

## Single-machine batch scheduling minimizing weighted flow times and delivery costs with job release times

Mohammad Mahdavi Mazdeh<sup>a\*</sup>, Ali Naji Esfahani<sup>a</sup>, Seyyed Ershad Sakkaki<sup>a</sup> and Amir Ebrahimzadeh Pilerood<sup>b</sup>

<sup>a</sup>Department of Industrial Engineering, Iran University of Science & Technology, Tehran, Iran

<sup>b</sup>Department of Industrial Engineering, Iran University of Science & Technology, Tehran, Iran

### ARTICLE INFO

#### Article history:

Received 25 November 2011

Accepted 11 January 2012

Available online

22 January 2012

#### Keywords:

Scheduling

Single machine

Batch delivery

Branch and bound

Weighted flowtimes

### ABSTRACT

This paper addresses scheduling a set of weighted jobs on a single machine in presence of release date for delivery in batches to customers or to other machines for further processing. The problem is a natural extension of minimizing the sum of weighted flow times by considering the possibility of delivering jobs in batches and introducing batch delivery costs. The classical problem is NP-hard and then the extended version of the problem is NP-hard. The objective function is that of minimizing the sum of weighted flow times and delivery costs. The extended problem arises in a real supply chain network by cooperation between two layers of chain. Structural properties of the problem are investigated and used to devise a branch-and-bound solution scheme. Computational experiments show the efficiency of suggested algorithm for solving instances up to 40 jobs.

© 2012 Growing Science Ltd. All rights reserved

## 1. Introduction

In this paper, a specific type of batch scheduling problems named "batch delivery problems" is studied. In batch-delivery scenario, a job may be immediately deliver to the customer after its process as a single job batch or it may stay to be delivered to customer later with other jobs as a batch with more than one job. In the first state, the number of delivery costs increases while in the second state the completion times increases. This kind of problem was first introduced by Cheng & Kahlbacher. They solved the problem for a single machine in order to minimize the sum of total weighted earliness and delivery costs when the jobs were to be dispatched to the customers in batches (Kahlbacher & Cheng, 1993). This paper addresses the problem of minimizing the sum of total weighted flow time and delivery costs in presence of release date when jobs are to be delivered to different customers in batches i.e.  $(1|r_j|\sum F_i w_i + \sum \delta_j d_j)$ .

The problem of scheduling jobs with release dates on a single machine to minimize the total weighted completion time has been studied, extensively. Smith showed that the simplest form of the problem, i.e. while there is no release date and the weights of jobs are the same and equal to 1, will be solved

\* Corresponding author. Tel.: +98 912 350 68 63  
E-mail: mazdeh@iust.ac.ir (M Mahdavi Mazdeh)

simply by using SPT rule and if the weights are not the same, i.e.  $(1|\sum F_i w_i)$ , it can be solved by using WSPT method (Smith, 1956). Lenstra et al. (1997) showed that this problem is strongly NP-Hard in presence of release date when the weights are either the same or different. The complexity of the problem leads authors to use heuristics and Branches and Bounds (B&B) methods. Dessouky and Deogun (1981), Chu (1992), Chandra (1979) presented a branch and bound algorithm for the problem of  $(1|r_i|\sum F_i)$  and other authors including Bianco and Ricciardelli (1982), Belouadah et al. (1992), Harriri and Potts (1983) presented a branch and bound algorithm for the weighted version of the problem, i.e.  $(1|r_i|\sum F_i w_i)$ .

Problems that address objective functions with both machine scheduling and delivery costs appear to be rather complex, though, they are more practical than those which involve just one of these two factors. These types of combined optimizations are often encountered when a real-world supply chain management is considered.

Hall and Potts (2003) offered a dynamic programming solution to solve the problem of  $(1|\sum F_i + \sum \delta_j d_j)$  to minimize the total flow time and delivery costs, simultaneously. Mahdavi et al (2007) presented a branch and bound algorithm for the same problem. Hall and Potts (2003) also solved the problem when release dates are applied to each job and with the assumption of  $p_1 < p_2$  then  $r_1 < r_2$  and showed that the complexity of problem is  $(On^{3h})$ , when  $n$  and  $h$  show the number of jobs and customers, respectively. Mahdavi et al. (2008) offered a branch and bound algorithm for the same problem and showed an advantage over the dynamic programming solution of Hall and Potts.

Ji et al. (et al., 2007) studied the problem of minimizing the sum of total weighted flow time and delivery costs when there is no release date and all jobs are processed for one customer i.e.  $(1|\sum F_i w_i + \sum \delta_j d_j)$ . They showed that this problem is NP-hard even if the number of batches is constant. Mahdavi et al. (2011) offered a branch and bound algorithm for this problem and the extended problem, i.e. the situation that there is no constraint on the number of batches.

In this paper, we extend the same problem in the presence of release date and when the number of customers is more than one with the assumption that for each pair of jobs  $i$  and  $j$ , whenever  $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$

then  $r_i \leq r_j$ , and for each customer the job with the smaller  $\frac{p}{w}$  should be delivered earlier where,  $p$ ,  $w$  and  $r$  denote the processing times, weights and release times of jobs, respectively. This assumption may appear restrictive at first sight. However, within the framework of supply chain management, this condition may be enforced as part of the coordination between the supplier (upstream stage) and the manufacturer.

In this paper, we study the structural properties of the problem, derive an efficient upper and lower bounds, offer a branch-and-bound solution scheme for solving it and consider the efficiency of computational results.

## 2. Problem definition

Let  $n$  be the jobs to be processed on a single machine, which can process at most one job at the moment and no preemption is allowed. Each job  $i$  has a processing time  $p_i$  and a release date and a weight shown by  $r_i$  and  $w_i$ , respectively. The jobs are to be delivered to customers in batches. A group of jobs, which are destined for a customer, may form a batch if they are all delivered to their customer together. The number of jobs in a batch identifies the size of that batch and a batch with size 1 will be considered as single job batch. Let  $d_j$  be the non-negative cost of delivering a batch to

customer  $j$ . The objective function is to minimize total weighted flow time plus delivery costs in the presence of release dates. Thus, according to the standard classification scheme for scheduling problems introduced by Graham (Graham, 1979), the objective function is  $(\sum F_i w_i + \sum \delta_j d_j)$ , where  $\delta_j$  identifies the number of batches delivered to customer  $j$ . Delivering each job separately in a single job batch will decrease the amount of first term of the objective function ( $\sum F_i w_i$ ). However, delivering the jobs in the batches with the greater size will decrease the second term ( $\sum \delta_j d_j$ ). Therefore, the problem is to decide whether to form a batch or to deliver jobs, separately. According to Mahdvi et al. (2008) there are two strategies to form batches: One is to form continuous batches to be processed, successively when there is no idle time. Another is that of forming discontinuous batches, where the jobs are processed, separately but delivered together. In this scenario, there is at least one job which belongs to another customer or an idle time during the time in which the batch is being formed. In this paper, we offer an efficient branch-and-bound algorithm with the assumption that for each pair of jobs  $i$  and  $j$ , whenever  $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$  then  $r_i \leq r_j$  and for each customer the job with the smaller  $\frac{p}{w}$  should be delivered earlier.

### 3. Propositions

#### Property 1

When two jobs ' $i$ ' and ' $j$ ', with  $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$ , form a continuous batch, the batch is ready for processing at time  $R_b$ , where  $R_b = \text{Max}\{r_i, r_j - p_i, t\}$  and ' $t$ ' is defined as machine ready time. This property is easily extendable to more than two jobs.  $\square$

#### Proposition 1

Consider a set of continuous batches with constraint on the release dates,  $R$ , such that whenever  $T_i < T_j$  then  $R_i < R_j$ , where  $T$  is the batch effective time defined as  $T = \frac{F_b}{W_b}$ , where  $F_b$  is the total processing time of the batch and  $W_b$  is the total weight of the batch. The sequence ordered by the weighted shortest effective batch time ( $WSEBT$ ) is optimal in terms of total weighted flow time.

#### Proof

Consider a schedule " $s$ " that is not ordered by  $WSEBT$ . In this schedule, there must be at least two adjacent batches, say " $Y$ " is followed by " $X$ " when  $R_x < R_y$  and  $T_x < T_y$ . Assume machine is ready for processing each one of the two batches at time " $t$ ". Total weighted flow time of the partial schedule composed of the two batches is:

$$F = (\text{Max}\{t, R_y\} + F_y) \times W_y + (\text{Max}\{t, R_x\} + F_y + F_x) \times W_x.$$

Now perform adjacent pairwise interchange on batches " $Y$ " and " $X$ " to form a new sequence, " $s'$ ", with all other batches remaining in their original positions. The completion times of all preceding batches remain unchanged. However, the completion times of the two batches interchanged and all succeeding batches need to be considered. The new total flow time of the partial schedule of the two jobs is as follows,

$$F' = (\text{Max}\{t, R_x\} + F_x) \times W_x + (\text{Max}\{R_y, \text{Max}\{t, R_x\} + F_x\} + F_y) \times W_y.$$

We introduce parameters  $A$  and  $B$  as follows:

$$A = (\text{Max}\{t, R_x\} + F_x) \times W_x + (\text{Max}\{R_y, \text{Max}\{t, R_y\} + F_x\} + F_y) \times W_y,$$

$$B = (\text{Max}\{t, R_y\} + F_y) \times W_y + (\text{Max}\{t, R_x\} + F_x + F_x) \times W_x.$$

Since  $R_x < R_y$ , then  $F' \leq A$  and  $F \geq B$ ,

After simplifying, we get

$$A = (\text{Max}\{t, R_x\} + F_x) \times W_x + (\text{Max}\{t, R_y\} + F_x + F_y) \times W_y.$$

On the other hand,  $B - A = F_y W_x - F_x W_y$ , and  $\frac{F_x}{W_x} < \frac{F_y}{W_y}$ , thus  $B > A$  and  $F > F'$ .

Thus, interchanging the positions of "X" and "Y" reduces overall weighted flow time.  $\square$

By considering each batch as a single job batch, the following corollary then follows immediately.

### Corollary 1

For a set of jobs, under the assumptions that for each pair of jobs,  $r_i < r_j$  whenever  $\frac{p_i}{w_i} < \frac{p_j}{w_j}$ , the sequence ordered by *WSPT* is optimal in terms of total weighted flow time.

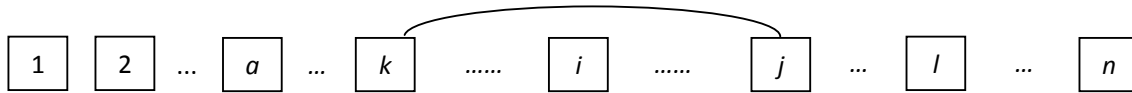
### Proposition 2

For a set of discontinuous batches, with the following two constraints, the sequence ordered by weighted shortest effective discontinuous batch time (*WSEDBT*) is optimal in terms of total weighted flow time:

- First: whenever  $Q_i \leq Q_j$  then  $r_i < r_j$ , where  $Q$  is effective discontinuous batch time defined as  $Q_i = \frac{p_i}{W_b}$ , where  $p_i$  is the processing time of job "i" and  $W_b$  is the total weight of the discontinuous batch that includes job "i" or it is a single job batch with weight  $w_i$ , and "r" is the release date,
- Second: for each job "k" of each discontinuous batch,  $F_k \leq r_{k+1}$ , where  $F_k$  is the finish time of job k, and  $r_{k+1}$  is the release date of the succeeding job of job "k" in the sequence.

*Proof:*

Consider the following sequence. Without any restriction, assume there is a set of single job batches where only one discontinuous batch,  $d$ , which is formed by jobs "k" and "j".



According to the assumptions, we have:

$$\frac{p_1}{w_1} < \frac{p_2}{w_2} < \dots < \frac{p_a}{w_a} < \dots < \frac{p_k}{W_b = (w_k + w_j)} < \dots < \frac{p_i}{w_i} < \dots < \frac{p_j}{W_b = (w_k + w_j)} < \dots < \frac{p_l}{w_l} < \dots < \frac{p_n}{w_n}$$

while

$$r_1 < r_2 < \dots < r_a < \dots < r_k < \dots < r_i < \dots < r_j < \dots < r_l < \dots < r_n$$

Four cases need to be distinguished:

1. Moving job "k" to a new position in the right side of the original position.  
This case is not possible according to the second part of the constraints  $F_k \leq r_{k+1}$ .
2. Moving job "k" to a new position in the left side of the original position.  
Since the completion time of batch "b" is equal to the completion time of the last job in the batch, i.e. job "j", thus moving the position of job "k" to the left side cannot decrease the total completion time of batch while, the completion times of jobs that move after job "k" will increase and therefore the total weighted flow time of sequence will obviously increase.
3. Moving job "j" to a new position in the left side of the original position.
4. Moving job "j" to a new position in the right side of the original position.

It can be shown by standard interchange argument that moving job "j" to the left, if it is possible, or moving it to the right side will increase the total flow time of sequence.

Thus, the proof is completed and it can be extended in the same manner for the situation where there is more than one discontinuous batch or while the sizes of batches are greater. □

However, it is worth to note that by establishing a discontinuous batch when the second assumption of the proposition 2 is not true, moving the position of job "k" to the right side may decrease the total weighted flow time. To make this matter clear, let us consider the following example.

*Example:*

Consider the following 3 jobs when job (1) and job (3) form a discontinuous batch. We show that the sequence ordered by WSEDBT is not optimal in terms of total weighted flow time.

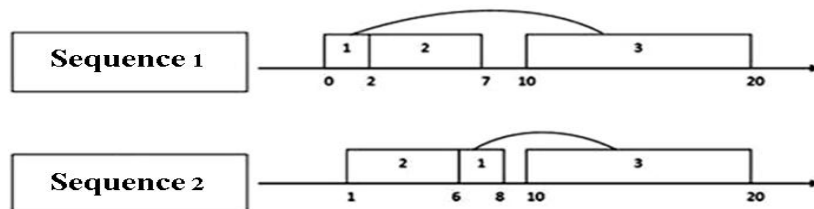
Jobs	Processing time	Weight	Release date	$\frac{P_i}{W_b}$
1	2	5	0	$\frac{2}{7}$
2	5	5	1	$\frac{5}{5}$
3	10	2	10	$\frac{10}{7}$

Sequencing the jobs according to WSEDBT leads to sequence (1) in which job 1 and job 3 form a discontinuous batch. The total weighted flow time of this sequence is as follows,

$$(1): \quad 5 \times 7 + (5 + 2) \times 20 = 175$$

This is while the interchange of jobs 1 and 2 lead to Sequence (2) with the following total weighted flow time:

$$(2) \quad 5 \times 6 + (5 + 2) \times 20 = 170$$



**Fig. 1.** Sequence 1 and sequence 2.

As it is shown in Fig. 1, the batch (Chandra, 1979) will remain discontinues, because job 3 is not immediately ready for processing after the completion time of job 1.

When the second assumption of the proposition 2 does not hold, the following algorithm will lead to the optimum sequence:

**Algorithm 1:**

Step 1: Make sure that all single job batches are ordered according to WSEDBT,

Step 2: Consider a discontinuous batch that consists of jobs  $l$  to  $m$

For  $i = m-1$  to  $l$

Consider job (i) in position (k) of the sequence

While:

a. There is at least one job between job  $i$  and next job of the batch.

b.  $F_k > r_{k+1}$ ,

Do: Swap jobs in positions  $k$  and  $k+1$ ,

Calculate the total weighted flow time,

If: it has improved,

$k = k+1$ ,

Continue while,

Else:

Undo the change.

Exit while.

The same process of step 2 must be repeated for all discontinuous batches of the sequence. □

In general, the sequence ordered by "Weighted Shortest Effective Remaining Batch Time" (WSERBT) is not optimal for the preemptive case and thus does not necessarily provide a lower bound on the optimal solution for the problem without preemption (Chou, 2004). But this on-line algorithm yields an optimal schedule under some circumstances (Labetoulle et al., 1984). In the next proposition, one of these circumstances will be considered.

**Proposition 3**

Consider a set of continuous batches, in the absence of any constraints on the inter-relationships among job release times. Assume that for each pair of jobs  $i$  and  $j$ , whenever  $\frac{p_i}{w_i} \leq \frac{p_j}{w_j}$  then  $r_i \leq r_j$ , if

preempt-resume is allowed then the sequence ordered by the weighted shortest effective remaining batch time (WSERBT) is optimal in terms of total weighted flow time, with batch effective time being  $T = \frac{F_b}{W_b}$ .

*Proof:*

Generalizing an idea from Baker for minimizing the sum of flow times for a set of jobs (Baker, 1974) and in the light of Proposition 1, we can show that when preempt-resume prevails, the optimal rule for a set of batches is always to keep the machine assigned to the available batch with minimum remaining weighted effective batch time. □

**Proposition 4**

In a partial schedule, where some batches have been formed, but no decision has been taken yet on batching the remaining 'un-batched' jobs, a lower bound on the sum of job flow times will be

computed by considering each un-batched job as a single-job batch and sequencing all batches in the order of *WSEBT* (by virtue of proposition 1), or *WSEDBT* (by virtue of proposition 2 and algorithm 1) or *WSERBT* (by virtue of proposition 3).

*Proof:*

By batching un-batched jobs, we will necessarily delay some jobs. Hence, considering each job as a single-job batch ensures no delay, after that, sequencing the batches by virtue of propositions 1, 2, 3 and algorithm 1 (if necessary) ensures that the resulting schedule minimizes the total weighted flow time.  $\square$

### Proposition 5

In an optimal solution, any batch "*b*" with more than one job, destined for a customer "*j*" will have the property that  $(W_b - w_l) \times p_l \leq d_j$ , where  $W_b$  is defined as the sum of weights of the jobs in the batch "*b*" and  $w_l$  and  $p_l$  are the weight and processing time of the last job in batch "*b*", respectively.

*Proof:*

Consider a batch, which does not have the indicated property. By removing the last job and delivering it in a single batch, by its own, the overall objective function will be decreased at least by  $(W_b - w_l) \times p_l - d_j$ .  $\square$

### Corollary 2

In an optimal solution, any job that has the property that  $p_k > d_j$ , will form a single job batch.

### Proposition 6

In completing a partial schedule, a "batching penalty",  $\Delta_k$ , attaches to each un-batched job *k*. Moreover,  $\Delta_k = \text{Min} \{W_b p_k, w_k p_{\text{Min}}, d_j\}$  where  $d_j$  is the delivery cost of customer "*j*" and  $p_k$  and  $w_k$  are the processing time and weight of job "*k*" and  $W_b$  is the sum of weights of jobs in the considered batch and  $p_{\text{Min}}$  is the minimum processing time among unscheduled jobs belonged to customer "*j*".

*Proof:*

Let job "*k*" be an un-scheduled job. Adding this job to the last formed batch with the same customer will increase the total weighted flow time by at least  $W_b p_k$  (Mazdeh et al., 2011). If job "*k*" may not be added to a formed batch, it would form a new batch by its own. In this situation, two cases should be distinguished:

First case: There may, at least, one of the other un-scheduled jobs (say job "*x*") with the same customer form a batch with job "*k*". It is clear that in this case the total weighted flow time will increase by  $w_k p_k$ . As we do not know in advance the processing time of job "*x*" then we choose the job with the minimum processing time. Therefore, the minimum penalty is  $w_k p_{\text{min}}$ .

Second case: job "*k*" remains as a single-job batch and then an additional batch delivery cost will be incurred.

So the 'batching penalty' is the smallest of the three.  $\square$

## 4. Branch and bound

All the jobs are sorted according to *WSPT* rule. The root of the tree is taken to be a single-job batches made of the first job of each customer in the sequence. The search tree to explore the solution area is structured as a trivalent 0-1-2 tree, where each node is branched to 3 nodes. The first one indicates the start of a new batch (0), the second one indicates that a new job is added to an open batch continuously (1), and the third one indicates adding a job to an open batch discontinuously (2). The tree is constructed in a depth-first fashion. In the beginning, all the jobs to form a single job batch by virtue of corollary 2, are identified and separated from other jobs and their corresponding variable is set to zero. We will present the other components of the branch and bound scheme in the following subsections.

### 4.1. Fathoming and backtracking a node

A node is fathomed when:

1. The lower bound calculated in the node exceeds the current upper bound.
2. We reach a leaf node i.e. all variables are fixed

Fathoming initiates backtracking to the parent of current node and the search will terminate when all nodes are fathomed.

### 4.2. Upper bound

An efficient upper bound is vital for branching. A low, sharp and accurate one will fathom more nodes, and reduce the time of calculation of branch and bound. In this paper, a heuristic algorithm to calculate the upper bound is offered. It consists of the following steps:

*Algorithm 2:*

- Form all jobs into single-job batches and sequence the batches in WSEBT (which, in this case, is equivalent to *WSPT* order).
- Calculate the initial start time ( $S_{(i)}$ ) and initial finish time ( $F_{(i)}$ ) of all jobs according to the following:

$$\begin{aligned} \text{If } i=1 \text{ then } S_{(i)} &= R_{(i)} \quad \text{and} \quad F_{(i)} = S_{(i)} + P_{(i)} \\ \text{Else } S_{(i)} &= \text{Max}[R_{(i)}, F_{(i-1)}] \quad \text{and} \quad F_{(i)} = S_{(i)} + P_{(i)} \end{aligned}$$

- Treating the sequence as a circular array, start with the first batch belonging to the current customer;

#### **Repeat**

Scan forward until the first batch that may profitably be joined with the current batch is found. If  $[(s_i + p_i) - (s_k + p_k)] \times w_k < d_j$ , join two batches.

Calculate the finish time and delivery cost of new batch. Move the newly formed batch forward to restore Algorithm 1 order, if necessary.

Move to the next single job batch.

**Until** a complete scan of all batches results in no improvement.

If the upper bound found is less than the incumbent one, the former replaces the latter.

**end.**

### 4.3. Lower bound

In each node of the decision tree, a lower bound is calculated. The lower bound consists of three parts, which are as follows,

**LBF:** LBF is the lower bound on total weighted flow time, which can be calculated in virtue of proposition 4.

**LBD:** LBD is the batch delivery costs of the batches already formed.

**LBP:** LBP is the sum of the lower bounds on the batching penalties, which can be calculated by applying the logic of Proposition 6.

### 5. Numerical Example

Consider the following two-customer problem, with delivery costs of 2000 and 1800, respectively:

customer	1				2	
Job	1	2	3	4	5	
$p_i$	15	30	150	20	70	
$w_i$	5	6	18	5	10	
$r_i$	0	40	80	10	70	
$\frac{p_i}{W_i}$	3	5	8.3	4	7	

Each batch is shown by  $|$  and if it is a discontinuous batch it is shown by  $\{ \}$ .

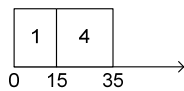
#### Upper Bound

UB is calculated according to the method indicated before. The order is  $\{ \{4,5\}, \{1,2\}, |3 \}$  that brings  $UB= 13890$ .

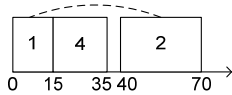
#### Branch and bound

As it was mentioned before, the tree is depth-first.

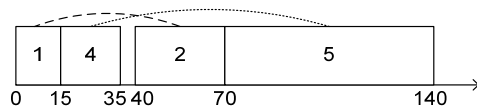
$$S_0 = |1, |4|; \text{LBF}= 7290, \text{LBD}= 3800, \text{LBP}=150+540+350=1040; \text{LB}= 12130.$$



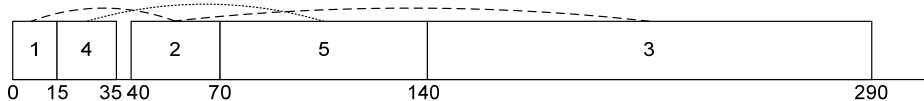
$$S_1 = \{ \{1, 2\} \}, |4|; \text{LBF}= 7565, \text{LBD}= 3800, \text{LBP}=1650+350= 2000; \text{LB}= 13365.$$



$$S_2 = \{ \{1, 2\} \}, \{ \{4, 5\} \}; \text{LBF}= 8090, \text{LBD}= 3800, \text{LBP}=1650; \text{LB}= 13540.$$

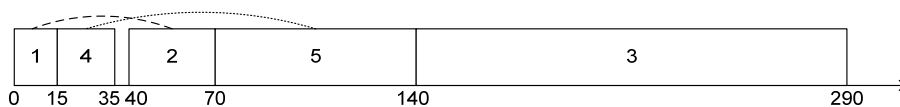


$$S_3 = \{ \{1, 2, 3\} \}, \{ \{4, 5\} \}; \text{LB}= 14310 > \text{UB}, \text{Backtrack}.$$

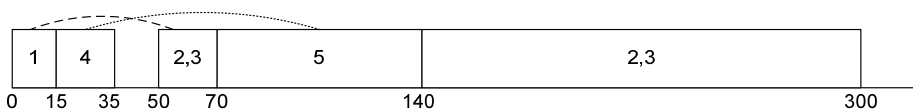


Explanation: The batches are sorted in WRSPT order by virtue of Proposition 3. At  $t = 80$  job 3 is ready, but as  $\frac{60}{15} < \frac{150}{29}$  then the machine continues the process of job "5".

$$S_4 = |\{1, 2\}|, |\{4, 5\}|, |3|; \text{LB} = 13890 = \text{UB}, \text{Backtrack.}$$



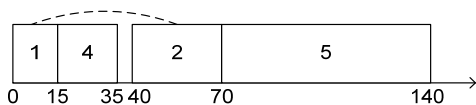
$$S_5 = |\{1, |2, 3|\}|, |\{4, 5\}|; \text{LB} = 14600 > \text{UB}, \text{Backtrack.}$$



Explanation:  $|2, 3|$ :  $p = 180, w = 24, r = 50$ .

The batches are sorted in WRSPT order by virtue of Proposition 3. At  $t = 70$  job 5 is ready, as  $\frac{70}{15} < \frac{160}{29}$ , then by using preempt- resume machine starts the process of job "5".

$$S_6 = |\{1, 2\}|, |4|, |5|; \text{LBF} = 7565, \text{LBD} = 5600, \text{LBP} = 1650; \text{LB} = 14815 > \text{UB}, \text{Backtrack.}$$



$$S_7 = |\{1, 2\}|, |4, 5|; \text{LBF} = 8750, \text{LBD} = 3800, \text{LBP} = 1650; \text{LB} = 14200 > \text{UB} \text{ Backtrack.}$$

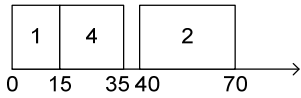


Explanation:  $|4, 5|$ :  $p = 90, w = 15, r = 50$ .

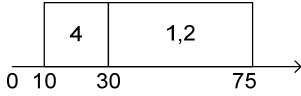
As the first constraint  $\left(\frac{15}{11} < \frac{30}{11} < \frac{90}{15} \text{ and } 0 < 40 < 50\right)$  and the second constraint  $(F_k \leq r_{k+1})$  of

Proposition 2 are satisfied, then we have used WSEDBT.

$$S_8 = |1|, |4|, |2|; \text{LBF} = 7290, \text{LBD} = 5800, \text{LBP} = 900 + 350 = 1250; \text{LB} = 14340 > \text{UB}, \text{Backtrack.}$$



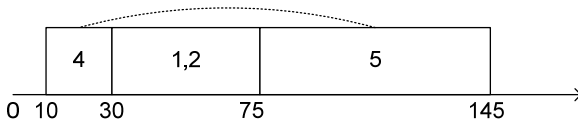
$$S_9 = |4|, |1, 2|; \text{LBF} = 7735, \text{LBD} = 3800, \text{LBP} = 1650 + 350 = 2000; \text{LB} = 13535.$$



Explanation:  $|1, 2|: p = 45, w = 11, r = 25.$

As the constraint of proposition 1  $\left(\frac{20}{5} < \frac{45}{11} \text{ and } 10 < 25\right)$  is satisfied, then we have used WSEBT.

$$S_{10} = |1, 2|, |\{4, 5\}|; \text{LBF} = 8310, \text{LBD} = 3800, \text{LBP} = 1650; \text{LB} = 13760.$$

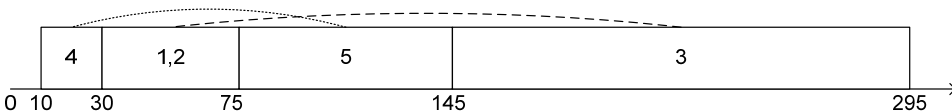


Explanation:  $|1, 2|: p = 45, w = 11, r = 25.$

The first constraint of proposition 2  $\left(\frac{20}{15} < \frac{45}{11} < \frac{70}{15} \text{ and } 10 < 25 < 70\right)$  is satisfied but the second constraint  $\left[(F_4 = 30) > (r_{|1,2|} = 25)\right]$  is not satisfied: According to algorithm 1, job 4 is swapped with batch  $|1, 2|$  and the total weighted flow time is calculated in both cases:

1. The sequence ordered by WSEBT:  $\text{LBF} = 8310.$
2. The sequence after swapping job 4 with batch  $|1, 2|$ :  $\text{LBF} = 8750.$

$$S_{11} = |\{1, 2, 3\}|, |\{4, 5\}|; \text{LB} = 14530 > \text{UB}, \text{Backtrack}.$$

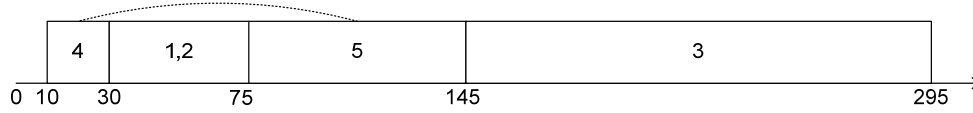


Explanation:  $|1, 2|: p = 45, w = 11, r = 25.$

The batches are sorted in WRSPT order by virtue of Proposition 3. At  $t = 25$  batch "1,2" is ready, but as  $\left(\frac{5}{15} < \frac{45}{29}\right)$  then the machine continues the process of . At  $t = 70$  job "5" is ready, but as

$\left(\frac{5}{29} < \frac{70}{15}\right)$  then the machine continues the process of job"5". At  $t=80$  job "3" is ready, but as  $\left(\frac{65}{15} < \frac{150}{29}\right)$  then the machine continues the process of job"5".

$S_{12} = |1, 2|, |\{4, 5\}|, |3|$ ; LB= 14110 > UB, Backtrack.



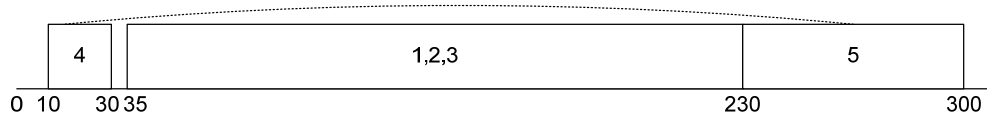
Explanation:  $|1, 2|$ :  $p = 45, w = 11, r = 25$ .

The first constraint of proposition 2  $\left(\frac{20}{15} < \frac{45}{11} < \frac{70}{15} < \frac{150}{18} \text{ and } 10 < 25 < 70 < 80\right)$  is satisfied, but

the second constraint  $\left[(F_4 = 30) > (r_{|1,2|} = 25)\right]$  is not satisfied: According to algorithm 1, job 4 is swapped with batch  $|1, 2|$  and the total weighted flow time is calculated in both cases:

1. The sequence ordered by WSEBT: LBF=14110.
2. The sequence after swapping job 4 with batch  $|1, 2|$ : LBF=14550.

$S_{13} = |1, 2, 3|, |\{4, 5\}|$ ; LB= 14970 > UB, Backtrack.



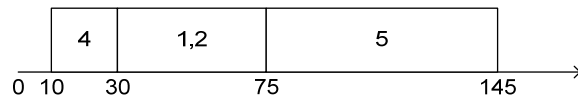
Explanation:  $|1, 2, 3|$ :  $p = 195, w = 29, r = 35$ .

The batches are sorted in WRSPT order by virtue of Proposition 3. At  $t = 70$  job 5 is ready, as  $\left(\frac{70}{15} < \frac{160}{29}\right)$ , then by using preempt- resume machine must start the process of job"5", however,

since this node is a leaf, there must be a feasible sequence of the problem, therefore, preempt-resume is not allowed. So at this node both cases are calculated, according to permutation rule.

1. LB= 15615
2. LB= 14970

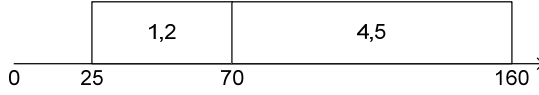
$S_{14} = |4|, |1, 2|, |5|$ ; LBF= 7735, LBD= 5600, LBP=1650; LB= 14985 > UB, Backtrack.



Explanation:  $|1, 2|$ :  $p = 45, w = 11, r = 25$ .

As the constraint of proposition 1  $\left( \frac{20}{5} < \frac{45}{11} < \frac{70}{10} \text{ and } 10 < 25 < 70 \right)$  is satisfied, then we have used WSEBT.

$$S_{15} = |1, 2|, |4, 5|; \text{LBF} = 8750, \text{LBD} = 3800, \text{LBP} = 1650; \text{LB} = 14200 > \text{UB}, \text{Backtrack.}$$



Explanation:  $|1, 2|: p = 45, w = 11, r = 25, |4, 5|: p = 90, w = 15, r = 50.$

As the constraint of proposition 1  $\left( \frac{45}{11} < \frac{90}{10} \text{ and } 25 < 50 \right)$  is satisfied, then we have used WSEBT.

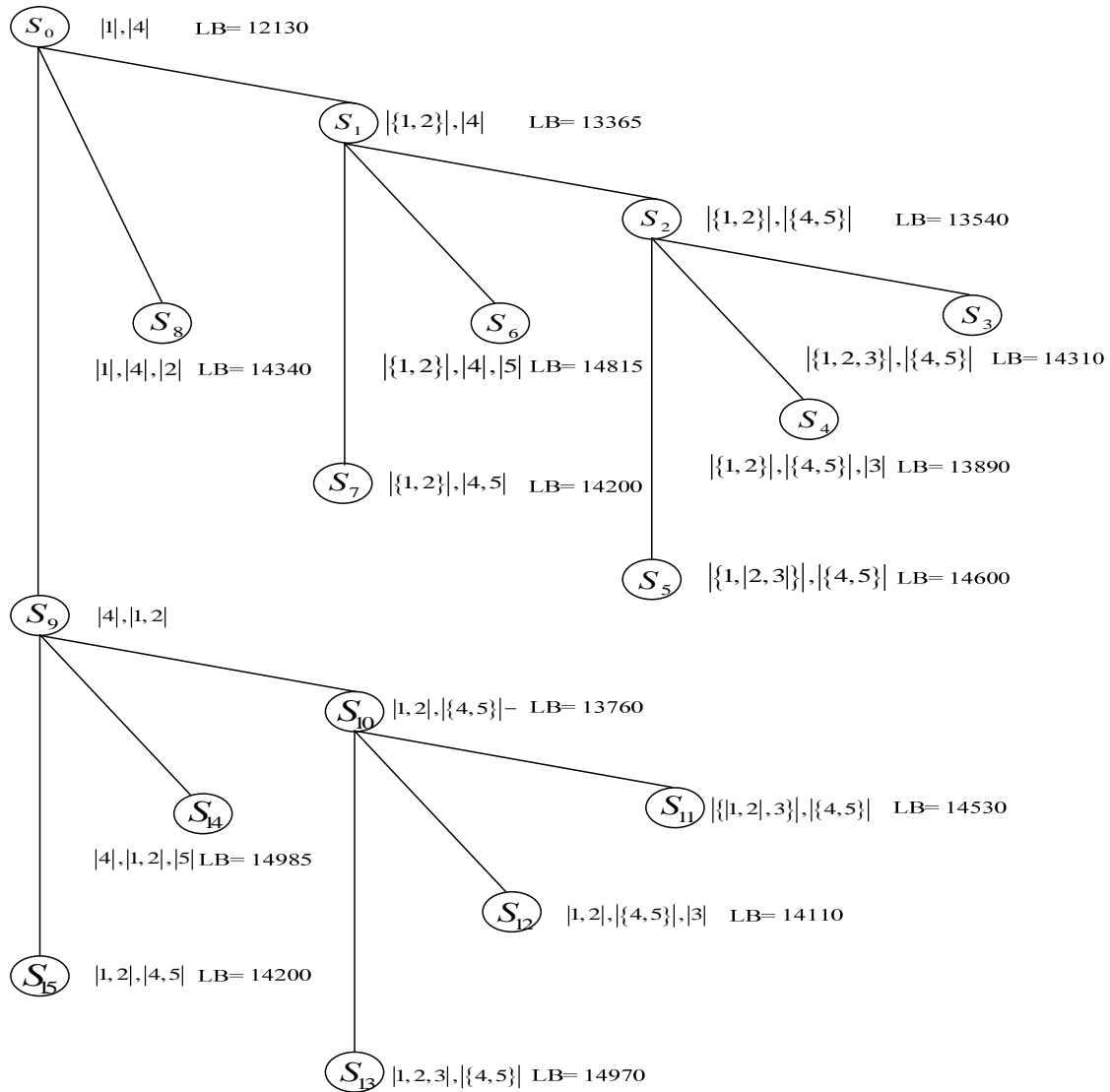


Fig. 2. Search tree

## 6. Computational Results

The computational tests were run with Pentium(R) 4 CPU 3.20 GHZ and 1.00 GB RAM computer. The branch & bound algorithm was coded in Visual basic application.

There are no benchmark instances for B&B test implications. We, therefore, generate a set of problem instances, with 5-40 jobs randomly distributed among customers, where each customer is assigned at least two jobs. The number of customers varied from 2 to 10. Processing times, weights and release dates of jobs were randomly generated integer inside [1-100], [1-10] and [1-100], respectively. Because of the interaction between delivery cost and job weighted flow time that may affect the problem hardness, two classes of instances were generated. In class A, the delivery cost of each customer is randomly generated inside [1001-10000], which is far away from the job processing times, while in class B the delivery costs are randomly generated inside [1-1000]. To ensure that results are representative, each subset consists of 10 instances.

**Table 1**

Running times(Av. = average, SD = standard deviation, Min. = Minimum, Max. = Maximum) class A

Running Times (seconds)					
Number of Customers	Number of Jobs	Av. (seconds)	SD	Min.	Max.
2	5	0.01	0.02	0.00	0.06
	7	0.02	0.01	0.01	0.03
	10	1.28	0.44	0.53	2.27
	13	40.86	38.42	4.78	120.87
	15	204.28	67.98	119.88	331.26
	17	411.34	102.46	275.36	585.12
3	7	0.01	0.02	0.00	0.06
	10	0.23	0.09	0.10	0.36
	13	27.16	14.32	1.66	40.56
	15	185.89	119.81	41.19	361.24
	17	329.96	58.27	235.21	422.12
	20	630.36	166.24	466.52	1073.21
4	8	0.01	0.01	0.00	0.03
	10	0.06	0.05	0.01	0.19
	12	4.11	2.51	1.51	9.45
	15	115.62	169.31	49.12	596.54
	17	300.29	84.22	125.61	381.23
	20	558.73	71.17	452.15	705.63
	23	734.98	95.24	552.53	850.26
5	10	0.02	0.03	0.00	0.09
	15	29.27	4.78	21.23	37.39
	17	262.85	109.81	39.02	363.22
	20	429.88	66.98	312.32	525.62
	23	623.78	42.53	536.56	692.52
7	14	0.82	0.21	0.63	1.33
	17	80.88	200.53	9.72	651.32
	20	497.03	49.15	425.63	572.15
	23	605.24	75.89	492.15	692.15
	25	825.97	106.73	591.23	992.12
10	20	37.70	22.19	12.45	78.12
	23	453.67	106.59	272.15	632.15
	25	612.29	63.36	525.90	692.12

Each of the running times in Tables 1 and 2 represent the average over 10 appropriate instances. The results show the efficiency of B&B algorithm for solving instances with 10 customers and up to 25 jobs for class A and up to 40 jobs in class B.

**Table 2**

Running times(Av. = average, SD = standard deviation, Min. = Minimum, Max. = Maximum) class B

Running times						Running times					
NoC	NoJ	Mean	SD	Min.	Max.	NoC	NoJ	Mean	SD	Min.	Max.
2	5	0.00	0.00	0.00	0.01	5	10	0.02	0.02	0.00	0.05
	7	0.02	0.02	0.00	0.06		15	12.94	7.32	3.52	25.22
	10	0.12	0.19	0.00	0.46		17	36.03	25.89	8.15	84.52
	13	4.14	9.92	0.13	32.15		20	112.46	55.85	7.25	185.22
	15	32.80	44.43	0.13	121.56		23	164.68	83.81	12.35	294.21
	17	60.32	88.68	1.26	282.26		25	296.19	180.48	4.85	525.26
	20	146.86	122.50	13.52	323.46		27	377.23	262.18	15.16	885.82
	22	275.66	160.78	21.52	552.55						
3	7	0.00	0.00	0.00	0.02	7	14	0.38	0.34	0.00	0.80
	10	0.09	0.12	0.00	0.36		17	31.79	77.23	0.77	251.32
	13	11.23	11.87	0.33	32.24		20	103.71	71.19	14.15	243.26
	15	25.98	25.14	1.05	72.32		23	148.57	134.26	14.52	435.41
	17	42.76	30.98	12.57	100.33		25	289.87	260.60	17.33	732.51
	20	116.24	120.92	15.22	355.13		27	307.19	200.37	13.51	712.15
	22	211.50	183.42	49.25	605.47		30	367.67	220.29	45.21	663.15
							35	371.45	234.72	72.86	792.15
4	8	0.00	0.01	0.00	0.02	10	20	14.39	20.15	0.00	55.12
	10	0.01	0.01	0.00	0.03		23	142.86	55.75	62.15	235.45
	12	1.35	1.06	0.00	3.21		25	274.99	177.16	67.41	645.12
	15	14.50	12.25	0.54	32.54		27	289.25	218.06	67.85	732.51
	17	38.73	27.39	7.12	102.51		30	355.06	225.60	72.59	774.85
	20	112.77	82.04	4.52	208.45		35	354.80	214.52	121.25	732.52
	23	177.82	178.44	9.13	552.35		40	410.82	278.63	92.15	961.51
	25	356.15	244.31	10.25	743.15						

NoC: Number of customer, NoJ: Number of Jobs

The maximum number of nodes ( $N$ ) that potentially could be extracted by search tree is countable by using  $N = 1 + 3 + 3^2 + \dots + 3^{n-k}$ , where  $n$  is the number of jobs and  $k$  is the number of customers. Considering the effectiveness of B&B algorithm, the average number of nodes searched for each subset is compared with the maximum number of nodes, which potentially could be extracted.

**Table 3**

Average number of nodes searched in compare with the maximum number of nodes - class A

NoC	NoJ	$N$	ANNS	PRND	NoC	NoJ	$N$	ANNS	PRND	
2	5	40	33	17.50%	5	10	364	247	32.10%	
	7	364	278	23.57%		15	88573	56513	36.20%	
	10	9841	7070	28.16%		17	797161	485934	39.04%	
	13	265720	185195	30.30%		20	21523360	12048845	44.02%	
	15	2391484	1585172	33.72%		23	581130733	313084613	46.12%	
	17	21523360	13965032	35.12%						
3	7	40	33	18.74%	7	14	3280	1976	39.74%	
	10	121	88	27.69%		17	88573	50567	42.91%	
	13	3280	2330	28.96%		20	2391484	1301179	45.59%	
	15	88573	61736	30.30%		23	64570081	34015488	47.32%	
	17	797161	518957	34.90%		25	581130733	286155910	50.76%	
	20	7174453	4472073	37.67%						
4	23	193710244	113432676	41.44%	10	20	88573	46218	47.82%	
	8	121	88	27.16%		23	2391484	1221631	48.92%	
	10	1093	750	31.41%		25	21523360	10395902	51.70%	
	12	9841	6381	35.16%						
	15	265720	165142	37.85%						
	17	2391484	1465171	38.73%						
20	64570081	37098228	42.55%							
23	1743392200	981048849	43.73%							

NoC: Number of customer, NoJ: Number of Jobs, ANNS: Average number of nodes searched, PRND: Percents of reduced nodes

Tables 3 and 4 show that up to 95% of nodes is reduced by using the B&B algorithm. Table 5 shows the average relative error, calculated as  $(\frac{UB}{Optimal} - 1)$  for all replications of all instances for each number of customers. It can be seen that on average it produces solutions that are within 0.26% of the optimum. Moreover, the error is smaller for instances in class B. Effectiveness of the upper bound makes it possible to use it as a fast heuristic.

**Table 4**

Average number of nodes searched in compare with the maximum number of nodes - class B

NoC	NoJ	$N$	ANNS	%	NoC	NoJ	$N$	ANNS	%
2	5	40	22	44.00%	5	10	364	127	65.00%
	7	364	178	51.00%		15	88573	10629	88.00%
	10	9841	984	90.00%		17	797161	49424	93.80%
	13	265720	23915	91.00%		20	21523360	1162261	94.60%
	15	2391484	167404	93.00%		23	581130733	27313144	95.30%
	17	21523360	1485112	93.10%		25	5230176601	240588124	95.40%
	20	581130733	33124452	94.30%		27	47071589413	2071149934	95.60%
	22	5230176601	235357947	95.50%					
3	7	121	54	55.00%	7	14	3280	328	90.00%
	10	3280	321	90.20%		17	88573	5580	93.70%
	13	88573	7706	91.30%		20	2391484	126749	94.70%
	15	797161	53410	93.30%		23	64570081	2905654	95.50%
	17	7174453	459165	93.60%		25	581130733	25569752	95.60%
	20	193710244	10847774	94.40%		27	5230176601	230127770	95.60%
	22	1743392200	78452649	95.50%		30	141214768240	5648590730	96.00%
						35	34315188682441	137260754730	99.60%
4	8	121	48	60.00%	10	40	8338590849833280	8338590849833	99.90%
	10	1093	108	90.10%		20	88573	17715	80.00%
	12	9841	876	91.10%		23	2391484	160229	93.30%
	15	265720	17538	93.40%		25	21523360	1162261	94.60%
	17	2391484	153055	93.60%		27	193710244	8910671	95.40%
	20	64570081	3551354	94.50%		30	5230176601	203976887	96.10%
	23	1743392200	83682826	95.20%		35	1270932914164	11438396227	99.10%
	25	15690529804	721764371	95.40%		40	308836698141973	926510094426	99.70%

NoC: Number of customer, NoJ: Number of Jobs, ANNS: Average number of nodes searched, PRND: Percents of reduced nodes

**Table 5**

Average percentage relative error for upper bound heuristic in class A and B

# of jobs	5	7	10	13	15	17	20	22	25	30	35	40
Class A	0.11	0.19	0.20	0.21	0.22	0.24	0.26	0.25				
Class B	0.01	0.02	0.05	0.05	0.06	0.06	0.06	0.14	0.12	0.11	0.06	0.06

## 7. Conclusions

A branch-and-bound algorithm for scheduling a set of jobs with specified weight and release times to be processed on a single machine for delivery in batches to customers has been presented. The objective function is that of minimizing the sum of weighted flow times and delivery costs. The problem arises in a real supply chain network by cooperation between two layers of chain.

The branch-and-bound algorithm proved to be efficient for solving instances with 10 customers and up to 25 jobs, when the delivery cost is far away from the job processing times and up to 40 jobs while the delivery costs are randomly generated inside [1-1000].

Future work aimed at solving larger problem instances would have to concentrate mainly on sharpening the lower bounds and tightening the upper bounds. However, solving much larger problem instances, albeit approximately, would require developing effective heuristics.

## Acknowledgment

This paper was financially supported by a Iran university grant and the authors would like to thank for the support. The authors would like to thank the anonymous referees for their constructive comments on earlier version of this paper.

## References

- Baker KR. (1974). *Introduction to Sequencing and Scheduling*. Wiley, NewYork, Wiley.
- Belouadah, H., Posner, M.E., & Potts, C.N. (1992). Scheduling with release dates on a single machine to minimize total weighted completion time. *Discrete Applied Mathematics*, 36(3), 213-231.
- Bianco, L. & Ricciardelli, S. (1982). Scheduling of a single machine to minimize total weighted completion time subject to release dates. *Naval Research Logistics Quarterly*, 29(1), 151-167.
- Chandra, R. (1979). On  $n/1/F|$  dynamic deterministic problems. *Naval Research Logistics Quarterly*, 26(3), 537-544.
- Chou, C.F.M. (2004). Asymptotic performance ratio of an online algorithm for the single machine scheduling with release dates. *IEEE Transactions on Automatic Control*, 49(5), 772-776.
- Chu,C. (1992). A branch-and-bound algorithm to minimize total flow time with unequal release dates. *Naval Research Logistics (NRL)*, 39(6), 859-875.
- Dessouky, M. I. & Deogun, J. S. (1981). Sequencing Jobs with Unequal Ready Times to Minimize Mean Flow Time. *SIAM Journal on Computing*, 10(1), 192-202.
- Graham, R.L., Lawler, E.L., Lenstra, J.K., & Kan, A.H.G.R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. InP.L.Hammer (Ed.), *Annals of Discrete Mathematics Discrete Optimization II*, Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium co-sponsored by IBM Canada and SIAM Banff, Aha. and Vancouver (287-326). Elsevier.
- Hall, N. & Potts, C.N. (2003). Supply chain scheduling: batching and delivery. *Operations Research*, 51, 566-584.
- Hariri, A.M.A. & Potts, C.N. (1983). An algorithm for single machine sequencing with release dates to minimize total weighted completion time. *Discrete Applied Mathematics*, 5(1), 99-109.
- Ji,M., He,Y., & Cheng, T.C.E. (2007). Batch delivery scheduling with batch delivery cost on a single machine. *European Journal of Operational Research*, 176(2), 745-755.
- Kahlbacher,H.G. & Cheng,T.C.E. (1993). Parallel machine scheduling to minimize costs for earliness and number of tardy jobs. *Discrete Applied Mathematics*, 47(2), 139-164.
- Labetoulle, J, Lawler, Eugene L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1984). Preemptive scheduling of uniform machines subject to release dates. *Stichting Mathematisch Centrum*, 245-261.
- Lenstra, J.K., Rinnooy Kan, A.H.G., & Brucker,P. (1977). Complexity of Machine Scheduling Problems. InP.L.Hammer (Ed.), *Annals of Discrete Mathematics Studies in Integer Programming*, 343-362.
- Mahdavi Mazdeh, M., Sarhadi, M., & Hindi, K.S. (2008). A branch-and-bound algorithm for single-machine scheduling with batch delivery and job release times. *Computers & Operations Research*, 35(4), 1099-1111.

- Mazdeh, M.M., Sarhadi, M., & Hindi, K.S. (2007). A branch-and-bound algorithm for single-machine scheduling with batch delivery minimizing flow times and delivery costs. *European Journal of Operational Research*, 183(1), 74-86.
- Mazdeh, M.M., Shashaani, S., Ashouri, A., & Hindi, K.S. (2011). Single-machine batch scheduling minimizing weighted flow times and delivery costs. *Applied Mathematical Modelling*, 35(1), 563-570.
- Smith, W.E. (1956). Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2), 59-66.