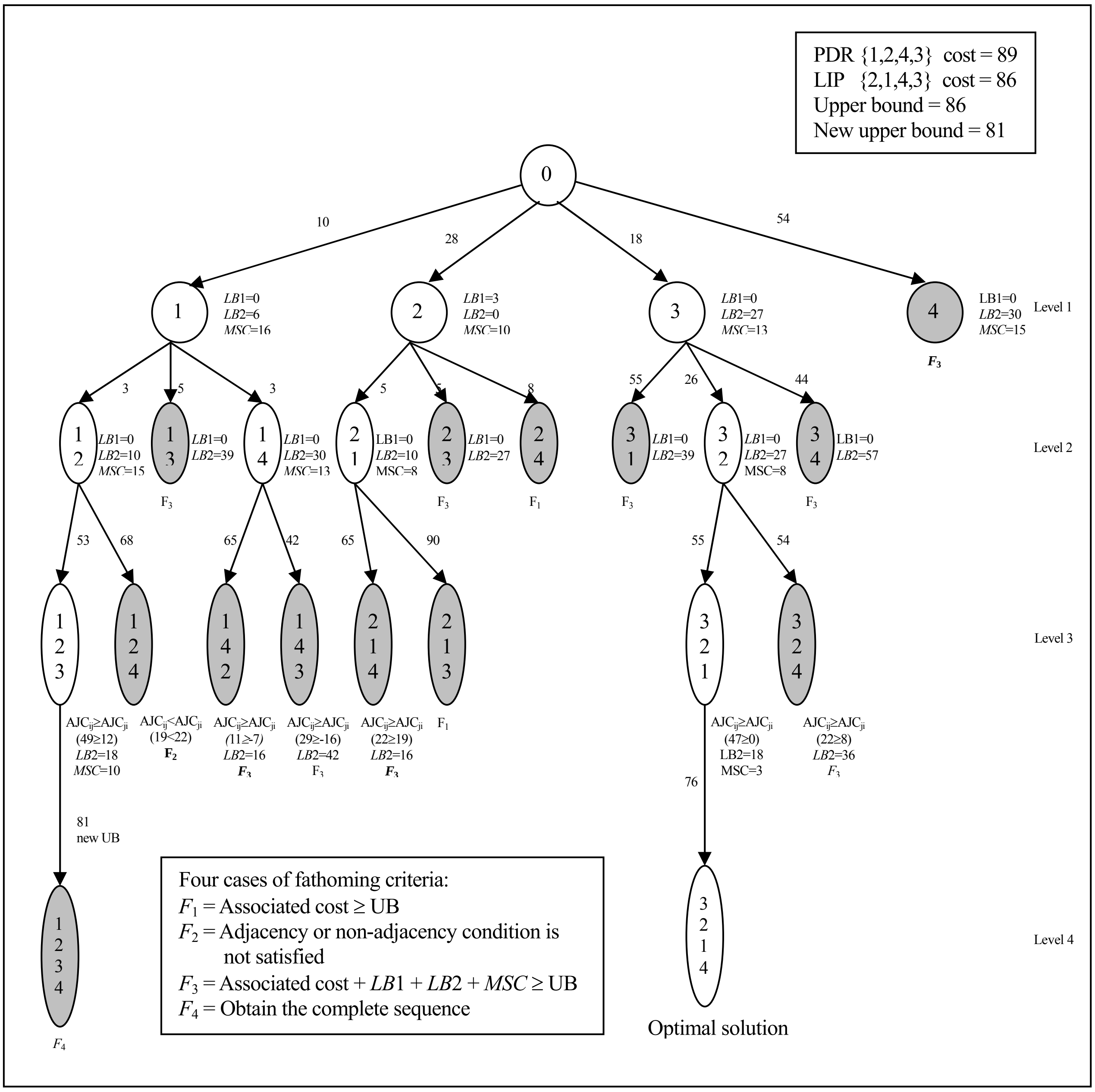

Figure 3. Network representation of numerical example

Since node {1,2} has the lowest cost, place job 2 in the second order. Branch from node {1,2} for the remaining jobs, which are jobs 3 and 4. At level 3, the dominance rule is applied. Since node {1,2,4} does not satisfy the adjacency condition, the node is fathomed. Branch from the node {1,2,3} to obtain the first complete sequence of {1,2,3,4} with the objective function value of 81. The obtained objective function value is less than the current upper bound; thus, set the new upper bound equal to the objective function value. Fathom all nodes that have their associated costs greater than the new upper bound. Repeat the iteration steps until all nodes are fathomed. After the algorithm is terminated, the optimal sequence is {3,2,1,4} with the objective function value of 76.

8. ANALYSIS OF COMPUTATIONAL PERFORMANCE

The proposed branch and bound algorithm is implemented and computationally tested on a Pentium IV computer, 1.8 GHz CPU speed, 512MB RAM. The algorithms are coded in C language. The test problems are randomly generated based on the parameters as shown in Table 4. For each job, the values of p_i, h_i, and w_i, are the same as those in Li (1997) and Liaw (1999). The setup costs are carefully selected to ensure that they are not dominated by early/tardy penalties. For due date setting, Liaw (1999) analyzed the influence of average lateness factor and relative range of due date on problem hardness. The results indicated that the problems are most difficult when a lateness factor (*LF*) = 0.6, and increasing the range of due date (*RDD*) seems to make the problem harder. For the test problems, *LF* and *RDD* are equal to 0.5 and 1, respectively, which result in high problem hardness. The look-ahead parameter (k) is equal to 3 according to a recommendation of Liaw (1999).

The first test (Test 1) evaluates the performance of the algorithm against the number of jobs. Six test problems with different numbers of jobs (15, 20, 25, 30, 35, and 40 jobs) are randomly generated. Each problem is tested by 30 randomly generated data sets. To evaluate the performance of upper bound, lower bounds, and dominance criteria, the second test (Test 2) compares the performance of the algorithm with and without implementation of these bounds and criteria for the problem of 20 and 30 jobs with the same parameter settings. For both tests, the program will terminate when the problem is not solved within 3,600 seconds in order to avoid excessive computational time. The number of unsolved problems is reported.

The performance measures are computational time, number of generated nodes, and number of generated complete solutions. Table 5 and figure 4 show that the required computational time increases exponentially with the problem size (number of jobs). This is an indication of an NP-hard problem. The propose algorithm can solve

the problem having up to 40 jobs in a reasonable

Table 4. Parameter settings for the test problems.

Parameters	Range of value
SC_{ji}	*Int U*[1, 40]
p_i	*Int U*[1, 10]
d_i	*Int U*[P(1- LF - RDD/2) , P(1 – LF + RDD/2)] where, LF = 0.5 and RDD = 1
h_i	*Int U*[1, 10]
w_i	*Int U*[1, 10]
k	3

Int U[a, b] = a set of integer numbers generated from a discrete uniform distribution within an interval [a, b]
LF = Lateness factor
RDD = Range of due date

Table 5. Computational time required by the test problems (Test1).

No. of jobs	Avg. CPU time (sec)	Avg. no. of generated nodes	Avg. no. of generated completed solutions	Total no. of nodes	Total no. of solutions	No. of problems unsolved (out of 30)
15	0.413	13,838.34	67.83	$3.55 * 10^{12}$	$1.307 * 10^{12}$	0
20	4.48	94,351.67	161.27	$6.61 * 10^{18}$	$2.43 * 10^{18}$	0
25	37.67	$1.17 * 10^{6}$	925.34	$4.21* 10^{25}$	$1.55 * 10^{25}$	0
30	98.66	$6.96 * 10^{6}$	2,827.54	$7.21 * 10^{32}$	$2.65 * 10^{32}$	0
35	885.26	$4.87 * 10^{7}$	17,410.67	$2.808 * 10^{40}$	$1.03 * 10^{40}$	4
40	2847.33	$2.83 * 10^{8}$	89,455.37	$2.21 * 10^{48}$	$8.15 * 10^{47}$	10

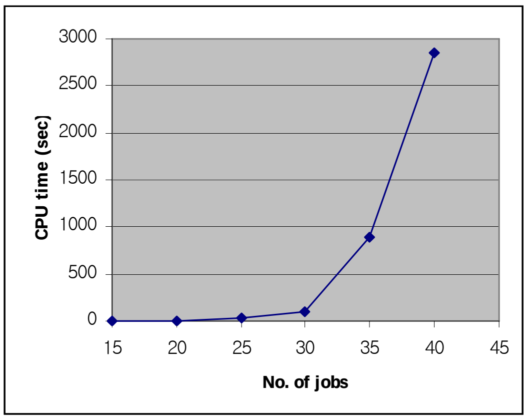

Figure 4. Relationship between the computational time and number of jobs.

Table 6. Comparison of efficiencies of each algorithm feature (Test 2).

Algorithm settings	No. of jobs	Avg. CPU time (sec)	Avg. no. of generated nodes	Avg. no. of generated completed solutions	No. of problems unsolved (out of 30)
UB is not used	20	7.75	391,702.5	899.75	0
	30	349.17	$1.34 * 10^7$	126,399.57	1
Dominance criteria are not used	20	28.12	963,951.42	195.73	0
	30	1426.18	$8.27 * 10^7$	4,974.76	3
LB is not used	20	3,012.34	$1.29 * 10^8$	8,112.25	25
	30	Too long	Unavailable	Unavailable	30
All features are used	20	3.67	89,135.34	148.34	0
	30	78.34	$5.84 * 10^6$	2,639.67	0

time (2,847 seconds on the average, not including unsolved problems). Comparing between the average number of generated nodes and the total number of nodes, and between the average number of generated complete solutions and the total number of solutions, it is obvious that the algorithm is very efficient in fathoming the nodes.

The efficiencies of upper bound, dominance criteria, and lower bound are summarized in Table 6. The results indicated that the lower bound is the most important feature of the algorithm. If the lower bound does not exist, branch and bound algorithm cannot efficiently solve even small sized problems. The dominance criteria are the second most important feature of the algorithm. The exclusion of dominance criteria results in a considerable increase in the computation time and the number of generated nodes for both problem sizes. The upper bound is the least important feature. However, for relatively large problems (No. of jobs = 30), the effect of upper bound is still significant.

The upper bound derived from the two-phase heuristic procedures is the least important feature because it is only useful at the early stage of the computation. It is applied to obtain the upper bound since the initialization step of the algorithm. If the upper bound is not calculated at the initialization step, the equivalent or better one can still be obtained after the algorithm has found a sufficiently good incumbent solution and has spent some computational time. However, the good upper bound can save this amount of computational time and allow a number of nodes to be eliminated at the beginning of the search tree, especially for large-sized problem.

On the other hand, the lower bound and dominance criteria are repeatedly calculated at every node, so they are capable of eliminating a larger number of nodes than the upper bound. Note that the upper bound derived from the two-phase heuristic procedures is effective only at the early stage of the computation while the lower bound and dominance criteria are effective for all stages of computation. The lower bound calculates the least cost that

will incur or the best objective value of that node if it is branched further. When adding the lower bound to the associated cost of the particular node, there is a good chance that the cost of branching further will be greater than the current upper bound and the node will be eliminated. The dominance criteria consider only the precedence relations of the adjacent and non-adjacent jobs within the node (the scheduled jobs). They do not consider the effect of other jobs which are still not included in the sequence (the unscheduled jobs). Therefore, the lower bound has a greater effect on the performance of the algorithm than the dominance criteria.

9. CONCLUSIONS

The network representation and branch and bound algorithm are developed to determine a global optimal production schedule on a single machine that minimizes total sequence-dependent setup cost and weighted earliness/tardiness penalties. It is assumed that the idle time cannot be inserted into the schedule. The problem is NP-hard, indicating that finding an optimal solution is difficult. Efficient lower bounds for three sub-problems are presented. The lower bounds of weighted early and tardy sub-problems are based on the algorithm proposed by Li (1997) whereas the lower bound of the sequence-dependent setup sub-problem is based on the proposed heuristic. Dominance criteria developed by Liaw (1999) are modified in order to incorporate sequence-dependent setup cost which is shown to be efficient in reducing the number of nodes in the branch and bound search tree. The two-phase heuristic procedures, including the priority dispatching by Ow and Morton (1989) and the modification of local improvement procedure of Liaw (1999), are used for determining the upper bound. The lower bounds, upper bound, and dominance criteria developed for the branch and bound algorithm are tested and proven effective. The computational results indicate that the lower bound greatly affects the performance of the branch and bound algorithm.

Most of single-machine scheduling problems found in literature are special cases of the problem under consideration in this paper. The early/tardy problems with up to 50 jobs have been optimally solved using the branch and bound algorithm (Li, 1997; Liaw, 1999). The problem addressed in this paper is more general and more complex since it also considers the sequence-dependent setup cost in the objective function. It is capable of solving problem with up to 40 jobs in reasonable time (1 hour). Parameters used in the experiment are set such that three cost elements (setup, early, tardy penalties) do not dominate each other, which makes the problem difficult to solve. It is expected that if one cost element dominates the others, the algorithm will be able to solve bigger problems. Moreover, the scheduling method in this paper can solve some special cases in the literature by only modifying some input parameters.

For the problem investigated in this paper, it is assumed that the idle time cannot be inserted into the schedule. This is the case when the cost of keeping machine idle is greater than the earliness penalties, the capacity of machine is less than the demand, or the algorithm is used for scheduling the bottleneck machine. This paper also assumes that the setup cost is sequence-dependent but the setup time is sequence-independent. The most recent paper by Sourd (in press) considered the problem where the idle time can be inserted and the setup time is sequence-dependent. The computational test has shown that the proposed branch and bound algorithm is limited to the problem with no more than 20 jobs, which is a half of the problem size in this paper. Since the problem introduced by Sourd (in press) is more complex, it is limited to relatively small-sized problems. The algorithm presented in this paper is suitable in the situation when the idle time is not allowed and the setup time is sequence-independent. The application of either algorithm depends on the characteristics of the problem being considered in order to obtain the solution using less time and effort.

In conclusion, for the weighted earliness/tardiness penalties with sequence-dependent setup cost problem where the idle time is not allowed in the schedule, the algorithm presented in this paper appears to work well for reasonable-sized problems. For large-sized problems, however, a heuristic approach is more practical. The algorithm developed in this paper can be used to provide a benchmark for the evaluation of heuristic algorithms. It is also useful for developing a heuristic based on the branch and bound approach. Recent developments in simulated annealing and tabu-search methods provide interesting approaches for finding good heuristic solutions.

ACKNOWLEDGEMENT

This research is supported by the Royal Golden Jubilee Ph.D. Program of Thailand Research Fund, contract number PHD/0084/2544.

APPENDIX A. Proof of Adjacency condition.

Let $c(ij)$ be the cost of sub-sequence $\{i,j\}$ and $c(ji)$ be the cost of inverse sub-sequence $\{j,i\}$

There are six cases, which may occur for job i and j in the sequence as follows:

1. Both jobs are early in both positions.
2. One job is early in both positions, but another is early in the first position and tardy in the second position.
3. Both jobs are early in the first position and tardy in the second position.
4. One job is tardy in both positions, but another is early in both positions.
5. One job is tardy in both positions, but another is

early in the first position and tardy in the second position.

6. Both jobs are tardy in both positions

The illustrations of all cases are shown in figure 5. The proof is similar to that in Ow and Morton(1989) with an addition of sequence-dependent setup cost.

Case 1. Both jobs are early in both positions.

Position of jobs i and j can be interchanged without affecting the cost of other jobs in the sequence. If adjacency condition holds, then $c(ij) \leq c(ji)$.

$$TSC_{ij} = SC_{xi} + SC_{ij} + SC_{jy} \quad (40)$$

$$c(ij) = h_i(d_i - t - p_i) + h_j(d_j - t - p_i - p_j) + TSC_{ij} \quad (41)$$

$$c(ij) = h_i(d_i - t - p_i) + h_j(d_j - t - p_j) - h_j p_i + TSC_{ij} \quad (42)$$

$$TSC_{ij} - h_j p_i = c(ij) - h_i(d_i - t - p_i) - h_j(d_j - t - p_j) \quad (43)$$

Similarly,

$$TSC_{ji} = SC_{xj} + SC_{ji} + SC_{iy} \quad (44)$$

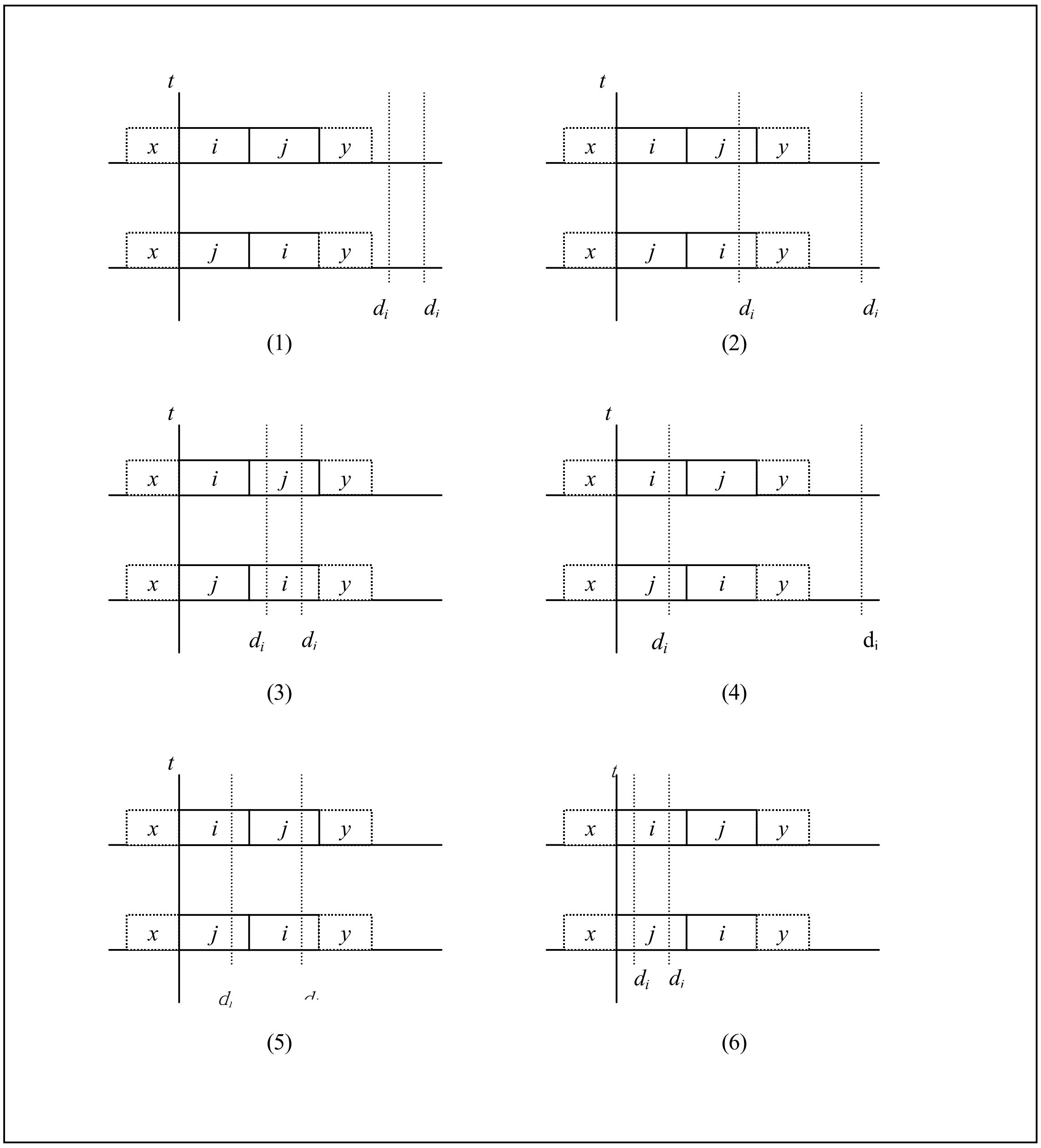

Figure 5. Illustration of adjacency condition cases 1 - 6

$$c(ji) = h_j(d_j - t - p_j) + h_i(d_i - t - p_i - p_j) + TSC_{ji} \quad (45)$$

$$c(ji) = h_j(d_j - t - p_j) + h_i(d_i - t - p_i) - h_i p_j + TSC_{ji} \quad (46)$$

$$TSC_{ji} - h_i p_j = c(ji) - h_i(d_i - t - p_i) - h_j(d_j - t - p_j) \quad (47)$$

Since job i is early in both positions (d_i - t - $p_i \geq p_j$) thus, $\Omega_{ij} = p_j$. Similarly, job j is early in both positions (d_j - t - $p_j \geq p_i$) thus, $\Omega_{ji} = p_i$. Substituting these values into adjacency condition gives:

$$TSC_{ji} + w_i p_j - p_j(w_i + h_i) \geq w_j p_i - p_i(w_j + h_j) + TSC_{ij} \quad (48)$$

By simplifying,

$$TSC_{ji} - h_i p_j \geq - h_j p_i + TSC_{ij} \quad (49)$$

Substituting (43) and (47) into (49) gives:

$$c(ij) \leq c(ji) \quad (50)$$

Case 2. One job is early in both positions, but another is early in the first position and tardy in the second position.

$$c(ij) = h_i(d_i - t - p_i) + h_j(d_j - t - p_i - p_j) + TSC_{ij} \quad (51)$$

$$c(ij) = h_i(d_i - t - p_i) + h_j(d_j - t - p_j) - h_j p_i + TSC_{ij} \quad (52)$$

$$TSC_{ij} - h_j p_i + h_i(d_i - t - p_i) = c(ij) - h_j(d_j - t - p_j) \quad (53)$$

and,

$$c(ji) = h_j(d_j - t - p_j) + w_i(t + p_i + p_j - d_i) + TSC_{ji} \quad (54)$$

$$c(ji) = h_j(d_j - t - p_j) - w_i(d_i - t - p_i) + w_i p_j + TSC_{ji} \quad (55)$$

$$TSC_{ji} + w_i p_j - w_i(d_i - t - p_i) = c(ji) - h_j(d_j - t - p_j) \quad (56)$$

Since job i is early in the first position only ($s_i < p_j$), thus $\Omega_{ij} = s_i$. While job j is early in both positions ($s_j \geq p_i$), thus $\Omega_{ji} = p_i$. Substitute these values in adjacency condition and then simplify the expression as presented in formula (57).

$$TSC_{ji} + w_i p_j - s_i(w_i + h_i) \geq w_j p_i - p_i(w_j + h_j) + TSC_{ij} \quad (57)$$

Substituting $s_i = d_i - t - p_i$ gives:

$$\begin{aligned} &TSC_{ji} + w_i p_j - (d_i - t - p_i)(w_i + h_i) \\ &\geq w_j p_i - w_j p_i - h_j p_i + TSC_{ij} \end{aligned} \quad (58)$$

By simplifying

$$\begin{aligned} &TSC_{ji} + w_i p_j - w_i(d_i - t - p_i) \\ &\geq - h_j p_i + h_i(d_i - t - p_i) + TSC_{ij} \end{aligned} \quad (59)$$

Similarly, substituting (53) and (56) into (59) gives:

$$c(ij) \leq c(ji) \quad (60)$$

The same procedure is repeated for the remaining cases of i and j.

APPENDIX B. Proof of Non-adjacency condition.

Similar to the proof of adjacency condition, there are also six cases which can occur for jobs i and j in the sequence. Non-adjacency condition can be applied only when the processing time of non-adjacent pairs of jobs are equal. Therefore, changing the position of jobs i and j are not affecting the cost of other jobs in the sequence. If non-adjacency condition holds, then $c(ij) \leq c(ji)$.

Case 1. Both jobs are early in both positions as shown in figure 6.

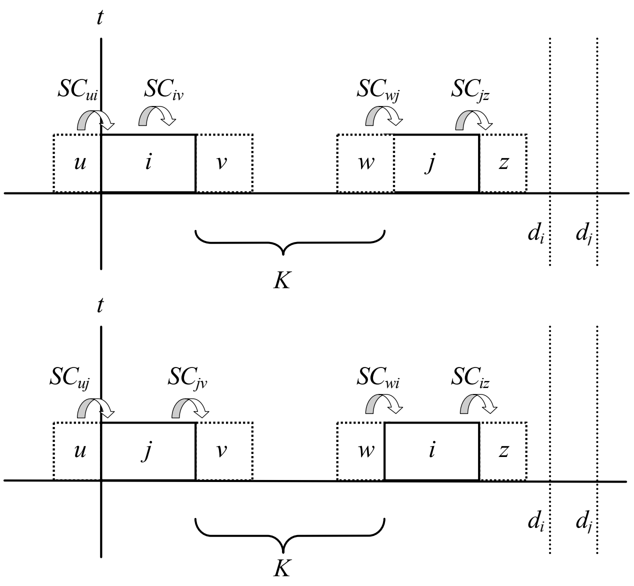

Figure 6. Illustration of non-adjacency condition case 1.

$$TSC_{ij} = SC_{ui} + SC_{iv} + SC_{wj} + SC_{jz} \quad (61)$$

$$c(ij) = h_i(d_i - t - p_i) + h_j(d_j - t - p_i - K - p_j) + TSC_{ij} \quad (62)$$

$$c(ij) = h_i(d_i - t - p_i) + h_j(d_j - t - p_j) - h_j(p_i + K) + TSC_{ij} \qquad (63)$$

$$TSC_{ij} - h_j(p_i + K) = c(ij) - h_i(d_i - t - p_i) - h_j(d_j - t - p_j) \qquad (64)$$

and,

$$TSC_{ji} = SC_{uj} + SC_{jv} + SC_{wi} + SC_{iz} \qquad (65)$$

$$c(ji) = h_j(d_j - t - p_j) + h_i(d_i - t - p_j - K - p_i) + TSC_{ji} \qquad (66)$$

$$c(ji) = h_j(d_j - t - p_j) + h_i(d_i - t - p_i) - h_i(p_j + K) + TSC_{ji} \qquad (67)$$

$$TSC_{ji} - h_i(p_j + K) = c(ji) - h_i(d_i - t - p_i) - h_j(d_j - t - p_j) \qquad (68)$$

Since job i is early in both positions (d_i - t - $p_i \geq p_j$ + K), thus, $\Delta_{ij} = p_j + K$. Similarly, job j is early in both positions (d_j - t - $p_j \geq p_i + K$) thus, $\Delta_{ji} = p_i + K$. Substituting these values into non-adjacency condition gives:

$$\begin{aligned} &TSC_{ji} + w_i(p_j + K) - (p_j + K)(w_i + h_i) \\ &\geq w_j(p_i + K) - (p_i + K)(w_j + h_j) + TSC_{ij} \end{aligned} \qquad (69)$$

This can be reduced to:

$$TSC_{ji} - h_i(p_j + K) \quad \geq \quad - h_j(p_i + K) + TSC_{ij} \qquad (70)$$

Similarly, substituting (64) and (68) into (70) gives:

$$c(ij) \quad \leq \quad c(ji) \qquad (71)$$

The same procedure can be repeated for the remaining cases of jobs i and j.

REFERENCES

Adul-Razaq, T.S., and Potts, C.N. (1988) Dynamic programming state-space relaxation for single-machine scheduling, *Journal of Operation Research Society*, **39**, 141-152.

Allahverdi, A., Gupta, J.N.D., and Aldowaisan T. (1999) A review of scheduling research involving setup considerations, *OMEGA The International Journal of Management Science*, **27**, 219-239.

Azizoglu, M., and Webster, S. (1997) Scheduling job families about an unrestricted common due date on a single machine, *International Journal of Production Research*, **35**, 1321-1330.

Baker, K.R., and Scudder, G.D. (1990) Sequencing with earliness and tardiness penalties: A review, *Operations Research*, **38**, 22-35.

Chang, P.C. (1999) A branch and bound approach for single machine scheduling with earliness and tardiness penalties, *Computers and Mathematics with Applications*, **37**, 133-144.

Chen, J.Y., and Lin, S.F. (2002) Minimizing weighted earliness and tardiness penalties in single machine scheduling with idle time permitted, *Naval Research Logistics*, **49**, 760-780.

Coleman, B.J. (1992) Technical note: A simple model for optimizing the single machine early/tardy problem with sequence-dependent setups, *Production and Operation Management*, **1**, 225-228.

Driscoll, W.C., and Emmons, H. (1977) Scheduling production on one machine with changeover costs, *AIIE Transaction* , **9**, 388-395.

Glassey, C.R. (1968) Minimum changeover scheduling of several products on one machine, *Operation Research* , **16**, 342-352.

Hu, T.C., Kuo, Y.S., and Ruskey, F. (1987) Some optimum algorithms for scheduling problems with changeover costs, *Operations Research*, **35**, 94-99.

Ibaraki, T., and Nakamura, Y. (1994) A Dynamic programming method for single machine scheduling, *European Journal of Operation Research*, **76**, 72-82.

Li, G. (1997) Single machine earliness and tardiness scheduling, *European Journal of Operational Research*, **96**, 546-558.

Liaw, C.F. (1999) A branch and bound algorithm for the single machine earliness and tardiness scheduling problem, *Computer & Operations Research*, **26**, 679-693

Mondal, S.A., and Sen, A.K. (2001) Single machine weighted earliness-tardiness penalty problem with commom due date. *Computers and Operations Research*, **28**, 649-669.

Morton, T.E., and Pentico, D.W. (1993) *Heuristic Scheduling Systems, with applications to production systems and project management*, John Wiley & Sons, INC, 172

Ow, P.S., and Morton, T.E. (1989) The single machine early/tardy problem, *Management Science*, **35**, 177-191.

Potts, C.N., and Van Wassenhove, L.N. (1985) A branch and bound algorithm for the total weighted tardiness problem, *Operations Research*, **33**, 363-377.

Rabadi, G., Mollaghasemi, M., and Anagnostopoulos, G.C. (2004) A branch and bound algorithm for the early/tardy machine scheduling problem with a common due date and sequence-dependent setup time, *Computers & Operations Research*, **31**, 1727-1751.

Shaller, J. (2004) Single machine scheduling with early and quadratic tardy penalties, *Computers & Industrial Engineering,* **46**, 511-532.

Sourd, F. (in press) Earliness-tardiness scheduling with setup considerations, *Computers & Operations Research*.

Ventura, J.A., and Radhakrishnan, S. (2003) Single machine scheduling with symmetric earliness and tardiness penalties, *European Journal of Operational Research*, **144**, 598-612.

Yano, C.A. and Kim, Y.D. (1991) Algorithms for a class of single-machine weighted tardiness and earliness problems, *European Journal of Operational Research*, **52**, 167-178.