Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/mbe

MBE, 19(11): 10731–10740.
DOI: 10.3934/mbe.2022502
Received: 28 May 2022
Revised: 10 July 2022
Accepted: 25 July 2022
Published: 28 July 2022

Research article

Efficient algorithms for scheduling equal-length jobs with processing set restrictions on uniform parallel batch machines

Shuguang Li*

School of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China

* **Correspondence:** Email: sgliytu@hotmail.com; Tel: +8618753509226.

Abstract: We consider the problem of scheduling jobs with equal lengths on uniform parallel batch machines with non-identical capacities where each job can only be processed on a specified subset of machines called its processing set. For the case of equal release times, we give efficient exact algorithms for various objective functions. For the case of unequal release times, we give efficient exact algorithms for minimizing makespan.

Keywords: scheduling; uniform parallel batch machines; processing set restrictions; equal job lengths; exact algorithms

1. Introduction

The problem of scheduling uniform parallel batch machines with processing set restrictions can be defined as follows. Let $\mathcal{J} = \{1, 2, \ldots, n\}$ be a set of jobs and $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ be a set of uniform parallel batch machines. Job j ($j = 1, 2, \ldots, n$) becomes available at its *release time* $r_j \geq 0$ and requires $p_j \geq 0$ units of processing called its *length*. For each job j, let $\mathcal{M}_j \subseteq \mathcal{M}$ be the set of the *eligible machines* which are capable of processing the job, called its *processing set*. Each job will be assigned to exactly one machine and job preemption is not allowed. Machine M_i ($i = 1, 2, \ldots, m$) has a *speed* $v_i \geq 1$ and a *capacity* $K_i < n$. The impact of the speed is that M_i can carry out v_i units of processing in one time unit. That is, if job j is assigned to machine M_i, then it requires p_j/v_i *processing time* to be completed. Machine M_i can process several jobs as a batch simultaneously as long as the total number of these jobs does not exceed K_i. The *length* of a batch is the maximum of the lengths of the jobs belonging to it. Jobs in the same batch have a common start time and a common completion time. The goal is to schedule the jobs on the machines in a manner that optimizes one or more objective functions.

Two classes of objectives are considered: the min-sum objective and the min-max objective.

Specifically, let $f_j : [0, +\infty) \to [0, +\infty)$ $(j = 1, 2, \ldots, n)$ be a non-decreasing function. Additional parameters that may be included for job j are its *due date* d_j and its *weight* w_j. For a particular schedule σ, let $C_j(\sigma)$ denote the *completion time* of job j in σ. Let $T_j(\sigma) = \max\{C_j(\sigma) - d_j, 0\}$ denote the tardiness of job j in σ. Let $U_j(\sigma) = 1$ if $C_j(\sigma) > d_j$ and $U_j(\sigma) = 0$ otherwise. (In the rest of this paper, we safely ignore σ in the notations without causing confusion.) The objectives of minimizing $\sum_{j=1}^{n} f_j(T_j)$ and $\max_{j=1,2,\ldots,n}\{f_j(T_j)\}$ will be considered. Following [1, 2], the models can be denoted as $Q|r_j, d_j, \mathcal{M}_j, p - batch, K_i| \sum f_j(T_j)$ and $Q|r_j, d_j, \mathcal{M}_j, p - batch, K_i| \max\{f_j(T_j)\}$.

Many popular scheduling objectives are covered by the two models, such as total weighted completion time ($\sum w_j C_j$) minimization, total weighted tardiness ($\sum w_j T_j$) minimization, weighted number of tardy jobs ($\sum w_j U_j$) minimization, makespan ($C_{\max} = \max C_j$) minimization, and maximum weighted tardiness ($\max\{w_j(T_j)\}$) minimization. As shown in [3–5], most of such problems are NP-hard even for the special cases where all $v_i = 1$, all $K_i = 1$ and all $\mathcal{M}_j = \mathcal{M}$. Thus, we are interested in polynomial time exact algorithms for some important special cases of the problems [6].

In this paper, we focus on an important special case where all jobs have equal lengths (and equal release times). The problems under study can be denoted as $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \sum f_j(T_j)$ (the special case of $Q|r_j, p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \sum f_j(T_j)$ where all $r_j = 0$) and $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \max\{f_j(T_j)\}$ (the special case of $Q|r_j, p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \max\{f_j(T_j)\}$ where all $r_j = 0$). Li [7] presented polynomial time algorithms for uniform parallel machine scheduling problems $Q|p_j = p, d_j, \mathcal{M}_j| \sum f_j(T_j)$ and $Q|p_j = p, d_j, \mathcal{M}_j| \max\{f_j(T_j)\}$ (the special cases of $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \sum f_j(T_j)$ and $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \max\{f_j(T_j)\}$ where all $K_i = 1$). We extend the results obtained in [7] to uniform parallel batch machines, allowing the machines to have non-identical capacities. Moreover, for minimizing makespan, we allow unequal release times and get an algorithm for arbitrary processing set restrictions.

Although the above problem setting appears simple, it captures important aspects of a wide range of applications. There are many problems arise as its special or similar cases in networking and information systems. For example, Low [8] studied the problem in the context of retrieving data blocks from disks in video on demand systems. Suri et al. [9] considered the problem in peer-to-peer systems. The problem was also studied for workload balancing among packet queues [10], for data aggregation in wireless sensor networks [11]. Recently, Champati and Liang [12] studied a very similar problem where each machine has its own convex cost functions, aiming to minimize the sum cost and the maximum differential cost of the machines.

The remainder of this paper is organized as follows. In Section 2, the related researches are reviewed. In Section 3, we consider the case of equal release times. We present an algorithm with running time $O(n^3 m + n^2 m \log(mn))$ for $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \sum f_j(T_j)$, as well as an algorithm with running time $O(n^{5/2} m^{3/2} \log(mn))$ for $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i| \max\{f_j(T_j)\}$. In Section 4, we consider the case of unequal release times. We present an algorithm with running time $O(n^{5/2} m^{3/2} \log(mn))$ for $Q|r_j, p_j = p, \mathcal{M}_j, p - batch, K_i|C_{\max}$. Section 5 presents the conclusions and future directions of research.

2. Literature review

Leung and Li [13] discussed several special cases of processing set restrictions. The processing sets of the jobs are *inclusive*, if for any two jobs j_1 and j_2, either $\mathcal{M}_{j_1} \subseteq \mathcal{M}_{j_2}$, or $\mathcal{M}_{j_2} \subseteq \mathcal{M}_{j_1}$. The processing sets are *nested*, if for any two jobs j_1 and j_2, either $\mathcal{M}_{j_1} \cap \mathcal{M}_{j_2} = \emptyset$, or $\mathcal{M}_{j_1} \subseteq \mathcal{M}_{j_2}$, or $\mathcal{M}_{j_2} \subseteq \mathcal{M}_{j_1}$. The processing sets are *interval*, if for any job j, $\mathcal{M}_j = \{M_{a_j}, M_{a_j+1}, \ldots, M_{b_j}\}$ for some $1 \leq a_j \leq b_j \leq m$. The processing sets are *tree-hierarchical*, if each machine is represented by a tree node, and each job j is associated with a tree node M_{a_j}, such that $\mathcal{M}_j$ is exactly the set of the machines consisting of all the nodes on the unique path from M_{a_j} to the root of the tree.

The problem studied in this paper combines two important sub-fields of scheduling theory: scheduling with processing set restrictions and parallel batch scheduling. The two sub-fields have received intense study in the literature, see the survey papers [13] and [14–16] respectively. There are also a few papers which combined the two-subfields into a unified framework [17–23]. In the problems studied in these papers except the last two, each job has a size and a machine can process several jobs simultaneously as a batch as long as the total size of these jobs does not exceed its capacity. Any machine cannot process the jobs whose sizes are larger than its capacity. Thus, for each job, the machines whose capacities are not less than its size form its processing set. Clearly, the processing sets of the jobs are inclusive. In [22], Li presented two algorithms with approximation ratios 3 and 9/4 for the problem of minimizing makespan on parallel batch machines with inclusive processing set restrictions, where the jobs have arbitrary lengths and the machines have the same speed. In [23], Li studied parallel batch scheduling with nested processing set restrictions to minimize makespan, and presented a $(3 - 1/m)$-approximation algorithm for the case of equal release times and a polynomial time approximation scheme (PTAS) for the case of unequal release times.

For scheduling with processing set restrictions, the review focuses on the case of equal job lengths. Lin and Li [24] obtained an algorithm for $P|p_j = p, \mathcal{M}_j|C_{\max}$ (the special case of $Q|r_j, p_j = p, \mathcal{M}_j|C_{\max}$ where all $r_j = 0$ and all $v_i = 1$) that runs in $O(n^3 \log n)$ time, and generalized the algorithm to solve $Q|p_j = p, \mathcal{M}_j|C_{\max}$ in $O(n^3 \log(nv_{lcm}))$ time, where v_{lcm} denotes the least common multiple of $v_1, v_2, \ldots, v_m$. They also obtained an algorithm for $P|p_j = p, \mathcal{M}_j(interval)|C_{\max}$ (the special case of $P|p_j = p, \mathcal{M}_j|C_{\max}$ with interval processing set restrictions) that runs in $O(m^2 + mn)$ time. Harvey et al. [25] independently developed an algorithm for $P|p_j = p, \mathcal{M}_j|C_{\max}$ that runs in $O(n^2 m)$ time. Brucker et al. [26] presented algorithms running in $O(n^2 m(n + \log m))$ time for $Q|p_j = p, d_j, \mathcal{M}_j|\sum w_j T_j$, $Q|p_j = p, d_j, \mathcal{M}_j|\sum w_j U_j$, $P|r_j, p_j = 1, d_j, \mathcal{M}_j|\sum w_j T_j$ and $P|r_j, p_j = 1, d_j, \mathcal{M}_j|\sum w_j U_j$. Li [7] presented an algorithm for $Q|p_j = p, d_j, \mathcal{M}_j|\sum f_j(T_j)$ that runs in $O(n^3 m + n^2 m \log(mn))$ time. For the special cases where $f_j(T_j) = C_j$ or $f_j(T_j) = U_j$, the running time of the algorithm can be improved to $O(n^{5/2} m \log n)$. He also presented an algorithm for $Q|p_j = p, d_j, \mathcal{M}_j|\max\{f_j(T_j)\}$ that runs in $O(n^{5/2} m \log(mn))$ time. For the special cases where $f_j(T_j) = C_j$ (i.e., $Q|p_j = p, \mathcal{M}_j|C_{\max}$), the running time of the algorithm can be improved to $O(n^2(m + \log(nv_{\max})) \log n)$, where $v_{\max} = \max\{v_j\}$. Lee et al. [27] showed that $Q|r_j, p_j = p, \mathcal{M}_j|C_{\max}$ can be solved in $O(m^{3/2} n^{5/2} \log(mn))$ time, and $P|r_j, p_j = p, \mathcal{M}_j|C_{\max}$(the special case of $Q|r_j, p_j = p, \mathcal{M}_j|C_{\max}$ where all $v_i = 1$) can be solved in $O(m^{3/2} n^{5/2} \log n)$ time. Shabtay et al. [28] obtained various results for several problems of scheduling uniform machines with equal length jobs, processing set restrictions and job rejection. Hong et al. [29] studied $P|r_j, p_j = p, \mathcal{M}_j|\sum C_j$. For the problem with a fixed number of machines, they provided a polynomial time dynamic programming

algorithm. For the general case, they presented two polynomial time approximation algorithms with approximation ratios 3/5 and 5/7 respectively. Jiang et al. [30] presented a comprehensive overview (including their new findings) of ideal schedules for various scheduling problems in different machine environments and with different job characteristics. An ideal schedule is a schedule that simultaneously minimizes the total completion time and makespan. They pointed out that in most problems that are known to have an ideal schedule, the jobs have equal processing times. Along with other results, they proved that any optimal schedule for $Q|p_j = p, \mathcal{M}_j|\sum C_j$ also minimizes makespan. Jing et al. [31] studied the problem of scheduling high multiplicity jobs to minimize makespan on parallel machines with processing set restrictions, setup times and machine available times. High multiplicity means that jobs are partitioned into several groups and in each group all jobs are identical. Whenever there is a switch from processing a job of one group to a job of another group, a setup time is needed. They formulated the problem as a mixed integer programming and proposed a heuristic for it.

Pinedo [32] and Glass and Mills [33] presented algorithms for $P|p_j = p, \mathcal{M}_j(\text{nested})|C_{\max}$ (the special case of $P|p_j = p, \mathcal{M}_j|C_{\max}$ with nested processing set restrictions) that run in time $O(n \log n)$ and $O(m^2)$ time respectively. Li and Li [34] presented algorithms for $P|r_j, p_j = p, \mathcal{M}_j(inclusive)|C_{\max}$ and $P|r_j, p_j = p, \mathcal{M}_j(\text{tree})|C_{\max}$ (the special cases of $P|r_j, p_j = p, \mathcal{M}_j|C_{\max}$ with inclusive and tree-hierarchical processing set restrictions) that run in $O(n^2 + mn \log n)$ time. For uniform machines, they showed that $Q|r_j, p_j = p, \mathcal{M}_j(inclusive)|C_{\max}$ and $Q|r_j, p_j = p, \mathcal{M}_j(\text{tree})|C_{\max}$ can be solved in $O(mn^2 \log m)$ time. Later, Li and Lee [35] developed an improved algorithm for $P|r_j, p_j = p, \mathcal{M}_j(inclusive)|C_{\max}$ that runs in $O(min\{m, \log n\}n \log n)$ time, and an improved algorithm for $P|r_j, p_j = p, \mathcal{M}_j(\text{tree})|C_{\max}$ that runs in $O(mn \log n)$ time.

For parallel batch scheduling, the review focuses on equal job lengths or uniform parallel batch machines. Liu et al. [36] presented an algorithm for $P|r_j, p_j = p, p - batch, B|C_{\max}$ (the special case of $Q|r_j, p_j = p, \mathcal{M}_j, p-batch, K_i|C_{\max}$ where all $\mathcal{M}_j = \mathcal{M}$, all $K_i = B$ and all $v_i = 1$) that runs in $O(n \log n)$ time. Ozturk et al. [37] presented a 2-approximation algorithm for the problem of scheduling jobs with equal lengths, unequal release times and sizes on identical parallel batch machines (all $K_i = B$) to minimize makespan. Wang and Leung [18] studied the problem of scheduling jobs with equal lengths and arbitrary sizes on parallel batch machines (all $v_i = 1$) with non-identical capacities. The problem fits into the model of scheduling with inclusive processing set restrictions, since each job can only be processed by the machines whose capacities are not less than the size of the job. Wang and Leung [18] presented a 2-approximation algorithm, as well as an algorithm with asymptotic approximation ratio 3/2 for the problem. Li et al. [38] proposed several heuristics for the problem of scheduling jobs with unequal lengths, release times and sizes on uniform parallel batch machines with identical capacities to minimize makespan. Zhou et al. [39] presented an effective discrete differential evolution algorithm for the problem of scheduling jobs with unequal lengths and sizes on uniform parallel batch machines with non-identical capacities to minimize makespan. Both [38] and [39] have not included processing set restrictions.

3. Equal release times

In this section, we consider the case of equal release times. We will present algorithms for $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i|\sum f_j(T_j)$ and $Q|p_j = p, d_j, \mathcal{M}_j, p - batch, K_i|\max\{f_j(T_j)\}$.

Since f_j $(j = 1, 2, \ldots, n)$ is a non-decreasing function, for both $Q|p_j = p, d_j, \mathcal{M}_j, p-batch, K_i|\sum f_j(T_j)$ and $Q|p_j = p, d_j, \mathcal{M}_j, p-batch, K_i|\max\{f_j(T_j)\}$, there is an optimal schedule in which the first batch on each machine starts at time zero, and the batches on each machine are processed successively. Moreover, we can assume that there are n_i empty batches on machine M_i $(i = 1, 2, \ldots, m)$ to which the jobs may be assigned, where n_i denotes the smallest integer such that $n_i K_i \geq n$. The k-th batch on M_i,$B_{k,i}$, completes at time kp/v_i, $k = 1, 2, \ldots, n_i$. To find a feasible schedule, we need only to assign the jobs to $\sum_{i=1}^{m} n_i$ empty batches such that all jobs obey the processing set restrictions.

First, we consider $Q|p_j = p, d_j, \mathcal{M}_j, p-batch, K_i|\sum f_j(T_j)$.

If job j is assigned to $B_{k,i}$ and $M_i \in \mathcal{M}_j$, then the cost incurred is defined to be $c_{jki} = f_j(\max\{kp/v_i - d_j, 0\})$, $j = 1, 2, \ldots, n$, $k = 1, 2, \ldots, n_i$, $i = 1, 2, \ldots, m$. Let $C = \max_{j,k,i} c_{jki}$. By regarding each job $j \in \mathcal{J}$ as a vertex in X, and each empty batch $B_{k,i}$ as K_i vertices $y_{k1}, y_{k2}, \ldots, y_{kK_i}$ in Y, we construct a bipartite graph G with bipartition (X, Y), where j is joined to $y_{k1}, y_{k2}, \ldots, y_{kK_i}$ if and only if $M_i \in \mathcal{M}_j$, and the incurred costs are equal to c_{jki}, $j = 1, 2, \ldots, n$, $k = 1, 2, \ldots, n_i$, $i = 1, 2, \ldots, m$. Then we use the Successive Shortest Path algorithm to solve the *bipartite weighted matching problem* [40] and get a matching of minimum cost that saturates every vertex in X. From this matching we can construct an optimal schedule easily.

The Successive Shortest Path algorithm runs in $O(|X| \cdot S(|X| + |Y|, |X||Y|, C))$ time, where $S(|X| + |Y|, |X||Y|, C)$ is the time for solving a shortest path problem with $|X| + |Y|$ vertices, $|X||Y|$ edges (these edges have non-negative costs), and maximum coefficient C. Currently, $S(u, a, C) = O(a+u\log u)$ [41]. Note that $|X| = n$ and $|Y| \leq 2mn$. We get:

Theorem 3.1. *There is an exact algorithm for $Q|p_j = p, d_j, \mathcal{M}_j, p-batch, K_i|\sum f_j(T_j)$ that runs in $O(n^3 m + n^2 m\log(mn))$ time.*

Next, we consider $Q|p_j = p, d_j, \mathcal{M}_j, p-batch, K_i|\max\{f_j(T_j)\}$. Let OPT denote the objective value of an optimal schedule.

Recall that we are focusing on an optimal schedule in which machine M_i $(i = 1, 2, \ldots, m)$ processes n_i batches (some batches may be empty), where n_i denotes the smallest integer such that $n_i K_i \geq n$. Each batch on M_i has K_i positions to accommodate jobs, and there are at most $2n$ positions on M_i, $i = 1, 2, \ldots, m$. Therefore, there are at most $2mn$ positions in total. Since each position has at most n choices of accommodating a job, there are at most $2mn^2$ possible values for OPT. We can sort these values in ascending order in $O(mn^2\log(mn))$ time. Then, we perform a binary search in the interval to determine OPT in $O(\log(mn))$ iterations.

For each value λ selected, we test whether there is a feasible schedule whose objective value is no more than λ. To this end, we construct a bipartite graph G with bipartition (X, Y) as follows. Regard each job $j \in \mathcal{J}$ as a vertex in X, and each empty batch $B_{k,i}$ as K_i vertices $y_{k1}, y_{k2}, \ldots, y_{kK_i}$ in Y, where j is joined to $y_{k1}, y_{k2}, \ldots, y_{kK_i}$ if and only if $M_i \in \mathcal{M}_j$ and $f_j(kp/v_i - d_j) \leq \lambda$, $j = 1, 2, \ldots, n$, $k = 1, 2, \ldots, n_i$, $i = 1, 2, \ldots, m$. Then we use the algorithm in [42] to solve the *maximum cardinality bipartite matching problem*. If the obtained matching saturates every vertex in X, then the procedure succeeds and we search the lower half of the interval. Otherwise, the procedure fails and we search the upper half of the interval.

The algorithm in [42] runs in $O(\sqrt{|X| + |Y|} \cdot |X||Y|)$ time. Note that $|X| = n$ and $|Y| \leq 2mn$. We get:

Theorem 3.2. *There is an exact algorithm for* $Q|p_j = p, d_j, \mathcal{M}_j, p-batch, K_i|\max\{f_j(T_j)\}$ *that runs in* $O(n^{5/2}m^{3/2}\log(mn))$ *time.*

4. Unequal release times

In this section, we consider the case of unequal release times. We will present an algorithm for $Q|r_j, p_j = p, \mathcal{M}_j, p-batch, K_i|C_{\max}$, which generalizes the one in [27] for $Q|r_j, p_j = p, \mathcal{M}_j|C_{\max}$ (the special case of $Q|r_j, p_j = p, \mathcal{M}_j, p-batch, K_i|C_{\max}$ where all $K_i = 1$). Let OPT denote the makespan of an optimal schedule for $Q|r_j, p_j = p, \mathcal{M}_j, p-batch, K_i|C_{\max}$. Let $\Lambda = \{\lambda|\lambda = r_j + kp/v_i; j, k \in \{1, 2, \ldots n\}$ and $i \in \{1, 2, \ldots m\}\}$. The following lemma, adopted from [27], still holds for $Q|r_j, p_j = p, \mathcal{M}_j, p-batch, K_i|C_{\max}$.

Lemma 4.1. *The set* Λ*, as defined above, contains all candidates for* OPT.

Since $|\Lambda| \leq mn^2$, there are at most mn^2 possible values for OPT. We can sort these values in ascending order in $O(mn^2\log(mn))$ time. Then, we perform a binary search to determine OPT in $O(\log(mn))$ iterations.

For each value λ selected, we use the following procedure, AssignJobs, to test whether there is a feasible schedule whose makespan is no more than λ.

AssignJobs (λ):

Step 1. Assign $b_i = \min\{n_i, \lfloor\lambda \cdot v_i/p\rfloor\}$ empty batches of length p and capacity K_i to machine M_i, where n_i denotes the smallest integer such that $n_iK_i \geq n$. The k-th batch on M_i,$B_{k,i}$, starts at time $\lambda - (b_i - k + 1)p/v_i$ and completes at time $\lambda - (b_i - k)p/v_i$, $k = 1, 2, \ldots, b_i$, $i = 1, 2, \ldots, m$.

Step 2. Construct a bipartite graph G with bipartition (X, Y) as follows. Regard each job $j \in \mathcal{J}$ as a vertex in X, and each empty batch $B_{k,i}$ as K_i vertices $y_{k1}, y_{k2}, \ldots, y_{kK_i}$ in Y, where j is joined to $y_{k1}, y_{k2}, \ldots, y_{kK_i}$ if and only if $M_i \in \mathcal{M}_j$ and $r_j \leq \lambda - (b_i - k + 1)p/v_i$, $j = 1, 2, \ldots, n$, $k = 1, 2, \ldots, b_i$, $i = 1, 2, \ldots, m$.

Step 3. Use the algorithm in [42] to solve the maximum cardinality bipartite matching problem. If the obtained matching saturates every vertex in X, then the procedure succeeds and terminates. Otherwise, the procedure fails and terminates.

Lemma 4.2. *If* $OPT \leq \lambda$*, then AssignJobs will generate a feasible schedule in* $O(n^{5/2}m^{3/2})$ *time for* $Q|r_j, p_j = p, \mathcal{M}_j, p-batch, K_i|C_{\max}$ *whose makespan is at most* λ.

Proof. Let σ^* denote an optimal schedule. Consider machine M_i, $i = 1, 2, \ldots, m$. Since $OPT \leq \lambda$, in σ^*, M_i processes at most b_i batches. Without loss of generality, assume that there are b_i batches (some of which may be dummy empty batches) processed on M_i in σ^*, denoted as $B^*_{1,i}, B^*_{2,i}, \ldots B^*_{b_i,i}$.

Modify σ^* as follows. Let $B^*_{b_i,i}$ be completed at time λ, $B^*_{b_i-1,i}$ be completed at time $\lambda - p/v_i$, ..., $B^*_{1,i}$ be completed at time $\lambda - (b_i - 1)p/v_i$, $i = 1, 2, \ldots, m$. Denote by $\tilde{\sigma}^*$ the modified schedule. Since each batch in $\tilde{\sigma}^*$ starts no earlier than its corresponding batch in σ^*, $\tilde{\sigma}^*$ is a feasible schedule. The makespan of $\tilde{\sigma}^*$ is exactly λ. Clearly, AssignJobs will generate a feasible schedule which is no worse than $\tilde{\sigma}^*$.

The algorithm in [42] runs in $O(\sqrt{|X| + |Y|} \cdot |X|\,|Y|)$ time. Since $|X| = n$ and $|Y| \leq 2mn$, the running time of AssignJobs is $O(n^{5/2}m^{3/2})$.

□

We get:

Theorem 4.3. *There is an exact algorithm for* $Q|r_j, p_j = p, \mathcal{M}_j, p - batch, K_i|C_{\max}$ *that runs in* $O(n^{5/2}m^{3/2}\log(mn))$ *time.*

5. Conclusions

We studied the problem of scheduling jobs with equal lengths and processing set restrictions on uniform parallel batch machines with non-identical capacities. For the case of equal release times, we gave efficient exact algorithms for various objective functions. For the case of unequal release times, we gave efficient exact algorithms for minimizing makespan. The findings extend previous results for uniform machines counterparts.

Future research should focus on extending the algorithms to the case of unequal release times for objective functions other than makespan. It would also be interesting (and difficult) to extend the techniques to other related models, for general or special cases of processing set restrictions, such as scenario-dependent processing times and/or due dates [43], two-agent scheduling problems [44], or multitasking scheduling problems [45].

Acknowledgments

This work is supported by Natural Science Foundation of Shandong Province China (No. ZR2020MA030).

Conflict of interest

The authors declare there is no conflict of interest.

References

1. R. L. Graham, E. L. Lawler, J. K. Lenstra, A. R. Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Ann. Discrete Math.*, **5** (1979), 287–326. https://doi.org/10.1016/S0167-5060(08)70356-X
2. P. Brucker, *Scheduling Algorithms*, 5th edition, Springer, 2007.
3. M. R. Garey and D. S. Johnson, Computers and intractability: a guide to the theory of NP-completeness (michael r. garey and david s. johnson), *SIAM Rev.*, **24** (1982), 90. https://doi.org/10.1137/1024022
4. E. L. Lawler, J. K. Lenstra, A. R. Kan, D. B. Shmoys, Sequencing and scheduling: Algorithms and complexity, *Handb. Oper. Res. Manage. Sci.*, **4** (1993), 445–522. https://doi.org/10.1016/S0927-0507(05)80189-6
5. J. Y. T. Leung, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, CRC Press, 2004.
6. C. H. Papadimitriou, K. Steiglitz, *Combinatorial optimization: Algorithms and Complexity*, Courier Dover Publications, 1998.

7. C. L. Li, Scheduling unit-length jobs with machine eligibility restrictions, *Eur. J. Oper. Res.*, **174** (2006), 1325–1328. https://doi.org/10.1016/j.ejor.2005.03.023
8. C. P. Low, An efficient retrieval selection algorithm for video servers with random duplicated assignment storage technique, *Inf. Process. Lett.*, **83** (2002), 315–321, 2002. https://doi.org/10.1016/S0020-0190(02)00210-7
9. S. Suri, C. D. Toth, Y. Zhou, Selfish load balancing and atomic congestion games, *Algorithmica*, **47** (2007), 79–96.
10. S. Kittipiyakul, T. Javidi, Delay-optimal server allocation in multiqueue multiserver systems with time-varying connectivities, *IEEE Trans. Inf. Theory*, **55** (2009), 2319–2333. https://doi.org/10.1109/TIT.2009.2016051
11. M. Shan, G. Chen, D. Luo, X. Zhu, X. Wu, Building maximum lifetime shortest path data aggregation trees in wireless sensor networks, *ACM Trans. Sensor Networks*, **11** (2014), 1–24. https://doi.org/10.1145/2629662
12. J. P. Champati, B. Liang, Efficient minimization of sum and differential costs on machines with job placement constraints, in *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, (2017), 1–9. https://doi.org/10.1109/INFOCOM.2017.8057085
13. J. Y. T. Leung, C. L. Li, Scheduling with processing set restrictions: A literature update, *Int. J. Prod. Econ.*, **175** (2016), 1–11. https://doi.org/10.1016/j.ijpe.2014.09.038
14. C. N. Potts, M. Y. Kovalyov, Scheduling with batching: a review, *Eur. J. Oper. Res.*, **120** (2000), 228–249. https://doi.org/10.1016/S0377-2217(99)00153-8
15. M. Mathirajan, A. Sivakumar, A literature review, classification and simple meta-analysis on scheduling of batch processors in semiconductor, *Int. J. Adv. Manuf. Technol.*, **29** (2006), 990–1001. https://doi.org/10.1007/s00170-005-2585-1
16. L. Monch, J. W. Fowler, S. Dauzere-Peres, S. J. Mason, O. Rose, A survey of problems, solution techniques, and future challenges in scheduling semiconductor manufacturing operations, *J. Scheduling*, **14** (2011), 583–599. https://doi.org/10.1007/s10951-010-0222-9
17. P. Damodaran, D. A. Diyadawagamage, O. Ghrayeb, M. C. Vlez-Gallego, A particle swarm optimization algorithm for minimizing makespan of nonidentical parallel batch processing machines,*Int. J. Adv. Manuf. Technol.*, **58** (2012), 1131–1140. https://doi.org/10.1007/s00170-011-3442-z
18. J. Q. Wang, J. Y. T. Leung, Scheduling jobs with equal-processing-time on parallel machines with non-identical capacities to minimize makespan, *Int. J. Prod. Econ.*, **156** (2014), 325–331. https://doi.org/10.1016/j.ijpe.2014.06.019
19. Z. h. Jia, K. Li, J. Y.-T. Leung, Effective heuristic for makespan minimization in parallel batch machines with non-identical capacities, *Int. J. Prod. Econ.*, **169** (2015), 1–10. https://doi.org/10.1016/j.ijpe.2015.07.021
20. Z. h. Jia, T. T. Wen, J. Y. T. Leung, K. Li, Effective heuristics for makespan minimization in parallel batch machines with non-identical capacities and job release times, *J. Ind. Manage. Optim.*, **13** (2017), 977–993. http://dx.doi.org/10.3934/jimo.2016057

21. S. Li, Approximation algorithms for scheduling jobs with release times and arbitrary sizes on batch machines with non-identical capacities, *Eur. J. Oper. Res.*, **263** (2017), 815–826. https://doi.org/10.1016/j.ejor.2017.06.021
22. S. Li, Parallel batch scheduling with inclusive processing set restrictions and non-identical capacities to minimize makespan, *Eur. J. Oper. Res.*, **260** (2017), 12–20, 2017. https://doi.org/10.1016/j.ejor.2016.11.044
23. S. Li, Parallel batch scheduling with nested processing set restrictions, *Theor. Comput. Sci.*, **689** (2017), 117–125. https://doi.org/10.1016/j.tcs.2017.06.003
24. Y. Lin, W. Li, Parallel machine scheduling of machine-dependent jobs with unit-length, *Eur. J. Oper. Res.*, **156** (2004), 261–266. https://doi.org/10.1016/S0377-2217(02)00914-1
25. N. J. Harvey, R. E. Ladner, L. Lovasz, T. Tamir, Semi-matchings for bipartite graphs and load balancing, *J. Algorithms*, **59** (2006), 53–78. https://doi.org/10.1016/j.jalgor.2005.01.003
26. P. Brucker, B. Jurisch, A. Kramer, Complexity of scheduling problems with multi-purpose machines, *Ann. Oper. Res.*, **70** (1997), 57–73. https://doi.org/10.1023/A:1018950911030
27. K. Lee, Y. T. Leung, M. L. Pinedo, Scheduling jobs with equal processing times subject to machine eligibility constraints, *J. Scheduling*, **14** (2011), 27–38. https://doi.org/10.1007/s10951-010-0190-0
28. D. Shabtay, S. Karhi, D. Oron, Multipurpose machine scheduling with rejection and identical job processing times, *J. Scheduling*, **18** (2015), 75–88. https://doi.org/10.1007/s10951-014-0386-9
29. J. Hong, K. Lee, M. L. Pinedo, Scheduling equal length jobs with eligibility restrictions, *Ann. Oper. Res.*, **285** (2020), 295–314. https://doi.org/10.1007/s10479-019-03172-8
30. X. Jiang, K. Lee, M. L. Pinedo, Ideal schedules in parallel machine settings, *Eur. J. Oper. Res.*, **290** (2021), 422–434. https://doi.org/10.1016/j.ejor.2020.08.010
31. C. Jing, W. Huang, L. Zhang, H. Zhang, Scheduling high multiplicity jobs on parallel multi-purpose machines with setup times and machine available times, *Asia Pac. J. Oper. Res.*, **2022** (2022), 2250012. https://doi.org/10.1142/S0217595922500129
32. M. L. Pinedo, *Scheduling: Theory, Algorithms and Systems*, Spring, 2018.
33. C. A. Glass, H. R. Mills, Scheduling unit length jobs with parallel nested machine processing set restrictions, *Comput. Oper. Res.*, **33** (2006), 620–638. https://doi.org/10.1016/j.cor.2004.07.010
34. C. L. Li, Q. Li, Scheduling jobs with release dates, equal processing times, and inclusive processing set restrictions, *J. Oper. Res. Soc.*, **66** (2015), 516–523. https://doi.org/10.1057/jors.2014.22
35. C. L. Li, K. Lee, A note on scheduling jobs with equal processing times and inclusive processing set restrictions, *J. Oper. Res. Soc.*, **67** (2016), 83–86. https://doi.org/10.1057/jors.2015.56
36. L. Liu, C. Ng, T. Cheng, Scheduling jobs with release dates on parallel batch processing machines to minimize the makespan, *Optim. Lett.*, **8** (2014), 307–318. https://doi.org/10.1007/s11590-012-0575-4
37. O. Ozturk, M. L. Espinouse, M. D. Mascolo, A. Gouin, Makespan minimisation on parallel batch processing machines with non-identical job sizes and release dates, *Int. J. Prod. Res.*, **50** (2011), 1–14. https://doi.org/10.1080/00207543.2011.641358

38. X. Li, H. Chen, B. Du, Q. Tan, Heuristics to schedule uniform parallel batch processing machines with dynamic job arrivals, *Int. J. Comput. Integr. Manuf.*, **26** (2012), 474–486. https://doi.org/10.1080/0951192X.2012.731612

39. S. Zhou, M. Liu, H. Chen, X. Li, An effective discrete differential evolution algorithm for scheduling uniform parallel batch processing machines with non-identical capacities and arbitrary job sizes, *Int. J. Prod. Econ.*, **179** (2016), 1–11. https://doi.org/10.1016/j.ijpe.2016.05.014

40. R. K. Ahuja, T. L. Magnanti, J. B. Orlin, *Network Flows, Theory, Algorithms, and Applications*, Prentice Hall, Englewood Cliffs, 1993.

41. M. L. Fredman, R. E. Tarjan, Fibonacci heaps and their uses in improved network optimization algorithms, *J. ACM*, **34** (1987), 596–615. https://doi.org/10.1145/28869.28874

42. J. E. Hopcroft, R. M. Karp, An $n^{5/2}$ algorithm for maximum matching in bipartite graphs, *SIAM J. Comput.*, **2** (1973), 225–231. https://doi.org/10.1137/0202019

43. C. C. Wu, D. Bai, X. Zhang, S. R. Cheng, J. C. Lin, Z. L. Wu, et al., A robust customer order scheduling problem along with scenario-dependent component processing times and due dates, *J. Manuf. Syst.*, **58** (2021), 291–305. https://doi.org/10.1016/j.jmsy.2020.12.013

44. C. C. Wu, J. N. Gupta, W. C. Lin, S. R. Cheng, Y. L. Chiu, J. H. Chen, et al., Robust scheduling of two-agent customer orders with scenario-dependent component processing times and release dates, *Mathematics*, **10** (2022), 1545. https://doi.org/10.3390/math10091545

45. C. C. Wu, A. Azzouz, J. Y. Chen, J. Xu, W. L. Shen, L. Lu, et al., A two-agent one-machine multitasking scheduling problem solving by exact and metaheuristics, *Complex Intell. Syst.*, **8** (2022), 199–212. https://doi.org/10.1007/s40747-021-00355-4

AIMS Press