Decision Science Letters 10 (2021) 231–240

Contents lists available at GrowingScience

Decision Science Letters

homepage: www.GrowingScience.com/dsl

A single machine multi-job integer batch scheduling problem with multi due date to minimize total actual flow time

Rinto Yusriski[a*], Budi Astuti[a], Damawijaya Biksono[b] and Tika Ayu Wardani[a]

[a]*Department of Industrial Engineering, Universitas Jenderal Achmad Yani, Bandung, Bogor, Indonesia*
[b]*Departement of Mechanical Engineering, Universitas Jenderal Achmad Yani, Bandung, Indonesia*

C H R O N I C L E

Article history:
Received February 20, 2021
Received in revised format:
April 19, 2021
Accepted April 19 2021
Available online
April 19, 2021

Keywords:
Integer Batch Scheduling
Multi-Item
Multi Due Dates

A B S T R A C T

This research deals with a multi-job Integer batch scheduling problem on a single machine with different due dates. Every job demanded one or more parts, and the single machine processed the job into a number of batches. The objective is to minimize total actual flow time, defined as the total flow time of all jobs starting from the arrival to the common due date. The decisions are to determine the sequence of jobs, the number of batches, batch size, and sequence of all batches on a single machine. This research proposes three algorithms, developed based on the longest due date rule (The P1-LDD Algorithm), the adjacent pairwise interchange method (The P2-API Algorithm), and the permutation method (The P3-PM Algorithm). The numerical experience shows that the three algorithms produce an outstanding solution. The P1-LDD Algorithm fits to solve a simple problem. The P2-API Algorithm has superior to solve a big complicated problem. The P3-PM Algorithm has the best performance to solve small complicated problems.

1. Introduction

We discuss a batch scheduling of multi-job with multiple delivery dates on a single machine where every job consists of a number of parts along with a respective due date. The objective is to minimize the total actual flow time, defined by Halim et al. (1994) as the time interval of all parts on all batches to flow on the shop, starting from when that arrives until the due date. This research is relevant to the company that adopts the Just-In-Time system where the company can manage the arrival of the part when the machine is already to process it. This problem is categorized as the serial batch problem where the length of a batch is equal to the sum of processing time of part in it, and common setup time is needed before the machine processes a new batch (Baptiste, 2000). The motivation of this research is to solve the problem in an Indonesian plastic company. A single machine, the so-called extruder machine, is assigned to process a number of jobs where every job consists of a number of units with a respective due date. We identify that the processing time of jobs is different from each other. Since the warehouse capacity is limited, the companies manage the arrival of material to be a number of batches. The company should minimize the time of jobs flowing in the shop and keep on-time delivery (It is the same as minimizing the total actual flow time). The problems determine the number of batches for every job, the arrival time of the baches, the batch sizes, the schedule of the resulting batches, and the schedule of the jobs.

2. Literatur Review

Halim et al. (1994) have discussed batch scheduling with multi-jobs and delivery of all completed jobs at a common due date. This research assumes that the machine can process different parts in a batch and deliver all completed products at the common due date. The researcher proposes some properties and algorithms to solve the problems with the objective is to minimize the total actual flow time. Hazir and Sidhoum (2014) also study the problem to minimize the sum of the weighted earliness and tardiness penalty and the setup cost to create a new batch. They assume that there are K planned customer

* Corresponding author.
E-mail address: yusarisaki@yahoo.co.id (R. Yusriski)

doi: 10.5267/j.dsl.2021.4.002

orders, which all orders ready to process by a single machine at time zero, and all completed jobs delivered on a common due date. They propose some properties to determine the optimal schedule and construct the optimal algorithm to solve the problem. In the literature of batch scheduling, there is a situation where a multi-job is scheduled on a single machine, and the completed job is delivered in a batch (batch delivery). Basir and Karimian (2018) have discussed that problem to minimize tardiness cost and delivery cost simultaneously. They assume that the company can control both delivery time and the quantity of product to the customer. Its decision could decrease the delay cost; however, it also increases the number of transportation. Researchers propose batch delivery to accommodate the trade-off between both tardiness cost and delivery cost. They construct a mixed-integer linear programming (MILP) model and solve it by the genetic algorithm. Another researcher, among of Yin et al. (2013), Rasti-Barzoki et al. (2013), Hamidinia et al. (2012) and Mazdeh et al. (2011) has discussed the similar problem using various objective and construct the solution by dynamic algorithm, branch and bound and Genetic Algorithm. Research on Yusriski et al. (2015a) studies a single machine batch scheduling problem with a common due date, especially on integer batch size cases. The model is similar to Hazir and Sidhum (2014); the difference is that research assumes the delivery of finished parts must coincide with the due date. The researchers adopt the backward approach to schedule the resulting batches, and using the objective the so-called total actual flow time, defined in Halim et al. (1994) as "the total of parts in all batches flowing on the shop start from that arrival to the common due date." The researcher solves the problem using the proposed algorithm and shows that the algorithm produces an optimal solution after comparing tests with an enumeration one. The reader can find similar research in Yusriski et al. (2015b), Yusriski et al. (2016), Yusriski et al. (2018), and Yusriski et al. (2019).

The research on batch scheduling problems considering due date can be found in Xie and Wang (2014). A study by Xie and Wang (2014) discusses a single-machine scheduling problem with batch delivery. The researcher assumes that there are a number of jobs with identical processing times so that the machine can process and then deliver the completed jobs together as a batch. The problem is determining the sequencing of the jobs and the batch sizes (the set of jobs contained in every batch) to minimize the sum of total weighted earliness, tardiness, holding, due date, and delivery costs. The researcher proposes an algorithm developed based on the dynamic programming approach and proved that the optimal sequence is found by batch-LPT order. They also show that the complexity of the algorithm is $O(n^3)$. For a similar study, we refer the reader to the work of Prasetyaningsih et al. (2017) and Maulidya et al. (2019). Mohri et al. (2011) have studied a batch scheduling multi-job with multi due dates. The research is motivated by a real problem in the food manufacturing industry, which is bread production. They describe the situation where the factory makes a number of bread batches and delivers them to several stores with different due dates. Since the bread has expired so the quality of bread should deteriorate along with time, the flow time and delivery coincide at the due dates are very important. The researchers construct the algorithm and find the optimal solution to minimize the total flow time. Nurainun et al. (2016) have deals with batch scheduling on dynamic flow shop. In that research, the researcher considers the multi-item case. Albers and Bruker (1993), Ji et al. (2007) show that most of the batching problem is now polynomially solvable and some of the cases known as NP-Hard. Yet, the literature on this problem is small relatively.

3. The Problem Formulation

Let there be a number of jobs $(J_h, h = 1,2,3,..,K)$ ordered by the customer to be scheduling on a single machine. The processing time of every job is not identical (t_h). Each order consist of a set of demand (n_h) to be processed into a number of batches (N_h) respectively. Hence, The batch sizes per jobs are $(Q_{h[i]}, i = 1,2,3, \dots, N_h)$. There is a common setup time (s_h) before a new batch to be processed by the machine. Also, all completed products of J_h must be delivered to the customer together at their respective common due date (d_h). We assume that the company can manage the arrival of materials at the time when the machine is ready to process a batch $(B_{h[i]})$. The objective is to minimize the sum of the total actual flow time of jobs (TF^a). The decision is to determine Jobs' sequence, the number of batches, batch sizes, and the sequence of the resulting batches.

4. A Mathematical Model

In this section, we describe a mathematical model of the problem. Firstly, let us understand the notation to be used in the model as follows.

$[h]$	:	The index of the sequence of the job, $(h = 1, \dots, K)$ using a backward approach
$[i]$	:	The index of the batch sequence, $(i = 1, \dots, N_h)$ using a backward approach
$J_{[h]}$	:	The job to be processed in the $[h]$-th
$t_{[h]}$	:	The processing time of $J_{[h]}$
$s_{[h]}$	:	The setup time of $J_{[h]}$
$d_{[h]}$	:	The due date of $J_{[h]}$
$d_{[h]}$	:	The number of demand for $J_{[h]}$
N_h	:	The number of batches of $J_{[h]}$
$Q_{[h][i]}$	:	The batch sizes of $J_{[h]}$
$B_{[h][i]}$	:	The starting time of the machine already to process a $Q_{[h][i]}$

TF^a : the total actual Flow time

A mathematical model as shown as follows

$$\text{minimize } TF^a = \sum_{h=1}^{K}\left[\sum_{i=1}^{N_h}\left\{\sum_{j=1}^{i}\left(s_{[h]} + t_{[h]}Q_{[h][j]} + \varphi_{[h]}\right)\right\}Q_{[h][i]}\right] \quad (1)$$

subject to

$$\sum_{h=1}^{K}\left[\sum_{i=1}^{N_{h\cdot}} t_{[h]}Q_{[h][i]} + s_{[h]}\right] \le d_{[1]}, \quad (2)$$

$$\sum_{i=1}^{N_{h\cdot}} Q_{[h][i]} = n_{[h]}, \quad h = 1, \dots, K, \quad (3)$$

$$B_{[h][1]+}t_{[h]}Q_{h[1]} + \varphi_{[h]} = d_{[h]}, \quad h = 1, \dots, K;\ \varphi_{[h]} \ge 0, \quad (4)$$

$$Q_{[h][i]} \ge 1;\ N_{[h]} \ge 1;\ i = 1, \dots, N_h;\ h = 1, \dots, K. \quad (5)$$

Eq. (1) describes the objective function of the total actual flow time. Constraint (2) shows the makespan of all jobs scheduled in the machine must be shorter than the schedule period (start from time zero to the longest due date). It shows that the batches are scheduled using a backward approach. The Constraint (3) means material balances of jobs: the total parts on batches of a job must be the same as their demand. Constraint (4) describes that the first batch in every job could complete at the due date or before it, which means that the first batch's scheduled ready time depends on the due date and the time the previous job is scheduled. The time lag between the scheduled ready time of a job and their due date notated by $\varphi_{[h]}$, where the value of $\varphi_{[h]} \ge 0$. Constraint (5) shows both the batch size minimum and the minimum number of bathes.

5. The Solution Methods

The problem of batch scheduling with multiple jobs and multiple due dates can be viewed as a single job batch scheduling problem with recurring common due dates. However, this problem's decision is not only in determining the number of batches, batch size, and batch scheduling but also in deciding on the sequence of jobs to be scheduled. The simple case is shown by determining the jobs' sequence where the distance between consecutive due dates is sufficient to schedule the jobs. However, in some cases, the distance between the two or more consecutive due dates is very narrow and close together, so it is necessary to determine which job priority should be scheduled first.

5.1 The sequence of jobs

In this section, we are determining the job sequence influenced by the characteristics of the job. There are two characteristics, namely independent and dependent jobs. Independent jobs could be scheduled without being influenced by another job. It is due to the job's completion time, including setup time $(C_{[h]} + s_{[h]})$, is less than the available time or the length of schedule $(L_{[h]})$. Meanwhile, dependent jobs always need to be scheduled simultaneously, and sharing capacity is required. The completion time of a job is the total time it takes for a machine to complete the demand. Completion time job can be calculated using the formula

$$C_{[h]} = \sum_{i=1}^{N} Q_{[i]}t_{[i]}, \qquad i = 1,2,3, \dots., N;\ h = 1,2,3, \dots., K \quad (6)$$

However, the Length of the schedule is the distance-time between two consecutive due dates. The length of the schedule period can be calculated as follows

$$L_{[h]} = d_{[h]} - d_{[h+1]}, \qquad h = 1,2,3, \dots., K \quad (7)$$

A job can be scheduled without being affected by other jobs if that completion time is shorter than its length. However, if the job completion time is longer than the length, the job scheduling decision needs to consider other jobs. In the backward scheduling approach, it is necessary to view jobs with the next index.

Proposition 1. Suppose there are *K* jobs with their respective due date scheduled on a single machine using the backward scheduling approach. Assume that the completion time plus setup time of jobs scheduled is less than the length $(C_{[h]} + s_{[h]} \le L_{[h]})$. Minimizing the total actual flow time can be obtained by scheduling the jobs to non-increasing due dates in a backward scheduling approach.

$$d_{[1]} \ge d_{[2]} \ge \cdots \ge d_{[K]}$$

Proof. Let be considered a schedule of K jobs $(h = 1,2,3,..K)$ with the assumption that $C_{[h]} + s_{[h]} \le L_{[h]}$. Since $L_{[h]} = d_{[h]} - d_{[h+1]}$, so $d_{[h]} \ge d_{[h+1]}$. It keeps the positive value of length. Since we adopted a backward scheduling approach, where the first position job on a sequence is a job close to the due date, the job must be arranged by a non-increasing due date.

$d_{[1]} \ge d_{[2]} \ge \cdots \ge d_{[K]}$ ■ (proven)

Example 1. Suppose there are three jobs (J_1, J_2, and J_3) to be scheduled on a single machine with the backward scheduling approach. Parameters for among that jobs and the values for $C_{[h]} + s_{[h]}$ and $L_{[h]}$ can be seen in Table 1. It can be observed

in Table 1 that $C_{[h]} + s_{[h]}$ is shorter or equal to $L_{[h]}$. Base on Proposition 1, the job sequence must arrange according to the non-increasing due date (Longest due date, LDD). It shows that the arrangement is 1-2-3 ■

Table 1
Job's parameter for Example 1

Job (h=1,2,3)	$n_{[h]}$	$t_{[h]}$	$s_{[h]}$	$d_{[h]}$	$C_{[h]}$	$C_{[h]} + s_{[h]}$	$L_{[h]}$
J_1	3	1	1	13	3	4	5
J_2	2	2	1	8	4	5	5
J_3	1	1	1	3	1	2	3

Proposition 2. Suppose there are *K* jobs with their respective due date scheduled on a single machine using the backward scheduling approach. Let be assume that there are one or more completion time plus setup time of jobs scheduled is longer than the length ($C_{[h]} + s_{[h]} \leq L_{[h]}$). It leads that some of the jobs must be grouped. The group's result shows that the sum of their length must lower than the sum of their completion time and setup time. The new length can be created from the sum of the job's length in the group multiple by the proportion of the job's completion time. The new length is formulated as follows.

$$L'_{[h]} = \left(\sum_{h=1}^{G} L_{[h]}\right)\left\{C_{[h]} + s_{[h]} / \sum_{h=1}^{G}\left(C_{[h]} + s_{[h]}\right)\right\},\ h = 1,2,3,\ldots,G;$$

Proof. Let be considered a schedule of G jobs ($h = 1,2,3,..G$) with the assumption that $C_{[h]} + s_{[h]} > L_{[h]}$. Since the job could be scheduled if $C_{[h]} + s_{[h]} \leq L_{[h]}$ so we should combine (grouped) some job by counting some of their completion time job and length to meet the condition, that is $\sum_{h=1}^{G}\left(C_{[h]} + s_{[h]}\right) \leq \sum_{h=1}^{G} L_{[h]}$. Since we get the sum of both completion time plus setup time and the sum of length job in the group, we can calculate the proportion of length job, the so-called new length, notated by $L'_{[h]}$. The formula for calculating that as follows

$$L'_{[h]} = \left(\sum_{h=1}^{G} L_{[h]}\right)\left\{C_{[h]} + s_{[h]} / \sum_{h=1}^{G}\left(C_{[h]} + s_{[h]}\right)\right\},\ h = 1,2,3,\ldots,G;\ ■$$

Example 2. Suppose three jobs (J_1, J_2, and J_3) are scheduled on a single machine with the backward scheduling approach. Parameters for among that jobs can be seen in Table 2

Table 2
Job's parameter for Example 2

Job (h=1,2,3)	$n_{[h]}$	$t_{[h]}$	$s_{[h]}$	$d_{[h]}$	$C_{[h]} + s_{[h]}$	$L_{[h]}$	$\sum_{h=1}^{G}\left(C_{[h]} + s_{[h]}\right)$	$\sum_{h=1}^{G} L_{[h]}$	L'
Job 1	3	1	1	13	4	3	4	3	4.73
Job 2	2	2	1	10	5	1	9	7	5.91
Job 3	1	1	1	9	2	9	11	13	2.36

Proposition 3. Suppose there are *K* jobs scheduled on a single machine using the backward scheduling approach, with their respective due date. Let be assume that the completion time of each job is less than the new length ($C_{[h]} + s_{[h]} > L'_{[h]}$), so minimizing the actual flow time can be obtained by arranging jobs using the adjacent pairwise interchange (API) method. The formula for evaluated the adjacent pairwise interchange method as follows

$$\{\left(d_j - d_i\right) + s_i\}/n_i + t_i \ \leq \{\left(d_i - d_j\right) + s_j\}/n_j + t_j,$$

Proof. Let's define adjacent pair jobs, *i* and *j* from K-job. Let S and S' be two feasible schedules for the two jobs using the backward scheduling approach. The first Schedule (S) places *i* following by *j* and the second schedule (S') is vice versa. The actual flow time for S and S' can be calculated as follows

$$\text{Under S:}\quad TF^a(S) = \{d_i - (d_i - t_i n_i)\}n_i + \left\{d_j - \left(d_i - t_i n_i - s_i - t_j n_j\right)\right\}n_j$$

$$\text{Under S':}\quad TF^a(S') = \left\{d_j - \left(d_j - t_j n_j\right)\right\}n_j + \left\{d_i - \left(d_j - t_j n_j - s_j - t_i n_i\right)\right\}n_i$$

Solving the equation under S and S' found a formula for evaluated pair adjacent job, that is scheduled *i* and following by *j* (it mean job *i* is dominant than *j*) if only if $\{\left(d_j - d_i\right) + s_i\}/n_i + t_i \ \leq \{\left(d_i - d_j\right) + s_j\}/n_j + t_j$. Otherwise, it is vice versa. If there are K-job, there are {K(K-1)/2} number of pairwise jobs. The number of jobs was found by counting the number of dominant jobs and arranging that by a non-increasing number. ■ (proven)

Example 3. Consider the problem similar in Tabel 3. Let be applied Proposition 3 to evaluate the pair of adjacent jobs. The application of the adjacent pairwise interchange method shows in Table as follows.

Table 3
The result of the adjacent pairwise interchange (API) method

Job	1	2	3	The number of asterisks
1	-	0.33*	0*	2
2	4	-	2*	1
3	6	3	-	0

There are three resulting pairs: 1-2, 1-3, and 2-3. Each pair of jobs has two possible sequences: 1-2 and 2-1, 1-3 and 3-1, 2-3 and 3-2. For this case, we combine the value between each pair. For example, we compare 1-2 and 2-1, and we found the value of 1-2 less than 2-1, so we give asterisks to 1-2. The complete sequence is arranged by arranging the job by a non-increasing number of asterisks, that is 1-2-3. ■

5.2 the number of the batches, the batch size, and the sequence of the resulting batches

The next step is to calculate the number of the batches and also the batch sizes. We adopt the formula in Halim et al. (1994) and Yusriski et al. (2015a). Assume there are G jobs (h=1,2,3,…, G), the formulas for calculating both the number of batches and the batch sizes for every single job as shown in equation () and () respectively.

$$N_{[h]} = \left(-Z_{[h]}/s_{[h]}\right) + \left(\sqrt{Z_{[h]} + 2n_{[h]}s_{[h]}t_{[h]}}\right)/s_{[h]} \tag{8}$$

where

$$Z_{[h]} = n_{[h]}t_{[h]}/2 + \left\{n_{[h]}s_{[h]}/\left(L_{[h]}/t_{[h]} + s_{[h]}/t_{[h]} - n_{[h]}\right)\right\} - L_{[h]}/2 - s_{[h]}/2$$

$$N_{max} = \left\lfloor 1/2 + \sqrt{1/4 + 2n_{[h]}t_{[h]}s_{[h]}} \right\rfloor \tag{9}$$

where $\lfloor \quad \rfloor$ means the maximum integer less than or equal to the value

$$Q_{[h][i]} = \begin{cases} max\left\{\left\lceil \frac{n_{h[i]}}{i} + \frac{(i+1)s_h}{2t_h} - \frac{s_h i}{t_h} \right\rceil, 1\right\}, \\ \qquad i = 1,2,..,N-1 \\ n_{[h]} - \sum_{i=1}^{N-1} Q_{[h][i]}, \\ \qquad i = N \end{cases} \tag{10}$$

$$F^a = \left(d - B_{[h][i]}\right)Q_{[h][i]}, \tag{11}$$

where $B_{[h][i]} = \sum_{j=1}^{i}\left(Q_{[h][j]}t_{h[j]}\right) - (i-1)\,s_h, j \in i$

The multi-item problem could be seen as the repetition of a common due date single item problem. We can prove that non-increasing batch sizes obtain the optimal sequence of the resulting batches on the backward scheduling approach.

Proposition 4. Suppose N batches (i = 1, .., N) with a constant setup time between two consecutive batches to be scheduled on a single machine. Minimizing The total actual flow time found by arranging the N-batches to non-increase the batch size.

Proof. Assuming that there are N batches (i = 1, .., N) to be scheduled on a single-machine with constant setup time. Let S and S' be two feasible schedules for N batches arranged using a backward scheduling approach. The first Schedule (S) places the first batch at the first position and the second batch at the second position, and the second schedule (S') is vice versa. The sequence is the same for the other bathes under either S or S'. That batch begins and completes at the exact moment. Hence, the total actual flow time is the same for both S and S'. Let F as a sum of the actual flow time of the last (N-2) batches ($F = \sum_{i=3}^{N} F_i^a$), the relevance of the total actual flow time, both S and S' are the following.

Under S: $$F^a(S) = \left(tQ_{[1]}\right)Q_{[1]} + \left(tQ_{[1]} + s + tQ_{[2]}\right)Q_{[2]} + F$$

$$F^a(S) = \left(tQ_{[1]}^2\right) + \left(tQ_{[1]}Q_{[2]} + sQ_{[2]} + tQ_{[2]}^2\right) + F$$

Under S': $$F^a(S') = \left(tQ_{[2]}\right)Q_{[2]} + \left(tQ_{[2]} + s + tQ_{[1]}\right)Q_{[1]} + F$$

$$F^a(S') = \left(tQ_{[2]}^2\right) + \left(tQ_{[1]}Q_{[2]} + sQ_{[1]} + tQ_{[1]}^2\right) + F$$

Under S and S', we found that $F^a(S) - F^a(S') = \left(Q_{[2]} - Q_{[1]}\right)$

The total actual flow time of $F^a(S)$ is less than or equal to the schedule of $TF^a(S')$ if only if $TF^a(S) - TF^a(S') \leq 0$, or $\left(Q_{[2]} - Q_{[1]}\right) \leq 0$. It is showed by $Q_{[1]} \geq Q_{[2]}$. We must sequence the batches in a backward approach by non-increasing the batch sizes.

$$Q_{[1]} \geq Q_{[2]} \geq ... \geq Q_{[N-1]} \geq Q_{[N]}$$ ■ (proven)

Halim et al. (1994) has proposed the algorithm, the so-called common due date single machine (CSM) algorithm, to solve a common due date single machine batch scheduling problem to minimize total actual flow time. Yusriski et al. (2015a) have developed a similar on Halim et al. (1994) on integer batch case. We develop both of the two algorithms on Halim and Yusriski with an adjustment; the last batch setup could be scheduled on a length of schedule period. The adjustment CSM algorithm, namely ICSM-A-Algorithm, is shown as follows.

ICSM-A Algorithm

Step 1: Compute N_{max} using Eq (9). Continue to Step 2
Step 2: Compute N^0 using Eq (8).
If $N^0 < 1$, then set N = 1, continue to Step 3.
If $N^0 > N_{max}$, then N = N_{max}, continue to Step 3.
If $1 < N^0 < N_{max}$, set $N_u^0 = \lceil N^0 \rceil$, where $\lceil N^0 \rceil$means the minimum integer greater than or equal to N^0. If $N^0 s + nt < d$, then $N = N^0$, Otherwise $N = N^0 - 1$. Continue to Step 3.
Step 3: Use Eq (10) to calculate $Q_{[i]}$. Continue to Step 4.
Step 4: Base on Proposition 4, sequence the resulting batches in non-increasing $Q_{[i]}$.
Continue to Step 5.
Step 5: Compute F^ausing expression (11).■

5.3 The proposed algorithm

This section discusses the proposed algorithm for solving problems. There are three algorithms. The First is the P1-LDD algorithm; It is developed based on the Latest Due Date (LDD) approach. The second is the so-called P2-API algorithm; It is built based on a combination of LDD with the adjacent pairwise interchange (API) approach. The latest algorithm is the P3-PM algorithm, which is developed using the permutation (PM) method.

The P1-LDD Algorithm

Step 0: input parameters $K, h, n_{h,} t_{h,} s_{h,} d_h$. Continue to Step 1
Step 1: Based on Proposition 1, sequence the jobs using the Latest Due Date (LDD) rules. Set that as the initial sequence.
Continue to Step 2
Step 2: Assume that each job is processed in one batch. For each job, calculate $C_{[h]} + s_{[h]}$ dan $L_{[h]}$ using Eq (6) and (7).
Continue to Step 3
Step 3: Compare the values of $C_{[h]} + s_{[h]}$ and $L_{[h]}$.
If $C_{[h]} + s_{[h]} \leq L_{[h]}$ for all jobs set Set $d_{[h]} = L_{[h]}$, continue to Step 6,
Otherwise, go to Step 4
Step 4: Start from job position [1] to [K] on the initial sequence, Grouping jobs with the following conditions:
a. Set the independent job group, if only if $C_{[h]} + s_{[h]} \leq L_{[h]}$.
The number of the job of each independent group is just one job
b. Set the dependent group, if only if $C_{[h]} + s_{[h]} > L_{[h]}$.
The number of job in a group determine by Proposition 2
Continue to Step 5
Step 5: Apply Proposition 2 to calculate $L'_{[h]}$. Set $d_{[h]} = L'_{[h]}$, continue to Step 6.
Step 6: Start from job [1] to [K], calculate $N_{[h]}, Q_{[h][i]}$ $for\ i = 1,2,..,N$ and $F^a_{[h]}$ of each job using the ICSM-A Algorithm.
Step 7. Calculate the sum of total actual flow time $\left(TF^a = \sum_{h=1}^{K} F^a_{[h]}\right)$ ■

Example 4. Suppose three jobs (J_1, J_2, J_3, and J_4) are scheduled on a single machine with the backward scheduling approach. Determine the sequence of jobs, the number of baches each job, and the batches' sequence with the objective is to minimize the total actual flow time. Parameters for among that jobs can be seen in Table 4

Table 4
Job's parameters for Example 4

Job (h=1,2,3)	$n_{[h]}$	$t_{[h]}$	$s_{[h]}$	$d_{[h]}$
Job 1	5	2	1	58
Job 2	6	1	1	39
Job 3	5	2	1	30
Job 4	5	1	1	25

The problem was solved using The P1-LDD Algorithm. The result of that is shown as follows,

Step 1: The sequence is J_1-J_2-J_3-J_4.
Step 2 to Step 4: we found the jobs grouped into three groups.
The first group consists of { J_1},
The second group consist of {J_2},
The third group consists of two jobs, that is (J_3-J_4).
Step 5: The value of length J_1, J_2, J_3, and J_4 are 19, 9, 18.3, 11.7 consecutively.
Step 6: The result of both $N_{[h]}$ and $Q_{[h][i]}$
Job 1: N=5, $Q_{[1]} = 1, Q_{[2]} = 1, Q_{[3]} = 1, Q_{[4]} = 1, Q_{[5]} = 1$

Job 2: N=4, $Q_{[1]} = 2, Q_{[2]} = 2, Q_{[3]} = 1, Q_{[4]} = 1$
Job 3: N=5, $Q_{[1]} = 1, Q_{[2]} = 1, Q_{[3]} = 1, Q_{[4]} = 1, Q_{[5]} = 1$
Job 4: N=2, $Q_{[1]} = 3, Q_{[2]} = 2$
The actual flow time of J_1, J_2, J_3, and J_4 are 40, 30, 40, 73 consecutively.
Step 7: The total actual flow time is 183

The P2-API Algorithm

Step 0: input parameters $K, h, n_{h,} t_{h,} s_{h,} d_h$. Continue to Step 1
Step 1: Based on Proposition 1, sequence the jobs using the Latest Due Date (LDD) rules. Set that as the initial sequence. Continue to Step 2
Step 2: Assume that each job is processed in one batch. For each job, calculate $C_{[h]} + s_{[h]}$ dan $L_{[h]}$ using Eq (6) and (7). Continue to Step 3
Step 3: Compare the values of $C_{[h]} + s_{[h]}$ and $L_{[h]}$.
If $C_{[h]} + s_{[h]} \leq L_{[h]}$ for all jobs set Set $d_{[h]} = L_{[h]}$, continue to Step 6,
Otherwise, go to Step 4
Step 4: Start from job position [1] to [K] on the initial sequence, Grouping jobs with the following conditions:
a. Set the independent job group, if only if $C_{[h]} + s_{[h]} \leq L_{[h]}$.
The number of the job of each independent group is just one job
b. Set the dependent group, if only if $C_{[h]} + s_{[h]} > L_{[h]}$.
The number of job in a group determine by Proposition 2
Continue to Step 5
Step 5: Apply Proposition 2 to calculate $L'_{[h]}$. Set $d_{[h]} = L'_{[h]}$, continue to Step 6.
Step 6: Focus on all the dependent group's job. Base on Proposition 3, sequence the jobs on each group using the adjacent pairwise interchange (API) method. Use minimizing the total actual flow time as an objective to evaluate each pair. Continue to Step 7
Step 7: Determine the final sequence by combining all sequences in each group. Continue to Step 8
Step 8: Start from job [1] to [K], calculate $N_{[h]}, Q_{[h][i]}\ for\ i = 1,2,..,N$ and $F^a_{[h]}$ of each job using the ICSM-A Algorithm. Continue to Step 9
Step 9. Calculate the sum of total actual flow time $\left(TF^a = \sum_{h=1}^{K} F^a_{[h]}\right)$■

Example 5. Suppose the problem similar to Example 4.

The problem was solved using the P2-API Algorithm. The result of that is shown as follows.
Step 1: The sequence is J_1-J_2-J_3-J_4.
Step 2 to Step 4: we found the jobs grouped into three groups.
The first group consists of { J_1},
The second group consist of {J_2},
The third group consists of two jobs, that is (J_3-J_4).
Step 5: The value of length J_1, J_2, J_3, and J_4 are 19, 9, 18.3, 11.7 consecutively.
Step 6: Focus on jobs in Group 3 (J_3-J_4). The number of asterisks by pairwise interchange method shows in Table 5 as follows.

Table 5
The result of the adjacent pairwise interchange (API) method

Job	3	4	The number of asterisks
3	-	1.2*	1*
4	2.2	-	0

Since 1.2 less than 2.2 so the sequence is J_3-J_4
Step 7: the final sequence is J_1-J_2-J_3-J_4.
Step 8: The actual flow time of jobs are 40, 30, 40, 73.
Step 9: The total actual flow time is 183

The P3-PM Algorithm

Step 0: input parameters $K, h, n_{h,} t_{h,} s_{h,} d_h$. Continue to Step 1
Step 1: Generate all alternative job sequences using the permutation method. Save all alternatives into List alternative sequence. It could be K! alternate sequences. Continue to Step 2.
Step 2: Start from sequence [1] to [K] on the alternative list, calculate $N_{[h]}, Q_{[h][i]}\ for\ i = 1,2,..,N$ and $F^a_{[h]}$ of each job using the ICSM-A Algorithm. Continue to Step 3.
Step 3: If there are negative positions of beginning time on the sequence, remove that sequence from the alternative list. Continue to Step 4.

Step 4: Calculate the sum of $TF^{a}_{[h]}$ for each alternative jobs on the list.

Step 5: Set the alternative sequence with the smallest value of the sum of $TF^{a}_{[h]}$ as a solution ■

Example 6. Suppose the problem similar to Example 4.

The problem was solved using the P3-PM Algorithm. The result of that is shown as follows.

Step 1: There are 24 sequence alternatives: The alternative of the sequence as follows.

J_1-J_2-J_3-J_4, J_1-J_2-J_4-J_3, J_1-J_3-J_2-J_4, J_1-J_3-J_4-J_2, J_1-J_4-J_2-J_3, J_1-J_4-J_3-J_2,
J_2-J_1-J_3-J_4, J_2-J_1-J_4-J_3, J_2-J_3-J_1-J_4, J_2-J_3-J_4-J_1, J_2-J_4-J_1-J_3, J_2-J_4-J_3-J_1,
J_3-J_1-J_2-J_4, J_3-J_1-J_4-J_2, J_3-J_2-J_1-J_4, J_3-J_2-J_4-J_1, J_3-J_4-J_1-J_2, J_3-J_4-J_2-J_1,
J_4-J_1-J_2-J_3, J_4-J_1-J_3-J_2, J_4-J_2-J_1-J_3, J_4-J_2-J_3-J_1, J_4-J_3-J_1-J_2, J_4-J_3-J_2-J_1,

Step 2 until Step 5: The total actual flow time minimum is 183 founded by sequence J_1-J_2-J_3-J_4 with the decision of $N_{[h]}, Q_{[h]}$ as follows

Job 1: N=5, $Q_{[1]} = 1, Q_{[2]} = 1, Q_{[3]} = 1, Q_{[4]} = 1, Q_{[5]} = 1$
Job 2: N=4, $Q_{[1]} = 2, Q_{[2]} = 2, Q_{[3]} = 1, Q_{[4]} = 1$
Job 3: N=5, $Q_{[1]} = 1, Q_{[2]} = 1, Q_{[3]} = 1, Q_{[4]} = 1, Q_{[5]} = 1$
Job 4: N=2, $Q_{[1]} = 3, Q_{[2]} = 2$

Solving a similar case (Example 4) using the thee algorithm (the P1-LDD algorithm, the P2-API algorithm, and the P3-PM algorithm) has the same result. We can conclude that the thee proposed algorithm fit to solve the problem.

6. Numerical Experience

This section has been done to check the proposed algorithm's performance by numerical experience test. We coded among the proposed algorithm by Microsoft Visual C# 8 with .NET Framework 4.8 and reported the numerical experience test using the computer processor Intel ® i7 ® 860 @ 2,80 GHz (4 Cores - 8 Threads) @ 8 GB RAM. We show 25 problems as a sample and analysis that by the total actual flow time and the time unit computation. The parameter is generated by randomly using the Mersenne Twister algorithm. We set a random generator for the parameter as follows: the number of jobs between 8 to 10, the demand between 10 to 35, the processing time between 2 to 5, the set up time between 2 to 5, and the due dates between 700 to 1500. The result of the test is shown in Table 6.

Table 6
The result of the numerical experience test

Case-j	Total Actual Flowtime (TF^a) Histories			Time to compute Histories		
	The P1-LDD Algorithm (A)	The P2-API Algorithm (B)	The P3-PM Algorithm (C)	The P1-LDD	The P2-API	The P3-PM
				* Computing time is measured in milliseconds		
1	27062.11	27062.11	25356.56	27062.11	27062.11	25356.56
2	54225.82	53044.87	51922.12	54225.82	53044.87	51922.12
3	35407.34	33954.88	32809.82	35407.34	33954.88	32809.82
4	38894.89	38894.89	38054.63	38894.89	38894.89	38054.63
5	14892.27	14892.27	14573.52	14892.27	14892.27	14573.52
6	17239.42	17239.42	16558.00	17239.42	17239.42	16558.00
7	27496.75	27496.75	26401.26	27496.75	27496.75	26401.26
8	26888.93	26344.51	26344.51	26888.93	26344.51	26344.51
9	16406.06	16406.06	15509.89	16406.06	16406.06	15509.89
10	14944.64	14944.64	14646.17	14944.64	14944.64	14646.17
11	14755.23	14755.23	14755.23	14755.23	14755.23	14755.23
12	49272.79	47403.10	42774.13	49272.79	47403.10	42774.13
13	13289.65	13289.65	13235.28	13289.65	13289.65	13235.28
14	15763.35	15763.35	14145.13	15763.35	15763.35	14145.13
15	36427.00	36427.00	34511.35	36427.00	36427.00	34511.35
16	19316.84	19316.84	18415.24	19316.84	19316.84	18415.24
17	42897.54	41675.21	35176.22	42897.54	41675.21	35176.22
18	34053.79	34053.79	29576.70	34053.79	34053.79	29576.70
19	19276.70	19276.70	18874.10	19276.70	19276.70	18874.10
20	37547.67	37547.67	33441.23	37547.67	37547.67	33441.23
21	47027.56	47027.56	46232.64	47027.56	47027.56	46232.64
22	26809.36	26212.82	26212.82	26809.36	26212.82	26212.82
23	48814.77	47391.13	41975.68	48814.77	47391.13	41975.68
24	24367.18	23788.48	23529.47	24367.18	23788.48	23529.47
25	35784.86	35784.86	35272.93	35784.86	35784.86	35272.93
Σ	738862.51	729993.78	690304.63	738862.51	729993.78	690304.63
$\bar{x}$	29554.50	29199.75	27612.19	199158.68	199680.88	337587.10
s	12636.98	12275.85	11290.03	564332.64	564262.13	1062738.29

For analyzing the data in Table 6, we are using the Bonferroni approach to compare the three algorithms' performance: the total actual flow time. It could be three pairwise comparisons for three methods: The P1-LDD Algorithm versus The P2-API Algorithm (A VS B), The P1-LDD Algorithm versus The P3-PM Algorithm (A VS C), and The P2-API Algorithm versus The P3-PM Algorithm (B VS C). We use α=6%, so $\alpha_i = 2\%$ (i=1,2,3). The hypothesis is shown as follows.

H_0 : $\mu_1 = \mu_2 = \mu_3 = \mu$

H_1 : $\mu_1 \neq \mu_2 \neq \mu_3$ (minimum there is one deferent pair of the algorithm)

The P1-LDD Algorithm VS The P2-API Algorithm

($\alpha_i = 2\%, , n = 25, s_{A-B} = 590.26, \bar{x}_{A-B} = 354.75$)

Calculate $t_{n-1,\alpha_i/2} = t_{24,001}$= 1.316 (From T-student)

Calculate $hw = \left(t_{24,001} s_{(A-B)} \right)/\sqrt{25}$ = 155.35

Calculare $\bar{x}$ (A-B) - $hw \leq \mu_{(A-B)} \leq \bar{x}$ (A-B) + hw, it found $199.39 \leq \mu_{(A-B)} \leq 510.11$

Since minimizing case, if The P1-LDD Algorithm VS The P2-API Algorithm, chose The P2-API Algorithm

The P1-LDD Algorithm VS The P3-PM Algorithm

($\alpha_i = 2\%, , n = 25, s_{A-C}$ =2227.28, $\bar{x}_{A-C}$ =1942.32)

Calculate $t_{n-1,\alpha_i/2} = t_{24,001}$= 1.316 (From T-student)

Calculate $hw = \left(t_{24,001} s_{(A-C)} \right)/\sqrt{25}$ = 586.22

Calculare $\bar{x}$ (A-C) - $hw \leq \mu_{(A-C)} \leq \bar{x}$ (A-C) + hw, it found $1356.10 \leq \mu_{(A-B)} \leq 2528.53$

Since minimizing case, if The P1-LDD Algorithm VS The P3-PM Algorithm, chose The P3-PM Algorithm

The P2-API Algorithm VS The P3-PM Algorithm

($\alpha_i = 2\%, , n = 25, s_{B-C}$ =1870.40, $\bar{x}_{B-C} = 1587.57$)

Calculate $t_{n-1,\alpha_i/2} = t_{24,001}$= 1.316 (From T-student)

Calculate $hw = \left(t_{24,001} s_{(B-C)} \right)/\sqrt{25}$ = 492.29

Calculare $\bar{x}$ (B-C) - $hw \leq \mu_{(A-C)} \leq \bar{x}$ (B-C) + hw, it found $1095.28 \leq \mu_{(A-B)} \leq 2079.86$

Since minimizing case, if The P2-API Algorithm VS The P3-PM Algorithm, chose The P3-PM Algorithm

Based on the Bonferroni comparison method, we can observe that The P3-PM Algorithm is better than another, followed by The P2-API Algorithm and The P1-LDD Algorithm consecutively. However, the comparison of time to compute shows vise versa. The P1-LDD Algorithm is shortest than another, followed by The P2-API Algorithm and The P3-PM Algorithm.

We have been done to check more than 1000 cases. We found that the P1-LDD Algorithm has the best performance when solving a simple problem, including more groups with one job in every group. The P3-PM Algorithm has the best performance when solving small complicated problems (the problem has more groups with fewer jobs in a group with fewer than ten jobs). However, The P2-API Algorithm has the best performance when solving a complicated problem, including more groups and more jobs in every group.

7. Concluding Remarks

This research deals with a batch scheduling of multi-job with multiple delivery dates on a single machine where every job consists of a number of parts. The objective is to minimize the total actual flow time. A problem addressing the combination of multi-job and multi-due dates is rather complicated and known as NP-Hard. We proposed three algorithms to solve the problem based on the longest due date rule (The P1-LDD Algorithm), the adjacent pairwise interchange method (The P2-API Algorithm), and the permutation method (The P3-PM Algorithm). The numerical experience test found insight: The P1-LDD Algorithm fit to solve a simple problem, including more groups with one job in every group; The P3-PM Algorithm has the best performance when solving small complicated problems (the problem has more groups with fewer jobs in a group less than ten jobs); The P2-API Algorithm has outstanding performance when solving a complicated problem, including more groups and more jobs in every group.

The problem in this research assumes that jobs are scheduled on a single machine. There are cases in a real system that the jobs should be scheduled on a serial or parallel machine. The model can be extended to discuss scheduling jobs on flow shop, job shop, or parallel machine.

Acknowledgments

The authors wish to thank the editors and the referees for their constructive comments and suggestions. This research was supported by the University of Jenderal Achmad Yani (UNJANI) Bandung-Cimahi.

References

Albers, S., & Brucker, P. (1993). The complexity of one-machine batching problems. *Discrete Applied Mathematics*, *47*, 87-107.

Baptiste, P. (2000). Batching Identical Jobs. *Mathematical Methods of Operational Research, 52*(3), 355-367.

Basir, S. A., & Karimian, Y. (2018). A Green Mathematical Model for a Single-Machine Scheduling Problem with Batch Delivery System. *The-12th International NCM Conference: Challenges in Industrial Engineering & Operation Management*. 34-37.

Halim, A.H., Miyazaki, S., & Ohta, H. (1994). Batch scheduling problem to minimize actual flow times of component through the shop under JIT environment. *European Journal of Operational Research*, *72,* 529-544.

Hamidinia, A., Khakabimamaghani, S., Mazdeh, M. M., & Jafari, M. (2012). A genetic algorithm for minimizing total tardiness/earliness of weighted jobs. *Computers & Industrial Engineering*. *65*, 29-38.

Hazir, Ö., & Sidhoum, S. K. (2014). Batch sizing and just-in-time scheduling with common due date. *Annals of Operations Research, 213*, 187–202.

Maulidya, R., Suprayogi., Wangsaputra, R., & Halim, A. H. (2019). Batch Scheduling for Multi Due Date Heterogeneous Machines with Reentrant Flow to Minimize Total Tardiness. *IOP Conf. Series: Materials Science and Engineering*, *528* (1), 1-8.

Mazdeh, M. M., Hamidini, A., & Karamouzian, A. (2011). Mathematical model for weighted tardy jobs scheduling problem with a batched delivery system. *International Journal of Industrial Engineering Computations*, *2*, 491–498.

Min 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*, 745–755.

Mohri, S., Masuda, T., & HiroakiIshii. (2011). Batch Scheduling Problem with Multiple Due-dates Constraints*. Industrial Engineering & Management Systems*, *10*(1), 1-6.

Nurainun, T., Fudholi, A., Hartati, M., Yendra, R., & Kusumanto, I. (2016) A multi due date batch scheduling model on dynamic flow shop to minimize total production cost. *Contemporary Engineering Sciences*, *9*(7), 315 – 324.

Prasetyaningsih, E., Suprayogi, S., Samadhi, T. M. A. A., & Halim, A. H. (2017). Production and Delivery Batch Scheduling with Multiple Due Dates to Minimize Total Cost. *Journal of Engineering and Technological Sciences*, *49*(1), 16-36.

Rasti-Barzoki, M., Hejazi, S.R., & Mazdeh, M.M.(2013). A branch and bound algorithm to minimize the total weighed number of tardy jobs and delivery costs. *Applied Mathematical Modelling*, *37*(7), 4924-4937.

Xie, X., & Wang X. (2014). Single-machine batch delivery scheduling and common due date assignment with identical processing times. *Applied Mechanics and Materials*, 635-637, 1884-1889.

Yin, Y., Cheng, T.C.E., Cheng, S-R., & Wu, C-C (2013) Single-machine batch delivery scheduling with an assignable common due date and controllable processing times. *Computers & Industrial Engineering, 65*, 652–662.

Yusriski, R., Astuti, B., & Ilham, M. (2019). Integrated Batch Production and Multiple Preventive Maintenance Scheduling on A Single Machine to Minimize Total Actual Flow Time. *IOP Conf. Series: Materials Science and Engineering*, *598* (1), 1-8.

Yusriski, R., Astuti, B., Samadhi, T. M. A. A., & Halim, A. H. (2015a). Integer batch scheduling problems for a single-machine to minimize total actual flow time. *Procedia Manufacturing*, *2*, 118-123.

Yusriski, R., Sukoyo, S., Samadhi, T. M. A. A., & Halim, A. H. (2015b). Integer batch scheduling problems for a single-machine with simultaneous effect of learning and forgetting to minimize total actual flow time. *International Journal of Industrial Engineering Computations, 6*(3), 365-378.

Yusriski, R., Sukoyo., Samadhi, T. M. A. A., & Halim, A. H. (2016). An integer batch scheduling model for a single machine with simultaneous learning and deterioration effects to minimize total actual flow time. *IOP Conf. Series: Materials Science and Engineering, 114*(1), 1-10.

Yusriski, R., Sukoyo., Samadhi, T. M. A. A., & Halim, A. H. (2018). An integer batch scheduling model considering learning, forgetting, and deterioration effects for a single machine to minimize total inventory holding cost. *IOP Conf. Series: Materials Science and Engineering, 319*(1), 1-7.