International Journal of Industrial Engineering, 30(2), 350-372, 2023

A DECOMPOSITION-BASED HEURISTIC ALGORITHM FOR PARALLEL BATCH PROCESSING PROBLEM WITH TIME WINDOW CONSTRAINT

Anh H. G. Nguyen[1] and Gwo-Ji Sheen[2, *]

[1]School of Industrial Engineering Management
International University
Ho Chi Minh, Vietnam

[2]Institute of Industrial Management
National Central University
Taoyuan, Taiwan
*Corresponding author's e-mail: gjsheen@mgt.ncu.edu.tw

This study considers a parallel batch processing problem to minimize the makespan under constraints of arbitrary lot sizes, start time window and incompatible families. We first formulate the problem with a mixed-integer programming model. Due to the NP-hardness of the problem, we develop a decomposition-based heuristic to obtain a near-optimal solution for large-scale problems when computational time is a concern. A two-dimensional saving function is introduced to quantify the value of time and capacity space wasted. Computational experiments show that the proposed heuristic performs well and can deal with large-scale problems efficiently within a reasonable computational time. For the small-size problems, the percentage of achieving optimal solutions by the DH is 94.17%, which indicates that the proposed heuristic is very good in solving small-size problems. For large-scale problems, our proposed heuristic outperforms an existing heuristic from the literature in terms of solution quality.

Keywords: Scheduling; Parallel Batch Processing Problem; Time Window Constraint; Decomposition Approach; Saving Method.

(Received on December 6, 2022; Accepted on February 20, 2023)

1. INTRODUCTION

We consider an identical parallel batch processing machine (BPM) scheduling problem when minimizing the makespan under various constraints. The BPM scheduling problems with incompatible families were addressed by several researchers, such as Uzsoy (1995); Koh *et al.* (2004); Bilyk *et al.* (2014); and Jia *et al.* (2016). However, time window constraints were not considered in the aforementioned studies. Our study considers time window constraints which have been studied for several problems such as cross-docking problem (Li *et al.*, 2004), parallel machine scheduling problem (Bard and Rojanasoonthon, 2006; Brucker and Kravchenko, 2008; Lee *et al.*, 2018), realistic cyclic scheduling problem (Shirvani *et al.*, 2014), traveling salesperson problem (Hungerländer and Truden, 2018), and vehicle routing problem (Hashemi *et al.*, 2020). It is shown that time window constraints are essential in the production environment but earn less attention in the context of parallel BPMs. In recent years, researchers have shown significant interest in parallel BPMs. Due to the NP-hardness of the parallel BPMs, most researchers solved their parallel BPMs by heuristic approaches. Note that we use the $\alpha \mid \beta \mid \gamma$ notation suggested by (Graham *et al.*, 1979) to describe each scheduling problem. The α field describes the machine environment, the β field provides different process restrictions, and the γ field presents the performance measures. We discuss scheduling problems with respect to these three dimensions in the remaining sections. Uzsoy (1995) presented heuristics based on the longest processing time first and earliest due date first rules for the problems $Pm|batch|C_{max}$ and $Pm|batch|L_{max}$. Koh *et al.* (2004) proposed several rule-based heuristics and designed a random key-based genetic algorithm (GA) for the problems $Pm|batch, incompatible, s_i|C_{max}(\sum w_i C_i)$. Jia *et al.* (2016) also solved the problem $Pm|batch, incompatible, s_i|C_{max}$ by developing a metaheuristic based on max-min ant system. Chang *et al.* (2004) applied a simulated annealing algorithm to address the problem $Pm|batch, s_i|C_{max}$ which was also solved by a hybrid genetic heuristic in Kashan *et al.* (2008). Balasubramanian *et al.* (2004) solved the problem $Pm|batch, incompatible|\sum w_i T_i$ by developing different decomposition approaches, which combined several dispatching rules with a proposed GA. Similar to Balasubramanian *et al.* (2004), Mönch *et al.* (2005) proposed two decomposition approaches to deal with the problem $Pm|batch, incompatible|\sum w_i T_i$. Chiang *et*

DOI: 10.23055/ijietap.2023.30.2.8769 ISSN 1943-670X

al. (2010) addressed the problem $Pm|batch, incompatible, r_i|\sum w_i T_i$ by a memetic algorithm. In this memetic algorithm, they proposed to encode batch formation and batch sequence simultaneously in the proposed chromosome, while machine assignment was done during the decoding. Malve and Uzsoy (2007) combined iterative improvement heuristics with a GA using the random key representation to solve the problem $Pm|batch, incompatible, r_i|L_{max}$. Chung *et al*. (2009) proposed a mixed integer programming (MIP) model, a MIP-based algorithm, and three constructive heuristics to address the problem $Pm|batch, compatible, s_i, r_i|C_{max}$. Ozturk *et al*. (2014) presented a branch and bound-based heuristic for solving the problem $Pm|batch, s_i, r_i, p_i{=}p|C_{max}$. Zhou *et al*. (2018) developed a GA based on the random keys representation to address the problem $Rm|batch, s_i, r_i|C_{max}$. Besides, The aforementioned studies solved different parallel BPMs by either mathematical models or near-optimal heuristics. The heuristic-based problem-solving approaches include genetic algorithm, ant colony optimization, simulated annealing, tabu search, variable neighborhood search or decomposition approach.

According to Mathirajan and Sivakumar (2006), the decomposition approach, which has received considerable attention, is one of the typical approaches for solving BPMs. The decomposition approach has also been applied successfully to solve the BPMs with incompatible families by other researchers. Reichelt and Mönch (2006) proposed the three-phase approach, including batch formation, batch assignment and batch sequencing, to solve the problem $\boldsymbol{Pm|batch, incompatible, r_i|\sum w_i T_i, C_{max}}$. Chung *et al*. (2009) developed two heuristics consisting of two phases, batch formation and batch scheduling, for solving the problem $\boldsymbol{Pm|batch, compatible, s_i, r_i|C_{max}}$. Almeder and Mönch (2011) proposed to solve the problem $\boldsymbol{Pm|batch, incompatible|\sum w_i T_i}$ by metaheuristics which are hybridized with a decomposition heuristic and a local search. Cheng *et al*. (2014) proposed a polynomial time heuristic which is based on a two-phase decomposition approach for the problem $\boldsymbol{Pm|batch, s_i|C_{max}}$. Arroyo and Leung (2017a) developed two-phase decomposition heuristics to address the problem $\boldsymbol{Rm|batch, s_i, r_i|C_{max}}$. It shows that the decomposition approach divides a parallel BPM into several sub-problems and solves sub-problem by sub-problem to obtain a solution for the original problem. Based on its advantages, in this study, we also propose a decomposition-based heuristic to obtain a near-optimal solution for our large-scale problem.

The above literature indicates that the parallel BPMs with incompatible families have been one of an interesting topic for many researchers. This paper contributes to the literature for parallel BPMs with incompatible families by further considering the practical start time window constraints. To solve the studied problem, a MIP model is first proposed to obtain optimal solutions. We then develop an efficient decomposition-based heuristic, which includes two phases - batch formation and batch scheduling, to deal with the large-scale problem. Extended from the idea of the saving method of Clarke and Wright (1964), a new two-dimensional saving function is introduced to quantify the saving space of time and capacity, which is a basis for the batch formation in our proposed heuristic while two priority rules are proposed to address the batch scheduling phase. The paper is structured as follows. Our problem description and the MIP formulation are given in Section 2. A decomposition-based heuristic is developed in Section 3. In Section 4, the computational result for randomly generated instances is reported. Concluding remarks are given in Section 5.

2. PROBLEM DESCRIPTION AND MIP MODEL FORMULATION

2.1. Problem description and assumptions

Our studied problem is motivated by the wafer fabrication procedure in semiconductor manufacturing. In wafer fabrication, multiple diffusion work centers provide similar processing capabilities, and each diffusion work center consists of multiple identical machines. Each diffusion machine can process several lots at the same time, which means that the diffusion machines in wafer fabs are an example of batch-processing machines. Each lot contains a fixed number of wafers and is classified into a specific product family/recipe according to its processing temperature, steps and chemical characteristics required for the diffusion process. The diffusion processes are long and allow batching of lots with the same family/recipe. The batching process is allowed only of lots of the same recipe, and a batch has a capacity limit that is recipe-dependent. According to Mönch *et al*. (2012), in wafer fabrication, time constraints between consecutive process steps are important restrictions. For instance, there is often a time restriction between operations in the etch work area and oxidation/diffusion work area. The time windows are installed to prevent native oxidation and contamination effects on the wafer surface. To derive time constraints to the scheduling problem, lots are recommended their own time windows to be processed. Lots that cannot be processed during the recommended time windows will be eliminated or scrapped when rework is generally not allowed for the scrapped lots. In this study, diffusion machines are assumed to model as parallel batch processing machines with incompatible job families and time window constraints.

Moreover, according to Mönch *et al*. (2012), there are several performance measures for the entire wafer fabs, but the most important among them are cycle time, throughput, and on-time delivery performance measures. Increasing throughput or bottleneck utilization leads to smaller cost per wafer, reducing cycle time results in lower financial holding costs and enhancing on-time delivery performance increases customer satisfaction. In our study, the throughput measure is derived for the studied scheduling problem by considering the objective to minimize makespan. As defined in Pinedo (1995), makespan is equivalent to the completion time of the last job to leave the system. Thus minimizing makespan results in a higher throughput value and a lower wafer production cost which is one of the important targets of the fabrication process. According to Graham *et al*. (1979), our strongly NP-hard problem can be expressed as $Pm|batch, incompatible, s_i, time\ window|C_{max}$.In addition, the following assumptions are considered for the problem formulation as follows:

- There are M parallel machines to batch and process N lots. All the data, including lot processing times p_i, release time r_i, remaining lifetime R_i and lot size s_i are deterministic and are known in advance.
- All the batch-processing machines are identical in nature.
- The machines are available at the beginning of the scheduling.
- Each machine can only process one batch at a time.
- Preemption and machine breakdown are not allowed.
- Each lot i must be processed within its start time window $[r_i, r_i + R_i]$; otherwise, the lot will be scrapped.
- Suppose that lot i is in batch B_b ($b = 1, \dots, B$), the batch B_b has its start time window [ES_b^B, LS_b^B] with ES_b^B= $max\ \{r_i | i \in B_b\}$ and $LS_b^B = min\{(r_i + R_i) | i \in B_b\}$.
- All lots with the same recipe have the same processing time.
- A batch can only consist of lots with the same recipe.
- The size of each lot cannot exceed the capacity of any batch.

2.2. MIP model formulation

In this section, our problem is formulated by a MIP model. The notations used are presented in **Appendix A**.

$$\text{Minimize} \quad C_{max} \tag{1}$$

Subject to

$$\sum_{j=1}^{M}\sum_{b=1}^{B} X_{j,b,i} = 1 \qquad \forall i \tag{2}$$
$$\boldsymbol{L}Y_{j,b,e} \geq \sum_{i=1}^{N} h_{i,e}X_{j,b,i} \qquad \forall j,b,e \tag{3}$$
$$\sum_{i=1}^{N} h_{i,e}X_{j,b,i} \geq Y_{j,b,e} \qquad \forall j,b,e \tag{4}$$
$$\sum_{i\in J_e} s_iX_{j,b,i} \leq UB_eY_{j,b,e} \qquad \forall j,b,e \tag{5}$$
$$\sum_{e=1}^{E} Y_{j,b,e} \leq 1 \qquad \forall j,b \tag{6}$$
$$S_{j,b} \geq r_iX_{j,b,i} \qquad \forall j,b,i \tag{7}$$
$$S_{j,b} \leq r_i + R_i + \boldsymbol{L}\left(1 - X_{j,b,i}\right) \qquad \forall j,b,i \tag{8}$$
$$F_{j,b} - S_{j,b} \geq p_iX_{j,b,i} \qquad \forall j,b,i \tag{9}$$
$$F_{j,b} \leq S_{j,b+1} \qquad \forall j, 1 \leq b \leq B-1 \tag{10}$$
$$S_{j,b+1} \leq F_{j,b} + \boldsymbol{L}\sum_{i=1}^{N} X_{j,b+1,i} \qquad \forall j, 1 \leq b \leq B-1 \tag{11}$$
$$F_{j,b} \leq C_{max} \qquad \forall j,b \tag{12}$$

Objective (1) is to minimize the makespan. Constraint (2) imposes that each lot can be assigned to only one batch. Constraints (3) and (4) ensure that recipe *e* is processed by batch *b* on machine *j* when lot *i* using recipe *e* is assigned to batch *b* on machine *j*. Constraint (5) guarantees that total lot sizes in a batch cannot exceed the batch capacity. Constraint (6) ensures that each batch can have at most one recipe. Constraints (7) and (8) ensure that if lot *i* is assigned to batch *b* on machine j, then batch b's start time window must satisfy lot i's start time window. Constraint (9) indicates that if lot *i* is assigned to batch b on machine j, then lot i's processing time is within the range of batch b's start and finish time. Constraints (10) and (11) ensure that under the same machine, batch b's finish time cannot be greater than batch (*b*+1)'s start time. Constraint (12) restricts that the objective makespan cannot be less than any batch's finish time on machine j.

3. DECOMPOSITION-BASED HEURISTIC ALGORITHM

Here, our studied problem is decomposed into two sub-problems, which are solved separately. A decomposition-based heuristic (DH) algorithm with two phases is proposed, and each phase addresses one corresponding sub-problem. Phase I is for batch formation, in which lots are grouped into several batches, and phase II is for batch scheduling, in which the formed batches are scheduled on the parallel batch processing machines. In phase I, lots are grouped into batches in consideration of the lot size, the incompatible families and the start time windows. We propose to form batches by an approach based on the saving value. Clarke and Wright (1964) proposed a one-dimensional saving function to calculate the saved distance from each pair of demand points. Then, starting from the largest saving value, the shipments are consolidated for a vehicle when the consolidation does not violate the constraints such as vehicle capacity and time window. A similar one-dimensional saving function can be found in several researchers, such as Çatay (2010); Pamosoaji *et al*. (2019); Segerstedt (2014); and Tarhini *et al*. (2020). However, because our study needs to address not only the capacity but also the time, a new two-dimensional saving function is proposed to calculate the saving value between any two batches in our batch formation phase. The concept of two-dimensional function was also discussed in some studies, such as Chen *et al*. (2011), Jia and Leung (2014), Zhou *et al*. (2014), and Jia *et al*. (2015). Before presenting our saving function, we need the following definitions. The notations used in this section are presented in **Appendix B**.

Definition 1: A batch with unit size and unit processing time is defined as a 'unit batch'. A batch B_b with processing time p_b^B and size s_b^B is composed of $p_b^B s_b^B$ unit batches. Unit batch is a base for measuring time and capacity area wasted by a batch.

Definition 2: Time and capacity wasted area for a batch is caused by the joint effect of the batch residual capacity and the lot delay time in the batch processing. For any two batches B_k and B_l $(k, l = 1, \ldots, N; k < l)$, there are two cases for calculating time and capacity wasted area:

Case 1: When batches B_k and B_l are processed separately, time and capacity wasted area for each batch is only affected by the batch residual capacity (see Figures 1-a, b). Time and capacity wasted area $TC'_{k,l}$ is the sum of time and capacity wasted areas for batches B_k and B_l as follows:

$$\begin{aligned} TC'_{k,l} &= [(\text{Capacity of batch } B_k - \text{total size of lots in batch } B_k) \times \text{Processing time of batch } B_k] + \\ &\quad [(\text{Capacity of batch } B_l - \text{total size of lots in batch } B_l) \times \text{Processing time of batch } B_l] \\ &= \left[\left(UB_{re_k^B} - s_k^B\right) p_k^B\right] + \left[\left(UB_{re_l^B} - s_l^B\right) p_l^B\right]. \end{aligned} \tag{13}$$

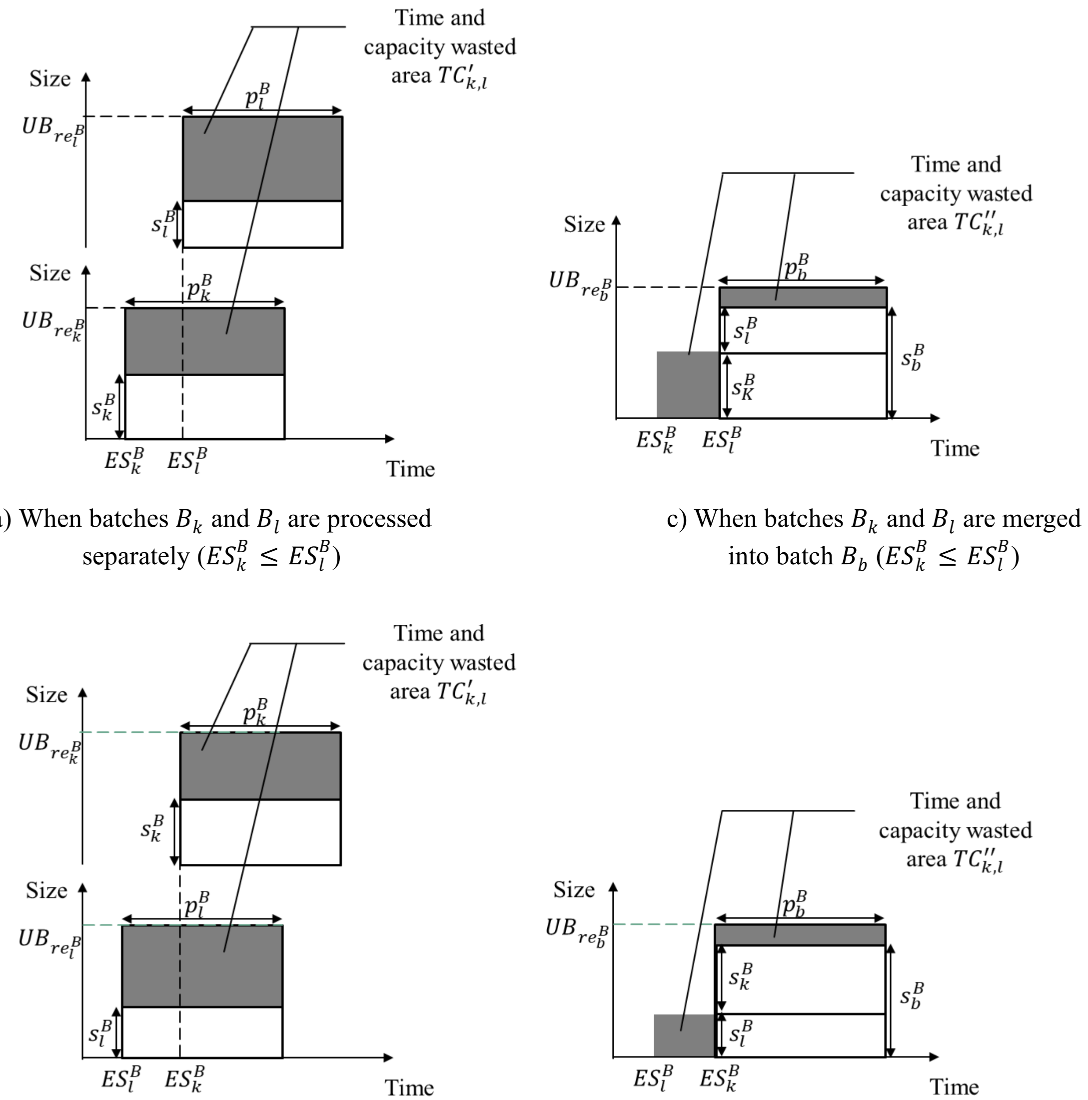
Case 2: When batches B_k and B_l are merged into batch B_b, there are two different situations needed to be considered (see **Figures 1-c, d**). Each situation results in a different way to calculate time and capacity wasted area $TC''_{k,l}$. They are stated as follows:

$$\begin{aligned} TC''_{k,l} &= [(\text{Capacity of batch } B_b - \text{total size of lots in batch } B_b) \times \text{Processing time of batch } B_b] + \\ &\quad [\text{Difference between the earliest start time of two merged batches} \times \text{Size of the batch} \\ &\quad \text{with smaller earliest start time}] \\ &= \begin{cases} \left(UB_{re_b^B} - (s_k^B + s_l^B)\right) p_b^B + (ES_l^B - ES_k^B) s_k^B & \text{if } ES_k^B \le ES_l^B \\ \left(UB_{re_b^B} - (s_k^B + s_l^B)\right) p_b^B + (ES_k^B - ES_l^B) s_l^B & \text{if } ES_k^B > ES_l^B \end{cases}. \end{aligned} \tag{14}$$

Definition 3: Saving space value $S_{k,l}$ is the value of space saved regarding time and capacity wasted area after merging batches B_k and B_l $(k, l = 1, \ldots, N; k < l)$ together; namely, a saving space value is the difference between $TC'_{k,l}$ and $TC''_{k,l}$. There is no saving space value for any two batches which cannot be merged together. The two-dimensional saving space value $S_{k,l}$ is calculated as follows:

$$\begin{aligned} S_{k,l} &= \begin{array}{c}\text{Time and capacity wasted area when batches } B_k \\ \text{and } B_l \text{ are processed separately}\end{array} - \begin{array}{c}\text{Time and capacity wasted area when batches } B_k \\ \text{and } B_l \text{ are merged into batch } B_b\end{array} \\ &= TC'_{k,l} - TC''_{k,l} \end{aligned}$$

$$= \begin{cases} \left[\left(UB_{re_k^B} - s_k^B\right) p_k^B\right] + \left[\left(UB_{re_l^B} - s_l^B\right) p_l^B\right] - \left[\left(UB_{re_b^B} - (s_k^B + s_l^B)\right) p_b^B + (ES_l^B - ES_k^B) s_k^B\right] & \text{if } ES_k^B \le ES_l^B, re_k^B = re_l^B; \\ \left[\left(UB_{re_k^B} - s_k^B\right) p_k^B\right] + \left[\left(UB_{re_l^B} - s_l^B\right) p_l^B\right] - \left[\left(UB_{re_b^B} - (s_k^B + s_l^B)\right) p_b^B + (ES_k^B - ES_l^B) s_l^B\right] & \text{if } ES_k^B > ES_l^B, re_k^B = re_l^B; \\ 0 & \text{, otherwise.} \end{cases} \tag{15}$$



a) When batches B_k and B_l are processed separately ($ES_k^B \leq ES_l^B$)

c) When batches B_k and B_l are merged into batch B_b ($ES_k^B \leq ES_l^B$)

b) When batches B_k and B_l are processed separately ($ES_k^B > ES_l^B$)

d) When batches B_k and B_l are merged into batch B_b ($ES_k^B > ES_l^B$)

Figure 1. Illustration of time and capacity wasted area

Phase I, the batch formation, is summarized as follows. Each lot initially forms its own batch. We then compute the saving space value for each pair of batches based on the saving space value function Equation (15). Two batches, starting with the largest positive saving value, are merged in consideration of constraints such as lot start time windows and batch capacity. After a new batch is formed, the saving value for each pair of batches is then re-calculated. The batching process continues until no positive saving values exist.

Definition 4: Two priority rules are proposed for Phase II of the DH algorithm. Here, ES (resp. LS) stands for the earliest start time (resp. the latest start time).

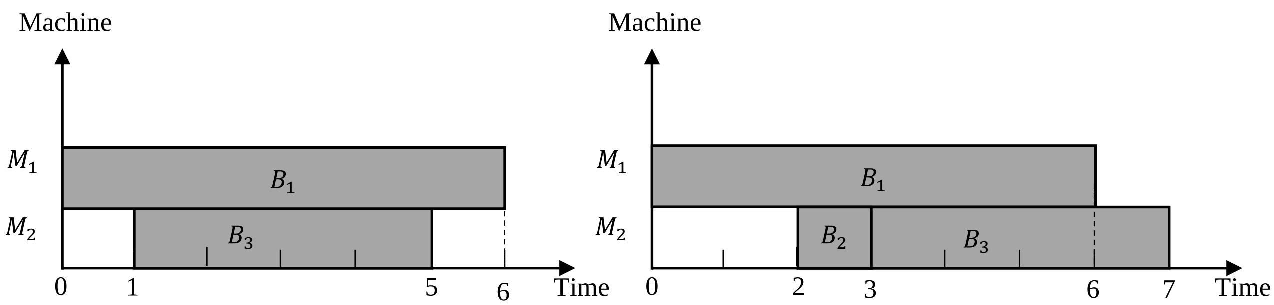
1. EST (earliest start time) rule: When a machine is freed, the batch with the smallest ES among those not yet processed is put on the machine.
2. LST (latest start time) rule: When a machine is freed, the batch with the smallest LS among those not yet processed is put on the machine.

Phase II, the batch scheduling, consists of two procedures. In the first procedure, the formed batches from phase I are assigned to machines according to the EST rule. This procedure gives non-delay schedules, where a machine is never left idle when a batch is available for processing. The EST rule is applied to assign batches to machines by several literatures, such as Chung *et al*. (2009), Damodaran and Vélez-Gallego (2012), and Arroyo and Leung (2017a,b). These papers have in common assigned batches to machines by using the batch earliest start times. As discussed above, our study further considers the batch start time window constraint when assigning batches to machines. Thus, we introduce a feasibility condition for a solution found by the first procedure to ensure that all the batch start time windows are satisfied. The feasibility condition, namely **Condition 1,** is shown in detail as follows.

For instance, $\boldsymbol{I}$ of our problem, let sequence $\mathcal{L}^{EST}$ be a sorted sequence according to the non-decreasing order of batch earliest start time for un-assigned batches. Let batch b_1^{EST} be the first batch in sequence $\mathcal{L}^{EST}$ and $LS^B_{b_1^{EST}}$ be the latest start time of batch b_1^{EST}. Let $T_{j^*}^{EST}$ be the earliest available time among machines such that $T_{j^*}^{EST} = \min_{j=1,\dots,M}\{T_j^{EST}\}$ when assigning batch b_1^{EST}.

Condition 1: *If the inequality* $LS^B_{b_1^{EST}} \geq T_{j^*}^{EST}$ *holds for every batch in sequence* $\mathcal{L}^{EST}$*, then there exists a feasible solution for instance* $\boldsymbol{I}$.

However, the first procedure, which uses only the batch earliest start times for making decisions, may not find a feasible solution but actually there exists one. Consider the 3-batch, 2-machine example with batch information: $ES_1^B = 0$, $ES_2^B = 2$, $ES_3^B = 1$, $LS_1^B = 2$, $LS_2^B = 3$, $LS_3^B = 4$, $p_1^B = 6$, $p_2^B = 1$ and $p_3^B = 4$. According to the EST rule, we have $\mathcal{L}^{EST} = (B_1, B_3, B_2)$. Then, batch B_1 is assigned to machine M_1 and batch B_3 is assigned to machine M_2, leading to later violating the start time window of batch B_2. This is because when assigning batch B_2, machine M_2 is the machine with the earliest available time but $T_2^{EST} > LS_2^B$ (i.e., $5 > 4$). Thus, as stated in **Condition 1**, no feasible solution is found by the first procedure (see Figure 2-a). But there does exist a feasible solution for the example, as depicted in Figure 2-b. Figure 2 is used to demonstrate the two situations for the described example. One is for assigning batches to machines according to the EST rule in which no feasible solution is found. Another is for the assignment of batches to machines according to the LST rule, where a feasible solution can be obtained.



a) No feasible solution is found by the EST rule b) There exists a feasible solution

Figure 2. Illustrative example for the situation when the EST rule cannot find a feasible solution

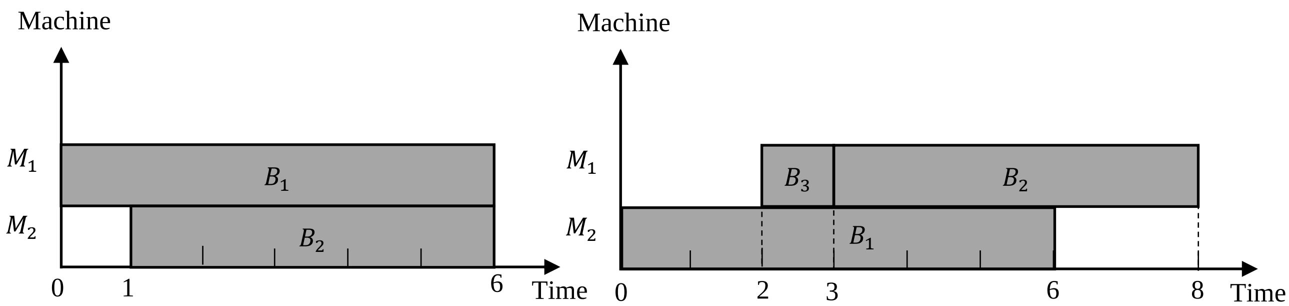
When the first procedure cannot find a feasible solution, we will switch to use the second procedure. The second procedure basically applies the LST rule when considering **Condition 2** and **Condition 3**. The LST rule is motivated by a well-known dispatching rule, Earliest Due Date first (EDD), where a batch with an earlier due date has a higher priority.

Let sequence $\mathcal{L}^{LST}$ be a sorted sequence according to non-decreasing order of batch latest start time for un-assigned batches. Let batch b_1^{LST} be the first batch in sequence $\mathcal{L}^{LST}$ and $LS^B_{b_1^{LST}}$ be the latest start time of batch b_1^{LST}. Let $T_{j^*}^{LST}$ be the earliest available time among machines when assigning batch b_1^{LST}. At time $T_{j^*}^{LST}$, let batch b_2^{LST} be the critical batch such that $max\left(ES^B_{b_2^{LST}}, T_{j^*}^{LST}\right) + p^B_{b_2^{LST}} = \min_{b \text{ in } \mathcal{L}^{LST}, b \neq b_1^{LST}}\left(max\left(ES_b^B, T_{j^*}^{LST}\right) + p_b^B\right)$. Similar to **Condition 1**, **Condition 2** is the feasibility condition for a solution found by the second procedure to ensure that all the batch start time windows are satisfied. **Condition 3** is used to determine that at time $T_{j^*}^{LST}$, either the first batch b_1^{LST} or the critical batch b_2^{LST} in the sequence $\mathcal{L}^{LST}$ should be assigned next.

Condition 2: *If the inequality* $LS^B_{b_1^{LST}} \geq T_{j^*}^{LST}$ *holds for every batch in sequence* $\mathcal{L}^{LST}$*, then there exists a feasible solution for instance* $\boldsymbol{I}$.

Condition 3: *At time $T_{j^*}^{LST}$, batch b_1^{LST} is assigned to machine M_{j^*} if $LS_{b_1^{LST}}^B < max\left(ES_{b_2^{LST}}^B, T_{j^*}^{LST}\right) + p_{b_2^{LST}}^B$; otherwise, batch b_2^{LST} is assigned to machine M_{j^*}.*

The following example is used to illustrate the situation when the pure LST rule cannot find a feasible solution, but there does exist one. Consider the 3-batch, 2-machine example with batch information: $ES_1^B = 0$, $ES_2^B = 1$, $ES_3^B = 2$, $LS_1^B = 3$, $LS_2^B = 4$, $LS_3^B = 5$, $p_1^B = 6$, $p_2^B = 5$ and $p_3^B = 1$. According to the LST rule, we have $\mathcal{L}^{LST} = (B_1, B_2, B_3)$. Then, if batch B_1 is assigned to machine M_1 and batch B_2 is assigned to machine M_2, this will lead to no feasible solution because batch B_3 cannot be scheduled due to the violation of **Condition 2** (see **Figure 3-a)**. However, there exists a feasible solution for the example (see **Figure 3-b)**. **Figure 3** is used to illustrate the two situations for the above example. The first one is for assigning batches to machines according to the pure LST rule, where no feasible solution is obtained. The second is for the assignment of batches to machines according to the pure LST rule with, further considering **Condition 3**, where a feasible solution can be found.



a) No feasible solution is found by the pure LST rule

b) There exists a feasible solution

Figure 3. Illustrative example for the situation when the pure LST rule cannot find a feasible solution

Next, the pseudo-code of our proposed DH algorithm is presented:

DH algorithm

Input: $re_i^L, s_i^L, p_i^L, r_i^L, R_i^L$ for $i = 1, \dots, N$; UB_e for $e = 1, \dots, E$; $h = N$.
Output: The solution with its makespan.
Phase I:
Assign each lot to a batch separately:
For $b = 1, \dots, N$ **do**
    $B_b = \{b\}$; $re_b^B = re_b^L$, $s_b^B = s_b^L$, $p_b^B = p_b^L$, $ES_b^B = r_b^L$, $LS_b^B = r_b^L + R_b^L$;
End For
Let $\boldsymbol{\mathcal{B}} = \{1, \dots, N\}$;
Calculate saving value for every pair of batches in $\boldsymbol{\mathcal{B}}$:
For $k, l \in \boldsymbol{\mathcal{B}}$; $k < l$ **do**
    Calculate $S_{k,l}$ by **Eq. (15)**;
End For
Repeat
    Let k^*, l^* be such that $S_{k^*,l^*} = \max_{k,l \in \boldsymbol{\mathcal{B}};\, k<l} \{S_{k,l}\}$
    If $(max(ES_{k^*}^B, ES_{l^*}^B) \le min(LS_{k^*}^B, LS_{l^*}^B))$ and $(s_{k^*}^B + s_{l^*}^B \le UB_{re_{k^*}^B})$ **then**
        $h = h + 1$;
        Merge batches B_{k^*} and B_{l^*} to form a new batch B_h: $B_h = B_{k^*} \cup B_{l^*}$;
        Remove k^*, l^* from $\boldsymbol{\mathcal{B}}$, and add h to $\boldsymbol{\mathcal{B}}$;
        Determine batch B_h information:
            $re_h^B = re_{k^*}^B$, $p_h^B = p_{k^*}^B$, $s_h^B = s_{k^*}^B + s_{l^*}^B$;
            $ES_h^B = max(ES_{k^*}^B, ES_{l^*}^B)$, $LS_h^B = min(LS_{k^*}^B, LS_{l^*}^B)$;
        Re-calculate saving value for every pair of batches in $\boldsymbol{\mathcal{B}}$:
            For $k, l \in \boldsymbol{\mathcal{B}}$; $k < l$ **do**
                Calculate $S_{k,l}$ by **Eq. (15)**;

End For
End If
Until all $S_{k,l} = 0$ for $k, l \in \boldsymbol{\mathcal{B}}; k < l$.
Phase II:
Let $\boldsymbol{\mathcal{L}^{EST}}$ be a sequence of all formed batches in $\boldsymbol{\mathcal{B}}$, being re-indexed according to the non-decreasing order of batch earliest start time such that $ES_1^B \leq ES_2^B \leq \cdots \leq ES_{|\boldsymbol{\mathcal{B}}|}^B$;
$T_j^{EST} = 0$, $S_j^{EST} = (\,)$ for $j = 1, \dots, M$, $second\_run = False$;
While $\boldsymbol{\mathcal{L}^{EST}}$ is not empty **do**:
Let batch b_1^{EST} be the first batch in $\boldsymbol{\mathcal{L}^{EST}}$;
Let machine j^* be the machine such that $T_{j^*}^{EST} = \min_{j=1,\dots,M}(T_j^{EST})$;
If $LS_{b_1^{EST}}^B \geq T_{j^*}^{EST}$ **then**
Assign batch b_1^{EST} to machine j^* by appending batch b_1^{EST} to $S_{j^*}^{EST}$; Remove b_1^{EST} from $\boldsymbol{\mathcal{L}^{EST}}$;
Determine information of batch b_1^{EST}: $BS_{b_1^{EST}}^{j^*} = max\,(ES_{b_1^{EST}}^B, T_{j^*}^{EST})$; $BC_{b_1^{EST}}^{j^*} = BS_{b_1^{EST}}^{j^*} + p_{b_1^{EST}}^B$;
Update available time of machine j^*: $T_{j^*}^{EST} = BC_{b_1^{EST}}^{j^*}$;
Else:
$second\_run = True$;
Set $\boldsymbol{\mathcal{L}^{EST}}$ is empty;
End While

If $second\_run$ **then**
Let $\boldsymbol{\mathcal{L}^{LST}}$ be a sequence of all formed batches in $\boldsymbol{\mathcal{B}}$, being re-indexed according to non-decreasing order of batch latest start time; namely, $LS_1^B \leq LS_2^B \leq \cdots \leq LS_{|\boldsymbol{\mathcal{B}}|}^B$;
$T_j^{LST} = 0$, $S_j^{LST} = (\,)$ for $j = 1, \dots, M$;
While $\boldsymbol{\mathcal{L}^{LST}}$ is not empty **do**:
Let batch b_1^{LST} be the first batch in $\boldsymbol{\mathcal{L}^{LST}}$;
Let machine j^* be the machine such that $T_{j^*}^{LST} = \min_{j=1,\dots,M}(T_j^{LST})$;
If $LS_{b_1^{LST}}^B \geq T_{j^*}^{LST}$ **then**
Let batch b_2^{LST} be the batch such that $max\left(ES_{b_2^{LST}}^B, T_{j^*}^{LST}\right) + p_{b_2^{LST}}^B = \min_{b \text{ in } \boldsymbol{\mathcal{L}^{LST}}, b \neq b_1^{LST}}(max\left(ES_b^B, T_{j^*}^{LST}\right) + p_b^B)$;
If $LS_{b_1^{LST}}^B < max\left(ES_{b_2^{LST}}^B, T_{j^*}^{LST}\right) + p_{b_2^{LST}}^B$ **then**
Assign batch b_1^{LST} to machine j^* by appending batch b_1^{LST} to $S_{j^*}^{LST}$; Remove b_1^{LST} from $\boldsymbol{\mathcal{L}^{LST}}$;
Determine information of b_1^{LST} : $BS_{b_1^{LST}}^{j^*} = max\,(ES_{b_1^{LST}}^B, T_{j^*}^{LST})$; $BC_{b_1^{LST}}^{j^*} = BS_{b_1^{LST}}^{j^*} + p_{b_1^{LST}}^B$;
Update available time of machine j^*: $T_{j^*}^{LST} = BC_{b_1^{LST}}^{j^*}$;
Else:
Assign batch b_2^{LST} to machine j^* by appending batch b_2^{LST} to $S_{j^*}^{LST}$; Remove b_2^{LST} from $\boldsymbol{\mathcal{L}^{LST}}$;
Determine information of b_2^{LST}: $BS_{b_2^{LST}}^{j^*} = max\,(ES_{b_2^{LST}}^B, T_{j^*}^{LST})$; $BC_{b_2^{LST}}^{j^*} = BS_{b_2^{LST}}^{j^*} + p_{b_2^{LST}}^B$;
Update available time of machine j^*: $T_{j^*}^{LST} = BC_{b_2^{LST}}^{j^*}$;
Else:
No feasible solution is found.
End While
End If

The computational complexity of the proposed algorithm can be determined as follows: the time complexity of Phase I is O(n^3) while the time complexity of Phase II is O(n^2(log (n)). Thus, the proposed DH algorithm is then referred to as an O(n^3) algorithm.

4. COMPUTATIONAL RESULTS

We conduct experiments to evaluate the effectiveness of our proposed MIP model and DH heuristic. The proposed heuristic is coded in Python and run on an Intel(R) Core(TM) i7-8550UCPU at 1.8GHz with 8GB of RAM memory. The Gurobi 8.0.1 solver is used for the MIP model. To prevent excessive computation time, the running time limit is set to 3600 seconds (i.e., one hour).

4.1. Experiment design

Two computational tests are designed to evaluate the performance of the proposed algorithm. The first test is to evaluate the solution quality of the proposed algorithm, while the second test is used to show how well our proposed algorithm performs for large-size problems. Because the MIP can solve only small-size instances, this first test is conducted only on small-size problems and has 48 different combinations of factors. For each combination of the 48 combinations, we randomly generate 10 problem instances. The second test is designed to compare our proposed algorithm to the heuristics proposed in Koh *et al*. (2004) for the problem $Pm|batch, incompatible, s_i|C_{max}$ without time window constraints. Koh *et al*. (2004) proposed three simple heuristics and two GAs for solving the problem and indicated that the simple heuristic LFLT (largest job first fit batching and longest processing time sequencing) outperformed other heuristics and GAs. Therefore, we will only compare our algorithm with the LFLT heuristic. Note that LFLT is the heuristic in which batches are formed by the order of job sizes, and batch sequencing for the machines is based on the order of batch processing times. The testing instances are randomly generated according to the setting used in Koh *et al*. (2004). For each combination of 36 combinations, we randomly generate 20 problem instances. The factors and the levels for generating instances are shown in **Table 1**.

Table 1. Experimental factors for small-size and large-size problems

Factor	Small-size instances	Count	Large-size instances (Koh *et al*., 2004)	Count
Number of machines, M	2, 3	2	10, 30, 50	3
Number of lots, N	10, 15, 20	3	100, 200, 300	3
Number of recipes, E	3	1	5, 10, 15, 20	4
Processing time, P_e	$U[1, 10]$	1	$U[10e, 10e + 10]$	1
Lot size, s_i	$U\ [1, 15]$ & $U\ [15, 50]$	2	$U[1,100]/100$	1
Batch capacity, UB_e	$U\ [50, 70]$	1	1	1
Release time, r_i	$U\ [0, 30]$ & $U\ [0, 60]$	2	-	-
Remaining lifetime, R_i	αp_i (α =5, 10)	2	-	-
Number of factor combinations		48		36

4.2. Experimental results

4.2.1. Small-size instances

Tables 2, 3, and 4 present the experimental results obtained by the MIP model and the DH heuristic for small-size instances with 10, 15, and 20 lots, respectively. In each table, the results are grouped by the number of machines ($M = 2, 3$). Columns 1 and 12 represent the run code for the instance with the combination of lot release times ranges (ri), remaining lifetime (Ri), and lot sizes ranges (si), $i = 1, 2$. For example, "r1R1s1" represents the instance with release time within $U\ [0, 30]$, remaining lifetime with $\alpha = 5$ and lot size within $U\ [1, 15]$. For each combination, ten problem instances are randomly generated. The proposed heuristic's improvement is calculated by $IMP\ (\%) = \frac{Heu_{sol} - Min_{sol}}{Min_{sol}} \times 100$, where Heu_{sol} is the makespan value obtained by the DH heuristic and Min_{sol} is the makespan value obtained by the MIP model. For $M = 2$, columns 2-3 (13-14) report the C_{max} and run time produced by the MIP model, respectively. Columns 4-6 (15-17) report the C_{max}, run time and improvement obtained by the DH heuristic, respectively. While the corresponding columns 7-11 and 18-22 report the results for $M = 3$. Besides, Table 5 displays the performance comparison between the MIP model and the DH algorithm in terms of solution quality (namely, number of problem instances receiving the optimal solutions and the worst IMP) and computation time (namely, average run times).

The results from Tables 2-5 reveal that the proposed DH heuristic performs very efficiently and gets optimal solutions for almost all small-size problems in a very short run time. For a total of 480 instances for small-size problems, the percentage of achieving optimal solutions by the DH is 94.17%. (452 out of 480). The high percentage indicates that the proposed heuristic is very good in solving small-size problems. Even for the instances where the proposed heuristics cannot obtain optimal solutions, the solution found is still quite close to the optimal solution. By comparing the results of our heuristic with the optimal solutions, we can see that the worst IMP value for the DH is only 9.68%. Concerning computation time, it is shown that the proposed heuristic is significantly faster than the MIP model. The average run time of the MIP model on all the instances is about 326.27 seconds, while the DH heuristic requires only 0.02 seconds on average to solve an instance.

4.2.2. Large-size instances

Table 6 presents the comparative results obtained by the LFLT and the DH heuristic for the large-size instances. The performance of a heuristic is measured by $GAP(\%) = \frac{Heu_{sol} - \text{LB}}{\text{LB}} \times 100$, where Heu_{sol} is the makespan value obtained by the corresponding heuristic (e.g., DH or LFLT) and LB is the lower bound value. LB for each instance is used as the base value for the comparison of the results found by LFLT and DH. The table consists of 36 combinations according to the levels of N, M and E. For each combination, the results of 20 test instances are summarized by two kinds of values, one of which represents the average GAP of the corresponding heuristic, while the other is a standard deviation of the average GAP values. A smaller average GAP value indicates that the solution found by the corresponding heuristic is closer to the lower bound averagely. In Table 6, the average GAP of LFLT varies from 0 to 32.04%, while the average GAP of DH varies from 0 to 22.79%. It indicates that the DH heuristic performs better than the LFLT heuristic. The results clearly show that our proposed heuristic is efficient and produces results that are closer to the lower bound compared to the existing heuristic LFLT. From the perspective of computational effort, the run time to get a solution from LFLT is shorter than one second. While the run time of our proposed DH is longer and depends on the number of lots N. However, the run time of the heuristic is still in a reasonable range in every instance. The average run time of the DH is about 5 seconds when $N = 100$, about 70 seconds when $N = 200$, and about 350 seconds when $N = 300$.

In order to validate the obtained results, we conduct the one-way ANOVA test and use the GAP measure as the response variable. We have a null hypothesis stating that the mean GAP values of the two heuristics are equal. The ANOVA table in Figure 4 shows that the $p\text{-}value$ is 0.000, which is less than the significance level of 0.05; we reject the null hypothesis that the two heuristics have the same mean GAP values. We then use Tukey's test to do the pairwise comparisons between the two heuristics. The result of the Tukey test in Figure 4 shows that LFLT is in Group A while DH is in Group B at the 95% confidence level, and there is a statistically significant difference between GAP values of DH and LFLT. This indicates that the mean of DH is significantly lower than the mean of LFLT. (Please see **APPENDIX C** for the results of all instances).

One-way ANOVA: LFLT, DH

Analysis of Variance

Source	DF	Adj SS	Adj MS	F-Value	P-Value
Factor	1	10933	10932.6	153.84	0.000
Error	1438	102188	71.1		
Total	1439	113120			

Tukey Pairwise Comparisons

Grouping Information Using the Tukey Method and 95% Confidence

Factor	N	Mean	Grouping
LFLT	720	21.620	A
DH	720	16.109	B

Means that do not share a letter are significantly different.

Figure 4. ANOVA table and Tukey test table for LFLT and DH

Table 2. Computational results for small-size problems with 10 lots

Run code	$M = 2$					$M = 3$				
	MIP		DH			MIP		DH		
	C_{max}	Run time	C_{max}	Run time	IMP	C_{max}	Run time	C_{max}	Run time	IMP
r1R1s1	**38**	0.49	**38**	0.005	0	**27**	10.76	**27**	0.004	0
	38	0.45	**38**	0.003	0	**36**	0.45	**36**	0.009	0
	48	0.65	**48**	0.003	0	**34**	0.67	**34**	0.008	0
	35	1.99	**35**	0.006	0	**32**	0.19	**32**	0.007	0
	34	0.37	**34**	0.004	0	**36**	7.98	**36**	0.009	0
	31	2.61	**31**	0.005	0	**31**	0.32	**31**	0.005	0
	34	1.12	**34**	0.006	0	**30**	0.98	**30**	0.007	0
	32	0.81	**32**	0.004	0	**32**	1.87	**32**	0.005	0
	38	0.35	**38**	0.006	0	**33**	2.09	**33**	0.004	0
	33	1.97	**33**	0.004	0	**36**	10.02	**36**	0.015	0
r1R1s2	**30**	2.35	**30**	0.003	0	**38**	5.22	**38**	0.006	0
	32	2.65	**32**	0.004	0	**37**	6.98	**37**	0.009	0
	31	2.66	**31**	0.004	0	**36**	10.26	**36**	0.013	0
	36	1.75	**36**	0.003	0	**35**	9.28	**35**	0.012	0
	36	0.63	**36**	0.004	0	**38**	0.19	**38**	0.008	0
	34	4.53	**34**	0.005	0	**38**	29.18	**38**	0.011	0
	33	0.82	**33**	0.004	0	**35**	0.35	**35**	0.003	0
	31	35.29	**31**	0.005	0	**35**	2.19	**35**	0.004	0
	34	43.12	**34**	0.005	0	**36**	0.96	**36**	0.004	0
	33	2.41	**33**	0.005	0	**34**	0.35	**34**	0.005	0
r1R2s1	**31**	1.69	**31**	0.004	0	**40**	6.44	**40**	0.004	0
	34	2.87	**34**	0.005	0	**35**	7.19	**35**	0.008	0
	38	2.06	**38**	0.005	0	**33**	9.18	**33**	0.009	0
	34	0.67	**34**	0.004	0	**34**	69.18	**34**	0.011	0
	33	2.97	**33**	0.005	0	**33**	0.87	**33**	0.007	0
	43	11.21	**43**	0.004	0	**37**	3.09	**37**	0.004	0
	32	5.36	**32**	0.008	0	**28**	1.77	**28**	0.005	0
	36	0.52	**36**	0.006	0	**38**	0.23	**38**	0.004	0
	37	0.66	**37**	0.004	0	**34**	1.78	**34**	0.005	0
	33	4.08	**33**	0.005	0	**39**	0.27	**39**	0.006	0
r1R2s2	**39**	50.01	**39**	0.004	0	**38**	1070.12	**38**	0.015	0
	37	107.82	**37**	0.006	0	**30**	3.37	**30**	0.004	0
	33	0.73	**33**	0.003	0	**44**	9.19	**44**	0.012	0
	61	15.86	**61**	0.004	0	**34**	3.19	**34**	0.013	0
	34	1.45	**34**	0.003	0	**37**	28.19	**37**	0.012	0
	34	2.19	**34**	0.004	0	**38**	0.12	**38**	0.005	0
	39	0.39	**39**	0.004	0	**35**	5.19	**35**	0.006	0
	35	3.19	**35**	0.005	0	**39**	0.28	**39**	0.004	0
	61	2.15	**61**	0.004	0	**33**	1.99	**33**	0.006	0
	33	3.85	**33**	0.003	0	**39**	7.93	**39**	0.003	0

Run code	$M = 2$					$M = 3$				
	MIP		DH			MIP		DH		
	C_{max}	Run time	C_{max}	Run time	IMP	C_{max}	Run time	C_{max}	Run time	IMP
r2R1s1	**62**	1.72	**62**	0.004	0	**65**	1.75	**65**	0.003	0
	63	0.23	**63**	0.004	0	**60**	0.98	**60**	0.008	0
	57	0.39	**57**	0.005	0	**65**	3.89	**65**	0.012	0
	60	2.18	**60**	0.004	0	**60**	103.19	**60**	0.011	0
	61	0.89	**61**	0.005	0	**61**	87.19	**61**	0.012	0
	61	5.35	**61**	0.004	0	**41**	10.05	**41**	0.004	0
	69	9.21	**69**	0.004	0	**64**	8.17	**64**	0.004	0
	68	2.31	**68**	0.006	0	**44**	0.11	**44**	0.005	0
	63	0.69	**63**	0.005	0	**62**	0.63	**62**	0.006	0
	59	0.53	**59**	0.005	0	**50**	3.77	**50**	0.003	0
r2R1s2	**69**	0.65	**69**	0.003	0	**53**	5.64	**53**	0.003	0
	45	8.33	**45**	0.004	0	**65**	9.18	**65**	0.009	0
	54	0.89	**54**	0.007	0	**49**	56.19	**49**	0.012	0
	60	0.65	**60**	0.005	0	**55**	9.18	**55**	0.008	0
	61	3.89	**61**	0.005	0	**54**	29.18	**54**	0.013	0
	61	29.78	**61**	0.005	0	**59**	1.28	**59**	0.006	0
	54	3.14	**54**	0.006	0	**60**	102.19	**60**	0.004	0
	58	0.87	**58**	0.005	0	**61**	0.76	**61**	0.003	0
	68	0.58	**68**	0.004	0	**65**	0.25	**65**	0.006	0
	58	0.98	**58**	0.003	0	**67**	9.91	**67**	0.007	0
r2R2s1	**60**	1.23	**60**	0.006	0	**59**	6.94	**59**	0.004	0
	61	10.89	**61**	0.007	0	**63**	0.67	**63**	0.013	0
	64	0.29	**64**	0.006	0	**64**	31.98	**64**	0.012	0
	60	3.45	**60**	0.006	0	**56**	11.18	**56**	0.008	0
	61	0.87	**61**	0.005	0	**57**	0.67	**57**	0.015	0
	56	1.18	**56**	0.006	0	**65**	5.89	**65**	0.011	0
	64	0.78	**64**	0.005	0	**59**	0.72	**59**	0.005	0
	55	2.16	**55**	0.006	0	**63**	119.01	**63**	0.004	0
	60	0.67	**60**	0.006	0	**61**	2.66	**61**	0.004	0
	62	1.98	**62**	0.004	0	**64**	0.18	**64**	0.004	0
r2R2s2	**60**	2.96	**60**	0.004	0	**47**	3.06	**47**	0.005	0
	61	0.28	**61**	0.004	0	**73**	5.09	**73**	0.007	0
	66	3.23	**66**	0.005	0	**66**	0.87	**66**	0.008	0
	64	0.82	**64**	0.005	0	**66**	11.19	**66**	0.011	0
	50	4.56	**50**	0.005	0	**60**	38.19	**60**	0.009	0
	65	7.12	**65**	0.007	0	**53**	111.87	**53**	0.003	0
	57	7.19	**57**	0.005	0	**64**	0.57	**64**	0.004	0
	52	1.09	**52**	0.003	0	**61**	7.34	**61**	0.006	0
	68	0.78	**68**	0.005	0	**62**	19.28	**62**	0.003	0
	55	3.11	**55**	0.004	0	**59**	0.53	**59**	0.004	0

Note: "*" represents the best result found within 3600 seconds.
Bold numbers represent the optimal solutions for each run code.

Table 3. Computational results for small-size problems with 15 lots

Run code	$M=2$ MIP C_{max}	$M=2$ MIP Run time	$M=2$ DH C_{max}	$M=2$ DH Run time	$M=2$ IMP	$M=3$ MIP C_{max}	$M=3$ MIP Run time	$M=3$ DH C_{max}	$M=3$ DH Run time	$M=3$ IMP
r1R1s1	**35**	773.67	**35**	0.011	0	**37**	14.48	**37**	0.009	0
	39	35.13	**39**	0.013	0	**36**	34.19	**36**	0.016	0
	39	20.21	**39**	0.013	0	**36**	78.18	**36**	0.017	0
	38	128.15	**38**	0.013	0	**67**	134.19	**67**	0.015	0
	40	231.19	**40**	0.019	0	**37**	2091.18	**37**	0.016	0
	36	0.51	**36**	0.013	0	**34**	1523.19	**34**	0.067	0
	36	2.56	**36**	0.012	0	**35**	11.67	**35**	0.076	0
	34	3.17	**34**	0.014	0	**34**	524.18	**34**	0.075	0
	38	120.16	**38**	0.013	0	**32**	1092.19	**32**	0.078	0
	27	191.28	**27**	0.011	0	**33**	18.72	**33**	0.072	0
r1R1s2	**48**	*3600.00	49	0.008	2.08	**36**	29.81	**36**	0.008	0
	40	576.76	**40**	0.011	0	**34**	78.19	35	0.012	2.94
	53	891.13	54	0.018	1.89	**29**	53.19	**29**	0.013	0
	42	189.13	**42**	0.018	0	**34**	150.12	**34**	0.014	0
	36	1367.18	**36**	0.012	0	**41**	200.18	44	0.013	7.32
	38	100.27	**38**	0.011	0	**35**	532.65	**35**	0.006	0
	49	2151.29	**49**	0.021	0	**32**	197.76	**32**	0.005	0
	43	50.17	**43**	0.016	0	**39**	*3600.00	41	0.005	5.13
	32	23.19	**32**	0.015	0	**35**	1201.12	**35**	0.008	0
	30	12.87	**30**	0.012	0	**36**	91.19	**36**	0.005	0
r1R2s1	**34**	9.04	**34**	0.017	0	**36**	12.99	**36**	0.013	0
	37	10.56	**37**	0.019	0	**38**	34.19	**38**	0.012	0
	40	138.17	**40**	0.082	0	**35**	9.28	**35**	0.019	0
	32	459.19	**32**	0.09	0	**36**	109.18	**36**	0.021	0
	35	34.17	**35**	0.087	0	**36**	387.92	**36**	0.017	0
	35	760.91	**35**	0.072	0	**35**	687.34	**35**	0.038	0
	36	156.18	**36**	0.061	0	**38**	76.16	**38**	0.026	0
	40	37.67	**40**	0.078	0	**35**	10.08	**35**	0.041	0
	34	21.18	**34**	0.083	0	**36**	1001.97	**36**	0.024	0
	38	8.19	**38**	0.085	0	**31**	32.18	**31**	0.066	0
r1R2s2	**31**	15.49	34	0.005	9.68	**39**	7.01	41	0.009	5.13
	60	1310.13	**60**	0.011	0	**29**	21.15	**29**	0.013	0
	38	415.69	**38**	0.009	0	**47**	6.27	**47**	0.015	0
	34	345.18	**34**	0.029	0	**37**	105.28	38	0.027	2.70
	71	3108.12	**71**	0.052	0	**38**	70.12	40	0.019	5.26
	61	*3600.00	62	0.012	1.64	**33**	231.19	**33**	0.021	0
	40	160.89	**40**	0.029	0	**36**	334.17	**36**	0.051	0
	49	30.11	**49**	0.019	0	**31**	*3600.00	**31**	0.023	0
	38	134.18	**38**	0.053	0	**35**	99.44	**35**	0.072	0
	47	47.91	**47**	0.047	0	**39**	31.56	**39**	0.045	0

Run code	$M=2$ MIP C_{max}	$M=2$ MIP Run time	$M=2$ DH C_{max}	$M=2$ DH Run time	$M=2$ IMP	$M=3$ MIP C_{max}	$M=3$ MIP Run time	$M=3$ DH C_{max}	$M=3$ DH Run time	$M=3$ IMP
r2R1s1	**55**	4.19	**55**	0.009	0	**67**	1.15	**67**	0.012	0
	66	389.19	**66**	0.013	0	**65**	6.78	**65**	0.015	0
	67	98.13	**67**	0.017	0	**67**	3.19	**67**	0.014	0
	68	250.82	**68**	0.019	0	**67**	89.19	**67**	0.016	0
	64	270.89	**64**	0.016	0	**57**	51.89	**57**	0.018	0
	75	1091.78	**75**	0.118	0	**64**	10.28	**64**	0.015	0
	66	231.19	**66**	0.067	0	**68**	23.34	**68**	0.071	0
	62	129.18	**62**	0.071	0	**64**	5.23	**64**	0.019	0
	63	512.76	**63**	0.052	0	**61**	2018.19	**61**	0.025	0
	69	191.17	**69**	0.008	0	**67**	156.37	**67**	0.027	0
r2R1s2	68	28.59	**68**	0.011	0	**66**	9.18	**66**	0.011	0
	62	35.19	**62**	0.017	0	**53**	4.74	**53**	0.007	0
	65	129.18	**65**	0.011	0	**49**	29.19	**49**	0.012	0
	68	2130.89	69	0.011	1.47	**63**	309.19	**63**	0.014	0
	54	71.29	**54**	0.013	0	**49**	491.19	**49**	0.019	0
	73	159.28	**73**	0.016	0	**64**	22.66	**64**	0.024	0
	72	367.67	**72**	0.027	0	**63**	57.56	**63**	0.028	0
	73	210.16	**73**	0.031	0	**61**	19.78	**61**	0.037	0
	56	45.19	**56**	0.096	0	**64**	374.78	**64**	0.025	0
	57	134.19	**57**	0.018	0	**63**	269.74	**63**	0.038	0
r2R2s1	59	2.11	**59**	0.019	0	**65**	2.67	**65**	0.009	0
	60	318.17	**60**	0.014	0	**65**	72.19	**65**	0.009	0
	62	78.19	**62**	0.013	0	**70**	5.19	**70**	0.012	0
	66	134.19	**66**	0.019	0	**62**	193.12	**62**	0.012	0
	61	349.98	**61**	0.012	0	**68**	187.18	**68**	0.011	0
	57	15.16	**57**	0.014	0	**52**	239.66	**52**	0.008	0
	70	117.19	**70**	0.097	0	**67**	1291.76	**67**	0.037	0
	70	389.65	**70**	0.059	0	**63**	16.67	**63**	0.035	0
	65	71.23	**65**	0.124	0	**65**	9.23	**65**	0.048	0
	64	1119.87	**64**	0.129	0	**65**	209.54	**65**	0.051	0
r2R2s2	**77**	1045.11	**77**	0.014	0	**61**	17.75	**61**	0.008	0
	65	28.33	**65**	0.008	0	**64**	34.18	**64**	0.012	0
	57	*3600.00	**57**	0.086	0	**66**	43.19	**66**	0.013	0
	59	98.18	**59**	0.011	0	**65**	203.78	**65**	0.013	0
	58	211.28	**58**	0.017	0	**60**	134.87	**60**	0.015	0
	64	145.83	**64**	0.018	0	**48**	1027.56	**48**	0.008	0
	65	98.18	**65**	0.019	0	**63**	2981.55	64	0.014	1.59
	67	619.23	**67**	0.012	0	**58**	19.43	**58**	0.034	0
	70	160.29	**70**	0.086	0	**61**	88.54	**61**	0.023	0
	68	44.19	**68**	0.052	0	**62**	23.19	**62**	0.042	0

Table 4. Computational results for small-size problems with 20 lots

Run code	$M=2$ MIP C_{max}	Run time	DH C_{max}	Run time	IMP	$M=3$ MIP C_{max}	Run time	DH C_{max}	Run time	IMP
r1R1s1	41	24.45	41	0.022	0	39	141.26	39	0.021	0
	36	108.34	36	0.021	0	37	102.19	37	0.024	0
	34	87.12	34	0.028	0	38	59.12	38	0.021	0
	35	290.13	35	0.021	0	36	11.46	36	0.023	0
	35	10.88	35	0.022	0	35	0.12	35	0.021	0
	36	6.17	36	0.019	0	37	203.18	37	0.021	0
	41	4.59	41	0.018	0	37	182.17	37	0.017	0
	33	2.13	33	0.019	0	37	98.78	37	0.023	0
	37	10.12	37	0.029	0	29	365.19	29	0.019	0
	40	2108.17	40	0.017	0	30	711.67	30	0.016	0
r1R1s2	33	324.23	33	0.014	0	37	*3600.00	38	0.014	2.70
	37	99.82	37	0.014	0	32	201.19	32	0.027	0
	60	*3600.00	64	0.051	6.67	49	1802.19	51	0.019	4.08
	40	201.29	40	0.015	0	37	*3600.00	37	0.022	0
	30	*3600.00	30	0.031	0	37	132.19	38	0.022	2.70
	53	1976.11	56	0.017	5.66	38	49.81	38	0.021	0
	45	872.11	46	0.028	2.22	34	78.87	34	0.019	0
	30	20.17	30	0.016	0	32	761.56	32	0.035	0
	68	*3600.00	69	0.015	1.47	39	29.91	39	0.031	0
	42	2187.19	42	0.018	0	35	377.29	35	0.041	0
r1R2s1	31	43.54	31	0.021	0	39	117.91	39	0.025	0
	37	198.45	37	0.024	0	38	0.78	38	0.024	0
	33	8.54	33	0.024	0	37	5.19	37	0.018	0
	33	4.21	33	0.022	0	35	109.18	35	0.021	0
	35	68.13	35	0.026	0	32	35.19	32	0.023	0
	33	10.12	33	0.018	0	36	567.31	36	0.028	0
	35	36.19	35	0.021	0	34	1012.76	34	0.041	0
	38	90.17	39	0.018	2.63	32	29.18	32	0.023	0
	40	3.19	40	0.017	0	39	99.18	39	0.038	0
	37	5.21	37	0.016	0	38	355.27	38	0.025	0
r1R2s2	32	*3600.00	32	0.018	0	40	301.67	40	0.021	0
	33	313.34	33	0.013	0	37	123.18	37	0.022	0
	72	3385	73	0.021	1.39	33	8.19	33	0.023	0
	34	3344.12	34	0.016	0	40	299.18	40	0.037	0
	35	9.89	36	0.021	2.86	35	19.19	35	0.024	0
	48	103.98	48	0.027	0	39	516.88	40	0.036	2.56
	31	1219.18	31	0.021	0	34	91.02	34	0.021	0
	48	2103.12	51	0.021	6.25	32	316.28	32	0.028	0
	34	60.70	34	0.013	0	30	123.27	30	0.018	0
	48	401.50	48	0.015	0	38	1005.88	38	0.019	0

Run code	$M=2$ MIP C_{max}	Run time	DH C_{max}	Run time	IMP	$M=3$ MIP C_{max}	Run time	DH C_{max}	Run time	IMP
r2R1s1	59	33.61	59	0.018	0	65	42.82	65	0.022	0
	66	750.49	66	0.022	0	63	55.13	63	0.031	0
	69	431.13	69	0.023	0	62	9.28	62	0.022	0
	62	23.14	62	0.021	0	65	90.13	65	0.021	0
	66	31.14	66	0.022	0	60	109.19	60	0.025	0
	59	156.13	60	0.021	1.69	68	543.68	68	0.028	0
	62	29.17	62	0.022	0	65	165.28	65	0.036	0
	64	317.19	64	0.016	0	69	19.27	69	0.028	0
	69	209.28	69	0.017	0	65	1113.29	65	0.037	0
	60	51.17	60	0.036	0	62	75.37	62	0.036	0
r2R1s2	54	108.27	54	0.013	0	67	*3600.00	67	0.015	0
	66	36.53	66	0.018	0	66	52.77	66	0.013	0
	65	43.45	65	0.019	0	66	109.18	66	0.018	0
	63	1029.76	65	0.015	3.17	70	48.19	70	0.021	0
	67	2007.21	68	0.017	1.49	68	23.19	68	0.019	0
	68	*3600.00	68	0.014	0	67	9.19	67	0.022	0
	59	123.12	59	0.026	0	65	298.37	65	0.031	0
	65	256.13	65	0.019	0	57	766.39	57	0.019	0
	56	23.19	56	0.027	0	66	78.28	66	0.037	0
	63	65.18	63	0.018	0	67	31.77	67	0.024	0
r2R2s1	66	79.34	67	0.031	1.52	59	51.07	59	0.019	0
	70	32.14	70	0.024	0	63	23.19	63	0.024	0
	61	43.19	61	0.021	0	59	108.19	59	0.019	0
	66	234.56	66	0.021	0	64	68.19	64	0.022	0
	70	431.78	70	0.025	0	62	87.29	62	0.021	0
	55	201.18	55	0.027	0	51	268.34	51	0.029	0
	64	2031.91	64	0.023	0	63	41.65	63	0.018	0
	65	761.09	65	0.021	0	63	712.19	63	0.032	0
	66	29.18	66	0.018	0	65	35.65	65	0.027	0
	65	97.92	65	0.016	0	64	1768.25	64	0.031	0
r2R2s2	62	43.82	62	0.032	0	59	48.66	59	0.016	0
	60	659.18	60	0.017	0	63	198.19	63	0.025	0
	59	2191.10	59	0.024	0	67	37.19	67	0.019	0
	76	1271.19	76	0.015	0	66	92.19	66	0.032	0
	69	234.12	69	0.019	0	57	239.19	57	0.023	0
	66	367.12	66	0.013	0	70	11.37	70	0.032	0
	64	34.89	64	0.015	0	65	753.19	65	0.029	0
	60	102.15	60	0.044	0	67	55.28	67	0.037	0
	70	2819.11	70	0.031	0	66	1213.99	66	0.033	0
	70	367.12	70	0.018	0	64	2128.44	64	0.018	0

Table 5. Comparison between the MIP model and DH algorithm

Criteria		**MIP model**	**DH algorithm**
Total number of small-size instances		480.00	480.00
Solution quality	Number of instances receiving the optimal solutions	480.00	452.00
	Worst IMP (%)	0.00	9.68
Average run times (seconds)		326.27	0.02

Table 6. Performance comparison between our DH heuristic and LFLT heuristic

M	E		$N = 100$		$N = 200$		$N = 300$	
			LFLT	DH	LFLT	DH	LFLT	DH
10	5	Avg. GAP	26.42	19.00	28.27	19.11	28.36	18.40
		SD	3.47	3.32	2.71	2.47	1.51	2.43
	10	Avg. GAP	18.10	13.35	24.38	16.67	26.05	17.21
		SD	3.50	4.07	2.65	2.92	1.73	1.89
	15	Avg. GAP	14.11	9.50	19.88	13.11	23.53	15.82
		SD	3.59	2.88	2.43	2.38	2.10	1.41
	20	Avg. GAP	8.90	6.09	18.42	12.69	21.36	14.18
		SD	4.03	3.70	2.71	2.45	2.10	1.92
30	5	Avg. GAP	22.79	21.10	30.18	21.58	31.17	21.75
		SD	5.36	7.43	4.36	2.88	2.52	2.93
	10	Avg. GAP	23.26	22.32	27.27	21.32	28.16	18.98
		SD	4.22	4.89	2.92	2.56	1.80	3.16
	15	Avg. GAP	19.02	18.63	23.35	17.78	25.67	18.46
		SD	4.81	5.19	3.53	2.59	1.83	1.67
	20	Avg. GAP	16.57	16.83	21.18	16.11	23.25	16.61
		SD	3.76	3.90	2.40	3.16	2.20	1.85
50	5	Avg. GAP	4.79	0.56	28.34	21.47	32.04	22.27
		SD	6.14	2.48	7.22	2.91	5.47	3.00
	10	Avg. GAP	0.05	0	28.50	22.79	29.94	21.42
		SD	0.20	0	4.82	3.96	2.81	2.18
	15	Avg. GAP	0	0.82	26.30	22.29	28.46	21.14
		SD	0	2.35	3.67	2.74	2.68	3.22
	20	Avg. GAP	0	0.05	26.22	21.98	24.92	18.54
		SD	0	0.22	4.10	3.96	2.13	2.90

5. CONCLUSIONS

In this study, the parallel BPM problem with various constraints when minimizing makespan is investigated. This problem is motivated by the wafer fabrication procedure in the semiconductor industry. A MIP model is first proposed to obtain optimal solutions for our problem. To deal with the large-scale problem, a DH algorithm, which includes two phases - batch formation and batch scheduling, is proposed to obtain approximation solutions within a reasonable run time. A new two-dimensional saving function is introduced to quantify the saving space of time and capacity, which is a basis for batch formation in the DH algorithm. A comprehensive set of randomly generated small- and large-size instances are used to evaluate the performance of the proposed algorithm. The computational experiments show that the proposed heuristic performs well for small-size problems and can deal with large-scale problems efficiently within a reasonable computational time. For small-size problems, the percentage of achieving optimal solutions by the DH is 94.17%. The high percentage indicates that the proposed heuristic is very good in solving small-size problems. The DH heuristic requires only 0.02 seconds on average to solve an instance, while the average run time of the MIP model is about 326.27 seconds. The experiment for the large-scale problems is designed to compare our proposed heuristic to the existing heuristic LFLT proposed by Koh *et al*. (2004).

Computational results indicate that the average GAP of LFLT varies from 0 to 32.04%, while the average GAP of DH varies from 0 to 22.79%. It shows that the DH heuristic is efficient and outperforms the heuristic LFLT. In future research, further study can be directed to problems in job shop or flow shop environments. Other criteria, such as the total completion time or due date-related performance measures, are also worth studying.

Moreover, in order to illustrate the application of the proposed algorithm as interesting future research, it is worthwhile to further study the solution quality while the studied problem is involved with more practical assumptions, such as machine breakdowns, resource constrains and setup time constraints. Besides, since our proposed algorithm is a deterministic heuristic and only one solution is found for each instance, a local search algorithm can be applied to explore the neighborhood of the solution found by our proposed heuristic and to further improve the solution quality for large-size instances. Namely, it is worthwhile to treat the solution found by our heuristic as an initial solution when applying a local search strategy.

REFERENCES

Almeder, C. and L. Mönch. (2011). Metaheuristics for Scheduling Jobs with Incompatible Families on Parallel Batching Machines. *Journal of the Operational Research Society,* 62(12): 2083–96.

Arroyo, J. E C. and Leung, J. Y. T. (2017a). An Effective Iterated Greedy Algorithm for Scheduling Unrelated Parallel Batch Machines with Non-Identical Capacities and Unequal Ready Times. *Computers and Industrial Engineering,* 105: 84–100.

Arroyo, J. E C. and Leung, J. Y. T. (2017b). Scheduling Unrelated Parallel Batch Processing Machines with Non-Identical Job Sizes and Unequal Ready Times. *Computers and Operations Research,* 78: 117–28.

Balasubramanian, H., Mönch, L., Fowler, J. and Pfund, M. (2004). Genetic Algorithm Based Scheduling of Parallel Batch Machines with Incompatible Job Families to Minimize Total Weighted Tardiness.*International Journal of Production Research,* 42(8): 1621–38.

Bard, J. F. and Rojanasoonthon, S. (2006). A Branch-and-Price Algorithm for Parallel Machine Scheduling with Time Windows and Job Priorities. *Naval Research Logistics,* 53(1): 24–44.

Bilyk, A., Mönch, L. and Almeder, C. (2014). Scheduling Jobs with Ready Times and Precedence Constraints on Parallel Batch Machines Using Metaheuristics. *Computers and Industrial Engineering,* 78: 175–85.

Brucker, P. and Kravchenko, S. A. (2008). Scheduling Jobs with Equal Processing Times and Time Windows on Identical Parallel Machines. *Journal of Scheduling,* 11(4): 229–37.

Çatay, B. (2010). A New Saving-Based Ant Algorithm for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Expert Systems with Applications,* 37(10): 6809–17.

Chang, P. Y., Damodaran, P. and Melouk, S. (2004). Minimizing Makespan on Parallel Batch Processing Machines. *International Journal of Production Research,* 42(19): 4211–20.

Chen, H., Du, B. and Huang, G. Q. (2011). Scheduling a Batch Processing Machine with Non-Identical Job Sizes: A Clustering Perspective. *International Journal of Production Research,* 49(19): 5755–78.

Cheng, B., Cai, J., Yang, S. and Hu, X. (2014). Algorithms for Scheduling Incompatible Job Families on Single Batching Machine with Limited Capacity. *Computers and Industrial Engineering,* 75(1):116–20.

Chiang, T. C., Cheng, H. C. and Fu, L. C. (2010). A Memetic Algorithm for Minimizing Total Weighted Tardiness on Parallel Batch Machines with Incompatible Job Families and Dynamic Job Arrival. *Computers and Operations Research,* 37(12): 2257–69.

Chung, S. H., Tai, Y. T. and Pearn, W. L. (2009). Minimising Makespan on Parallel Batch Processing Machines with Non-Identical Ready Time and Arbitrary Job Sizes. *International Journal of Production Research,* 47(18): 5109–28.

Clarke, G. and Wright, J. W. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations*

Research, 12(4): 568–81.

Damodaran, P. and Vélez-Gallego, M. C. (2012). A Simulated Annealing Algorithm to Minimize Makespan of Parallel Batch Processing Machines with Unequal Job Ready Times. *Expert Systems with Applications,* 39(1): 1451–58.

Graham, R. L., Lawler, E. L., Lenstra, J. K. and Kan, A. R. (1979). Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey. *Annals of Discrete Mathematics*, 5: 287–326.

Hashemi, S., Salari, M. and Ranjbar, M. (2020). Multi-Trip Open Vehicle Routing Problem with Time Windows: A Case Study. *International Journal of Industrial Engineering : Theory Applications and Practice*, 27(1): 37–57.

Hungerländer, P. and Truden, C. (2018). Efficient and Easy-to-Implement Mixed-Integer Linear Programs for the Traveling Salesperson Problem with Time Windows. *Transportation Research Procedia*, 30: 157–66.

Jia, Z. H. and Leung, J. Y. T. (2014). An Improved Meta-Heuristic for Makespan Minimization of a Single Batch Machine with Non-Identical Job Sizes. *Computers and Operations Research,* 46: 49–58.

Jia, Z. H., Li, K. and Leung, J. Y. T. (2015). Effective Heuristic for Makespan Minimization in Parallel Batch Machines with Non-Identical Capacities. *International Journal of Production Economics,* 169: 1–10.

Jia, Z. H., Wang, C. and Leung, J. Y. T. (2016). An ACO Algorithm for Makespan Minimization in Parallel Batch Machines with Non-Identical Job Sizes and Incompatible Job Families. *Applied Soft Computing Journal,* 38: 395–404.

Kashan, A. H., Karimi, B. and Jenabi, M. (2008). A Hybrid Genetic Heuristic for Scheduling Parallel Batch Processing Machines with Arbitrary Job Sizes. *Computers and Operations Research,* 35(4): 1084–98.

Koh, S. G., Koo, P. H., Ha, J. W. and Lee, W. S. (2004). Scheduling Parallel Batch Processing Machines with Arbitrary Job Sizes and Incompatible Job Families. *International Journal of Production Research,* 42(19): 4091–4107.

Lee, J. Y., Kim, Y. D. and Lee, T. E. (2018). Minimizing Total Tardiness On Parallel Machines Subject To Flexible Maintenance. *International Journal of Industrial Engineering : Theory Applications and Practice*, 25(4): 472–89.

Li, Y., Lim, A. and Rodrigues, B. (2004). Crossdocking - JIT Scheduling with Time Windows. *Journal of the Operational Research Society,* 55(12): 1342–51.

Malve, S. and Uzsoy, R. (2007). A Genetic Algorithm for Minimizing Maximum Lateness on Parallel Identical Batch Processing Machines with Dynamic Job Arrivals and Incompatible Job Families. *Computers and Operations Research,* 34(10): 3016–28.

Mathirajan, M. and Sivakumar, A. I. (2006). A Literature Review, Classification and Simple Meta-Analysis on Scheduling of Batch Processors in Semiconductor. *International Journal of Advanced Manufacturing Technology*, 29: 990–1001.

Mönch, L., Balasubramanian, H., Fowler, J. W. and Pfund, M. E. (2005). Heuristic Scheduling of Jobs on Parallel Batch Machines with Incompatible Job Families and Unequal Ready Times. *Computers and Operations Research*, 32(11): 2731–50.

Mönch, L., Fowler, J. W. and Mason, S. J. (2012). *Production Planning and Control for Semiconductor Wafer Fabrication Facilities: Modeling, Analysis, and Systems*. Vol. 52. Springer Science & Business Media.

Ozturk, O., Begen, M. A. and Zaric, G. S. (2014). A Branch and Bound Based Heuristic for Makespan Minimization of Washing Operations in Hospital Sterilization Services. *European Journal of Operational Research*, 239(1): 214–26.

Pamosoaji, A. K., Dewa, P. K. and Krisnanta, J. V. (2019). Proposed Modified Clarke-Wright Saving Algorithm for Capacitated Vehicle Routing Problem. *International Journal of Industrial Engineering and Engineering Management,* 1(1): 9.

Reichelt, D. and Mönch, L. (2006). Multiobjective Scheduling of Jobs with Incompatible Families on Parallel Batch Machines. *In European Conference on Evolutionary Computation in Combinatorial Optimization*, 209–21.

Segerstedt, A. (2014). A Simple Heuristic for Vehicle Routing-A Variant of Clarke and Wright's Saving Method. *International Journal of Production Economics*, 157(1): 74–79.

Shirvani, N., Ruiz, R. and Shadrokh, S. (2014). Cyclic Scheduling of Perishable Products in Parallel Machine with Release Dates, Due Dates and Deadlines. *International Journal of Production Economics,* 156:1–12.

Tarhini, A., Danach, K. and Harfouche, A. (2020). Swarm Intelligence-Based Hyper-Heuristic for the Vehicle Routing Problem with Prioritized Customers. *Annals of Operations Research*, 1–22.

Uzsoy, R. (1995). Scheduling Batch Processing Machines with Incompatible Job Families. *International Journal of Production Research*, 33(10): 2685–2708.

Zhou, S., Chen, H., Xu, R., and Li, X. (2014). Minimising Makespan on a Single Batch Processing Machine with Dynamic Job Arrivals and Non-Identical Job Sizes. *International Journal of Production Research*, 52(8): 2258–74.

Zhou, S., Xie, J., Du, N. and Pang, Y. (2018). A Random-Keys Genetic Algorithm for Scheduling Unrelated Parallel Batch Processing Machines with Different Capacities and Arbitrary Job Sizes. *Applied Mathematics and Computation*, 334: 254–68.

APPENDIX A - Notations used in MIP model

Indices	
i	lot index $i = 1, \dots, N$,
j	machine index, $j = 1, \dots, M$,
e	recipe index, $e = 1, \dots, E$,
b	batch index, $b = 1, \dots, B$;
Parameters	
p_i	processing time of lot i,
r_i	release time of lot i,
R_i	remaining lifetime of lot i,
s_i	size of lot i,
UB_e	batch capacity with recipe e,
$\boldsymbol{L}$	a very large positive number,
$h_{i,e}$	1, if lot i uses recipe e; 0, otherwise;
Decision variables	
$X_{j,b,i}$	1, if lot i is assigned to batch b on machine j; 0, otherwise,
$Y_{j,b,e}$	1, if recipe e is processed on batch b on machine j; 0, otherwise,
$S_{j,b}$	start time of batch b on machine j,
$F_{j,b}$	finish time of batch b on machine j,
E_j	end time of machine j,
C_{max}	makespan.

APPENDIX B - Additional notations used in DH algorithm:

b, l, k, h	batch index,
j, u, v	machine index,
p_b^B	processing time of batch b,
p_i^L	processing time of lot i,
re_b^B	recipe of batch b,
re_i^L	recipe of lot i,
s_b^B	size of batch b,

s_i^L	size of lot i,
r_i^L	release time of lot i,
R_i^L	remaining lifetime of lot i,
ES_b^B	earliest start time of batch b,
LS_b^B	latest start time of batch b,
$\mathcal{L}^{\mathrm{EST}}(\mathcal{L}^{\mathrm{LST}})$	sorted sequence according to non-decreasing order of batch earliest start time (batch latest start time),
$b_1^{EST}(b_1^{LST})$	first batch in sequence $\mathcal{L}^{EST}(\mathcal{L}^{\mathrm{LST}})$,
$T_j^{EST}(T_j^{LST})$	available time of machine j when applying EST rule (LST rule),
$S_j^{EST}(S_j^{LST})$	batch sequence on machine j when applying EST rule (LST rule),
BS_b^j	start time of batch b on machine j,
BC_b^j	completion time of batch b on machine j,
S_j	batch sequence on machine j.

APPENDIX C - Details of the computational results for the large-size instances

Table C. Makespan results of LB, LFLT and DH

M	E		LB	$N = 100$		LB	$N = 200$		LB	$N = 300$	
				LFLT	DH		LFLT	DH		LFLT	DH
10	**5**	1	178	219	208	341	453	413	490	616	564
		2	188	227	222	369	466	429	544	700	652
		3	163	206	196	384	490	460	555	719	672
		4	190	230	225	345	427	400	508	653	599
		5	186	223	215	362	461	419	586	775	701
		6	179	234	215	345	446	413	539	689	646
		7	179	227	210	332	412	396	506	646	601
		8	184	236	221	385	478	446	555	716	659
		9	190	241	235	369	468	431	540	701	626
		10	215	265	249	350	458	428	494	638	607
		11	170	217	203	332	422	395	550	703	653
		12	174	226	212	353	458	435	520	663	600
		13	192	243	222	303	402	366	551	706	628
		14	170	214	205	399	513	483	526	680	634
		15	198	254	247	384	491	460	558	702	634
		16	187	244	229	376	475	443	509	659	609
		17	171	226	209	373	488	459	517	655	611
		18	198	257	233	342	439	406	499	643	600
		19	192	239	212	418	538	483	505	650	602
		20	175	221	208	339	448	408	557	706	661
10	**10**	1	336	387	354	630	788	747	976	1243	1173
		2	332	397	394	606	737	701	988	1222	1126
		3	316	371	377	586	743	690	960	1208	1122
		4	317	379	347	677	845	793	948	1207	1130
		5	278	334	317	630	784	740	956	1183	1102
		6	328	399	373	619	780	704	905	1133	1051
		7	314	370	344	668	843	797	899	1112	1063
		8	368	413	403	584	766	729	939	1190	1092
		9	339	422	388	649	779	745	899	1160	1083
		10	286	321	309	638	786	715	874	1102	1006
		11	389	453	440	604	747	711	938	1171	1114
		12	295	354	344	625	780	721	952	1205	1122
		13	322	376	363	634	769	719	929	1179	1080
		14	308	368	346	651	805	761	935	1162	1100
		15	320	368	353	572	721	677	940	1181	1126
		16	346	405	407	659	802	741	944	1184	1087
		17	345	402	407	682	830	790	902	1160	1065
		18	324	382	363	656	841	775	921	1192	1088
		19	309	359	346	680	849	779	947	1177	1093
		20	316	399	380	643	786	767	954	1205	1101
10	**15**	1	578	633	622	962	1113	1057	1392	1675	1578
		2	539	616	573	970	1173	1132	1330	1644	1546
		3	572	678	633	1011	1221	1165	1325	1640	1533
		4	579	650	618	987	1202	1138	1253	1530	1464
		5	577	670	645	923	1145	1046	1419	1703	1604

		6	527	605	590	954	1142	1087	1363	1644	1556
		7	543	609	592	963	1174	1098	1251	1563	1429
		8	578	604	588	912	1057	1028	1317	1613	1537
		9	555	638	596	847	1011	944	1373	1673	1596
		10	580	675	657	948	1111	1100	1388	1727	1624
		11	600	689	677	925	1098	1047	1379	1675	1578
		12	496	595	553	974	1167	1111	1291	1621	1514
		13	526	621	586	981	1169	1119	1367	1742	1592
		14	601	687	672	916	1137	1045	1291	1615	1509
		15	568	647	621	947	1142	1095	1344	1688	1584
		16	517	616	564	931	1135	1035	1389	1739	1606
		17	600	667	655	907	1096	1022	1363	1670	1567
		18	511	570	541	931	1099	1033	1327	1664	1555
		19	517	583	572	967	1123	1033	1387	1744	1595
		20	596	673	667	1017	1231	1129	1299	1591	1522
10	**20**	1	668	767	760	1189	1396	1372	1788	2183	2030
		2	703	743	734	1303	1582	1506	1824	2138	2066
		3	637	735	713	1208	1457	1396	1629	2039	1908
		4	594	675	677	1151	1332	1289	1856	2163	2074
		5	730	761	740	1267	1549	1414	1774	2134	2008
		6	661	711	713	1114	1359	1243	1749	2158	2029
		7	680	747	720	1263	1521	1430	1830	2256	2048
		8	770	804	797	1182	1383	1320	1653	2033	1878
		9	664	723	701	1306	1514	1384	1728	2079	1940
		10	661	778	706	1238	1454	1381	1773	2169	2025
		11	668	704	692	1269	1486	1371	1719	2102	2003
		12	699	738	708	1318	1503	1468	1720	2082	1997
		13	727	769	746	1283	1489	1452	1803	2215	2047
		14	660	747	700	1119	1386	1271	1876	2269	2157
		15	614	680	636	1288	1522	1464	1765	2154	2041
		16	609	661	666	1222	1404	1400	1729	2138	2034
		17	733	780	757	1158	1391	1333	1697	2031	1926
		18	694	734	740	1153	1344	1306	1812	2181	2089
		19	651	697	686	1214	1451	1392	1860	2241	2102
		20	742	796	778	1226	1446	1376	1876	2257	2073
30	**5**	1	70	83	83	136	172	162	170	231	214
		2	80	97	92	109	144	134	165	215	191
		3	66	83	88	121	152	145	167	219	202
		4	68	90	94	126	158	152	161	212	201
		5	70	86	80	129	174	161	200	261	241
		6	61	76	76	119	156	147	191	252	232
		7	66	85	83	113	152	136	184	240	233
		8	71	85	77	127	167	152	196	254	241
		9	73	84	84	117	156	142	190	255	226
		10	68	81	80	129	167	156	186	243	228
		11	64	81	81	116	148	140	185	240	229
		12	67	76	82	133	176	164	170	226	208
		13	69	87	85	120	144	140	186	244	231
		14	61	76	76	127	160	152	185	227	213
		15	81	94	94	126	162	156	165	218	199
		16	68	84	84	130	174	157	171	225	208
		17	69	81	75	131	175	168	182	239	223
		18	65	82	82	112	149	135	179	233	217
		19	72	88	88	118	149	139	169	224	202
		20	68	76	80	126	174	160	183	243	226
30	**10**	1	113	143	143	188	245	236	300	396	367
		2	113	138	138	194	240	235	312	389	361
		3	110	138	138	205	253	243	311	392	370
		4	113	140	137	205	254	248	323	414	391
		5	123	157	150	205	273	253	299	383	377
		6	119	142	136	221	287	276	292	376	344
		7	109	138	137	211	271	253	303	391	357
		8	114	135	139	210	266	246	303	389	358
		9	110	139	139	216	281	267	327	427	392
		10	132	166	166	201	246	231	319	403	381
		11	114	148	141	201	249	243	295	377	347
		12	109	126	131	228	292	278	298	388	340
		13	118	140	140	222	284	270	293	376	367
		14	125	148	148	213	270	253	310	397	351
		15	133	158	154	235	293	285	285	360	340

		16	104	130	130	218	285	270	288	371	347
		17	105	134	141	199	255	244	309	399	367
		18	121	146	153	206	261	252	305	388	364
		19	129	166	152	220	279	268	309	401	365
		20	121	145	139	208	270	252	302	379	351
30	**15**	1	184	211	218	319	385	366	455	579	540
		2	205	246	227	315	386	360	462	574	548
		3	192	232	216	318	382	364	441	550	536
		4	167	208	207	315	384	374	450	562	521
		5	184	215	215	297	370	354	450	556	529
		6	188	214	212	338	406	392	461	581	534
		7	188	230	228	318	407	380	466	580	553
		8	210	252	264	338	412	393	457	570	547
		9	172	204	213	311	396	369	458	589	555
		10	182	214	210	312	391	376	445	559	534
		11	183	217	222	345	407	388	452	580	538
		12	175	232	206	315	390	369	437	536	521
		13	204	227	242	335	399	401	427	534	503
		14	196	220	217	299	379	356	455	589	541
		15	202	238	231	311	391	359	447	562	517
		16	196	243	236	325	425	393	437	549	523
		17	187	223	245	321	386	380	461	568	542
		18	186	219	221	327	411	400	430	537	501
		19	181	209	213	326	410	384	424	538	507
		20	185	226	223	325	386	390	448	571	527
30	**20**	1	239	279	293	417	501	480	633	784	759
		2	221	270	263	409	480	461	593	735	695
		3	214	257	243	426	508	497	579	713	670
		4	241	274	277	417	506	483	574	732	673
		5	228	278	266	448	531	495	616	761	714
		6	211	237	239	422	523	497	599	718	694
		7	239	282	280	429	536	499	634	774	735
		8	222	272	262	404	498	486	567	708	669
		9	223	258	257	365	438	431	562	697	651
		10	231	262	277	390	461	426	582	721	674
		11	220	259	259	394	495	476	608	757	736
		12	224	246	245	385	467	454	582	708	672
		13	211	243	260	384	466	450	619	740	715
		14	256	306	284	426	517	502	618	769	724
		15	268	319	314	425	513	485	616	741	695
		16	239	281	269	468	555	526	607	742	698
		17	249	281	285	431	511	495	600	769	713
		18	229	272	279	395	491	461	585	721	688
		19	251	281	307	442	543	517	599	730	696
		20	210	236	245	392	478	474	596	729	686
50	**5**	1	59	59	59	76	107	98	107	145	132
		2	56	65	56	79	106	97	109	137	135
		3	55	58	55	82	95	100	108	141	129
		4	57	57	57	76	108	92	103	141	125
		5	60	60	60	73	88	89	101	136	123
		6	54	58	54	77	104	92	112	150	135
		7	54	54	54	86	106	98	118	155	143
		8	59	59	59	86	111	106	110	146	142
		9	58	58	58	78	98	94	110	145	135
		10	54	58	54	70	96	86	116	148	142
		11	53	55	53	78	105	94	109	144	132
		12	52	52	52	78	103	93	106	144	132
		13	51	51	51	74	95	91	98	142	121
		14	52	58	52	78	102	95	115	143	142
		15	55	55	55	69	83	83	110	147	136
		16	52	57	52	83	104	99	111	148	135
		17	58	58	58	85	105	102	114	149	138
		18	50	58	50	73	89	88	119	140	139
		19	55	56	55	80	98	98	104	140	121
		20	54	63	60	78	98	98	112	149	143
50	**10**	1	109	109	109	131	169	157	197	250	232
		2	110	110	110	126	167	168	191	244	231
		3	108	108	108	135	181	165	172	224	208
		4	100	100	100	138	176	168	185	243	223
		5	103	103	103	126	169	168	202	258	247

		6	110	111	110	127	148	155	175	225	223
		7	107	107	107	128	174	157	191	250	233
		8	105	105	105	135	170	167	182	239	220
		9	108	108	108	137	177	167	203	259	243
		10	102	102	102	126	162	150	179	232	219
		11	108	108	108	129	164	156	188	252	229
		12	108	108	108	123	154	148	179	238	218
		13	103	103	103	126	158	154	186	238	221
		14	105	105	105	128	155	150	191	255	234
		15	101	101	101	128	168	154	179	223	215
		16	103	103	103	137	176	168	185	232	229
		17	108	108	108	132	175	163	182	238	225
		18	108	108	108	138	169	169	183	239	221
		19	100	100	100	128	168	157	179	239	211
		20	110	110	110	126	166	156	182	243	223
50	**15**	1	168	168	168	203	258	247	274	368	353
		2	164	164	164	194	250	231	268	340	309
		3	170	170	170	211	267	254	269	345	319
		4	169	169	169	183	223	223	263	339	322
		5	167	167	167	193	246	241	263	334	320
		6	166	166	166	176	213	212	263	353	327
		7	168	168	168	178	214	214	279	362	346
		8	167	167	167	195	253	239	282	360	327
		9	162	162	162	182	241	225	260	327	313
		10	168	168	168	194	238	239	271	356	325
		11	167	167	167	197	252	243	268	335	321
		12	164	164	168	218	262	260	283	359	336
		13	164	164	180	188	244	229	270	348	335
		14	168	168	168	182	237	236	260	332	314
		15	168	168	168	197	255	241	263	338	321
		16	168	168	175	190	243	241	304	378	366
		17	166	166	166	207	260	250	283	356	352
		18	170	170	170	181	222	214	269	352	319
		19	166	166	166	187	241	233	273	349	324
		20	166	166	166	204	256	247	239	309	296
50	**20**	1	209	209	209	261	326	296	351	433	414
		2	202	202	202	262	322	315	354	438	426
		3	200	200	200	244	309	289	348	423	404
		4	209	209	209	248	317	309	360	450	426
		5	208	208	208	209	257	264	367	456	417
		6	207	207	209	239	314	295	353	434	401
		7	205	205	205	290	342	334	331	420	400
		8	210	210	210	251	317	313	346	421	394
		9	210	210	210	248	317	307	350	445	425
		10	209	209	209	240	310	281	382	489	452
		11	200	200	200	216	284	271	362	443	428
		12	206	206	206	251	295	295	357	450	431
		13	208	208	208	252	325	312	339	427	408
		14	208	208	208	261	334	326	372	468	460
		15	207	207	207	268	323	325	368	471	444
		16	209	209	209	253	316	312	343	434	422
		17	209	209	209	256	326	316	354	449	420
		18	210	210	210	243	317	312	354	431	414
		19	207	207	207	225	294	280	337	426	393
		20	205	205	205	258	327	309	372	462	437