Scheduling hybrid flowshop with parallel batching machines and compatibilities.

Adrien Bellanger, Ammar Oulamara

Scheduling hybrid flowshop with parallel batching machines and compatibilities

A. Bellanger, A. Oulamara

LORIA - INRIA Nancy Grand Est, Nancy université - INPL, Ecole des Mines de Nancy, Parc de Saurupt, 54042 Nancy, France.

December 7, 2007

Abstract

This paper considers a two-stage hybrid flowshop problem in which the first stage contains several identical discrete machines, and the second stage contains several identical batching machines. Each discrete machine can process no more than one task at time, and each batching machine can process several tasks simultaneously in a batch with the additional feature that the tasks of the same batch have to be compatible. A compatibility relation is defined between each pair of tasks, so that an undirected compatibility graph is obtained which turns out to be an interval graph. The batch processing time is equal to the maximal processing time of the tasks in this batch, and all tasks of the same batch start and finish together. The goal is to make batching and sequencing decisions in order to minimize the makespan. Since the problem is NP-hard, we develop several heuristics along with their worst cases analysis. We also consider the case in which tasks have the same processing time on the first stage, for which a polynomial approximation scheme (PTAS) algorithm is presented.

Keywords. Hybrid flowshop problem, batch processing machines, task compatibilities.

1 Introduction

In this paper we consider a two-stage hybrid flowshop scheduling problem, where each of n tasks is to be processed first at stage one and then at stage two. The first stage contains several identical discrete machines and the second stage contains several identical batching machines. Each discrete machine can process no more than one task at time, and each batching machine can process up to k tasks simultaneously in a batch. The batch processing time is equal to the maximal processing time of the tasks in this batch, and all tasks of the same batch start and finish together. The processing of task j on any machine of stage one requires p_j time units, and on any machine of stage two the processing time q_j lies in closed interval $[a_j, b_j]$. The terms initial and terminal endpoints will refer to a_j and b_j, respectively. On the second stage the tasks are processed in batches. A batch is a set of tasks with the additional constraint that the tasks of the same batch have to be compatible. A compatibility is a symmetric binary relation in which a pair (i, j) of tasks is compatible if they share a similar processing time on the second machine (i.e, $[a_i, b_i] \cap [a_j, b_j] \neq \emptyset$), so an undirected *compatibility graph* $G = (V, E)$

-which is an interval graph- is defined, where V is the set of tasks and a pair of tasks is an element of the edge set E if and only if they are compatible. The batch processing time on the batching machine is determined as the maximum initial endpoint a_j of compatible tasks. For each batch B, denote its processing time on the second machine as $q(B) = \max\{a_j \mid j \in B\}$. For a given schedule, task completion times C_j, $j = 1, \ldots, n$ can be calculated. If task j belongs to batch B on the second stage, then $C_j = C_2(B)$, where $C_2(B)$ is the time at which a machine on stage two finishes the processing of batch B. The objective is to find a schedule which minimizing the completion time of the latest batch $C_{max} = \max\{C_j \mid j = 1, \ldots, n\}$. Following the standard scheduling notation, see Graham et al.[8], we denote the problem formulated above as $FH2Bm_1, m_2|p - batch(II), G_p = INT, k < n|C_{max}$, where m_1 and m_2 represent the number of machines on first and second stage, respectively, $p - batch(II)$ means that the stage two is composed with parallel batching machines, $G_p = INT$ specifies that the compatibility graph is an interval graph, which depends on the processing times of tasks, and $k < n$ specifies that the capacity of batching machines is a variable and it is part of the input.

This problem is motivated by the scheduling of tire in the manufacturing industry. Indeed, making tires involves several steps, and requires a high degree of precision and control at each step of the process. This process begins by producing the gum-like material that will be milled and split into strips that become the sidewalls after adding steel and fabric cords to reinforce the rubber compound and provide strength. When all parts involved to make the tires are prepared, the main and difficult task can be started. A typical tire is built in a two-stage process. In the first stage (tire building), all components (sidewalls and tread) are assembled and radial tires are built on a round drum, which is a part of the tire building machine. The machine pre-shapes the tire into a form that is very close to the tire's final dimensions. The end result is called a *green tire* or *uncured tire*. In the second stage (tire curing), curing occurs through a series of chemical reactions. Tire curing is a high-temperature and high-pressure batch operation in which a pair of uncured tires is placed into a mold at a specified temperature. Each type of tire must be cured for a total duration in the interval of its total curing duration and total curing duration plus 4% of this value. Two kinds of tires can be cured together if they share a same value of total curing duration. After the mold is closed, the rubber compound flows into mold the shape and form the tread details and sidewall. The mold cannot be opened until the curing reaction is completed for both green tires on the same mold. As the rubber is a perishable material, the objective is to produce tires as soon as possible, which is equivalent to minimizing the completion time of the last product.

In the scheduling literature, intensive research involving a single batching machine without task compatibilities are studied for various objective functions and additional constraints. The main comprehensive study is that of Brucker et al. [4]. Potts and Van Wassenhove [23], Potts and Kovalyov [21] have published state-of-the-art surveys of the batch scheduling problem. A single batching machine with task compatibilities has been studied in Finke et al. [6] for general graph and also for some special graphs. Boudhar and Finke [3] consider the problem of minimizing makespan in general compatibility graph. They show that the problem with capacity $k = 2$ of the batching machine is solvable in polynomial time as a weighted matching problem. For $3 \leq k < n$ minimizing the makespan is NP-hard, even for split graph.

For flowshop system, Potts et al. [22] studied the flowshop problem with two batching machines. They offered a polynomial algorithm for the problem of minimizing batch completion time, when both batching machines can process an unbounded number of tasks in the same batch, and proved the NP-hardness of this problem when at least one of the machines can

process up to k tasks in the same batch $k < n$. Oulamara et al. [19] consider the two-machine flowshop problem where the first machine is a discrete machine and the second is a batching machine. The objective is to minimize the makespan. The tasks in the same batch have to be compatible. One of their result is used in our study. Given a batching machine with capacity k and each task j has processing time given as an interval $[a_j, b_j]$. Minimizing the total batch processing time is solvable in $O(nlogn)$ by listing the intervals in nonincreasing order of their initial endpoint a_j and forming the first batch from the first k compatible tasks, forming the second batch with the next k compatible tasks and so on. We call this rule, Full-Compatible-Batch-Largest-Processing-Time (FCBLPT). Oulamara et al. [20] studied the no-wait flowshop problem with two batching machines, and proposed a polynomial algorithm for the above problem, though they also extended their studies to the case of m batching machines. Other related results for the case of flowshop batching machine can be found in [17], [18].

Although extensive research has been carried out on flowshops with batching machines, to the best of our knowledge, the problem of hybrid flowshop involving batching machines and task compatibilities has not been considered before. The case of classical hybrid flowshop is extensively considered in the literature, Lee and Vairaktarakis [15], Gupta et al. [10], Guinet et al. [9], Allaoui and Artiba [2], Haouari and M'Hallah [12].

The reminder of this paper is organized as follows. In section 2, we introduce the notation used in this paper. In section 3 we establish lower bounds, heuristics with computational experiments for the considered problem. Section 5 presents a PTAS algorithm scheme in the case of uniform processing time on the first stage.

2 Notations

The following notations are used throughout the paper:

- m_1 (m_2) : The number of machines on the first (second) stage.
- $m = \max\{m_1;\ m_2\}$.
- p_j : The processing time of task j on any machine of the first stage.
- q_j : The processing time of task j on any machine of the second stage, the value of q_j lies within interval $[a_j, b_j]$.
- k : The batch capacity of machines of the second stage, $2 \leqslant k < n$.
- $q(B_l)$: The processing time of batch B_l on the second stage, $q(B_l) = \max\{a_j | j \in B_l\}$.

The example below illustrates the considered scheduling problem, with $n = 8$, $k = 2$. The processing times are given in Table 1, whereas Figure 1 shows the compatibility graph. A feasible schedule $S = \langle (T_1, T_8), (T_6, T_7), (T_3, T_5), (T_2, T_4) \rangle$ contains four batches, its Gantt chart is given in Figure 2.

[Insert Figure 1 about here]

[Insert Figure 2 about here]

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8
$p_{1,j}$	4	7	5	6	8	6	10	4
$p_{2,j}$	[5,15]	[3,6]	[7,10]	[3,11]	[9,12]	[11,16]	[15,18]	[14,19]

Table 1: Task processing times.

3 Several machines on each stage

In this section we consider the general case in which each stage contains several machines, denoted by $FH2B(m_1, m_2)$. We start by presenting three lower bounds and three heuristics along with their worst case analysis.

3.1 Lower bounds

In the following we derive three lower bounds to be used to evaluate the performance of approximate heuristics. The first and the second are machine-based bounds, whereas the third is job-based bound. Let C_{max} be the minimum makespan.

First, observe that all the processing on the first stage cannot be completed before time $\frac{\sum_i p_i}{m_1}$. Therefore,

$$C_{max} \geq LB_1 = \frac{\sum_{i=1}^{n} p_i}{m_1} + \min_{1 \leq j \leq n} a_j.$$

On the other hand, the processing at the second stage cannot be completed before the minimal load of that stage, i.e. the total processing on the second stage is greather than $C_{max}(B)$ divided by m_2, where $C_{max}(B)$ is obtained by the FBCLPT algorithm. Therefore,

$$C_{max} \geq LB_2 = \min_{1 \leq i \leq n} p_i + \frac{C_{max}(B)}{m_2}.$$

The last lower bound is based on the largest processing time, i.e.

$$C_{max} \geq LB_3 = \max_{1 \leq i \leq n} \{p_i + a_i\}.$$

3.2 Heuristics with worst-case performance

In this section, we investigate the worst-case analysis of three heuristic algorithms. These heuristics are based on the combination of well known rules for machine scheduling.

Before presenting these heuristics, we show the efficiency of the H-FCBLPT heuristic on scheduling of parallel batching machines $(PB(m)|G = INT, k < n|C_{max})$ that would be useful in the rest of the paper.

Algorithm H-FCBLPT

1. Apply FCBLPT rule to construct a list of batches.
2. Order the batches in nonincreasing order of their processing time and assign them to the machines as they become available.

Lemma 1 *The H-FCBLPT rule approximate the problem* $PB(m)|G_p = INT, k < n|C_{max}$ *with ratio of* $(\frac{4}{3} - \frac{1}{3m})$.

Proof. The H-FCBLPT heuristic starts by constructing a list of batches with minimal total processing times, Oulamara et al. [19], and it order the batches in nonincreasing order of their processing times and assigns them to the machines as they become available. In fact, the second step of H-FCBLPT heuristic applies the LPT rule to the $P||C_{max}$ problem obtained by aggregating the jobs in the batches, then the ratio $(\frac{4}{3} - \frac{1}{3m})$ of Graham [7]. □

In the rest of this section, we present three heuristics and their worst-case analysis when each stage contains several machines.

Heuristic H_{LPT}

Heuristic H_{LPT} schedules tasks at each stage independently, in a way that tasks are scheduled on the first stage following LPT rule, and then batched using the FCBLPT rule. Note that batches are carried out on the second stage in the LPT order.

Algorithm H_{LPT}

1. Reindex tasks in a LPT order and let L be the list of tasks. On the first stage, schedule each task of L in the order in which they appear in the list on the first available machine. Let t be the completion time of all tasks.
2. Apply FCBLPT rule to construct a list of batches and denote by L_B the obtained list.
3. Let B be the first batch of list L_B. Schedule a batch B on stage two on the first available machine starting from time t. Remove B from L_B. If $L_B = \emptyset$, then return the schedule, else go to step 3.

Theorem 1 *The heuristic* H_{LPT} *gives a schedule* $S_{H_{LPT}}$ *in* $O(n\ log\ n)$ *with a performance guarantee of* $\frac{8}{3} - \frac{2}{3m}$, *and this bound is tight.*

Proof. Let $S_{H_{LPT}}$ be the schedule produced by heuristic H_{LPT}. Let us consider the two stages independently. At the first stage, the LPT rule gives an approximation ratio of $(\frac{4}{3} - \frac{1}{3m_1})$. At the second stage, one has a scheduling problem with parallel batching machines $(PB(m)|G_p = INT, k < n|C_{max})$, which is approximated with a ratio of $(\frac{4}{3} - \frac{1}{3m_2})$ by FCBLPT heuristic (Lemma 1). Therefore,

$$\begin{aligned} C_{max}(S_{H_{LPT}}) &\leqslant C_{max}(S_{H_{Pm_1}}) + C_{max}(S_{H_{Bm_2}}) \\ &\leqslant \left(\tfrac{4}{3} - \tfrac{1}{3m_1}\right) C_{max}(S^*_{Pm_1}) + \left(\tfrac{4}{3} - \tfrac{1}{3m_2}\right) C_{max}(S^*_{Bm_2}). \end{aligned}$$

Let S^* be the optimal schedule for hybrid flowshop problem, and $C_{max}(S^*)$ its makespan, then

$$C_{max}(S^*) \geqslant C_{max}(S^*_{Pm_1}) \qquad and \qquad C_{max}(S^*) \geqslant C_{max}(S^*_{Bm_2}).$$

Hence,

$$\begin{aligned} C_{max}(S_{H_{LPT}}) &\leqslant \left(\tfrac{4}{3} - \tfrac{1}{3m_1}\right) C_{max}(S^*_{H_{Pm_1}}) + \left(\tfrac{4}{3} - \tfrac{1}{3m_2}\right) C_{max}(S^*_{Bm_2}) \\ &\leqslant \left(\tfrac{8}{3} - \tfrac{2}{3m}\right) C_{max}(S^*). \end{aligned}$$

Type	Tasks number	p_j	$[a_j, b_j]$
A_1	2	$2m-1$	$[\epsilon; 2m-1]$
A_2	2	$2m-2$	$[\epsilon; 2m-1]$
...			
A_{m-1}	2	$m+1$	$[\epsilon; 2m-1]$
A_m	3	m	$[\epsilon; 2m-1]$
B_1	2	ϵ	$[2m-1; 2m-1]$ and $[2m-1-\epsilon; 2m-1-\epsilon]$
B_2	2	ϵ	$[2m-2; 2m-2]$ and $[2m-2-\epsilon; 2m-2-\epsilon]$
...			
B_{m-1}	2	ϵ	$[m+1; m+1]$ and $[m+1-\epsilon; m+1-\epsilon]$
B_m	3	ϵ	$[m; m]$, $[m-\epsilon; m-\epsilon]$ and $[m-2\epsilon; m-2\epsilon]$
C	nb_C	ϵ	$[\epsilon; \epsilon]$

Table 2: Task processing times.

Therefore,

$$\frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} \leqslant \frac{8}{3} - \frac{2}{3m}.$$

Thus, the heuristic H_{LPT} provides a solution with a performance ratio bounded by $\frac{8}{3} - \frac{2}{3m}$. Now we present an instance for which this upper bound is reached. Denote by I an instance for the hybrid flowshop problem with m machines on stage one and two, composed of classes A, B and C containing $2m+1$, $2m+1$ and nb_C tasks, respectively. The processing times are given in Table 2 where $\epsilon < \min\{\frac{m}{3}; \frac{(m-1)^2}{2m+1}\}$. The capacity of batching machines is equal to $2m+1$. Without loss of generality, we assume that m is an even number.

[Insert Figure 3 about here]

[Insert Figure 4 about here]

The makespan $C_{max}(S_{H_{LPT}})$ for the schedule $S_{H_{LPT}}$, produced by the heuristic H_{LPT}, is equal to $C_{max}(S_{H_{LPT}}) = 8m - 2 - 3\epsilon$ (see Figure 3). While the optimal value of the makespan for the instance I is $C_{max}(S^*) = 3m + 2\epsilon$ (see Figure 4). $C_{max}(S^*)$ is optimal as it reachs the lower bound $LB_1 = \frac{\sum p_i}{m} + \min_j a_j$. Therefore,

$$\frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{8m - 2 - 3\epsilon}{3m + 2\epsilon}.$$

Hence

$$\lim_{\epsilon \to 0} \frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{8}{3} - \frac{2}{3m}.$$

If $nb_C > 0$, when ϵ tend toward 0, then the tasks of type C are insignificant. Thus the result. □

Heuristic H_{LBPT}

This heuristic starts by constructing a list L_B of batches using the FCBLPT rule. Then the tasks of each batch are scheduled at the Stage 1 according to LPT order. At Stage 2, batches are scheduled as soon as possible.

Algorithm H_{LBPT}

1. Apply the FCBLPT rule on the list of tasks, and let L_B be the list of batches, $i = 1$.
2. Let B be the first batch of L_B, and $L_B(B)$ be the list of tasks of B. Schedule tasks $L_B(B)$ at Stage 1 according to the LPT rule. Next, schedule the batch B on Stage 2 on the first available machine when all its tasks of B are completed at Stage 1. Remove B from L. If $L = \emptyset$, then return the schedule, otherwise go to step 2.

Theorem 2 *The heuristic* H_{LBPT} *gives a schedule* $S_{H_{LBPT}}$ *in* $O(nlogn)$ *with a performance guarantee of* $\frac{10}{3} - \frac{4}{3m}$.

Proof. Let $S_{H_{LBPT}}$ be the schedule produced by heuristic H_{LBPT}. Consider the two stages independently. The second stage is a parallel batching machines problem $(PB(m)|G_p = INT, k < n|C_{max})$. The FCBLPT algorithm generates a solution within $(\frac{4}{3} - \frac{1}{3m_2})$ of the optimal solution. The first stage is a classical parallel machines problem. It is known that the list scheduling gives a solution within $(2 - \frac{1}{m_1})$ of the optimal solution. Therefore,

$$\begin{aligned} C_{max}(S_{H_{LBPT}}) &\leqslant C_{max}(S_{H_{Pm_1}}) + C_{max}(S_{H_{Bm_2}}) \\ &\leqslant \left(2 - \tfrac{1}{m_1}\right) C_{max}(S^*_{Pm_1}) + \left(\tfrac{4}{3} - \tfrac{1}{3m_2}\right) C_{max}(S^*_{Bm_2}). \end{aligned}$$

Let S^* be the optimal schedule for the hybrid flowshop problem, and $C_{max}(S^*)$ its makespan. Then

$$C_{max}(S^*) \geqslant C_{max}(S^*_{Pm_1}) \qquad and \qquad C_{max}(S^*) \geqslant C_{max}(S^*_{Bm_2}).$$

Hence

$$\begin{aligned} C_{max}(S_{H_{LBPT}}) &\leqslant \left(2 - \tfrac{1}{m_1}\right) C_{max}(S^*_{Pm_1}) + \left(\tfrac{4}{3} - \tfrac{1}{3m_2}\right) C_{max}(S^*_{Bm_2}) \\ &\leqslant \left(\tfrac{10}{3} - \tfrac{4}{3m}\right) C_{max}(S^*). \end{aligned}$$

The tightness of this bound is an open question. □

Heuristic H_J

This heuristic is based on Johnson algorithm. It starts by constructing a list of batches using the FBCLPT rule, and reindex batches according to Johnson rule [13]. The batches are considered as tasks, and the processing time of each batch B is equal to $\frac{\sum_{j \in B} p_j}{m_1}$ and $\frac{Q(B)}{m_2}$ on Stage 1 and 2, respectively.

Algorithm H_J

1. Apply the FCBLPT rule on the list of tasks, and let L_B be the reindexed list of batches obtained by applying Johnson algorithm.

2. Denote B the first batch of L_B, and $L_B(B)$ be the list of tasks of B. Schedule tasks $L_B(B)$ at stage one according to the LPT rule. Then schedule batch B on the second stage on the first available machine, when all tasks of B are completed on the first stage. Remove B from L. If $L = \emptyset$, then return the schedule, otherwise go to step 2.

Theorem 3 *The heuristic H_J produces a schedule S_{H_J} in $O(nlogn)$ with a performance guarantee of $4 - \frac{2}{m}$.*

Proof. Let S_{H_J} be the schedule produced by heuristic H_J. Consider the two stages independently. The second stage is a parallel batching machines problem $(Bm|G_p = INT, k < n|C_{max})$, it is approximated with a ratio of $(2-\frac{1}{m})$ by list scheduling. At Stage 1, one uses a list scheduling to schedule tasks, which gives a ratio of $(2 - \frac{1}{m})$ to makespan. Therefore,

$$\begin{aligned} C_{max}(S_{H_J}) &\leqslant C_{max}(S_{H_{Pm}}) + C_{max}(S_{H_{Bm}}) \\ &\leqslant \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Pm}) + \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Bm}). \end{aligned}$$

Let S^* be the optimal schedule for the hybrid flowshop problem, and $C_{max}(S^*)$ its makespan. Then,

$$C_{max}(S^*) \geqslant C_{max}(S^*_{Pm}) \qquad and \qquad C_{max}(S^*) \geqslant C_{max}(S^*_{Bm}).$$

So,

$$\begin{aligned} C_{max}(S_{H_J}) &\leqslant \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Pm}) + \left(2 - \tfrac{1}{m}\right) C_{max}(S^*_{Bm}) \\ &\leqslant \left(4 - \tfrac{2}{m}\right) C_{max}(S^*). \end{aligned}$$

□

4 One machine at the second stage

In this section we consider the case in which the second stage contains only one batching machine. We denote this problem as $FH2B(m, 1)$. We present two heuristics with a performance guarantee.

Heuristic H_{LPT}

This heuristic schedules tasks at each stage independently, i.e. tasks are scheduled on the first stage according to the LPT rule. Then, on Stage 2, tasks are batched using FCBLPT rule, and batches are scheduled on the batching machine as soon as possible.

Algorithm H_{LPT}

1. Reindex tasks in LPT order and let L be the list of tasks. At the first stage, schedule each task of L in the order they appear in the list on the first available machine.

2. Apply FCBLPT algorithm, and let L_B be the list of the produced batches.

3. Schedule the first batch B of L_B on the batching machine as soon as possible, i.e. when all tasks of B are completed on Stage 1. Remove B from L_B. If $L_B = \emptyset$, then return the schedule, otherwise go to step 3.

Theorem 4 *The heuristic $S_{H_{LPT}}$ gives a schedule $S_{H_{LPT}}$ in $O(nlogn)$ with a performance guarantee of $(\frac{7}{3} - \frac{1}{3m})$ and this bound is tight.*

Proof. The first stage is classical parallel machine problem. It is known that the list scheduling generates a solution within $(\frac{4}{3} - \frac{1}{3m})$ of the optimal solution. At the second stage, we have a problem of one batching machine $(B|G_p = INT, k < n|C_{max})$, which is solved by FCBLPT rule ([19]). Using the same technical aspect as in the proof of theorem 1, we obtain the ratio $(\frac{7}{3} - \frac{1}{3m})$.

To show that this bound is tight, consider the following instance I of the problem with the capacity of the batching machine equal to $2m + 2$. The list of tasks and their processing times are given in Table 4. Without loss of generality, we consider an even number of machines.

Type	Tasks number	p_1	p_2
A_1	2	$2m-1$	$[\epsilon; 3m]$
A_2	2	$2m-2$	$[\epsilon; 3m]$
	...		
A_{m-1}	2	$m+1$	$[\epsilon; 3m]$
A_m	3	m	$[\epsilon; 3m]$
B	1	ϵ	$[3m; 3m]$
C	nb_C	ϵ	$[\epsilon; \epsilon]$

Table 3: Task processing times.

The makespan $C_{max}(S_{H_{LPT}})$ for schedule $S_{H_{LPT}}$ produced by heuristic H_{LPT} is equal to $C_{max}(S_{H_{LPT}}) = 7m-1$, Figure 5. The optimal value of the makespan is $C_{max}(S^*) = 3m+2\epsilon$, Figure 6. $C_{max}(S^*)$ is optimal it reaches the lower bound $LB_1 = \frac{\sum p_{1,i}}{m} + \min_j a_j$.

[Insert Figure 5 about here]

[Insert Figure 6 about here]

Therefore,

$$\frac{C_{max}(S_{H_{LPT}})}{C_{max}(S^*)} = \frac{7m-1}{3m+2\epsilon},$$

and

$$\lim_{\epsilon \to 0} \frac{7m-1}{3m+2\epsilon} = \frac{7}{3} - \frac{1}{3m}.$$

□

Heuristic H_J

This heuristic uses Johnson algorithm. It starts by applying FCBLPT rule to construct a list of batches, then a Johnson order [13] is used to sequence these batches.
Algorithm H_J

1. Apply FCBLPT rule to produce a list L of batches.

2. Reindex batches following Johnson order, where the processing time of each batch B is equal to $\frac{\sum_{j\in B} p_j}{m_1}$ and $\frac{Q(B)}{m_2}$ on Stages 1 and 2, respectively.

3. Let B be the first batch of L, and L_B be the list of tasks of B. At the first stage, schedule tasks L_B in LPT order on the first available machine. Then, schedule a batch B on the batching machine as soon as possible. Remove B from L. If $L = \emptyset$, then return the schedule, otherwise go to step 3.

Theorem 5 *Heuristic H_J generates a schedules S_{H_J} in $O(nlogn)$ with a performance guarantee of $(3 - \frac{1}{m})$, and this bound is tight.*

Proof. At the first stage, we have a parallel machines problem, where a list algorithm is used to schedule tasks within$(2 - \frac{1}{m})$ of the optimal solution. On stage 2, the FBCLPT rule produces an optimal solution. Therefore, heuristic H_J produces a schedule with ratio performance of $(3 - \frac{1}{m})$.

To show the tightness of this bound, consider the following instance I of problem with the capacity of the batching machine equal to $2m$. The list of tasks and their processing times are given in Table 4. Let L be an integer.

Type	Tasks number	p_1	p_2
A	$m-1$	$(m-1)L$	$[\epsilon; mL]$
B	$m-1$	L	$[\epsilon; mL]$
C	1	mL	$[\epsilon; mL]$
D	1	ϵ	$[mL; mL]$
E	nb_E	ϵ	$[\epsilon/2; \epsilon/2]$

Table 4: Task processing times.

The makespan $C_{max}(S_{H_J})$ for schedule S_{H_J} produced by heuristic H_J is equal to $C_{max}(S_{H_J}) = (3m-1)L$, Figure 7. The optimal value of the makespan is $C_{max}(S^*) = mL + 2\epsilon$, Figure 6. $C_{max}(S^*)$ is optimal as it reaches the lower bound $LB_1 = \frac{\sum p_{1,i}}{m} + \min_j a_j$.

[Insert Figure 7 about here]

[Insert Figure 8 about here]

Therefore,

$$\frac{C_{max}(S_{H_J})}{C_{max}(S^*)} = \frac{3m-1}{m+2\epsilon}.$$

And

$$\lim_{\epsilon\to 0} \frac{(3m-1)L}{mL+2\epsilon} = 3 - \frac{1}{m}.$$

This completes the proof.
□

5 One machine at the first stage

In this section we consider the case in which the first stage contains one machine and the second stage contains m parallel batching machines. We denote this problem as $FH2B(1, m)$. We present two heuristics along with their performance guarantee.

Heuristic H_{LBPT}

This heurstic starts by creating a list L of batches using FCBLPT rule. Then the tasks of each batch are scheduled on Stage 1, and on Stage 2, each batch is scheduled on the first available machine.

Algorithm H_{LBPT}

1. Apply FCBLPT rule and let L be the produced list of batches.
2. Let B be the first batch of L. Sequence tasks of B on the first machine. After their completion, schedule batch B at the second stage on the first available machine. Remove B from L, if $L = \emptyset$, then return the schedule, otherwise go to step 2.

Theorem 6 *Heuristic* H_{LBPT} *produces a schedule* $S_{H_{LBPT}}$ *in* $O(nlogn)$ *with a performance guarantee of* $(\frac{7}{3} - \frac{1}{3m})$*, and this bound is tight.*

Proof. Since at the second stage, the LPT order gives a $(\frac{4}{3} - \frac{1}{3m})$ approximation ratio for parallel batching machines problem then, including the first stage, we obtain the ratio of $(\frac{7}{3} - \frac{1}{3m})$.

To show that this bound is tight, consider the following instance I of problem with the capacity of the batching machines equal to 2. The list of tasks and their processing times are given in Table 5, and an even number of machines at second stage.

Type	Tasks number	p_1	p_2
A_1	2	ϵ	$[2m-1; 2m-1]$ et $[2m-1-\epsilon; 2m-1-\epsilon]$
A_2	2	ϵ	$[2m-2, 2m-2]$ et $[2m-2-\epsilon; 2m-2-\epsilon]$
		...	
A_{m-1}	2	ϵ	$[m+1; m+1]$ et $[m+1-\epsilon; m+1-\epsilon]$
A_m	3	ϵ	$[m; m]$, $[m-\epsilon; m-\epsilon]$ et $[m-2\epsilon; m-2\epsilon]$
B	1	$3m$	$[\epsilon; 2m]$
C	nb_C	ϵ	$[\epsilon; \epsilon]$

Table 5: Task processing times.

The makespan $C_{max}(S_{H_{LBPT}})$ for schedule $S_{H_{LBPT}}$ produced by heuristic H_{LBPT} is equal to $C_{max}(S_{H_{LBPT}}) = 7m - 1 - 2\epsilon$, Figure 9. The optimal value of the makespan is $C_{max}(S^*) = (2m + 2)\epsilon + 3m$, Figure 10. $C_{max}(S^*)$ is optimal since it reaches the lower bound $LB_1 = \sum p_{1,i} + \min_j a_j$.

[Insert Figure 9 about here]

[Insert Figure 10 about here]

Therefore,

$$\frac{C_{max}(S_{H_{LBPT}})}{C_{max}(S^*)} = \frac{7m-1-2\epsilon}{3m+(2m+2)\epsilon},$$

and

$$\lim_{\epsilon \to 0} \frac{7m-1-2\epsilon}{3m+(2m+2)\epsilon} = \frac{7}{3} - \frac{1}{3m}$$

This completes the proof. □

Heuristic H_J

This heuristic uses Johnson algorithm. It starts by applying FCBLPT algorithm to construct a list of batches. Then, a Johnson order is used to sequence these batches. The processing time of each batch B is equal to $\sum_{j \in B} p_j$ and $\frac{Q(B)}{m_2}$ on Stages 1 and 2, respectively.

Algorithm H_J

1. Apply FCBLPT rule to produce a list L of batches.
2. Reindex the batches according to Johnson order. The processing time of each batch B is equal to $\sum_{j \in B} p_j$ and $\frac{Q(B)}{m_2}$ on Stages 1 and 2, respectively.
3. Let B be the first batch of L, and L_B be the list of tasks of batch B. Sequence tasks of B on the first machine. After their completion, schedule the batch B on the first available machine. Remove B from L. If $L = \emptyset$, then return the schedule, otherwise go to 3.

Theorem 7 *Heuristic* H_J *gives a schedule* S_{H_J} *in* $O(nlogn)$ *with a performance guarantee of* $(3 - \frac{1}{m})$.

Proof. The proof is similar to that of theorem 6. Instead of an approximation ratio of $(\frac{4}{3} - \frac{1}{3m})$ obtained by the LPT order for parallel batching machines, we use here a list order that results a performance ratio of $(2 - \frac{1}{m})$ for parallel batching machines. Thus we obtain a performance guarantee ratio of $(3 - \frac{1}{m})$. □

6 Computational experiments

In this section we mesure the efficiency of the heuristics as follows by calculating the relative distance between the solution given by the heuristic and the best lower bound:

$$D^H_{moy} = \frac{C_{max}(H) - LB}{LB} \times 100$$

.

Experiments were done on two sets of 100 and 250 tasks, respectively. For each set of tasks, we have done two series of computational experiments. With regard to the first experiment, the initial endpoint a_j of the interval processing time of tasks on the batching machine are generated from a uniform distribution $[5, 100]$. In the second experiments, they are generated from a uniform distribution $[50, 500]$. The terminal endpoints b_j for the task j are given by $b_j = a_j + a_j \times \alpha$ where $\alpha \in \{0.05, 0.15, 0.25\}$. For both sets, the processing times of tasks on Stage 1 are generated from a uniform distribution $[5, 100]$. We tested instances with 3 different

couples of number of machines at each stage: the first case contains 1 machine on Stage 1 and 10 on Stage 2, denoted as 1-10. With the same notation, cases 10-1 and 10-10 denote the second and the third couple, respectively. For all instances, the capacity of batching machine, k, is equal to two. The comparative results are summarized in Tables 6 and 7.

$p_{2i} \in$	$m_1 - m_2$	α	H_{LPT}		H_{LPT_a}		H_{LPBT}		H_J	
			moy	worst	moy	worst	moy	worst	moy	worst
$[5, 100]$	1-10	0.05	2.962	4.007	0.244	0.273	0.244	0.273	0.244	0.273
		0.15	2.747	3.786	0.249	0.284	0.249	0.284	0.249	0.284
		0.25	2.434	3.481	0.253	0.283	0.253	0.283	0.253	0.283
	10-1	0.05	15.693	19.783	4.683	7.795	1.739	2.765	0.233	0.701
		0.15	14.552	18.804	6.103	9.187	1.848	3.316	0.255	0.613
		0.25	15.152	19.154	5.918	8.399	1.965	2.990	0.287	0.755
	10-10	0.05	40.713	53.332	-	-	8.326	12.101	8.338	11.933
		0.15	39.363	48.178	-	-	8.790	12.255	8.690	11.928
		0.25	38.446	50.842	-	-	7.559	9.760	7.480	9.955
$[50, 500]$	1-10	0.05	14.337	17.475	0.264	0.323	0.264	0.323	0.264	0.323
		0.15	14.105	18.844	0.282	0.435	0.282	0.435	0.282	0.435
		0.25	13.115	17.538	0.298	0.466	0.298	0.466	0.298	0.466
	10-1	0.05	2.879	4.075	0.791	1.405	0.408	0.703	0.060	0.176
		0.15	2.382	3.272	1.181	2.207	0.421	0.699	0.077	0.300
		0.25	1.860	3.244	1.393	1.950	0.494	0.779	0.067	0.177
	10-10	0.05	35.627	43.105	-	-	7.516	9.135	10.779	16.966
		0.15	36.412	44.111	-	-	8.037	9.570	12.528	18.478
		0.25	37.769	44.737	-	-	8.143	10.127	10.637	19.971

Table 6: Experiments on D_{moy}, for n=100

For the parameter setting in Tables 12 and 13, we can see that heuristic H_{LPT} is outperformed by heuristics H_{LPBT} and H_J, even when one of each stage contains only one machine. Heursitics H_{LPBT} and H_J generate similar results with a minimal advantage to H_J. This may be due to the fact that in H_J the flowshop environment is better taken into count.

7 Constant processing times at stage one

In this section, we analyze the case where each stage contains several machines (m_1 parallel machines on Stage 1 and m_2 parallel machines on Stage 2). The processing times of tasks are constant on Stage 1, and their interval processing times on Stage 2 are uniform, i.e., for all tasks i and j if $a_i \leq a_j$, then $b_i \leq b_j$. The uniform interval processing times on batching machines is a real assumption, as in the tire manufacturing (see section 1).

This scheduling problem can be considered as a problem of minimizing the makespan with generalized release dates on parallel batching machines. The processing tasks on Stage 1 are considered as generalized release dates. A generalized release date $\overline{r}_i$ means that at time $\overline{r}_i$ one has α_i available tasks. In our case, at release date $\overline{r}_1$, one has m_1 available tasks, i.e., m_1 tasks are completed on Stage 1, and available for processing on Stage 2. At $\overline{r}_2$ one has $2m_1$ available tasks, and so on. This probem is denoted by $PB(m_2)|\overline{r}, G_p = INT, k < n|C_{max}$ and $PB(\overline{r})$ for

$p_{2i} \in$	$m_1 - m_2$	α	H_{LPT}		H_{LPT_a}		H_{LPBT}		H_J	
			moy	worst	moy	worst	moy	worst	moy	worst
$[5, 100]$	1-10	0.05	3.862	4.242	0.096	0.102	0.096	0.102	0.096	0.102
		0.15	3.615	4.431	0.097	0.101	0.097	0.101	0.097	0.101
		0.25	3.632	4.402	0.097	0.102	0.097	0.102	0.097	0.102
	10-1	0.05	17.346	19.327	4.507	5.922	0.940	1.244	0.064	0.277
		0.15	17.462	20.333	4.540	6.407	0.791	1.269	0.066	0.190
		0.25	17.396	19.242	4.802	6.170	0.862	1.234	0.083	0.185
	10-10	0.05	43.802	48.198	-	-	3.230	5.270	3.250	5.391
		0.15	42.032	46.273	-	-	3.381	4.370	3.395	4.242
		0.25	42.600	46.323	-	-	3.162	4.496	3.166	4.560
$[50, 500]$	1-10	0.05	16.885	19.006	0.103	0.122	0.103	0.122	0.103	0.122
		0.15	16.676	18.983	0.106	0.174	0.106	0.174	0.106	0.174
		0.25	16.799	18.556	0.109	0.170	0.109	0.170	0.109	0.170
	10-1	0.05	3.199	4.249	0.626	1.004	0.220	0.304	0.015	0.044
		0.15	3.415	4.035	0.743	1.221	0.219	0.311	0.019	0.045
		0.25	3.127	4.226	0.808	1.578	0.189	0.298	0.015	0.045
	10-10	0.05	38.351	44.161	-	-	3.123	3.608	1.342	3.138
		0.15	40.447	43.943	-	-	2.935	3.540	1.409	2.361
		0.25	39.356	44.929	-	-	3.101	3.593	1.353	2.175

Table 7: Experiments on D_{moy}, for n=250

short. Since the generalized release dates in our model correspond to completion times of tasks on Stage 1, then $\overline{r}_i = i \times p$ where p is the constant processing time of tasks on Stage 1, and at time $\overline{r}_i$ there are $i \times m_1$ available tasks, where $1 \leq i \leq \alpha = \lceil \frac{n}{m_1} \rceil$. In the rest of this section, we use the term release dates instead of generalized release dates.

Since the problem $PB(\overline{r})$ is NP-hard, this section focusses on setting a polynomial time approximation scheme (PTAS, for short) for the $PB(\overline{r})$ problem. In other words, we need to find a solution within a $(1+\epsilon)$ factor of the optimal solution in polynomial time, over all $0 < \epsilon < 1$. To get such PTAS, we perform several transformations to simplify the input of an instance into one with a simple structure. Each transformation increases the objective function value by less than $1 + O(\epsilon)$. So with a constant number of transformations, the objective function value stays within $1 + O(\epsilon)$ of the original optimum.

Li et al. [16] exhibited an approximation scheme algorithm (PTAS) for the problem of minimizing makespan with release times on identical parallel batching machines. Afrati et al. [1], Hall et Shmoys [11], et Deng et al. [5] provide principle and results for PTAS algorithm.

Let $\overline{r}_{max} = \lceil \frac{n}{m_1} \rceil p$ be the maximum release date, $a_{max} = \max\{a_j | 1 \leq j \leq n\}$ be the maximum initial endpoint over all tasks, and $P(B)$ be the total processing times of batches obtained by applying the FCBLPT rule. Denote by *opt* the optimal value of the makespan. We have the following result.

Lemma 2 $\max\{\overline{r}_{max}, a_{max}, \frac{P(B)}{m_2}\} \leqslant opt \leqslant \overline{r}_{max} + a_{max} + \frac{P(B)}{m_2}$.

Proof. It is obvious that $\max\{\overline{r}_{max}, a_{max}, \frac{P(B)}{m_2}\}$ is a lower bound for the optimal solution of $PB(\overline{r})$, since each of r_{max}, a_{max} and $\frac{P(B)}{m_2}$ is a lower bound for the problem. To show

the correctness of the upper bound, we exhibit a schedule with an objective value of at most $\overline{r}_{max} + a_{max} + \frac{P(B)}{m_2}$. Consider the schedule obtained from the following algorithm: Let L be the list of batches obtained by the FCBLPT rule, and use a list scheduling algorithm to sequence batches, starting at time $\overline{r}_{max}$. Let B be the last scheduled batch. B starts no later than $\overline{r}_{max} + \frac{P(B)}{m_2}$, and must finish no later than $r_{max} + a_{max} + \frac{P(B)}{m_2}$. Thus, the correctness of the upper bound. □

Let $M = \epsilon \times \max\{\overline{r}_{max}, a_{max}, \frac{C(B)}{m_2}\}$, where $0 < \epsilon < 1$. From Lemma 2, we can deduce that any optimal schedule has a makespan value lower than $\frac{3}{\epsilon}M$.

Consider the first transformation on release dates. The idea is to transform each $\overline{r}_i$, $i = 1, \ldots, \alpha$, to a multiple integer of M, i.e. we round down each $\overline{r}_i$ to the nearest multiple of M. Let $\tilde{r}_i = M\lfloor \frac{\overline{r}_i}{M} \rfloor$ be the rounded value of $\overline{r}_i$. We have the following result.

Lemma 3 *With $(1+\epsilon)$ loss, we can assume that there are at most $(\frac{1}{\epsilon}+1)$ distinct release dates in the original problem.*

Proof. The gap between an original release date and its rounded value is at most M. So, a feasible schedule for the original problem can be obtained from the feasible solution of the rounded problem by moving forward all batches by an amount of at most M. Since $M \leqslant \epsilon \times opt$, the transformation involves a $(1+\epsilon)$ loss. Furthermore, we have $M \geqslant \epsilon \times \overline{r}_{max}$. Hence $\lfloor \frac{\overline{r}_i}{M} \rfloor \leqslant \frac{1}{\epsilon}$, for all i. Thus we obtain at most $(\frac{1}{\epsilon} + 1)$ distinct release times. □

Since the rounded problem contains $(\frac{1}{\epsilon} + 1)$ distinct release dates, where each release date is a multiple of M, we partition the time horizon interval $[0, \frac{3}{\epsilon}M)$ into $(\frac{1}{\epsilon}+1)$ disjoint intervals. We denote by $\delta_i = [(i-1)M, iM)$ the ith interval, $i = 1, \ldots, \frac{1}{\epsilon}$ and $\delta_{\frac{1}{\epsilon}+1} = [(\frac{1}{\epsilon}M, (\frac{3}{\epsilon})M)$. We assume that the $(\frac{1}{\epsilon} + 1)$ release dates are denoted by $\rho_1, \rho_2, ..., \rho_{\frac{1}{\epsilon}+1}$, with $\rho_i = (i-1)M$, $i = 1, 2, \ldots, \frac{1}{\epsilon} + 1$.

The second transformation concerns tasks. We split the set of tasks into two subsets, namely small tasks and large tasks. A task is called *small* if its initial endpoint of the interval processing time is less than $\epsilon \times M$. Otherwise, it is called *large*. A batch is called *large* if it contains at least one large task, and *small* otherwise.

Our polynomial time approximation scheme (PTAS) proceeds in two steps. First, we schedule large tasks optimally after one transformation on their processing times. In the second step, we schedule approximately the set of small tasks. Before proceeding further, let us present the following result.

Lemma 4 *There exists an optimal schedule with the following properties :*

1. *On any one machine, the batches started at the same time interval are sequenced in non-increasing batch processing times order.*
2. *On each interval, δ_1 to $\delta_{\frac{1}{\epsilon}+1}$, batches started at the same interval are filled in non-increasing batch processing times order, such as each of them consists of k (or as many as possible) largest compatible and available number of tasks.*
3. *On each interval, δ_1 to $\delta_{\frac{1}{\epsilon}+1}$, there is at most one large batch that contains small tasks.*

Proof. This lemma can be easily established by using exchange arguments. □

7.1 Small tasks

In this section, we restrict our discussion to the case where all tasks are small.

First, we apply FCBLPT rule to obtain a list L of batches. This list is ordered in non-increasing order of $\frac{P_{Bi}}{k_i}$ where P_{Bi} is the processing time and k_i is the number of tasks of batch B_i. Then, at the beginning of each interval δ_i, when there are enough number of tasks available to sequence the first batch of L, schedule this batch on the first available machine, otherwise we go to the next interval.

Algorithm ST

- Apply algorithm FCBLPT. Let L be the list of obtained batches. Reindex L in non-increasing $\frac{P_{Bi}}{k_{Bi}}$, where P_{Bi} denote the batch processing time and k_{Bi} the number of tasks of batch B_i, respectively. Let B be the first batch of list L, and $nb = 0$.
- For each interval δ_i
  1. If there is no enough available tasks to process a batch B (i.e. $m_1\lfloor\frac{(i-1)M}{p}\rfloor - nb \leq 0$) or if there is no available machine before the end of interval δ_i (except for the last interval), then go to the next interval δ_{i+1}
  2. If enough tasks are available to process batch B (i.e. $m_1\lfloor\frac{(i-1)M}{p}\rfloor - nb \geq k_B$) and there is an available machine, then schedule batch B on the first available machine. Remove B from L, $nb = nb + k_B$ go to step 1.

Lemma 5 *Algorithm ST is a PTAS when all tasks are small with at most* $1 + (2 + \frac{1}{m_2})\epsilon^2$ *loss.*

Proof. Denote by S and opt the optimal schedule and the optimal makespan, respectively. Let $\tilde{S}$ and $\tilde{opt}$ be the schedule obtained by Algorithm ST and the value of its makespan, respectively. Consider the last idle time interval Δ in schedule $\tilde{S}$, and let t be the terminal endpoint of Δ. It is easy to see that t must be one of the modified release date ρ_i, otherwise the batch scheduled after Δ can be shifted to the right. Since all batches in $\tilde{S}$ are small, then any batch that starts before t must be finished earlier than $t + \epsilon M$. Denote by $\tilde{A}$ the set of all batches in $\tilde{S}$ started after t. Since all batches of $\tilde{A}$ are scheduled without idle times from time $t + \epsilon M$ and $\tilde{S}$ is obtained by list scheduling algorithm, then an upper bound on $\tilde{opt}$ is

$$\tilde{opt} \leq t + \epsilon M + \frac{\sum_{B\in\tilde{A}} P_B}{m_2} + P_B^{max} \leq t + \frac{\sum_{B\in\tilde{A}} P_B}{m_2} + 2\epsilon M, \tag{1}$$

where P_B^{max} is the largest processing time of batches of set $\tilde{A}$.

Assume that $\tilde{B}$ is the delayed batch from δ_i to δ_{i+1}. We have that $\sum_{B\in\tilde{A}} P_B = \sum_{B\in\tilde{A}\setminus\tilde{B}} P_B + P_{\tilde{B}} \leq \sum_{B\in\tilde{A}\setminus\tilde{B}} P_B + \epsilon M$. $\tilde{A} \setminus \tilde{B}$ represents the set of all tasks to be scheduled from time t.

We denote by A the list of batches scheduled after time t in the optimal schedule S. Looking at the construction and the reindexing operation in Algorithm ST, it is easy to deduce that $\sum_{B\in\tilde{A}\setminus\tilde{B}} P_B \leq \sum_{B\in A} P_B$. Thus $\sum_{B\in\tilde{A}} P_B \leq \sum_A P_B + \epsilon M$. It then follows that,

$$t + \frac{\sum_{B\in\tilde{A}} P_B}{m_2} + 2\epsilon M \leq t + \frac{\sum_{B\in A} P_B + \epsilon M}{m_2} + 2\epsilon M$$

Since $t + \frac{\sum_{B\in A} P_B}{m_2}$ is a lower bound for opt then using inequality (1), we obtain $\tilde{opt} \leq opt + \frac{\epsilon M}{m_2} + 2\epsilon M$. Since $M \leq \epsilon \times opt$, we get $\tilde{opt} \leq (1 + (2 + \frac{1}{m_2})\epsilon^2)opt$. □

7.2 Large tasks

The scheduling of the large tasks is based on enumeration. In order to limit the number of feasible schedules, we use the technique of [1] to get a number of distinct processing times independent of number of tasks. We round the initial endpoint of all interval processing times by multiplying every initial endpoint by $(1+\epsilon)$. Next, we decrease it to the lower integer power of $(1+\epsilon)$. The following lemma is due to Afrati et al. [1].

Lemma 6 *With $(1+\epsilon)$ loss, we assume that all initial endpoint of interval processing times of large tasks are integer powers of $(1+\epsilon)$.*

This lemma ensures that the number of distinct processing times can be bounded by a constant number.

Lemma 7 *The number of distinct initial endpoint of processing times of large tasks, v, is bounded by $\lfloor log_{1+\epsilon}(1/\epsilon)+1 \rfloor$.*

Proof. Let j be the shortest large task, we have $a_j \geqslant \epsilon M \geqslant \epsilon^2 a_{max}$. From lemma 6, we may assume that $a_j = (1+\epsilon)^x$ and $a_{max} \geqslant (1+\epsilon)^{x+v-1}$ for some integer x. It follows that $a_j = (1+\epsilon)^x \geqslant \epsilon^2(1+\epsilon)^{v-1}(1+\epsilon)^x$. Hence $v \leq \lfloor 1 + log_{1+\epsilon}(1/\epsilon^2) \rfloor$. □

Let $\overline{a}_1 < \overline{a}_2 < \ldots < \overline{a}_v$ be the v distinct processing times of large tasks. We keep the compatibility relations between tasks. We use the concept of machine configurations and execution profiles, introduced by Hall and Shmoys [11].

Let σ be a feasible schedule. We delete from σ all tasks and small batches, and retain only empty large batches, which are represented by their processing times. For each machine, we call *machine configuration*, with respect of σ, the pair (λ, μ) with $\lambda = (\lambda_1, \ldots, \lambda_{\frac{1}{\epsilon}+1})$ and $\mu = (\mu_{1,1}, \ldots, \mu_{1,v}, \mu_{2,1}, \ldots, \mu_{2,v}, , \mu_{\frac{1}{\epsilon}+1,1}, \ldots, \mu_{\frac{1}{\epsilon}+1,v})$ such that λ_i is the total number of large batches started on that machine in interval δ_i, and $\mu_{i,j}$, $i = 1, \ldots, \frac{1}{\epsilon}+1$, $j = 1, \ldots, v$ is the number of empty large batches started in δ_i with processing time $\overline{a}_j$, thus $\sum_{i=1}^{v} \mu_{i,j} = \lambda_i$, $i = 1, \ldots, \frac{1}{\epsilon}+1$. Since the length of each interval δ_i, $i = 1, \ldots, \frac{1}{\epsilon}$, is M, except for $\delta_{\frac{1}{\epsilon}+1}$ which has length of $\frac{2}{\epsilon}M$, and each large batch has processing times greater than ϵM, then each δ_i, $i = 1, \ldots, \frac{1}{\epsilon}$ contains at most $\frac{1}{\epsilon}$ large batches and the last interval $\delta_{\frac{1}{\epsilon}+1}$ contains at most $\frac{2}{\epsilon}$ large batches. On the other hand, from Lemma 7, processing times of empty large batches are chosen from v values with $v \leq \lfloor 1 + log_{1+\epsilon}(1/\epsilon^2) \rfloor$. Then, when δ_i contains w large batches, the number of possibilities for each interval δ_i is at most v^w configurations. Hence, when w takes all values from zero to $\frac{1}{\epsilon}$, the total number of configurations of vector $\mu_i = (\mu_{i,1}, \ldots, \mu_{i,v})$ is equal to $1+v+\ldots+v^{1/\epsilon}$, and the number of machine configuration to consider for all intervals δ_i can be bounded by $(1+v+\ldots+v^{1/\epsilon})^{1/\epsilon} \times (1+v+\ldots+v^{2/\epsilon^2}) < 2^{1/\epsilon} \times v^{3/\epsilon}$, which is independent of n. Thus, for a given schedule, we define an *execution profile* as a tuple $(m_1, \ldots, m_\psi)$, where m_i is the number of machines with the configuration i for that schedule. Therefore, we may only consider $(m+1)^\psi$ execution profile.

7.3 A PTAS for PB($\overline{r}$)

In this section we merge the two types of tasks in the same schedule to produce a polynomial approximation scheme for the $P(\overline{r})$.

The main idea of PTAS Algorithm is as follows : given an execution profile σ and feasible assignement of large tasks to this profile, the objective is to schedule small tasks in intervals δ_i $(i = 1, \ldots, \frac{1}{\epsilon}+1)$ each time it is possible, i.e. there are enough number of available tasks at each begining interval δ_i and there is an idle time on machines in interval δ_i. When a scheduling of a small batch in an interval idle time δ_i crosses the interval δ_{i+1}, i.e. a small batch starts its execution in the interval δ_i and completed in interval δ_{i+1}, we can strech the end of the interval δ_i to make an extra space with length ϵM for a small batch such that it need not cross the interval. Since there are at most $\frac{1}{\epsilon}$ intervals streched, then the following lemma.

Lemma 8 *(Afrati et al. [1]) With $(1+\epsilon)$ loss, we restrict our attention to schedules in which no small batch crosses an interval.*

From Lemma 3, we know that among the large batches started in the same interval, only one large batch may contain small tasks. Since all these small tasks are compatible between each other, we can stretch each interval to make an extra space of length $\epsilon \times M$ to schedule the small tasks in the same small batch. Since there are $(\frac{1}{\epsilon}+1)$ intervals, we obtain the following lemma.

Lemma 9 *With $(1+\epsilon+\epsilon^2)$ loss, we assume that no small tasks is included in a large batch.*

Now we are ready to present the complete algorithm.

Algorithm LT-ST

- Get all possible execution profiles as mentioned in Section 7.2
- For each execution profile, set $nb = 0$
    1. Assign a configuration for each machine according to the profile. If it is not possible, delete the profile.
    2. For each machine and each interval, start the empty large batches specified by the execution profil as soon as possible in non-increasing processing times order. If some batch needs to be delayed to start in one of the next interval, then delete this profile.
    3. For each interval, fill the empty large batches started in this interval in the order of non-increasing batch processing times such that each of them consists of b (or as many as possible) compatible tasks that have greater initial endpoints and no more than the processing time of the batch. If one of the batches remains empty, than delete the profile. Otherwise, add to nb, the number of sequenced tasks, and update the rest of available tasks to be scheduled in this interval, i.e. $l_{\delta_i} = m_1.\lfloor (i-1)M/p \rfloor - nb$.
    4. Run algorithm ST to schedule small tasks in the spaces left by the large batches.
- Select the feasible schedule with the smallest makespan.

Theorem 8 *Algorithm LT-ST is a PTAS for problem $PB(\overline{r})$.*

Proof. Since an optimum schedule is associated with one of the $(m+1)^{\psi}$ execution profiles and all execution profiles are explored, then a schedule obtained by Algorithm LT-ST approximates the optimal solution. Given an execution profile, Algorithm LT-ST generates an optimal schedule of large tasks for this profile. Invoking Algorithm ST yields at most $1+(2+\frac{1}{m_2})\epsilon^2$ loss. Combining Lemmas 2, 5, 7 and 8, and taking the smallest one among the feasible schedules produced, Algorithm LT-ST generates an $(1+4\epsilon+(3+\frac{1}{m_2})\epsilon^2)$ loss, and its time complexity is $O(nlogn + n \times (m+1)^{\psi})$. □

8 Conclusion

In this paper we have considered a new application of two-stage hybrid flowshop problem that involves classical and batching machines. The batching machine can process several tasks per batch with the additional feature that the tasks of the same batch have to be compatible. A compatibility relation is defined between each pair of tasks. We have considered the makespan criterion. Since minimizing the makespan is NP-hard, we have presented three heuristics and their worst-case analysis. We have also considered two particular cases, namely, one discrete machine at the first stage and one batching machine at the second stage. When the processing times of the tasks are constant at the first stage, we have propossed a polynomial time approximation scheme.

References

[1] F. Afrati, E. Bampis, C. Chekuri, D. Kargerr, C. Kenyon, S. Khana., I. Milis, M. Queyranne, M. Skutella, C. Stein, and M. Sviridenko. Approximation schemes for minimizing average weighted completion time with release dates. In *Proceeding of the 40th Annual IEEE Synposium on Foundations of Computer Science*, pages 32–43, 1999.

[2] H. Allaoui and A. Artiba. Scheduling two-stage hybrid flow shop with availability constraints. *Computers & Operations Research*, (33):1399–1419, 2006.

[3] M. Boudhar and Finke G. Scheduling on a batch machine with job compatibility. *Belgian Journal of Operations Research, Statistics and Computer Science.*, 40:69–80, 2000.

[4] P. Brucker, A. Gladky.and J.A. Hoogeveen, M.Y. Kovalyov, C.N. Potts, T. Tautenhahn, and S.L. Van de Velde. Scheduling a batch processing machine. *Journal of Scheduling*, (1):31–54, 1998.

[5] X. Deng, H. Feng, G. Li, and B. Shi. A ptas for semiconductor burn-in scheduling. *Journal of Combinatorial Optimization*, (9):5–17, 2005.

[6] G. Finke, V. Jost, M. Queyranne, and A. Sebo. Batch processing with interval graph compatibilities between tasks. *Discrete Applied Mathematics*, In press, 2007.

[7] R. L. Graham. Bounds on multiprocessor timing anomalies. *SIAM J. appl. Math.*, 17:416–429, 1969.

[8] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G. Rimooy Kan. Optimization and approximation in deterministic sequencing and scheduling : a survey. *Annals of Discrete Mathematics*, (5):287–326, 1979.

[9] A. Guinet, M.M. Solomon, P.K. Kedia, and A. Dussauchoy. A computational study of heuristics for two-stage flexible flowshop. *Int. J. Prod. Res.*, 34(5):1399–1415, 1996.

[10] J.N.D Gupta, K. Kruger, V. Lauff, F. Werner, and Y. N. Sotskov. Heuristics for hybrid flow shops with controllable processing times and assignable due dates. *Computers & Operations Research*, 29(10):1417–1439, 2002.

[11] L.A. Hall and D.B. Shmoys. Approximation schemes for constrained scheduling problems. In *Proceedings of the 30th Annual IEEE Synposium on Foundations of Computer Science*, pages 134–139, 1989.

[12] M. Haouari and R. M'Hallah. Heuristic algorithms for the two-stage hybrid flowshop problem. *Operations Research Letters*, (21):43–53, 1997.

[13] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Res. Logist. Quart.*, 1:61–68, 1954.

[14] C-Y. Lee, R. Uzsoy, and L.A. Martin-Vega. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40(4):764–775, July-August 1992.

[15] C-Y. Lee and L. Vairaktarakis. Minimizing makespan in hybrid flowshop. *Operations Research Letters*, (16):149–158, 1994.

[16] D. Li, G. Li, and S. Zhang. Minimizing makespan with release times on identical parallel batching machines. *Discrete Applied Mathematics*, (148):127–134, 2005.

[17] B.M.T. Lin and T.C.E. Cheng. Batch scheduling in the no-wait two-machine flowshop to minimize the makespan. *Computers & Operations Research*, 28:613–624, 2001.

[18] A. Oulamara and G. Finke. Flowshop problems with batch processing machines. *International Journal of Mathematical Algorithms*, 2:269–287, 2001.

[19] A. Oulamara, G. Finke, and A. Kamgaing Kuiten. Flowshop scheduling problem with batching machine and task compatibilities. *Computers & Operations Research*, Accepted 2007.

[20] A. Oulamara, M.Y. Kovalyov, and G. Finke. Scheduling a no-wait flowshop with unbounded batching machines. *IIE Transactions on Scheduling and Logistics*, 37, 8:685–696, 2005.

[21] C.N. Potts and M.Y. Kovalyov. Scheduling with batching : a review. *European journal of Operational Research*, (120):228–249, 2000.

[22] C.N. Potts, V.A. Strusevich, and T. Tautenhahn. Scheduling batches with simultaneous job processing for two-machine shop problems. *Journal of Scheduling*, 4:25–51, 2001.

[23] C.N. Potts and L.N. Van Wassenhove. Integrating scheduling with batching and lotsizing : a review of algorithm and complexity. *Journal of Operational Research Society*, (43):395–406, 1992.

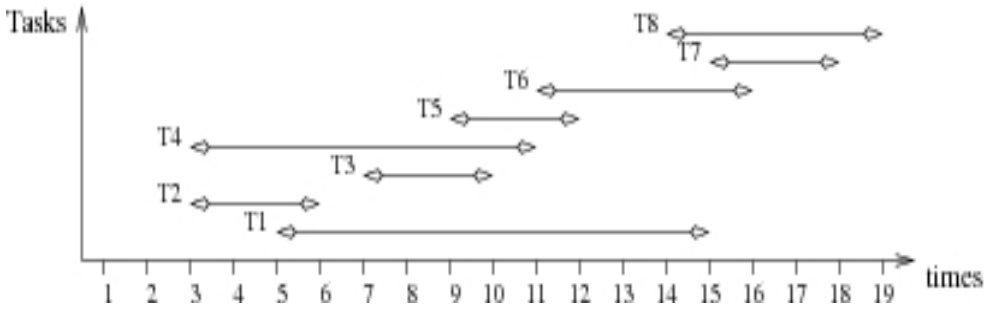

Figure 1: Graph compatibility between tasks.

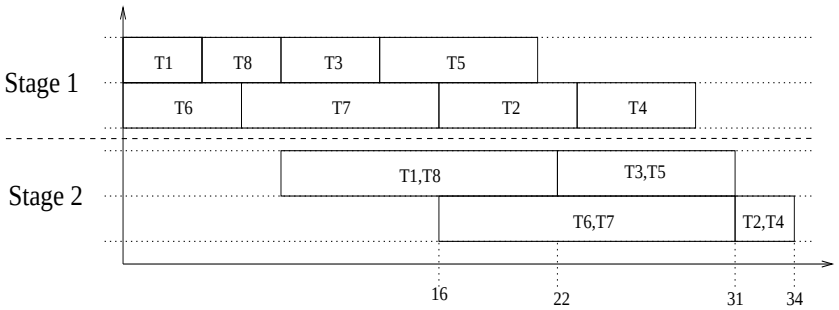

Figure 2: Feasible schedule S

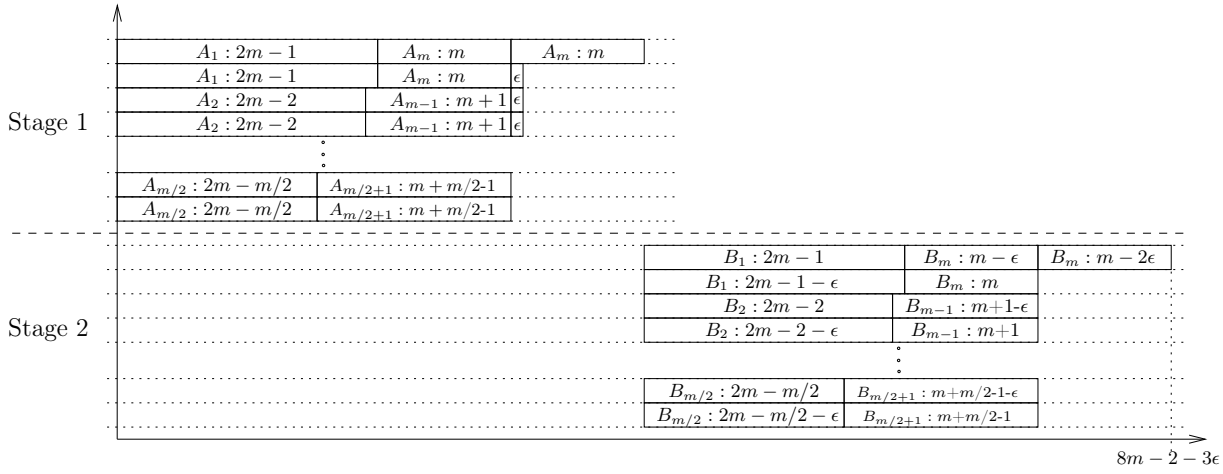

Figure 3: Schedule $S_{H_{LPT}}$

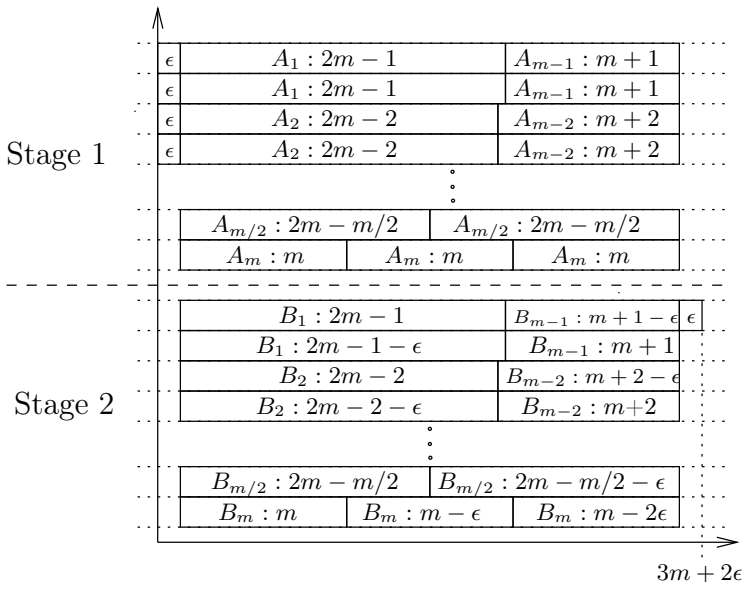

Figure 4: Optimal schedule S^*

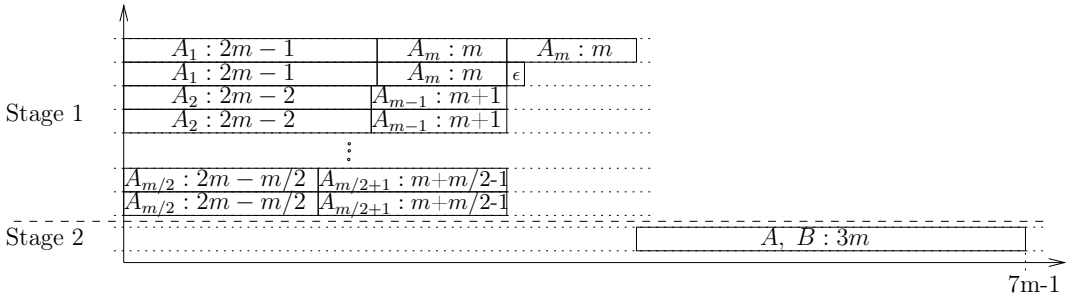

Figure 5: Schedule $S_{H_{LPT}}$

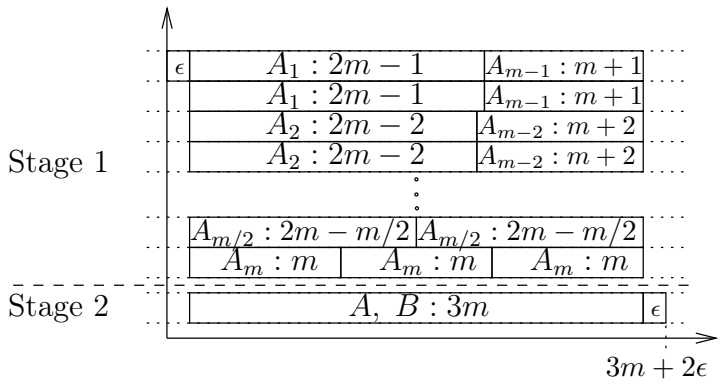

Figure 6: Optimal schedule S^*

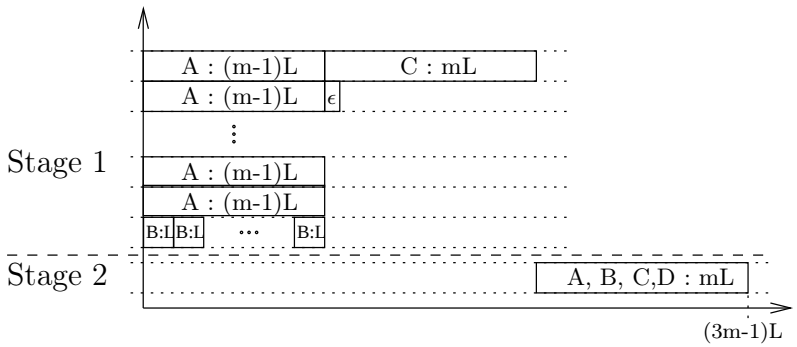

Figure 7: Schedule S_{H_J}

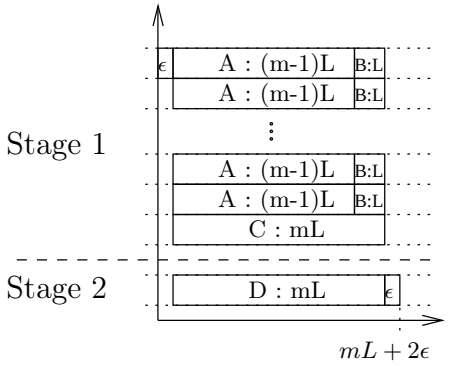

Figure 8: Optimal schedule S^*

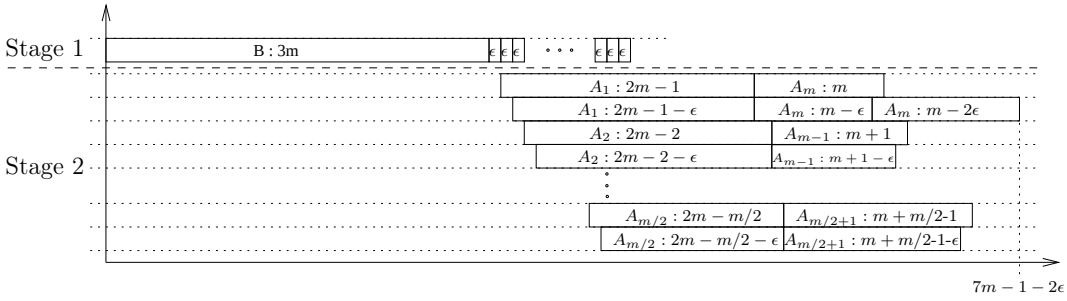

Figure 9: Schedule $S_{H_{LBPT}}$

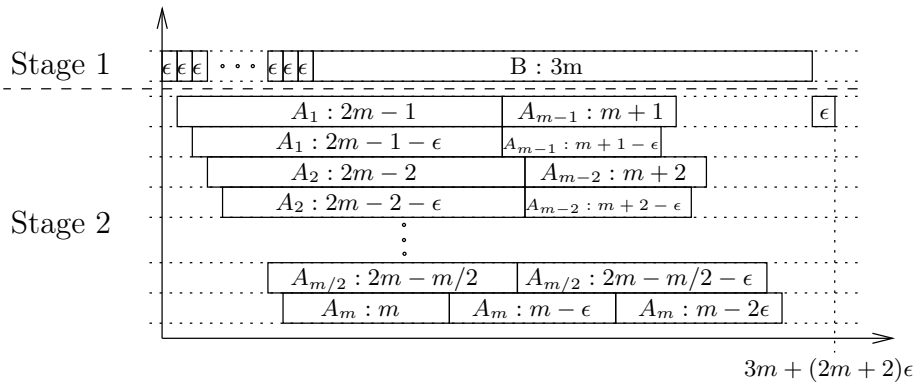

Figure 10: Optimal schedule S^*