Omega 108 (2022) 102567

Contents lists available at ScienceDirect

Omega

journal homepage: www.elsevier.com/locate/omega

Parallel batch scheduling: Impact of increasing machine capacity☆

Jun Xu[a,b], Jun-Qiang Wang[a,b,*], Zhixin Liu[c]

[a] Performance Analysis Center of Production and Operations Systems (PacPos), Northwestern Polytechnical University, Xi'an, Shaanxi 710072, PR China
[b] Department of Industrial Engineering, School of Mechanical Engineering, Northwestern Polytechnical University, Xi'an, Shaanxi 710072, PR China
[c] Department of Management Studies, College of Business, University of Michigan-Dearborn, 19000 Hubbard Drive, Dearborn, Michigan 48126-2638, USA

ARTICLE INFO

Article history:
Received 4 February 2021
Accepted 6 November 2021
Available online 15 November 2021

Keywords:
Scheduling
Parallel machines
Parallel batch
Capacity augmentation
Approximation algorithm
Worst case ratio

ABSTRACT

Scheduling performance naturally improves with increased machine capacity, but the per-unit improvement typically decreases or even keeps unchanged with excessive capacity. We consider parallel batch scheduling on identical machines to minimize the makespan, with and without preemption. The machine capacity is the maximum number of jobs that a machine can process simultaneously. We quantitatively analyze the impact of capacity augmentation on the makespan in the form of increasing machine capacity. Considering machines costs, we need to determine the machine capacity that minimizes the weighted sum of the makespan and capacity costs. First, we obtain an upper bound of the ratio between the minimum makespans of two scheduling problems with different machine capacities. Second, noting the intractability of the scheduling problem without preemption, we analyze the upper bound of the ratio between the obtained makespans by heuristics of two scheduling problems with different machine capacities. Third, for the preemptive case, we develop a polynomial time algorithm to obtain the optimal machine capacity, and also present another polynomial time algorithm to obtain the optimal machine capacity as well as for bounding the approximation ratio of the non-preemptive case. Fourth, we design an approximation algorithm using machine capacity found in the preemptive case to yield a schedule for the non-preemptive case, and analyze the worst case performance ratio of the algorithm. This research provides new insights into performance improvement with increased machine capacity in parallel batch scheduling, and analyzes the trade-off between the makespan and capacity costs.

1. Introduction

Parallel batch machines are encountered in many industries, such as semiconductor industry [1,2], aircraft industry [3,4], steel working industry [5,6] and glass manufacturing industry [7,8]. A batch machine can process multiple jobs simultaneously as a batch, which updates the conventional single-item machine on which at most one job can be processed at a time. A parallel batch machine is generally considered as a bottleneck for reasons such as high equipment cost, long production cycle, and high energy consumption [4,9].

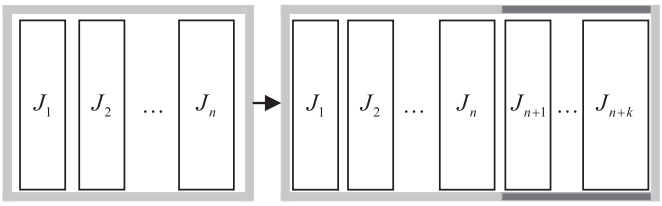
In classical problems, scheduling aims to optimize certain objectives under fixed resource constraints. In practice, however, resource capacity is not invariable and can be augmented. For batch machines, manufacturers improve scheduling performance by increasing *machine capacity* during reformation of an existing production line or implementation of a new production line. This can be achieved in two ways for different kinds of batch machines. First, for a reconfigurable batch machine with modular components, modular expansion of the batch machine can increase the volume of a batch machine according to capacity requirements. Second, for a batch machine with fixed volume, there is no choice but to keep the machine unchanged and try to excavate internal potential with adjustable tools and fixtures to improve space utilization for holding more jobs as a batch. We explain these two ways in detail as follows.

First, one can increase the volume of a batch machine by adding a certain modular chamber next to the existing chamber, forming an expansion to satisfy different capacity requirements, as shown in Fig. 1. Zhu and Yan [10] study a kind of variable-machine-capacity heat treatment furnace including fixed front furnace wall and removable rear furnace wall equipped with air preheater. By connecting a moveable furnace wall on the original furnace wall, the volume of the furnace is increased. Similarly, Huang [11] investigates a kind of removable transfiguration heat treatment furnace by setting separate fire doors to adjust the volume of the machine.

☆ Area: Production Management, Scheduling and Logistics. This manuscript was processed by Associate Editor Kovalyov.

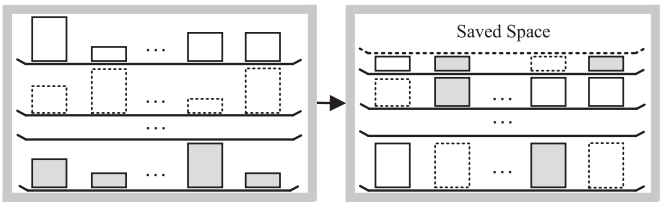
* Corresponding author at: Performance Analysis Center of Production and Operations Systems (PacPos), Northwestern Polytechnical University, Xi'an, Shaanxi 710072, PR China.
E-mail address: wangjq@nwpu.edu.cn (J.-Q. Wang).

https://doi.org/10.1016/j.omega.2021.102567
0305-0483/



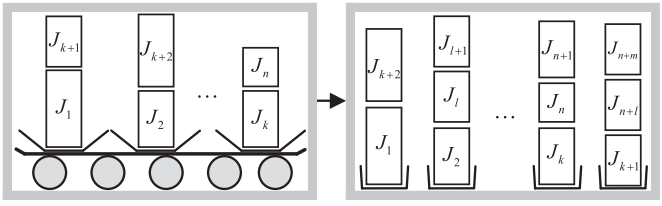
a. Chamber with fixed volume b. Chamber with extended modules

Fig. 1. Schematic diagram of modular expansion.



a. Arbitrarily grouped jobs on fixed trays b. Similar sized jobs grouped on removable trays

Fig. 2. Schematic diagram of adjusting tools and fixtures.



a. Jobs processed with shipping pallet b. Jobs processed with compact tools

Fig. 3. Schematic diagram of upgrading tools and fixtures.

Second, for excavating internal potential, there are two scenarios of adjusting, reducing and upgrading the related tools and fixtures to improve the space utilization for holding more jobs in a batch as shown below.

In the first scenario, when processing different sized jobs, the jobs are grouped in arbitrary sequence as a batch on the fixed trays, resulting in space waste and low space utilization, as shown in Fig. 2a. In order to improve space utilization, the similar sized jobs are grouped on a tray to make the trays more compact, and the removable multi-deck trays are introduced to divide the chamber of a batch machine into separate spaces. Thus, adjusting different combinations of grouping jobs can save more space to hold more jobs as a batch, as shown in Fig. 2b. This method can be widely used to expand capacity in different manufacturing industries, such as burn-in oven in semiconductor manufacturing [12], hardening oven in aircraft industry [13] and automobile gear manufacturing [14].

In the second scenario, a shipping pallet is put into a batch machine with the jobs for the convenience of centralized delivery, resulting in waste of space, as shown in Fig. 3a. Considering the batch processing time is longer than batch delivery time, it is worth sacrificing a little batch delivery time for great gains in freeing more space to hold more jobs simultaneously by using compact or upgraded tools and fixtures. Specifically, we use compact trays to hold jobs, and unload jobs from shipping pallets, as shown in Fig. 3b. After the jobs are finished processing, these trays are removed individually. This method can be applied to car type annealing furnaces in steel industry, well type annealing furnaces in copper manufacturing, and soaking operation in gear manufacturing [14].

Increasing machine capacity is a kind of *resource augmentation* which is a way to utilize extra resources to improve the performance of schedules. For scheduling with resource augmentation, the researchers relax resource constraints to obtain better performance of schedules. In terms of different types of resource augmentation, the related research can mainly be classified into four categories. The first category is to increase the number of machines [15,16] which is called machine augmentation model. The second category is to speed up the machine [17,18] which is called speed augmentation model. The third category is to allow preemption [19,20]. The fourth category is to increase capacity which is called *capacity augmentation* model [21]. The first three methods have been investigated on single-item machines, while the fourth method is proposed for batch machines in this paper.

Existing three methods focus on single-item machines and are hard to apply in the context of batch machines in production systems. It is hard to improve scheduling performance by increasing the number of batch machines due to the high equipment cost of batch machine [4,9]. It is impossible to shorten the processing of jobs by speeding up the machine due to the hard constraint that all jobs in a batch have to stay in a specific equipment for longer than its prescribed time. Preemption is not allowed for the reason that no jobs can be interrupted until the batch is completed. No research related to capacity augmentation has been investigated on batch scheduling in production.

For capacity augmentation, scheduling performance naturally improves to a certain extent, but the per-unit improvement typically decreases or even keeps unchanged with excessive resources. So it is realistic for production to make capacity augmentation decisions and investigate the capacity augmentation impact.

In this paper, we analyze improved scheduling performance with increased machine capacity for identical parallel batch machines. Our research lies in two aspects. First, we quantify the impact of increasing machine capacity on the makespan. Scheduling performance is promoted by increased machine capacity, but the per-unit improvement typically decreases or even keeps unchanged with overly large capacity. We need to measure the impact of increasing machine capacity and deduce the corresponding bound with different capacities. Second, we investigate the trade-off between improved scheduling performance and the cost of increased machine capacity, which is in view of the practice that it is not free to expand the capacity of a batch machine.

To address the first problem, we analyze the impact of increasing machine capacity by the *ratio* of the makespan incurred with the initial machine capacity over that with increased machine capacity. We theoretically establish upper bounds of the impact under optimal and approximation algorithms, for cases with and without preemption. The analysis of impact quantifies the improvement of the makespan with increased machine capacity, and provides guidelines to support machine capacity planning. To handle the second problem, considering usage costs of batch machines, we minimize the weighted sum of the makespan and capacity costs. For the preemptive case, we provide two algorithms to find the optimal machine capacity. For the non-preemptive case, we develop an approximation algorithm for capacity decision, and analyze its worst case performance. Our results provide insights on how to make a cost-effective choice of machine capacity.

The remainder of this paper is organized as follows. In Section 2, we review the literature related to parallel batch scheduling, scheduling with resource augmentation, and impact of resource augmentation. In Section 3, we formally define our problem with notations introduced. Section 4 analyzes the impact of increasing machine capacity. In Section 5, we develop algorithms for machine capacity decisions and analyze the performance of our algorithms. Finally, conclusion and future work are given in Section 6.

2. Literature review

This section reviews related literature on parallel batch scheduling in Section 2.1, scheduling with resource augmentation in Section 2.2, and impact of resource augmentation in Section 2.3.

2.1. Parallel batch scheduling

Related work on parallel batch scheduling can be classified into two categories according to volume requirements of jobs: unit size jobs and different size jobs. We refer readers to [22–24] for the survey papers on parallel batch scheduling, and next only review those closely relevant to ours.

For scheduling with unit size jobs on a single parallel batch machine, Brucker et al. [25] study batch scheduling problems to minimize regular objective functions for both unbounded model and bounded model. For an unbounded model, they design polynomial time algorithms to minimize the maximum cost, the number of tardy jobs, the maximum lateness and the total weighted completion time. They also prove the problems of minimizing the weighted number of tardy jobs and the total weighted tardiness are *NP*-hard. For a bounded model, they develop a dynamic programming algorithm in $O(n^{b(b-1)})$ time for minimizing the total completion time, and prove that scheduling problems with due date-based objectives are *NP*-hard. For a capacitated single parallel batch machine scheduling to minimize the total completion time, Poon and Yu [26] design an optimal algorithm where the running time is an exponential function of the machine capacity. Chandru et al. [27] develop a branch and bound algorithm and two approximation algorithms to solve a single parallel batch machine scheduling problem with total weighted completion time.

For scheduling with unit size jobs on multiple parallel batch machines, Lee et al. [1] develop a full batch list scheduling (FBLS) rule and full batch longest processing time (FBLPT) rule to minimize the makespan, with worst case performance ratios no greater than $b+1-1/m$ and $4/3-1/(3m)$, respectively, where m is the number of machines and b is the machine capacity. When the objective is to minimize the total completion time, Cheng et al. [28] propose an optimal dynamic programming algorithm with running exponential time to the number of machines, and Chandru et al. [27] analyze the worst case performance of two approximation algorithms. Hochbaum and Landy [29] study parallel batch scheduling to minimize total completion time with distinct job types, and develop approximation algorithms.

For scheduling with different size jobs on a single parallel batch machine, Uzsoy [30] proves problems of minimizing the makespan and total completion time are both *NP*-hard. The author integrates bin-packing results and existed results on unit size problem to present a number of heuristics, such as a longest processing time first fit (LPT-FF) and a shortest processing time first fit (SPT-FF). Zhang et al. [31] analyze the worst case ratios of heuristics proposed in [30]. They show some heuristics have unbounded worst case ratios while others have worst case ratios no greater than 2. In particular, they develop a 7/4-approximation algorithm. Malapert et al. [32] present a constraint programming approach to minimize the maximum lateness. Muter [33] proposes an exact algorithm based on decomposition in two levels to minimize the makespan. In the first level, the single batch machine scheduling problem is solved by a column-and-cut generation algorithm, which is regarded as a lower bound for identical batch machines problem. In the second level, a search mechanism is used to obtain the minimum makespan for identical batch machines. Emde et al. [34] study a single batch machine scheduling with the lateness objective and propose a novel exact algorithm. Considering a truncated batch-position-based learning effect, Cheng et al. [35] study two models with batch operations to minimize the makespan. In the first model, jobs have unit size, and they propose an optimal algorithm in $O(n\log n)$. In the second model, jobs have arbitrary sizes, and they propose an approximation algorithm with a worst case ratio less than 2. Polyakovskiy and M'Hallah [36] combine the two-dimensional bin packing and just-in-time batch scheduling problem. They adopt branch-and-check approaches to minimize the total weighted earliness and tardiness.

For scheduling with different size jobs on identical parallel batch machines, Jia and Leung [37] present a meta-heuristic algorithm and conduct experiments to show its efficiency. Li [38] investigates the problem of scheduling jobs with release times and different sizes to minimize the makespan on non-identical capacities batch machines. The author develops approximation algorithms for both cases of equal release times and different release times, and also considers processing set restrictions and incompatible families to present multiple approximation algorithms. For the problem of scheduling jobs with different release times and sizes, Ozturk [39] proposes a column generation algorithm to minimize the sum of completion times.

2.2. Scheduling with resource augmentation

We review four categories related to resource augmentation, i.e., machine augmentation model, speed augmentation model, allowing preemption, and capacity augmentation model.

In machine augmentation model, Phillips et al. [40] define this model in which the online algorithm can use more machines than the adversary to achieve offline optimal solution for the problem of hard-real-time scheduling and minimizing total flow time, i.e., the length of the time interval between release time and completion time of a job. This model also considers machine cost and optimizes related objectives when the number of machines is a decision variable. Dósa and Tan [41] consider an online scheduling problem where no machines are initially provided and the objective is to minimize the sum of the makespan and the cost of purchased machines. These authors develop an online algorithm with performance analysis. Rustogi and Strusevich [16] study identical parallel machines scheduling to minimize the makespan and total completion time. They design approximation algorithms to minimize the weighted total of scheduling objective and machine costs. Akaria and Epstein [42] consider an online scheduling problem with a general machine cost function. The objective is to minimize the makespan plus the cost of the purchased machines. They develop two deterministic algorithms with competitive ratio analysis. We refer readers to Li et al. [43] for surveys on scheduling with machine cost.

In the speed augmentation model, Kalyanasundaram and Pruhs [17] investigate how competitive ratios of online algorithms for two classical online scheduling problems can be improved through using faster speed machines. For the online scheduling problem on multiple machines, Chan et al. [44] analyze speed augmentation of the shortest remaining processing time first and shortest job first algorithms to minimize the total stretch, i.e., the ratio of the flow time to the processing time. To minimize the maximum flow time of the broadcast scheduling problem, Im et al. [21] develop an approximation scheme. Imreh [45] considers the problem with general machine cost functions for online scheduling problems, where machines at different speeds are with different costs. The author designs approximation algorithms and establishes lower bounds of competitive ratios of the algorithms.

For the machine augmentation model and speed augmentation model, there are some works studying both of them and comparing the differences between the two models. For the problem of online scheduling on m identical machines, Chekuri et al. [46] consider the flow time and stretch, and analyze the performance of two algorithms with $(1+\epsilon)m$ machines with unit speed or m ma-

chines with $(1+\epsilon)$ speed. Chan et al. [18] establish a relationship between extra speed and extra machines for an online scheduling problem to minimize the flow time and stretch on multiple parallel machines. These authors show that competitive results via faster machines can be transformed to similar results via extra machines.

For allowing preemption, the complexity of problems or algorithm design is investigated with increasing the number of preemption. Soper and Strusevich [47] design an optimal algorithm for problem $Q2|\#pmtn \leq 1|C_{\max}$, and also prove problem $R2|\#pmtn \leq 1|C_{\max}$ is binary *NP*-hard. The current related research mainly focuses on the impact of preemption which will review in the next subsection.

For the capacity augmentation model, there is little research on batch machine scheduling. Jaillet and Wagner [48] study online traveling salesman problem considering capacity augmentation that allows the online server to have a larger capacity than the offline server, and improve competitive ratios of online algorithms. To minimize the maximum flow time of the broadcast scheduling problem, Im et al. [21] design the first (1+ε)-approximations with using extra capacity or faster speed.

2.3. Impact of resource augmentation

For the impact of resource augmentation, researchers investigate the quantitative divergence of initial resource and augmented resource. For above four categories, the impact of machine augmentation and the impact of allowing preemption have been investigated.

For the impact of machine augmentation, Brehob et al. [15] analyze the divergence between algorithms with different numbers of machines. They study the problem of scheduling jobs on identical machines to minimize the makespan, and analyze the worst case performance of online and offline approximation algorithms relative to the performance of an optimal offline algorithm when the optimal offline algorithm assumes extra machines. For load balancing problems, Azar et al. [49] assume the online algorithm can be allowed to use more machines and examine the impact of machine augmentation. Rustogi and Strusevich [16] study the problem of processing jobs on identical parallel machines to minimize the makespan and the total completion time. These authors analyze the impact of extra machines on scheduling objectives.

For impact of allowing preemption, Braun and Schmidt [19] study the power of preemption over the scheduling objective and analyze the quantitative divergence between the optimal makespan of $i-$preemptive schedule and preemptive schedule for the problem $Pm||C_{\max}$. Jiang et al. [50] propose a simpler method to improve the results in [19]. Soper and Strusevich [51] study problems of scheduling on two and three uniform parallel machines to minimize the makespan, and deduce corresponding tight bounds on power of preemption. For the problem $Qm||\sum C_j$, Epstein et al. [52] analyze the power of preemption. When the number of machines $m = 2$ and $m \geq 3$, its values are equal to 1.2 and approximately 1.39795, respectively. We refer the readers to [20,47,53,54] for related research about power of preemption.

In conclusion, to our best knowledge, Jaillet and Wagner [48] consider servers with increasing capacity in routing problem and Im et al. [21] study broadcast scheduling problem related to capacity augmentation, which are relevant to this research. No research has been paid attention to analyzing the impact of capacity augmentation in parallel batch scheduling. To the best of our knowledge, our work is the first to quantitatively analyze the impact of capacity augmentation in the form of increasing machine capacity in parallel batch scheduling. The characteristics of the problem are mainly in two aspects: variable machine capacity and capacity costs. Thus, the difficulty of the problem is to analyze the impact of increasing machine capacity on the makespan, and make a decision on machine capacity when considering capacity costs. We show how to solve the problems in this paper.

3. Problem description

Our problem is formally described from three aspects: machine environment, job characteristic and objective function. We consider identical parallel batch machines which have the same processing speed and machine capacity. Let $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ be the set of all the machines. A parallel batch machine can process multiple jobs simultaneously as a batch, under the condition that the number of jobs in the batch does not exceed the machine capacity b. All the machines can be simultaneously upgraded through capacity augmentation. We denote the initial machine capacity and upgraded machine capacity by b and $\hat{b}$, respectively.

Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be the set of all the jobs. All jobs with unit size are available at time zero. Let p_j denote the processing time of job J_j and $p_{\max}$ denote the maximum processing time of all jobs. We analyze the impact of increasing machine capacity on the makespan for both preemptive and non-preemptive cases. With preemption, a job can be split into multiple sub-jobs and each sub-job can be grouped with other jobs or sub-jobs as a batch, but cannot overlap with other sub-jobs from the same job in processing. If a job is split into multiple sub-jobs, then the sum of processing times of these sub-jobs is equal to the job's original processing time. Without preemption, once a batch begins to process and the batch cannot be interrupted or added with other jobs until the processing is completed. Multiple jobs can be processed simultaneously as a batch. Jobs in a batch have the same start time and the same completion time. The processing time of each job in a batch is equal to the longest processing time of all jobs in the batch.

The scheduling objective considered in this work is the makespan, i.e., the maximum completion time of all the jobs. The capacity cost of a parallel batch machine is assumed to be a linear function of machine capacity. The total cost function is the weighted sum of the makespan and capacity costs.

Following the three-field notation [55], the scheduling problem we consider is denoted by $Pm|p\text{-}batch, b < n|C_{\max}$ in the case without preemption, where Pm means m identical parallel machines, $p\text{-}batch$ refers to parallel batch, $b < n$ denotes bounded machine capacity, and $C_{\max}$ specifies the makespan. For problem $Pm|p\text{-}batch, b < n|C_{\max}$, we analyze the impact over makespan when machine capacity is upgraded from b to $\hat{b}$, where $\hat{b} > b$.

For the problem $Pm|p\text{-}batch, b < n|C_{\max}$, if $C_{\max}(\sigma^*(b), I) = p_{\max}$, we have $C_{\max}(\sigma^*(\hat{b}), I) = p_{\max}$ for any $\hat{b} > b$. That means increasing machine capacity has no contribution to the performance improvement. In other words, the value of worst performance improvement is equal to 1. Thus, the worst case ratio is not suitable for our problem. By further analyzing the reason, the worst case ratio is used to evaluate the performance of an algorithm rather than the performance improvement of increasing machine capacity. Therefore, we concentrate on investigating the best performance improvement.

With reference to the idea proposed by Rustogi and Strusevich [16], we formally define the best case impact ratio over optimal schedule as follows.

Definition 1. The best case impact ratio over optimal schedule is defined as the supremum of the ratio between the optimal makespan value with initial capacity b and that of upgraded capacity $\hat{b}$ over all instances, i.e.,

$$\rho^*(b, \hat{b}) = \sup_{I \in \mathcal{I}} \frac{C_{\max}(\sigma^*(b), I)}{C_{\max}(\sigma^*(\hat{b}), I)},$$

where $\sigma^*(b)$ and $\sigma^*(\hat{b})$ denote the optimal schedules with the machine capacities b and $\hat{b}$, respectively. $C_{\max}(\sigma^*(b), I)$ and

$C_{\max}(\sigma^*(\hat{b}), I)$ are the makespan values of the optimal schedules $\sigma^*(b)$ and $\sigma^*(\hat{b})$ on instance I, respectively.

Since the scheduling problem $Pm|p\text{-}batch, b < n|C_{\max}$ is *NP*-hard [1], it is impossible to obtain an optimal solution in polynomial time unless $P = NP$. Instead, we use the makespan obtained by approximation algorithms to further analyze the impact of increasing machine capacity as follows.

Definition 2. The best case impact ratio over approximate schedule by algorithm A is defined as the supremum of the ratio between the makespan obtained by algorithm A with initial capacity b and that of upgraded capacity $\hat{b}$ over all instances, i.e.,

$$\rho^A(b, \hat{b}) = \sup_{I \in \mathcal{I}} \frac{C_{\max}(\sigma^A(b), I)}{C_{\max}(\sigma^A(\hat{b}), I)},$$

where $\sigma^A(b)$ and $\sigma^A(\hat{b})$ denote the two schedules obtained by algorithm A with the machine capacities b and $\hat{b}$, respectively. $C_{\max}(\sigma^A(b), I)$ and $C_{\max}(\sigma^A(\hat{b}), I)$ are the makespan values of schedules $\sigma^A(b)$ and $\sigma^A(\hat{b})$ on instance I, respectively.

4. Impact of machine capacity

In this section, we analyze the impact of increasing machine capacity for problem $Pm|p\text{-}batch, b < n|C_{\max}$, in both preemptive and non-preemptive cases. The preemptive case is prioritized to be considered because its results lay the foundation for solving the non-preemptive case. Specifically, the optimal makespan of the preemptive case is a lower bound for the non-preemptive case, which can be used to analyze the impact ratio of increasing machine capacity on the makespan.

4.1. Preemptive case

We first consider the problem $Pm|pmtn|C_{\max}$, where *pmtn* means preemption is allowed, can be solved optimally by a wrap-around algorithm in polynomial time by McNaughton [56]. For the problem $Pm|p\text{-}batch, pmtn, b < n|C_{\max}$, we design Algorithm 1 to solve it optimally, similar to the one developed by McNaughton [56], in polynomial time. The idea of our Algorithm 1 is to split a job into multiple sub-jobs for reducing the waste of space in each batch. Algorithm 1 finds the preemptive schedule and optimal makespan, as follows.

Algorithm 1

Step 1. Sort all jobs in non-increasing order of their processing times and compute $P = \sum_{j=1}^{n} p_j$. We modify every batch machine into a block with b unit capacity machines. Let $C_{\max}(\sigma_p^*(b)) = \max\{p_{\max}, P/(mb)\}$.

Step 2. We successively assign jobs to unit capacity machines until its completion time no greater than $C_{\max}(\sigma_p^*(b))$. If the processing time of a job exceeds $C_{\max}(\sigma_p^*(b))$, we preempt the job at time $C_{\max}(\sigma_p^*(b))$ and assign the remaining part to the following unit capacity machine as the first job.

Step 3. We reorder jobs processed in every block in the non-decreasing order of their completion times. For a job completed at time t, we preempt all other jobs processed in the same block at time t, and form these jobs as a batch. Continue this process for every block from time 0 to $C_{\max}(\sigma_p^*(b))$.

According to the above main idea, the following is a detailed description of Algorithm 1.

To illustrate Algorithm 1, we use the following example.

Example 1. Let $n = 7$, $m = 2$ and $b = 2$. The processing times of jobs are $p_1 = 5$, $p_2 = p_3 = p_4 = 4$, $p_5 = p_6 = 3$, $p_7 = 1$.

Following Algorithm 1, we yield the optimal schedule. In Step 1, all jobs are sorted in non-increasing order of their processing times. We compute $P = 24$ and $C_{\max}(\sigma_p^*(b)) = \max\{5, 6\} = 6$.

Algorithm 1: Preemptive optimal schedule.

Input : Set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, set of identical parallel batch machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$, and machine capacity b;

Output: The makespan $C_{\max}(\sigma_p^*(b))$ of the preemptive schedule $\sigma_p^*(b)$.

Sort all jobs in non-increasing order of their processing times and compute $P = \sum_{j=1}^{n} p_j$;
Modify batch machine M_i *into block* i *that with* b *unit capacity machines,* $i = 1$ *to* m;
Initialize $C_{\max}(\sigma_p^*(b)) = \max\{p_{\max}, P/(mb)\}$;
Let $C_{i,k}$ be the completion time of the kth unit capacity machine of the block i. Set $C_{i,k} = 0$, $i = 1, \ldots, m$, $k = 1, \ldots, b$;
Set $j = 1$;
for $i = 1$ *to* m **do**
  $k = 1$;
  while $C_{i,b} < C_{\max}(\sigma_p^*(b))$ *and* $j \le n$ **do**
    if $C_{i,k} + p_j < C_{\max}(\sigma_p^*(b))$ **then**
      Assign job J_j to the machine $M_{i,k}$;
      $C_{i,k} = C_{i,k} + p_j$;
      $j = j + 1$;
    else
      Assign the $C_{\max}(\sigma_p^*(b)) - t$ length of job J_j to the machine $M_{i,k}$; $C_{i,k} = C_{\max}(\sigma_p^*(b))$;
      $p_j = p_j - (C_{\max}(\sigma_p^*(b)) - t)$;
      $k = k + 1$;
    end
  end
end
Let $C_{[i,j]}$ be the completion time of job j processed in the block i. Set $C_{[i,j]} = 0$;
for $i = 1$ *to* m **do**
  Reorder jobs processed on machines $M_{i,1}, \ldots, M_{i,b}$ of the block i in the non-decreasing order of the completion times, i.e., $C_{[i,1]} \le \cdots \le C_{[i,n_i]}$;
  for $j = 1$ *to* n_i **do**
    Preempt other jobs processed in the block i at the time $C_{[i,j]}$;
    Schedule jobs in the time interval $[C_{[i,j-1]}, C_{[i,j]}]$ as a batch on the batch machine M_i;
  end
end

In Step 2, we assign job $J_1(5)$ and $J_2(1)$ on machine $M_{1,1}$, $J_2(3)$ and $J_3(3)$ on machine $M_{1,2}$, $J_3(1)$, $J_4(4)$ and $J_5(1)$ on machine $M_{2,1}$, $J_5(2)$, $J_6(3)$ and $J_7(1)$ on machine $M_{2,2}$. In Step 3, job $J_1(5)$ is split into $J_1(3)$ and $J_1(2)$ whose processing times are 3 and 2, respectively. $J_1(3)$ and $J_2(3)$ are grouped as batch $B_1 = \{J_1(3), J_2(3)\}$. Job $J_3(3)$ is split into $J_3(1)$ and $J_3(2)$. $J_1(2)$ and $J_3(2)$ are grouped as batch $B_2 = \{J_1(2), J_3(2)\}$. $J_2(1)$ and $J_3(1)$ are grouped as batch $B_3 = \{J_2(1), J_3(1)\}$. Job $J_5(2)$ is split into two sub-jobs with the same processing time, i.e., $J_5(1)$. $J_3(1)$ and $J_5(1)$ are grouped as batch $B_4 = \{J_3(1), J_5(1)\}$. Job $J_4(4)$ is split into $J_4(1)$ and $J_4(3)$. $J_4(1)$ and $J_5(1)$ are grouped as batch $B_5 = \{J_4(1), J_5(1)\}$. $J_4(3)$ and $J_6(3)$ are grouped as batch $B_6 = \{J_4(3), J_6(3)\}$. $J_5(1)$ and $J_7(1)$ are grouped as batch $B_7 = \{J_5(1), J_7(1)\}$. The makespan of the schedule is equal to 6. Fig. 4 shows a schedule for Example 1 by Algorithm 1.

For the time complexity of Algorithm 1, Step 1 requires $O(n \log n)$ time and remaining each step takes no greater than $O(n)$ time. Therefore, the overall time complexity of Algorithm 1 is

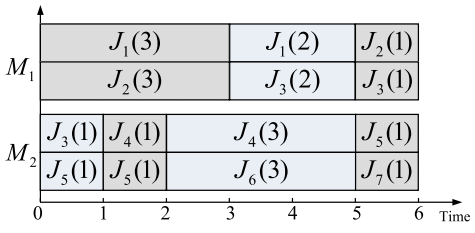


Fig. 4. Schedule obtained by Algorithm 1 for Example 1.

$O(n\log n)$. According to the description of Algorithm 1, the following Lemma can be seen to hold.

Lemma 1. *For problem $Pm|p\text{-}batch, pmtn, b < n|C_{\max}$, Algorithm 1 yields an optimal schedule σ^* in $O(n\log n)$ time, where $C_{\max}(\sigma_p^*(b)) = \max\{p_{\max}, P/(mb)\}$.*

With Lemma 1, we show how the minimum makespan can be improved when the machine capacity expands from b to $\hat{b}$.

Theorem 1. *For problem $Pm|p\text{-}batch, pmtn, b < n|C_{\max}$, when the machine capacity expands from b to $\hat{b}$, the best case impact ratio over optimal schedule of increasing machine capacity is equal to $\hat{b}/b$.*

Proof. Following Lemma 1, problem $Pm|p\text{-}batch, pmtn, b < n|C_{\max}$ has a minimum makespan of $\max\{p_{\max}, P/(mb)\}$. Thus, when machine capacity expands from b to $\hat{b}$, the best case impact ratio over optimal schedule of increasing machine capacity is no greater than $\hat{b}/b$.

We prove that this bound is tight by constructing an instance. We consider an instance with $n = m\hat{b}$ jobs having equal processing time p. When the machine capacity is $\hat{b}$, a schedule where each machine processes $\hat{b}$ jobs in a batch is optimal, achieving a makespan of p. When the machine capacity is b, Algorithm 1 obtains an optimal schedule with makespan $p\hat{b}/b$. Thus, the best case impact ratio over optimal schedule is equal to $\hat{b}/b$. □

4.2. Non-preemptive case

In this subsection, we analyze the best case impact ratio over optimal schedule of increasing machine capacity for the problem $Pm|p\text{-}batch, b < n|C_{\max}$ without preemption. We abbreviate makespans $C_{\max}(\sigma^*(b), I)$ and $C_{\max}(\sigma^*(\hat{b}), I)$ as $C^*_{\max}(b)$ and $C^*_{\max}(\hat{b})$, respectively.

Theorem 2. *For problem $Pm|p\text{-}batch, b < n|C_{\max}$ and $\hat{b} = ub + v$, where u and v are integers satisfying $u \geq 1$, $1 \leq v \leq b$, when the machine capacity expands from b to $\hat{b}$, the best case impact ratio over optimal schedule of increasing machine capacity is equal to $\left\lceil \hat{b}/b \right\rceil = u + 1$.*

Proof. Assume there are a_i batches on machine M_i of schedule $\sigma^*(\hat{b})$. Let batch $B^*_{ik}(\hat{b})$ be the kth batch on machine M_i of schedule $\sigma^*(\hat{b})$ and $P^*_{ik}(\hat{b})$ be the processing time of batch $B^*_{ik}(\hat{b})$, $1 \leq k \leq a_i$. Thus, we have $C^*_{\max}(\hat{b}) = \max_{1\leq i\leq m}\sum_{k=1}^{a_i} P^*_{ik}(\hat{b})$.

Based on schedule $\sigma^*(\hat{b})$, we construct a new schedule $\sigma(b)$ where each parallel batch machine processes the same set of jobs as in schedule $\sigma^*(\hat{b})$. Following the FBLPT rule, all jobs in batch $B^*_{ik}(\hat{b})$ can be grouped into multiple batches that each batch contains exactly b jobs except for the last one. Since the number of jobs in batch $B^*_{ik}(\hat{b})$ is no greater than $\hat{b}$ and $\hat{b} = ub + v$, batch $B^*_{ik}(\hat{b})$ can be split into at most $u + 1$ batches with machine capacity b, each with processing time no greater than $P^*_{ik}(\hat{b})$. We denote the makespan of schedule $\sigma(b)$ by $C_{\max}(b)$. By the construction process, the completion time of machine M_i of schedule $\sigma(b)$ is no greater than $(u+1)\sum_{k=1}^{a_i} P^*_{ik}(\hat{b})$. Thus, we have

$$C_{\max}(b) \leq \max_{1\leq i\leq m}(u+1)\sum_{k=1}^{a_i} P^*_{ik}(\hat{b}) = (u+1)C^*_{\max}(\hat{b}).$$

Therefore, we have

$$\rho^*(b,\hat{b}) \leq \frac{C_{\max}(b)}{C^*_{\max}(\hat{b})} \leq u+1.$$

We prove that this bound is tight by constructing an instance. Consider an instance with $n = m(ub+v) = m\hat{b}$ jobs having equal processing time p. When the machine capacity is $\hat{b}$, a schedule where each machine processes $\hat{b}$ jobs in one batch is optimal, with makespan $C^*_{\max}(\hat{b}) = p$. When the machine capacity is b, a schedule where each machine processes $ub+v$ jobs in $u+1$ batches is optimal, with the makespan $C^*_{\max}(b) = (u+1)p$. Thus, we have $C^*_{\max}(b)/C^*_{\max}(\hat{b}) = u+1$. □

Since problem $Pm|p\text{-}batch, b < n|C_{\max}$ is *NP*-hard [1], it is impossible to obtain an optimal schedule for a given instance in polynomial time unless *P=NP*. Instead, approximation solutions are used to analyze the best impact ratio over an approximate schedule of increasing machine capacity. As we know, there are three mainstream rules to yield approximation solutions as follows, the full batch list scheduling (FBLS) rule, the full batch longest processing time (FBLPT) rule and the full batch shortest processing time (FBSPT) rule. The FBLS rule assigns jobs with arbitrary sequence into full batches except possibly the last batch. The FBLS rule is a general method to solve batch scheduling problems. The FBLPT rule and the FBSPT rule are two specific FBLS rules with considering that jobs are sorted in non-increasing order and non-decreasing order of their processing times, respectively.

For the FBLPT rule, its worst case ratio has been investigated in [1], which is equal to $4/3 - 1/(3m)$. But for the FBSPT rule, there are no results on its worst case ratio. Here, we construct an instance as follows to show the worst case ratio of FBSPT rule is at least $2 - 1/m$.

There are $m(m-1)b$ jobs with processing time 1 and b jobs with processing time m. We use the FBSPT rule to obtain a schedule that $m(m-1)b$ jobs with processing time 1 are grouped as $m(m-1)$ batches which are dispatched averagely on m machines, and b jobs with processing time m are grouped as a batch which is processed on any available machine. The makespan of the schedule yielded by the FBSPT rule is equal to $(2m-1)$, as shown in Fig. 5a. In the optimal schedule, b jobs with processing time m are grouped as a batch which is processed on the first machine, and $m(m-1)b$ jobs with processing time 1 are grouped as $m(m-1)$ batches which are dispatched averagely on the last $(m-1)$ machines. The makespan of the optimal schedule is equal to m, as shown in Fig. 5b. The worst case ratio of FBSPT rule is at least $2 - 1/m$.

Overall, the worst case ratio of the FBLPT rule is equal to $4/3 - 1/(3m)$, and that of the FBSPT rule is at least $2 - 1/m$. Since $2 - 1/m \geq 4/3 - 1/(3m)$ for $m \geq 1$, we prioritize the FBLPT rule with better performance to analyze its impact ratio after increasing machine capacity. Therefore, we choose the FBLS rule and FBLPT rule to analyze in our paper.

For the best case impact ratio over approximate schedules by the FBLS rule and the FBLPT rule, we have the following Theorems 3 and 4, respectively.

Theorem 3. *For problem $Pm|p\text{-}batch, b < n|C_{\max}$, the best case impact ratio over approximate schedule by the FBLS rule is equal to $\hat{b} + 1 - 1/m$ when the machine capacity is upgraded from b to $\hat{b}$.*

Proof. For an instance I with machine capacity b, let $\sigma^{FBLS}(b, I)$ and $C^{FBLS}_{\max}(b, I)$, short for $\sigma^{FBLS}(b)$ and $C^{FBLS}_{\max}(b)$, respectively, be

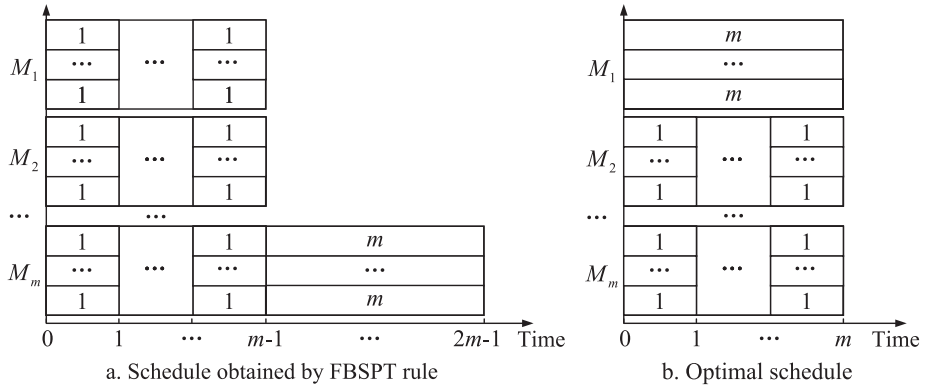


Fig. 5. Lower bound of the worst case ratio of the FBSPT rule.

the schedule and makespan obtained by the FBLS rule. We assume there are a_i $(1 \le i \le m)$ batches processed on machine M_i. Let $C_i^{FBLS}(b)$ be the completion time of schedule $\sigma^{FBLS}(b)$ on machine M_i. $C_i^{FBLS}(b)$ is equal to the total processing time of all batches on machine M_i. Let batch $B_{ik}(b)$ be the kth batch on machine M_i of schedule $\sigma^{FBLS}(b)$, and P_{ik} be the processing time of batch $B_{ik}(b)$.

Assume that machine M_l determines the makespan, and P_{la_l} is the processing time of the last batch on machine M_l. By the FBLS rule, the completion time of machine M_i $(i \ne l)$ is denoted by $C_i^{FBLS}(b)$ which is at least $C_{\max}^{FBLS}(b) - P_{la_l}$. Then, we have

$$\sum_{i=1}^{m} C_i^{FBLS}(b) = \sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik} \ge (m-1)\left(C_{\max}^{FBLS}(b) - P_{la_l}\right) + C_{\max}^{FBLS}(b). \quad (1)$$

Since the processing time of a batch is equal to the longest processing time of the jobs in the batch, the total processing time of all the batches is no greater than that of all the jobs. We have $\sum_{j=1}^{n} p_j \ge \sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik}$.

Further, according to inequality (1), we have

$$C_{\max}^{FBLS}(b) \le \frac{\sum_{j=1}^{n} p_j}{m} + \frac{(m-1)P_{la_l}}{m}.$$

We consider a schedule $\sigma^{FBLS}(\hat{b})$ obtained by the FBLS rule with machine capacity $\hat{b}$. Note that $C_{\max}^{FBLS}(\hat{b}) \ge \sum_{j=1}^{n} p_j/(m\hat{b})$. Also, $C_{\max}^{FBLS}(\hat{b}) \ge P_{la_l}$. Therefore, we have

$$\frac{C_{\max}^{FBLS}(b)}{C_{\max}^{FBLS}(\hat{b})} \le \hat{b} + 1 - \frac{1}{m}.$$

Specifically, we construct an instance to show the bound is tight as in Table 1.

When machine capacity is $\hat{b}$, we adopt the FBLS rule to yield batches in Table 2. The batch with processing time m is processed on the first machine. The $m(m-1)$ batches with processing time 1 are assigned averagely to the remaining $(m-1)$ machines.

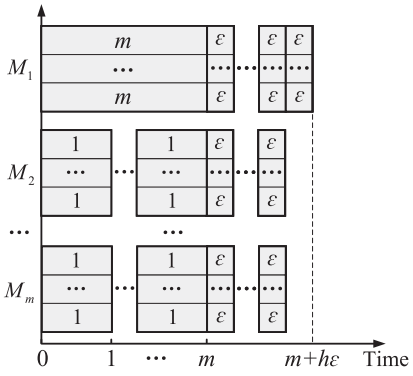


Fig. 6. Schedule obtained by the FBLS rule with machine capacity $\hat{b}$.

The remaining $(m^2-m+1)(b-1)$ batches with processing time ε are assigned successively to available machines. We obtain the makespan $C_{\max}^{FBLS}(\hat{b}) = m + h\varepsilon$, where $h = \lceil (m^2-m+1)(b-1)/m \rceil$, as shown in Fig. 6.

When machine capacity is b, we adopt the FBLS rule to yield batches in Table 3. We assign $(\hat{b}-1)$ batches with processing time m and $(m-1)$ batches with processing time 1 to the first machine. The $(m\hat{b}-1)(m-1)$ batches with processing time 1 are assigned averagely to the remaining $(m-1)$ machines. The last batch with processing time m is processed on any available machine. We obtain the makespan $C_{\max}^{FBLS}(b) = m\hat{b} + m - 1$, as shown in Fig. 7.

Therefore, we have the tight bound as follows.

$$\frac{C_{\max}^{FBLS}(b)}{C_{\max}^{FBLS}(\hat{b})} = \lim_{\varepsilon \to 0} \frac{m\hat{b} + m - 1}{m + h\varepsilon} = \hat{b} + 1 - \frac{1}{m}.$$

This proves the theorem. □

We investigate the best case impact ratio over an approximate schedule by the FBLPT rule.

Theorem 4. *For problem $Pm|p\text{-}batch, b < n|C_{\max}$, the best case impact ratio over approximate schedule by the FBLPT rule is no greater than $\lambda = \max\{\lceil \hat{b}/b \rceil, \hat{b}/b + 1/2 + 1/(2m) - 1/(mb)\}$ when the machine capacity is upgraded from b to $\hat{b}$.*

Table 1
Tight instance of Theorem 3.

Number of jobs	$\hat{b}$	$m\hat{b}(m-1)$	$\hat{b}(m^2-m+1)(b-1)$
Processing time of job	m	1	ε

Table 2
Processing time of batch with machine capacity $\hat{b}$.

Amount of batches	1	$m(m-1)$	$(m^2-m+1)(b-1)$
Processing time of batch	m	1	ε

Table 3
Processing time of batch with machine capacity b.

Amount of batches	$\hat{b}$	$m\hat{b}(m-1)$
Processing time of batch	m	1

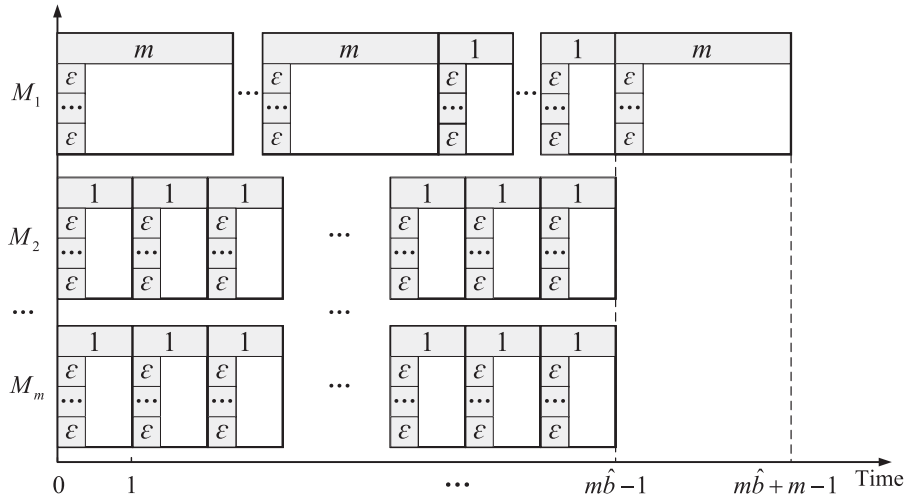


Fig. 7. Schedule obtained by the FBLS rule with machine capacity b.

Proof. We prove the result by contradiction. Assume that there exists counterexamples where the makespan obtained by the FBLPT rule with machine capacity b is strictly greater than λ times that with machine capacity $\hat{b}$. Consider a counterexample, denoted by instance I_1, with the smallest number of jobs. Suppose there are n_1 jobs in instance I_1.

For instance I_1, let $\sigma_1^{FBLPT}(b)$ and $\sigma_1^{FBLPT}(\hat{b})$ be the schedules obtained by the FBLPT rule with machine capacities b and $\hat{b}$, respectively. For schedule $\sigma_1^{FBLPT}(b)$, we assume that there are a_i $(1 \le i \le m)$ batches processed on machine M_i. Let $B_{ik}(b)$ denote the kth batch on machine M_i in schedule $\sigma_1^{FBLPT}(b)$. Let $C_{\max}^{FBLPT}(b)$ be the makespan obtained by the FBLPT rule for instance I_1. For the last completed batch in schedule $\sigma_1^{FBLPT}(b)$, we have the following claim.

Claim. Only one batch completes at $C_{\max}^{FBLPT}(b)$ and it consists of only one job.

We prove the claim by contradiction. Suppose the claim does not hold. Consider all batches that complete at $C_{\max}^{FBLPT}(b)$. Let B_{la_l} be the batch that determines the makespan with the earliest start time and suppose that the start time is t^*. Except for the longest job in batch B_{la_l}, delete all jobs that start no earlier than t^* from schedule $\sigma_1^{FBLPT}(b)$. We then obtain a new instance I_2 with a smaller number of jobs. The processing times of jobs that start earlier than t^* are larger than the longest job in batch B_{la_l} and all batches that start earlier than t^* are full batch. Therefore, when the machine capacity is b, the makespan of instance I_2 is equal to $C_{\max}^{FBLPT}(b)$. When the machine capacity is $\hat{b}$, the makespan of instance I_2 is no greater than that of instance I_1. Therefore, the impact ratio over the approximate schedule by the FBLPT rule for instance I_1 is no greater than that for instance I_2. Hence, instance I_2 is a counterexample with the number of jobs less than that of instance I_1, violating our assumption.

For schedule $\sigma_1^{FBLPT}(b)$, similar to inequality (1), we have

$$\sum_{i=1}^{m} C_i^{FBLPT}(b) = \sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik} \ge (m-1)\left(C_{\max}^{FBLPT}(b) - P_{la_l}\right) + C_{\max}^{FBLPT}(b). \tag{2}$$

Consider the batches obtained by removing the longest processing time job in each batch in schedule $\sigma_1^{FBLPT}(b)$. The total processing time of the remaining jobs in these batches is equal to $\sum_{j=1}^{n} p_j - \sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik}$. In schedule $\sigma_1^{FBLPT}(b)$, rearrange all the batches in a non-increasing order of their processing times, and denote the sequenced batches by $B_{[1]}, B_{[2]}, \ldots, B_{[r]}$, where $r = \sum_{i=1}^{m} a_i$. Let $P_{[l]}$ be the processing time of batch $B_{[l]}$, for $l = 1, \ldots, r$. Then, we have $\sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik} = \sum_{l=1}^{r} P_{[l]}$, $P_{[1]} = P_{11}$ and $P_{[r]} = P_{la_l}$. Since all the batches are obtained by the FBLPT rule, we have $P_{[1]} \ge P_{[2]} \ge \cdots \ge P_{[r]}$. The processing time of each job in batch $B_{[l]}$ is greater than or equal to the largest processing time of the job in batch $B_{[g]}$, where g is strictly greater than l. Thus, we have

$$\begin{aligned}\sum_{j=1}^{n} p_j - \sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik} &= \sum_{j=1}^{n} p_j - \sum_{l=1}^{r} P_{[l]} \\ &\ge (b-1)\sum_{l=2}^{r} P_{[l]} \\ &= (b-1)\left(\sum_{i=1}^{m}\sum_{k=1}^{a_i} P_{ik} - P_{11}\right).\end{aligned} \tag{3}$$

According to inequations (2) and (3), we obtain an upper bound of $C_{\max}^{FBLPT}(b)$

$$C_{\max}^{FBLPT}(b) \le \frac{\sum_{j=1}^{n} p_j}{mb} + \frac{(b-1)P_{11}}{mb} + \frac{m-1}{m} P_{la_l}. \tag{4}$$

Noting that $C_{\max}^{FBLPT}(\hat{b}) \ge P_{11}$ and $C_{\max}^{FBLPT}(\hat{b}) \ge \sum_{j=1}^{n} p_j/(m\hat{b})$, we have

$$\frac{C_{\max}^{FBLPT}(b)}{C_{\max}^{FBLPT}(\hat{b})} \le \frac{\hat{b}}{b} + \frac{1}{m} - \frac{1}{mb} + \frac{(m-1)P_{la_l}}{mC_{\max}^{FBLPT}(\hat{b})}.$$

We distinguish the following three cases to discuss.

Case 1. $C_{\max}^{FBLPT}(\hat{b}) \ge 2P_{la_l}$.

In this case, we have

$$\begin{aligned}\frac{C_{\max}^{FBLPT}(b)}{C_{\max}^{FBLPT}(\hat{b})} &\le \frac{\hat{b}}{b} + \frac{1}{m} - \frac{1}{mb} + \frac{(m-1)P_{la_l}}{mC_{\max}^{FBLPT}(\hat{b})} \\ &\le \frac{\hat{b}}{b} + \frac{1}{2} + \frac{1}{2m} - \frac{1}{mb}.\end{aligned}$$

Case 2. $C_{\max}^{FBLPT}(\hat{b}) < 2P_{la_l}$, $\hat{b} = ub$, where u is a positive integer.

When the machine capacity is $\hat{b}$, the job in batch B_{la_l} is the shortest job in set $\mathcal{J}$. Note that $C_{\max}^{FBLPT}(\hat{b}) < 2P_{la_l}$, which implies that each machine processes at most one batch. Thus, the makespan is determined by batch B_{11}, i.e., $C_{\max}^{FBLPT}(\hat{b}) = P_{11}$.

Using schedule $\sigma^{FBLPT}(\hat{b})$, we construct a schedule $\sigma(b)$ for the problem with the machine capacity b. Rearrange all jobs in batch $B_{i1}(\hat{b})$ using the FBLPT rule. Denote split batches processed

on machine M_i by $D_{i1}(b), D_{i2}(b), \ldots, D_{iu}(b)$. The processing time of batch $D_{i1}(b)$, denoted by $Q_{i1}(b)$, is no greater than $P_{i1}(\hat{b})$. Thus, the completion time of schedule $\sigma(b)$ on machine M_i is no greater than $uP_{i1}(\hat{b})$, i.e., $\sum_{s=1}^{u} Q_{is}(b) \le uP_{i1}(\hat{b})$. Since $\hat{b} = ub$ and u is an integer, the batches in schedule $\sigma(b)$ are exactly the batches in schedule $\sigma^{FBLPT}(b)$. Since batch $B_{11}(\hat{b})$ is the first batch of schedule $\sigma^{FBLPT}(\hat{b})$, the largest u batches in schedule $\sigma^{FBLPT}(b)$ are $D_{11}(b), D_{12}(b), \ldots, D_{1u}(b)$.

Since the number of jobs is no greater than $m\hat{b}$, there are at most mu batches in schedule $\sigma^{FBLPT}(b)$. If each machine processes at most u batches in schedule $\sigma^{FBLPT}(b)$, then we have $C_{\max}^{FBLPT}(b) = \sum_{k=1}^{a_l} P_{lk}(b) \le \sum_{s=1}^{u} Q_{ls}(b) \le uP_{l1} \le uC_{\max}^{FBLPT}(\hat{b})$. If a machine processes more than u batches in schedule $\sigma^{FBLPT}(b)$, then there exists a machine processing less than u batches in schedule $\sigma^{FBLPT}(b)$. Assume that machine M_t processes less than u batches in schedule $\sigma^{FBLPT}(b)$, i.e., $a_t \le u-1$. Let $P_{\min}$ denote the minimum processing time of batches in schedule $\sigma^{FBLPT}(b)$. According to the FBLPT rule, we have $C_{\max}^{FBLPT}(b) \le \sum_{k=1}^{a_t} P_{tk}(b) + P_{\min} \le \sum_{s=1}^{u} Q_{1s}(b) \le uC_{\max}^{FBLPT}(\hat{b})$.

Case 3. $C_{\max}^{FBLPT}(\hat{b}) < 2P_{la_l}$, $\hat{b} = ub + v$, $1 \le v \le b-1$, where u and v are integers.

Similarly to Case 2, we construct a schedule $\sigma(b)$ for the problem with the machine capacity b based on schedule $\sigma^{FBLPT}(\hat{b})$. In schedule $\sigma(b)$, every machine processes at most $u+1$ batches, where the last batch on each machine may not be full. The $u+1$ batches on machine M_1 of schedule $\sigma(b)$ are the longest $u+1$ batches of schedule $\sigma(b)$, and $\sum_{k=1}^{u+1} Q_{1k}(b) \le (u+1)P_{11}(\hat{b})$. The total processing time of the longest $u+1$ batches of schedule $\sigma(b)$ is equal to $\sum_{k=1}^{u+1} Q_{1k}(b)$. Similar to the discussion in Case 2, we have $C_{\max}^{FBLPT}(b) \le (u+1)C_{\max}^{FBLPT}(\hat{b})$.

Overall, we have that the impact ratio over the approximate schedule of instance I_1 is no greater than λ, which leads to a contradiction. □

5. Cost-effective choice of machine capacity

In the above section, we derive bounds on capacity augmentation impact that show how the makespan is affected by increased machine capacity. Increasing machine capacity can boost scheduling performance, but also leads to higher capacity costs. In this section, we consider both the makespan and capacity costs, and make a decision to optimize the total costs. The capacity cost is assumed to be a linear function of the machine capacity. The machine capacity b is a decision variable. We aim to minimize the weighted sum of the makespan and capacity costs.

Specifically, we assume that a batch machine with capacity b incurs a cost of Kb, where K is a given value. The total cost function consists of the makespan and capacity costs, namely $\Phi(b) = w_1 C_{\max}(\sigma(b)) + w_2 Kmb$. $C_{\max}(\sigma(b))$ denotes the makespan of schedule $\sigma(b)$, and w_1 and w_2 denote weights of makespan and capacity costs, respectively. The total objective function can be scaled as $\varphi(b) = C_{\max}(\sigma(b)) + \beta mb$, where $\beta = w_2 K/w_1$ is the scaled coefficient of capacity costs.

For the preemptive case, Algorithm 2 and Algorithm 3 are designed to capture the trade-off between the makespan and capacity costs, and obtain the preemptive optimal machine capacity b_p^* in polynomial time. To decide the schedule of the non-preemptive case, we need to determine the machine capacity b, which is *NP*-hard. We present an approximation algorithm that inputs the machine capacity b_p^* found by Algorithm 3 in the preemptive case.

5.1. Preemptive case

We study the scheduling problem with preemption to minimize the total cost function. We denote the total cost function by $\varphi_p(b)$, where the subscript p indicates preemption. By definition, $\varphi_p(b) = C_{\max}(\sigma_p(b)) + \beta mb$, where $\sigma_p(b)$ is a preemptive schedule and $C_{\max}(\sigma_p(b))$ is the makespan of schedule $\sigma_p(b)$. Let b_p^* denote the optimal machine capacity with preemption. We present Algorithm 2 to yield the optimal machine capacity b_p^* as follows.

Algorithm 2: Optimal machine capacity.

Input : Set of jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, set of identical parallel batch machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$, and coefficient of capacity costs β;

Output: The optimal machine capacity b_p^*.

Initialize $b = 1$ and compute $P = \sum_{j=1}^{n} p_j$;
for b= *1 to n* **do**
  if $p_{\max} \ge P/mb$ **then**
    compute $\varphi_p(b) = p_{\max} + \beta mb$;
  else
    compute $\varphi_p(b) = P/mb + \beta mb$;
  end
end
$b_p^* = \text{argmin}_{1 \le b \le n} \varphi_p(b)$. Output the optimal machine capacity b_p^*.

Example 2. We give an example using the same data from Example 1 and let β=0.3.

Following Algorithm 2, we obtain the optimal machine capacity. In Step 1, we initialize the machine capacity $b = 1$ and compute $P = \sum_{j=1}^{n} p_j = 24$. In Step 2, we compute $\varphi_p(1) = P/mb + \beta mb = 12.6$, $\varphi_p(2) = P/mb + \beta mb = 7.2$, $\varphi_p(3) = p_{\max} + \beta mb = 6.8$, $\varphi_p(4) = p_{\max} + \beta mb = 7.4$, $\varphi_p(5) = p_{\max} + \beta mb = 8$, $\varphi_p(6) = p_{\max} + \beta mb = 8.6$ and $\varphi_p(7) = p_{\max} + \beta mb = 9.2$. In Step 3, $b_p^* = \text{argmin}_{1 \le b \le n} \varphi_p(b) = 3$.

In order to use properties of the optimal machine capacity, we provide Algorithm 3 to obtain the candidate machine capacity set for bounding the approximation ratio of Algorithm 4 for non-preemptive case. Define b_1 as the minimum machine capacity by which the maximum job processing time is no less than the mean machine load, i.e., $b_1 = \lceil P/(mp_{\max}) \rceil$, where $P = \sum_{j=1}^{n} p_j$ and $p_{\max} = \max_{1 \le j \le n} p_j$. Following Lemma 1, we rewrite function $\varphi_p(b)$ as

$$\varphi_p(b) = \begin{cases} f_1(b), & b \ge b_1; \\ f_2(b), & 1 \le b < b_1, \end{cases}$$

where $f_1(b) = p_{\max} + \beta mb$, and $f_2(b) = P/(mb) + \beta mb$.

We provide Algorithm 3 to obtain the optimal machine capacity b_p^* for the preemptive case as follows.

Algorithm 3

Input: Jobs of set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$; identical parallel batch machines of set $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$; coefficient of capacity costs β.

Output: The optimal machine capacity b_p^*.

Step 1. Compute $b_0 = P/(mp_{\max})$ and $\mu = \sqrt{P/(\beta m^2)}$. Define $b_1 = \lceil b_0 \rceil$, $b_2 = b_1 - 1$, $b_3 = \lfloor \mu \rfloor$ and $b_4 = \lceil \mu \rceil$.

Step 2. If $b_0 \le \mu$, then go to Step 3. If $\mu < b_0 \le \lceil \mu \rceil$, go to Step 4. If $b_0 > \lceil \mu \rceil$, go to Step 5.

Step 3. Compute $f_2(b_2)$ and $f_1(b_1)$. If $f_2(b_2) \le f_1(b_1)$, then let $b_p^* = b_2$, otherwise $b_p^* = b_1$. Go to Step 6.

Step 4. Compute $f_2(b_3)$ and $f_1(b_1)$. If $f_2(b_3) \le f_1(b_1)$, then let $b_p^* = b_3$, otherwise $b_p^* = b_1$. Go to Step 6.

Step 5. Compute $f_2(b_3)$ and $f_2(b_4)$. If $f_2(b_3) \le f_2(b_4)$, then let $b_p^* = b_3$, otherwise let $b_p^* = b_4$. Go to Step 6.

Step 6. Output the optimal machine capacity b_p^*.

The $f_1(b)$ is a monotonically increasing function of b. The $f_2(b)$ is a first monotonically decreasing and then monotonically increas-

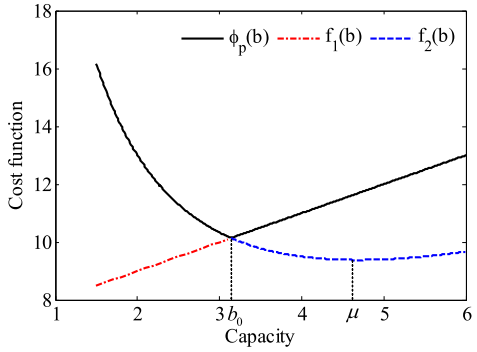


Fig. 8. Example 3.1 to illustrate Case 1.

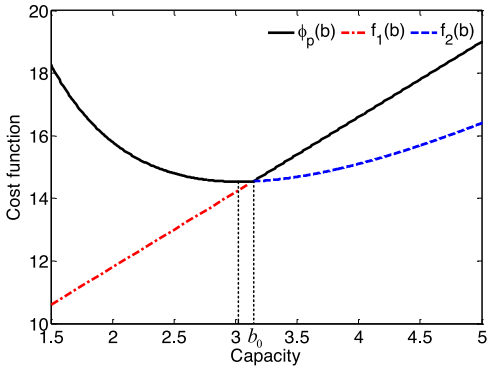


Fig. 9. Example 3.2 to illustrate Case 2.

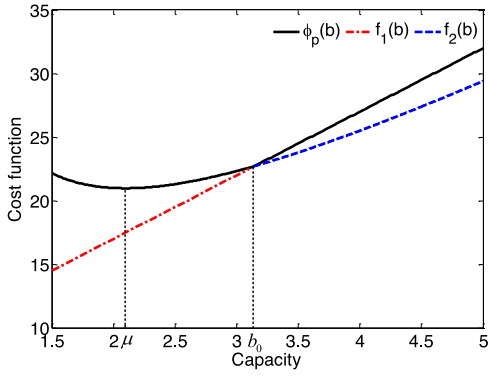


Fig. 10. Example 3.3 to illustrate Case 3.

ing function of b, and achieves its minimum either at $\lfloor\mu\rfloor$ or $\lceil\mu\rceil$. The $f_1(b)$ and $f_2(b)$ intersect at $b_0 = P/(mp_{\max})$. The total cost function $\varphi_p(b)$ consists of the right-hand side of $f_1(b)$ and the left-hand side of $f_2(b)$. We discuss the total cost function $\varphi_p(b)$ as following three cases.

$$\varphi_p(b_p^*) = \begin{cases} \min\{f_1(b_1), f_2(b_1-1)\}, & b_0 \le \mu; \\ \min\{f_2(\lfloor\mu\rfloor), f_1(b_1)\}, & \mu < b_0 \le \lceil\mu\rceil; \\ \min\{f_2(\lfloor\mu\rfloor), f_2(\lceil\mu\rceil)\}, & b_0 > \lceil\mu\rceil. \end{cases}$$

Case 1. $b_0 \le \mu$. When $b < b_0$, the function $\varphi_p(b)$ is monotonically decreasing with minimum value $f_2(b_1 - 1)$. While $b \ge b_0$, the function $\varphi_p(b)$ is monotonically increasing with minimum value $f_1(b_1)$. Thus, the minimum value of $\varphi_p(b)$ is $\min\{f_2(b_1 - 1), f_1(b_1)\}$. We give the Example 3.1 to illustrate case 1.

Example 3.1. Let $n = 10$, $m = 2$ and $\beta = 0.5$. The processing times of jobs are $p_1 = p_2 = 7$, $p_3 = p_4 = 6$, $p_5 = p_6 = 4$, $p_7 = p_8 = 3$, $p_9 = p_{10} = 2$.

Following Algorithm 3, we obtain the optimal machine capacity. In Step 1, we compute $b_0 = P/(mp_{\max}) \approx 3.14$ and $\mu = \sqrt{P/(\beta m^2)} \approx 4.69$. Let $b_1 = 4$ and $b_2 = b_1 - 1 = 3$. In Step 2, for $b_0 \le \mu$, go to Step 3. In Step 3, we compute $f_1(b_1) = 11$ and $f_2(b_2) \approx 10.33$. For $f_1(b_1) > f_2(b_2)$, let $b_p^* = b_2$. In Step 6, output the optimal machine capacity b_p^*, as shown in Fig. 8.

Case 2. $\mu < b_0 \le \lceil\mu\rceil$. When $b < \mu$, the function $\varphi_p(b)$ is monotonically decreasing with minimum value $f_2(\lfloor\mu\rfloor)$. While $b \ge \mu$, the function $\varphi_p(b)$ is monotonically increasing with minimum value $f_1(b_1)$. Thus, the minimum value of $\varphi_p(b)$ is $\min\{f_2(\lfloor\mu\rfloor), f_1(b_1)\}$. We give the Example 3.2 to illustrate case 2.

Example 3.2. We give an example using the same data from Example 3.1 and reset $\beta = 1.2$.

Following Algorithm 3, we obtain the optimal machine capacity. In Step 1, we compute $b_0 = P/(mp_{\max}) = 3.14$ and $\mu = \sqrt{P/(\beta m^2)} \approx 3.03$. Let $b_p^* = b_1 = 4$ and $b_3 = \lfloor\mu\rfloor = 3$. In Step 2, for $\mu < b_0 \le \lceil\mu\rceil$, go to Step 4. In Step 4, we compute $f_1(b_1) = 16.6$ and $f_2(b_3) \approx 14.53$. For $f_2(b_3) < f_1(b_1)$, let $b_p^* = b_3$. In Step 6, output the optimal machine capacity b_p^* as shown in Fig. 9.

Case 3. $b_0 > \lceil\mu\rceil$. When $b < \mu$, the function $\varphi_p(b)$ is monotonically decreasing with minimum value $f_2(\lfloor\mu\rfloor)$. While $b \ge \mu$, the function $\varphi_p(b)$ is monotonically increasing with minimum value $f_2(\lceil\mu\rceil)$. Thus, the minimum value of $\varphi_p(b)$ is $\min\{f_2(\lfloor\mu\rfloor), f_2(\lceil\mu\rceil)\}$. We give the Example 3.3 to illustrate case 3.

Example 3.3. We give an example using the same data from Example 3.1 and reset $\beta = 2.5$.

Following Algorithm 3, we obtain the optimal machine capacity. In Step 1, we compute $b_0 = P/(mp_{\max}) = 3.14$ and $\mu = \sqrt{P/(\beta m^2)} \approx 2.10$. Let $b_p^* = b_1 = 4$, $b_3 = \lfloor\mu\rfloor = 2$ and $b_4 = \lceil\mu\rceil = 3$. In step 2, for $b_0 > \lceil\mu\rceil$, go to Step 5. In Step 5, we compute $f_2(b_3) = 21$ and $f_2(b_4) \approx 22.33$. For $f_2(b_3) < f_2(b_4)$, let $b_p^* = b_3$. In Step 6, output the optimal machine capacity b_p^* as shown in Fig. 10.

Theorem 5. *For problem $Pm|p\text{-}batch, pmtn, b < n|\varphi$, Algorithm 3 finds an optimal machine capacity b_p^* and the corresponding minimum total cost $\varphi_p(b_p^*)$ in $O(n)$ time.*

5.2. Non-preemptive case

We consider capacity optimization for problem $Pm|p\text{-}batch, b < n|\varphi$. For the intractability of the problem, we design an approximation algorithm. We denote the total cost by $\varphi(b)$. By definition, we have $\varphi(b) = C_{\max}(\sigma(b)) + \beta mb$, where $\sigma(b)$ is a feasible non-preemptive schedule with capacity b. Let b^* denote the optimal capacity, i.e., $\varphi(b^*) \le \varphi(b)$ for $b \ge 1$. We start with the machine capacity found by Algorithm 3, and then use a specific rule to assign jobs to machines. We then compare the difference between the makespans of the obtained non-preemptive schedule and an optimal preemptive schedule for the same machine capacity. We discuss the *power of preemption* which is defined as the maximum ratio $C_{\max}(\sigma^*(b))/C_{\max}(\sigma_p^*(b))$ across all instances for problem $Pm|p\text{-}batch, b < n|C_{\max}$.

Lemma 2. *For problem $Pm|p\text{-}batch, b < n|C_{\max}$, the power of preemption is no greater than $2 - 1/(mb)$.*

Proof. For any instance, noting that $C_{\max}(\sigma^*(b)) \le C_{\max}(\sigma(b))$, we have

$$\frac{C_{\max}(\sigma^*(b))}{C_{\max}(\sigma_p^*(b))} \le \frac{C_{\max}(\sigma(b))}{C_{\max}(\sigma_p^*(b))},$$

where $C_{\max}(\sigma^*(b))$ and $C_{\max}(\sigma_p^*(b))$ denote the non-preemptive and preemptive optimal makespan, respectively.

We need to find a schedule $\sigma(b)$ that satisfies $C_{\max}(\sigma(b))/C_{\max}(\sigma_p^*(b)) \le 2 - 1/(mb)$.

$$C_{\max}(\sigma(b)) \le \frac{\sum_{j=1}^{n} p_j}{mb} + \frac{(b-1)P_{11}}{mb} + \frac{m-1}{m} P_{[a_l]}.$$

Note that a lower bound of $C_{\max}(\sigma_p^*(b))$ is $\max\{p_{\max}, \sum_{j=1}^{n} p_j/(mb)\}$. Thus, we have

$$\frac{C_{\max}(\sigma^*(b))}{C_{\max}(\sigma_p^*(b))} \le \frac{C_{\max}(\sigma(b))}{C_{\max}(\sigma_p^*(b))} \le 1 + \frac{b-1}{mb} + \frac{m-1}{m} = 2 - \frac{1}{mb}.$$

Hence, the lemma holds. □

In the following, we design an approximation algorithm to obtain the machine capacity and the schedule for the non-preemptive case. Thus, the machine capacity b and job sequence are needed to be determined. Specifically, we adopt the optimal machine capacity b_p^* of the preemptive case as the machine capacity of the non-preemptive case, and use the FBLPT rule to find a non-preemptive schedule.

Algorithm 4

Input: Jobs of set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$; identical parallel batch machines of set $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$; coefficient of capacity costs β.

Output: Machine capacity b and the corresponding total cost $\varphi(b)$.

Step 1. Use Algorithm 3 to find machine capacity b_p^* and define $b = b_p^*$.

Step 2. Use the FBLPT rule to find a non-preemptive schedule $\sigma(b)$. Output b and $\varphi(b)$.

Example 4. We give an example using the same data from Example 3.1.

Following Algorithm 4, we obtain the machine capacity and a non-preemptive schedule. In Step 1, we obtain the preemptive machine capacity $b_p^* = b_2 = 3$ by Algorithm 3 and set machine capacity $b = b_p^* = 3$. In Step 2, we use FBLPT rule to yield a non-preemptive schedule $\sigma(b)$ and compute the total cost $\varphi(b) = 12$. Output the machine capacity $b = 3$ and the total cost $\varphi(b) = 12$.

To capture the trade-off between the makespan and capacity costs for the non-preemptive case, we present Algorithm 4 as above and next analyze its worst case ratio. Specifically, the worst case performance ratio of Algorithm 4 depends on the value b_p^*, where b_p^* belongs to $\{b_1, b_2, b_3, b_4\}$ obtained by Algorithm 3. The optimal makespan of the preemptive case is used as a lower bound for the non-preemptive case. We also use the result of the power of preemption, i.e., the ratio between the makespan of non-preemptive and that of preemptive schedules, to scale inequality in case $b_p^* \in \{b_1, b_2\}$.

Lemma 3. *If $b_p^* \in \{b_1, b_2\}$ found by Algorithm 3, then Algorithm 3 delivers*

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le 2 - \frac{1}{mb}.$$

Proof. Since the makespan of an optimal non-preemptive schedule with capacity b is no smaller than that of an optimal preemptive schedule with the same machine capacity, we have that $\varphi^*(b^*) \ge \varphi_p(b^*) \ge \varphi_p(b_p^*)$. We consider two cases.

Case 1. $b_p^* = b_2 = b_1 - 1$. We have $C_{\max}(\sigma_p^*(b_2)) = P/(mb_2) > p_{\max}$, and then

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{\varphi(b_2)}{\varphi_p(b_2)} = \frac{C_{\max}(\sigma(b_2)) + \beta m b_2}{C_{\max}(\sigma_p^*(b_2)) + \beta m b_2} \le \frac{C_{\max}(\sigma(b_2))}{C_{\max}(\sigma_p^*(b_2))} \le 2 - \frac{1}{mb}.$$

Case 2. $b_p^* = b_1$. We have $C_{\max}(\sigma(b_1)) = C_{\max}(\sigma_p^*(b_1)) = p_{\max} > P/(mb_1)$, and then

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{\varphi(b_1)}{\varphi_p(b_1)} = \frac{C_{\max}(\sigma(b_1)) + \beta m b_1}{C_{\max}(\sigma_p^*(b_1)) + \beta m b_1} = 1 \le 2 - \frac{1}{mb}.$$

Hence, the lemma holds. □

Lemma 4. *If $b_p^* = b_3$ found by Algorithm 3, then Algorithm 4 delivers*

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{3}{2} + O\left(\frac{1}{b_p^*}\right).$$

Proof. When $b_p^* = b_3 = \lfloor \mu \rfloor$, we have $C_{\max}(\sigma_p^*(b_3)) = P/(mb_3) > p_{\max}$, and then

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{\varphi(b_3)}{f_2(b_3)} = \frac{C_{\max}(\sigma(b_3)) + \beta m b_3}{P/(mb_3) + \beta m b_3} \le \frac{(2 - 1/(mb_3))P/(mb_3) + \beta m b_3}{P/(mb_3) + \beta m b_3},$$

where the inequality follows the proof of Lemma 2.

For $P/(mb_4) < \beta m b_4$ and $b_4 = 1 + b_3$, we have $Pb_3/(m(b_3+1)^2) < \beta m b_3$, thus

$$\begin{aligned}\frac{\varphi(b)}{\varphi^*(b^*)} &\le \frac{(2 - 1/(mb_3))P/(mb_3) + \beta m b_3}{P/(mb_3) + \beta m b_3} \\ &\le \frac{(2 - 1/(mb_3))P/(mb_3) + Pb_3/(m(b_3+1)^2)}{P/(mb_3) + Pb_3/(m(b_3+1)^2)} \\ &= \frac{3}{2} + \frac{b_3 + 1/2 - (b_3+1)^2/(mb_3)}{2b_3^2 + 2b_3 + 1} \\ &= \frac{3}{2} + O\left(\frac{1}{b_p^*}\right).\end{aligned}$$

Hence, the lemma holds. □

Lemma 5. *If $b_p^* = b_4$ found by Algorithm 3, then Algorithm 4 delivers*

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{3}{2} - O\left(\frac{1}{b_p^*}\right).$$

Proof. When $b_p^* = b_4 = \lceil \mu \rceil$, we have $C_{\max}(\sigma_p^*(b_4)) = P/(mb_4) > p_{\max}$, and then

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{\varphi(b_4)}{f_2(b_4)} = \frac{C_{\max}(\sigma(b_4)) + \beta m b_4}{P/(mb_4) + \beta m b_4} \le \frac{(2 - 1/(mb_4))P/(mb_4) + \beta m b_4}{P/(mb_4) + \beta m b_4},$$

where the second inequality follows the proof of Lemma 2.

Noting that $b_4 \ge \mu = \sqrt{P/(\beta m^2)}$, we have $\beta m b_4 \ge P/(mb_4)$, and thus

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \frac{(2 - 1/(mb_4))P/(mb_4) + P/(mb_4)}{2P/(mb_4)} = \frac{3}{2} - O\left(\frac{1}{b_p^*}\right).$$

Hence, the lemma holds. □

Given Lemmas 3–5, we have the following Theorem.

Theorem 6. *For problem $Pm|p\text{-}batch, b < n|\varphi$, Algorithm 3 finds capacity b and a corresponding schedule with total costs satisfying*

$$\frac{\varphi(b)}{\varphi^*(b^*)} \le \max\left\{2 - \frac{1}{mb}, \frac{3}{2} + O\left(\frac{1}{b_p^*}\right)\right\}.$$

6. Conclusion and future work

We consider the problem of scheduling jobs on identical parallel batch machines to minimize the makespan. We study the impact of increasing machine capacity on the makespan for parallel

batch machines. Furthermore, considering capacity costs, we need to capture the trade-off between the makespan and capacity costs.

Our contributions are threefold. First, we define two performance ratios to theoretically characterize the impact of increasing machine capacity on the makespan for parallel batch machines, and deduce corresponding upper bound. For the purpose of making a cost-effective choice of machine capacity, we minimize the weighted sum of the makespan and capacity costs. Second, for the preemptive case, we provide optimal polynomial time algorithms to obtain the optimal machine capacity. Third, for the non-preemptive case, we develop an approximation algorithm that uses the optimal machine capacity found in the preemptive case and analyze its worst case performance.

For future work, there are some topics that need to be addressed. First, manufacturers can explore other heuristics and study corresponding impact ratios. Second, based on Lemma 5, the worst case performance of Algorithm 3 obtains a tighter bound in this scenario. It deserves investigation to sharpen the bound of Algorithm 3 established in Theorem 6 overall scenarios. Third, there are other scheduling objectives to study, such as minimizing the total weighted completion time.

CRediT authorship contribution statement

Jun Xu: Writing – original draft, Methodology, Formal analysis. **Jun-Qiang Wang:** Supervision, Conceptualization, Methodology, Writing – review & editing, Funding acquisition. **Zhixin Liu:** Methodology, Validation, Writing – review & editing.

Acknowledgments

We sincerely thank Editor-in-Chief Benjamin Lev, Associate Editor Kovalyov and the three anonymous reviewers for their constructive comments and valuable suggestions, which have significantly improved the quality of this work. The work of the first two authors was partly supported by the National Key R&D Program of China (Grant No. 2019YFB1703800), and the National Natural Science Foundation of China (Grant Nos. 52075453, 71931007 and 51675442).

References

[1] Lee C-Y, Uzsoy R, Martin-Vega LA. Efficient algorithms for scheduling semiconductor burn-in operations. Oper Res 1992;40(4):764–75.
[2] Wang J-Q, Leung JY-T. Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan. Int J Prod Econ 2014;156:325–31.
[3] Van der Zee DJ, Van Harten A, Schuur PC. Dynamic job assignment heuristics for multi-server batch operations-a cost based approach. Int J Prod Res 1997;35(11):3063–94.
[4] Wang J-Q, Fan G-Q, Zhang Y, Zhang C-W, Leung JY-T. Two-agent scheduling on a single parallel-batching machine with equal processing time and non-identical job sizes. Eur J Oper Res 2017;258(2):478–90.
[5] Tang L, Liu J, Rong A, Yang Z. A review of planning and scheduling systems and methods for integrated steel production. Eur J Oper Res 2001;133(1):1–20.
[6] Guo Q, Tang L. Modelling and discrete differential evolution algorithm for order rescheduling problem in steel industry. Comput Ind Eng 2019;130:586–96.
[7] Liu M, Yang X, Chu F, Zhang J, Chu C. Energy-oriented bi-objective optimization for the tempered glass scheduling. Omega 2020;90:101995.
[8] Lee Y, Lee K. Lot-sizing and scheduling in flat-panel display manufacturing process. Omega 2020;93:102036.
[9] Damodaran P, Chang P-Y. Heuristics to minimize makespan of parallel batch processing machines. Int J Adv ManufTechnol 2008;37(9–10):1005–13.
[10] Zhu C., Yan D.. A kind of variable capacity heat treatment furnace. https://patents.google.com/patent/CN204417540U/en/; 2015. [Online; accessed 27 February 2020].
[11] Huang H.. A kind of removable transfiguration heat-treatment furnace. https://patents.google.com/patent/CN107164626A/zh/; 2017. [Online; accessed 27 February 2020].
[12] Ponsignon T, Mönch L. Heuristic approaches for master planning in semiconductor manufacturing. Comput Oper Res 2012;39(3):479–91.
[13] Van der Zee D-J, Van Harten A, Schuur P. On-line scheduling of multi-server batch operations. IIE Trans 2001;33(7):569–86.
[14] Gokhale R, Mathirajan M. Heuristic algorithms for scheduling of a batch processor in automobile gear manufacturing. Int J Prod Res 2011;49(10):2705–28.
[15] Brehob M, Torng E, Uthaisombut P. Applying extra-resource analysis to load balancing. J Scheduling 2000;3(5):273–88.
[16] Rustogi K, Strusevich VA. Parallel machine scheduling: impact of adding extra machines. Oper Res 2013;61(5):1243–57.
[17] Kalyanasundaram B, Pruhs K. Speed is as powerful as clairvoyance. J ACM 2000;47(4):617–43.
[18] Chan H-L, Lam T-W, Liu K-S. Extra unit-speed machines are almost as powerful as speedy machines for flow time scheduling. SIAM J Comput 2008;37(5):1595–612.
[19] Braun O, Schmidt G. Parallel processor scheduling with limited number of preemptions. SIAM J Comput 2003;32(3):671–80.
[20] Soper AJ, Strusevich VA. Parametric analysis of the quality of single preemption schedules on three uniform parallel machines. Ann Oper Res 2021;298(1):469–95.
[21] Im S, Oh H, Shadloo M. Minimizing the maximum flow time in batch scheduling. Oper Res Lett 2016;44(6):784–9.
[22] Potts CN, Kovalyov MY. Scheduling with batching: a review. Eur J Oper Res 2000;120(2):228–49.
[23] Mönch L, Fowler JW, Dauzere-Peres S, Mason SJ, Rose O. A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations. J Scheduling 2011;14(6):583–99.
[24] Fowler JW, Mönch L. A survey of scheduling with parallel batch (p-batch) processing. Eur J Oper Res 2021.
[25] Brucker P, Gladky A, Hoogeveen H, Kovalyov MY, Potts CN, Tautenhahn T, et al. Scheduling a batching machine. J Scheduling 1998;1(1):31–54.
[26] Poon CK, Yu W. On minimizing total completion time in batch machine scheduling. Int J Found ComputSci 2004;15(4):593–607.
[27] Chandru V, Lee C-Y, Uzsoy R. Minimizing total completion time on batch processing machines. Int J Prod Econ 1993;31(9):2097–121.
[28] Cheng T, Chen Z-L, Kovalyov M, Lin B. Parallel-machine batching and scheduling to minimize total completion time. IIE Trans 1996;28(11):953–6.
[29] Hochbaum DS, Landy D. Scheduling semiconductor burn-in operations to minimize total flowtime. Oper Res 1997;45(6):874–85.
[30] Uzsoy R. Scheduling a single batch processing machine with non-identical job sizes. Int J Prod Res 1994;32(7):1615–35.
[31] Zhang G, Cai X, Lee C-Y, Wong CK. Minimizing makespan on a single batch processing machine with nonidentical job sizes. Nav Res Logist 2001;48(3):226–40.
[32] Malapert A, Guéret C, Rousseau L-M. A constraint programming approach for a batch processing problem with non-identical job sizes. Eur J Oper Res 2012;221(3):533–45.
[33] Muter İ. Exact algorithms to minimize makespan on single and parallel batch processing machines. Eur J Oper Res 2020;285(2):470–83.
[34] Emde S, Polten L, Gendreau M. Logic-based benders decomposition for scheduling a batching machine. Comput Oper Res 2020;113:104777.
[35] Cheng B, Zhu H, Li K, Li Y. Optimization of batch operations with a truncated batch-position-based learning effect. Omega 2019;85:134–43.
[36] Polyakovskiy S, M'Hallah R. Just-in-time two-dimensional bin packing. Omega 2021;102:102311.
[37] Jia Z-H, Leung JY-T. A meta-heuristic to minimize makespan for parallel batch machines with arbitrary job sizes. Eur J Oper Res 2015;240(3):649–65.
[38] Li S. Approximation algorithms for scheduling jobs with release times and arbitrary sizes on batch machines with non-identical capacities. Eur J Oper Res 2017;263(3):815–26.
[39] Ozturk O. A truncated column generation algorithm for the parallel batch scheduling problem to minimize total flow time. Eur J Oper Res 2020;286(2):432–43.
[40] Phillips CA, Stein C, Torng E, Wein J. Optimal time-critical scheduling via resource augmentation. In: Proceedings of the twenty-ninth annual ACM symposium on theory of computing; 1997. p. 140–9.
[41] Dósa G, Tan Z. New upper and lower bounds for online scheduling with machine cost. Discrete Optim 2010;7(3):125–35.
[42] Akaria I, Epstein L. An optimal online algorithm for scheduling with general machine cost functions. J Scheduling 2020;23(2):155–62.
[43] Li K, Xiao W, Yang S. Scheduling uniform manufacturing resources via the internet: a review. J Manuf Syst 2019;50:247–62.
[44] Chan W-T, Lam T-W, Liu K-S, Wong PW. New resource augmentation analysis of the total stretch of SRPT and SJF in multiprocessor scheduling. Theor Comput Sci 2006;359(1–3):430–9.
[45] Imreh C. Online scheduling with general machine cost functions. Discrete Appl Math 2009;157(9):2070–7.
[46] Chekuri C, Goel A, Khanna S, Kumar A. Multi-processor scheduling to minimize flow time with ε resource augmentation. In: Proceedings of the 36th annual ACM symposium on theory of computing. ACM; 2004. p. 363–72.
[47] Soper AJ, Strusevich VA. Schedules with a single preemption on uniform parallel machines. Discrete Appl Math 2019;261:332–43.
[48] Jaillet P, Wagner MR. Generalized online routing: new competitive ratios, resource augmentation, and asymptotic analyses. Oper Res 2008;56(3):745–57.
[49] Azar Y, Epstein L, van Stee R. Resource augmentation in load balancing. J Scheduling 2000;3(5):249–58.
[50] Jiang Y, Weng Z, Hu J. Algorithms with limited number of preemptions for scheduling on parallel machines. J Comb Optim 2014;27(4):711–23.

[51] Soper AJ, Strusevich VA. Single parameter analysis of power of preemption on two and three uniform machines. Discrete Optim 2014;12:26–46.
[52] Epstein L, Levin A, Soper AJ, Strusevich VA. Power of preemption for minimizing total completion time on uniform parallel machines. SIAM J Discrete Math 2017;31(1):101–23.
[53] Correa JR, Skutella M, Verschae J. The power of preemption on unrelated machines and applications to scheduling orders. Math Oper Res 2012;37(2):379–98.
[54] Epstein L, Levin A. The benefit of preemption for single machine scheduling so as to minimize total weighted completion time. Oper Res Lett 2016;44(6):772–4.
[55] Graham RL, Lawler EL, Lenstra JK, Kan AR. Optimization and approximation in deterministic sequencing and scheduling: a survey. In: Annals of discrete mathematics, vol. 5. Elsevier; 1979. p. 287–326.
[56] McNaughton R. Scheduling with deadlines and loss functions. Manage Sci 1959;6(1):1–12.