Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6611493, 6 pages
https://doi.org/10.1155/2021/6611493

Research Article

A Note on Multimachine Scheduling with Weighted Early/Late Work Criteria and Common Due Date

Kerang Cao,[1] Xin Chen ,[2] Kwang-nam Choi,[3] Yage Liang,[4] Qian Miao,[5] and Xingong Zhang[6]

[1]*Department of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang 110142, China*
[2]*School of Electronics and Information Engineering, Liaoning University of Technology, Jinzhou 121001, China*
[3]*NTIS Center, Korea Institute of Science and Technology Information, Seoul 34113, Republic of Korea*
[4]*School of Information Engineering, Jiujiang Vocational and Technical College, Jiujiang 332007, China*
[5]*School of Software, Dalian University of Technology, Dalian 116623, China*
[6]*Key Lab for OCME, College of Mathematical Science, Chongqing Normal University, Chongqing 401331, China*

Correspondence should be addressed to Xin Chen; chenxin.lut@hotmail.com

Received 21 December 2020; Revised 1 April 2021; Accepted 7 April 2021; Published 17 April 2021

Academic Editor: Bekir Sahin

In this note, we revisit two types of scheduling problem with weighted early/late work criteria and a common due date. For parallel identical machines environment, we present a dynamic programming approach running in pseudopolynomial time, to classify the considered problem into the set of binary NP-hard. We also propose an enumeration algorithm for comparison. For two-machine flow shop systems, we focus on a previous dynamic programming method, but with a more precise analysis, to improve the practical performance during its execution. For each model, we verify our studies through computational experiments, in which we show the advantages of our techniques, respectively.

1. Introduction

Scheduling with due date constraint [1, 2] has been widely studied in the field of combinatorial optimization, such as minimizing tardiness [3], lateness [4], or the number of late jobs [5]. Among all the criteria related to the due date, late work [6] and early work [7] are a pair of symmetrical objectives, in which the former one indicates the loss once a job is finished after its due date, while the latter one implies the profit when a job starts execution before this time.

The concept of late work was first proposed in 1984 by Błażewicz [8], who was motivated from the scene of data collection in a control system, and, therefore, he called this parameter "information loss" at that time. Then, in 1992, Potts and Van Wassenhove [9] claimed that this criterion can model not only information collection but also other situations where a perishable commodity is involved, so they suggested using the term "late work." After these pioneers, scheduling with late work minimization has attracted much attention from different research groups, for example, [10–14].

Although the formal investigation on the early work maximization problem was started from 2017 [7], this parameter has already appeared frequently in the literature since 2004, which was used as auxiliary metrics when analysing the optimal schedules for late work problems, for example, [10, 11]. Due to the symmetry of these two criteria, scheduling with late work minimization and early work maximization shares the same essence when the offline optimal solutions are constructed. However, since the difference when an approximation solution is evaluated [12], scheduling with early work criterion gradually became a new subject, and several studies were devoted to it [13, 14].

In this note, we revisit two types of scheduling problem with a common due date to maximize total weighted early work of all jobs or, equivalently, to minimize total weighted late work (in sense of complexity and optimal offline solution analysis).

We focus on the identical machines and two-machine flow shop environments, respectively, and, therefore, the problems considered in this paper could be presented as $Pm|d_j = d|\max(X_w)$ (or equally, $Pm|d_j = d|Y_w$) and $F2|d_j = d|\max(X_w)$ (or equally, $F2|d_j = d|Y_w$), using the classical three-field notation in the scheduling field [15].

The rest of this paper is organized as follows. In Section 2, we give the formal definitions of the studies problems, i.e., $Pm|d_j = d|\max(X_w)$ and $F2|d_j = d|\max(X_w)$, respectively. Section 3 is devoted to the former problem, where we propose a dynamic programming approach running in pseudopolynomial time and an enumeration algorithm in exponential time, on the basis of a property of an optimal schedule. In Section 4, we focus on a previous dynamic programming method in [16], by introducing some improvement techniques. Finally, the whole paper is concluded in Section 5.

2. Problem Formulation

The early work of a job J_j (denoted as X_j) is defined as the early part (if any) executed before its due date d_j. That is, let p_j and C_j be the processing and completion time of J_j, respectively; we have

$$X_j = \begin{cases} p_j, & d_j \geq C_j; \\ p_j - \left(C_j - d_j\right), & C_j - p_j < d_j < C_j; \\ 0, & d_j \leq C_j - p_j. \end{cases} \quad (1)$$

Correspondingly, the late work of J_j (denoted as Y_j) is the late part processing after d_j, i.e.,

$$Y_j = \begin{cases} 0, & d_j \geq C_j; \\ C_j - d_j, & C_j - p_j < d_j < C_j; \\ p_j, & d_j \leq C_j - p_j. \end{cases} \quad (2)$$

Based on the definitions of these two parameters, we have $X_j + Y_j = p_j$.

Thus, the problems we considered in this paper can be described as follows:

(1) $Pm|d_j = d|\max(X_w)$. Input: there are m identical machines and n jobs, and each job J_j is accompanied by its processing time p_j, weight w_j, as well as a common due date d $(1 \leq j \leq n)$.

Output: assign the n jobs to the m machines without preemption, with the goal of maximizing total weighted early work of all jobs (denoted as X_w), in which $X_w = \sum_{j=1}^{n} w_j X_j$.

As we mentioned before, the above problem could be presented as $Pm|d_j = d|\max(X_w)$ by the three-field notation, in which we use max in the γ field to emphasize that it is a maximization problem. Moreover, since the symmetry of early work maximization and late work minimization, we have $w_j Y_j = w_j p_j - w_j X_j$ (so does $\sum_{j=1}^{n} w_j Y_j = \sum_{j=1}^{n} w_j p_j - \sum_{j=1}^{n} w_j X_j$), which implies that the results on the complexities and the constructions of the optimal offline solutions could be directly transferred from $Pm|d_j = d|\max(X_w)$ to $Pm|d_j = d|Y_w$.

(2) $F2|d_j = d|\max(X_w)$. In this model, we are given a two-machine flow shop, and each job (also accompanied with p_{ji}-the processing time of J_j on M_i for $i \in \{1, 2\}$, a weight w_j, and a common due date d) has to be processed firstly on M_1 and then on M_2. Same as in the previous one, we are aimed to schedule these jobs into the system so that to maximize total weighted early work, i.e., $\sum_{j=1}^{n} w_j X_j$ (here we define $X_j = X_{j1} + X_{j2}$, in which X_{ji} is the early work of J_j on M_i, for $i \in \{1, 2\}$). It is worth to be mentioned that, when analysing complexity and designing optimal offline algorithms, the problem could be treated as the same as $F2|d_j = d|Y_w$, which is the original representation in [16].

3. Exact Approaches for $Pm|d_j = = d|\max(X_w)$

In this section, we first introduce a property of an optimal schedule for problem $Pm|d_j = d|\max(X_w)$, based on which we propose two exact approaches. Among them, the former one is a dynamic programming method running in pseudopolynomial time, so that we can decide that problem $Pm|d_j = d|\max(X_w)$ belongs to the set of binary NP-hard, while the latter is an enumeration one, which can help us to verify the correctness and efficiency of the dynamic programming by computational experiments.

3.1. Property of an Optimal Schedule. We now introduce a property of an optimal schedule in Lemma 1, which could help us to design the exact methods for problem $Pm|d_j = d|\max(X_w)$ latter.

Lemma 1. *There exists an optimal schedule for problem $Pm|d_j = d|\max(X_w)$ such that on each machine, jobs are sequenced in nonincreasing order of their weights.*

Proof. Suppose that there exists one of the optimal schedules π, in which on machine M_i $(1 \leq i \leq m)$, there are two jobs J_k and J_h such that $J_k \prec J_h$ while $w_k < w_h$, where $J_k \prec J_h$ means that J_k precedes (close to) J_h on M_i. Then, we can construct a new schedule π' by exchanging the positions of the two jobs and show that the criterion value X_w cannot decrease after this modification.

We prove the above declaration based on the possible values of the common due date, i.e., d. Denote S_j and C_j as the start and completion time of the job J_j in π, respectively $(1 \leq j \leq n)$. □

Case 1. $d \leq S_k$, which means that both jobs are late in π (also in π'). Then, the total weighted early work of these two jobs is equal to 0, in both π and π'.

Case 2. $d \geq C_h$, where both jobs are early in π (also in π'). Then, the criterion value of these two jobs is equal to $w_k \cdot p_k + w_h \cdot p_h$, in both π and π'.

Case 3. $S_k < d < C_h$. Then, the total weighted early work of these two jobs in π is equal to $w_k \cdot X_k + w_h \cdot X_h =$

$w_k \cdot \min\{p_k, d - S_k\} + w_h \cdot \max\{0, d - S_h\}$. We claim that the criterion value would be increased in π', since the length of the early parts of the two jobs remains the same (as in π), while the weight becomes higher after the exchanging.

Therefore, in any optimal schedule, by adjusting the positions of the two adjacent jobs which do not follow this order, we can finally get another optimal schedule, which concludes this lemma.

3.2. Dynamic Programming Approach. Based on Lemma 1, we could construct an optimal solution with the following process: (1) we first reorder the job sequence in nondecreasing order of jobs' weights, i.e., $w_1 \le w_2 \le \ldots \le w_n$; (2) then, we assign the jobs in the ordered sequence one by one to a suitable machine, using a dynamic programming approach defined as follows.

Let $f(j, E_1, E_2, \ldots, E_m)$ denote the maximal total weighted early work of the first j jobs scheduled on m machines, where the early parts are executed for at most $E_1, E_2, \ldots, E_m$ units on $M_1, M_2, \ldots, M_m$, respectively. The initial condition should be set as $f(0, E_1, E_2, \ldots, E_m) = 0$ for all the $0 \le E_i \le d$ $(1 \le i \le m)$, and the optimal criterion value is output as $f(n, d, d, \ldots, d)$.

The recurrence procedure runs according to the following formula:

$$f(j, E_1, E_2, \ldots, E_m) = \max\left\{\begin{array}{c} f\left(j-1, \max\left\{0, E_1 - p_j\right\}, E_2, \ldots, E_m\right) + w_j \cdot \min\left\{p_j, E_1\right\}, \\ f\left(j-1, E_1, \max\left\{0, E_2 - p_j\right\}, \ldots, E_m\right) + w_j \cdot \min\left\{p_j, E_2\right\}, \ldots, \\ f\left(j-1, E_1, E_2, \ldots, \max\left\{0, E_m - p_j\right\}\right) + w_j \cdot \min\left\{p_j, E_m\right\} \end{array}\right\}. \quad (3)$$

We give the complete procedure in Algorithm 1, denoted as Algo_DP$_p$.

The sorting process requires $O(n \log n)$ time, and the dynamic programming approach runs in $O(nd^m)$. Summing up, the time consumption of Algo_DP$_p$ is $O(\max\{n \log n, nd^m\})$, which is pseudopolynomial when m is fixed. Then, we have the following theorem.

Theorem 2. *Problem $Pm|d_j = d|\max(X_w)$ belongs the set of binary NP-hard.*

Proof (Proof of Theorem 2). This theorem holds since the existence of Algo_DP$_p$. □

3.3. Enumeration Algorithm. To further reveal the correctness and efficiency of Algo_DP$_p$, we also propose an enumeration algorithm for comparison. We first sort the jobs in nonincreasing order of their weights and then enumerate all the possible partitions to find the optimal solution. The procedure is presented in Algorithm 2 denoted as Algo_EM.

Since there are m machines and n jobs, it takes $O(m^n)$ time to check all the possible schedules. Therefore, Algo_EM runs in $O(\max\{n \log n, nd^m\})$ time.

3.4. Computational Experiments. To compare the two algorithms proposed in Sections 3.2 and 3.3, we design a set of numerical experiments in this part. Since both of them return the optimal solutions, we focus on the efficiencies of these approaches, i.e., their time consumption.

Motivated by [7], we set the experiment data by the following process. The job processing time p_j was randomly taken with a discrete uniform distribution from $[1, 20]$, and the weight w_j was taken also from this range. The common due date d is set to be $d = (\sum_{j=1}^{n} p_j/m)$. Moreover, due to the memory limitation, we set $n \in \{15, 20, 25, 30, 35, 40, 45, 50\}$ when $m = 2$, and $n \in \{5, 8, 10, 15, 20, 25, 30, 35\}$ when $m = 3$.

Both of the two algorithms were implemented in C++ with the IDE of Microsoft Visual Studio 2017 and executed on a PC with Intel Core i7-8550U 1.8 GHz CPU and 8 GB DDR4 RAM. For each pair of (m, n), we generate 25 random instances and show the results in Table 1, in which the running time of Algo_DP$_p$ and Algo_EM is reported in milliseconds (ms).

From Table 1, we can find that Algo_DP$_p$ works much more efficiently than Algo_EM. Basically, the time consumption of Algo_DP$_p$ is $O(\max\{n \log n, nd^m\})$ ($O(nd^m)$ for most of the case) while Algo_EM takes $O(m^n)$ time, which results in the fact that the latter one needs more time when processing the same setting of (m, n) than the former. More precisely, for a fixed m, the time consumption of Algo_DP$_p$ increases slowly while Algo_EM increases drastically, along with the increment of n. This is mainly because that nd^m could be considered as a linear function of n while m^n is an exponential one, when m is fixed. Finally, Algo_EM has to stop at $n = 25$ when m is 2, and $n = 15$ when m is 3, due to the limitation of memory. However, Algo_DP$_p$ could solve all the test sequences, mostly within 1 second in our experiments.

4. Precise Analysis on $F2|d_j = = d|\max(X_w)$

In this section, we focus on a previous dynamic programming approach proposed in [16] (denoted as Algo_DP$_f$ in this paper) for problem $F2|d_j = d|\max(X_w)$ and introduce several improvement techniques by carefully analysing the characteristics of some parameters. Through computational experiments, we show that these skills are effective to help us to obtain the optimal solutions faster than the original version.

4.1. Brief Introduction on Algo_DP$_f$. In [16], the authors considered problem $F2|d_j = d|Y_w$ (which is the same as $F2|d_j = d|\max(X_w)$ since they focused on the complexity

(1) Reorder the jobs by nondecreasing order of their weights, i.e., $w_1 \le w_2 \le \cdots \le w_n$;
(2) Assign J_1 to M_1;
(3) for $(2 \le j \le n)$
(4) do Calculate $f(j, d, d, \ldots, d)$ to find a suitable machine for J_j with the (current) maximal criterion value, say M_s;
(5) Schedule J_j on the first position of M_s, and move the already scheduled jobs on M_s (if any) backward;

Algorithm 1: Algo_DP$_p$.

and the construction of the offline optimal solutions) and proposed a dynamic programming approach Algo_DP$_f$ running in $O(n^2d^4)$. The result is very important since it can solve the problem optimally and clarify the problem complexity clearly (binary NP-hard).

For a job set $\mathcal{J}$, Algo_DP$_f$ adopts an enumeration with a recurrence process to find the first partially late job (denoted as J_*) and divides the remaining jobs into three subsets, which are $\mathcal{J}^E$ (all the totally early jobs), $\mathcal{J}^P$ (all the partially late jobs except for J_*), and $\mathcal{J}^L$ (all the totally late jobs). Then, Algo_DP$_f$ returns an optimal schedule as $\mathcal{J}^E \prec \{J_*\} \prec \mathcal{J}^P \prec \mathcal{J}^L$, in which

(1) Jobs in $\mathcal{J}^E$ are sequenced by Johnson's order [17]
(2) A special job in $\mathcal{J}^P$ should take the last position in this set if it exists, while the others are sequenced arbitrarily
(3) Jobs in $\mathcal{J}^L$ are scheduled arbitrarily

Algo_DP$_f$ enumerates each job one time as the potential J_*. For each selection (say $J_*^{\{i\}}$), it uses a recurrence process to obtain the maximal total weighted early work, under the assumption that $J_*^{\{i\}}$ is the "first partially late job." The recurrence process runs based on a state function $f_k(A, B, t, a)$, which denotes the maximum weighted early work among jobs $\{J_k, J_{k+1}, \ldots J_n\}$ (J_n is actually $J_*^{\{i\}}$ in the current case), when these jobs start exactly at time A on M_1 and not earlier than at time B on M_2. Moreover, parameter t means that there are exactly t time units remained on M_1 after $J_*^{\{i\}}$ and before the common due date d, while $a = 0$ (or 1) means that there is no job (or only one job) among $\{J_1, J_2, \ldots J_{k-1}\}$ partially executed on M_1 after $J_*^{\{i\}}$ and before d. For each selection of $J_*^{\{i\}}$, the initial state is set to be $f_n(A, B, t, a)$ for all $0 \le A$, B, $t \le d$, and $a \in \{0, 1\}$ based on several different conditions, while the optimal weighted early work under this assumption ($J_*^{\{i\}}$ is the first partially late job) is output by $f_1(0, 0, 0, 0)$. The above recurrence process needs $O(nd^4)$ time, and, therefore, considering the enumeration of every job as $J_*^{\{i\}}$, the whole procedure of Algo_DP$_f$ requires $O(n^2d^4)$ time. For further details of this approach, refer to [16] or [18].

4.2. Several Improvement Techniques. Algo_DP$_f$ is a very important result for $F2|d_j = d|\max(X_w)$ (so does for $F2|d_j = dY_w$) since it declared problem complexity more precisely. However, from the implementation point of view, we found that some details could be analysed, which can result in the fact that some unnecessary calculations could be avoided.

Based on the characteristic of flow shop scheduling, we claim the relationship of A and B in each state $f_k(A, B, t, a)$, in Property 1.

Property 1. For each state $f_k(A, B, t, a)$ in Algo_DP$_f$, $B \ge A + \min_{k \le j \le n} p_{j1}$, where p_{j1} means the processing time of J_j on M_1.

Moreover, due to the definitions of t, A, and d in the state function, we declare Property 2.

Property 2. For each state $f_k(A, B, t, a)$ in Algo_DP$_f$, $t \le d - (A + \sum_{j=k}^{n} p_{j1})$.

Therefore, for the states that violate the above properties, we set their values directly as "$-\infty$" without further calculations. Consequently, we denote the procedure introducing these two properties as Algo_DP$_f'$.

4.3. Computational Experiments. The same as in section 3.4, we introduce a series of numerical experiments to show the advantages of the properties in 4.2, i.e., we compare Algo_DP$_f'$ with Algo_DP$_f$.

We adopt the data set from [18] for our experiments, in which Algo_DP$_f$, an enumeration algorithm, and several heuristics for problem $F2|d_j = dY_w$ (also available for $F2|d_j = d|\max(X_w)$) were compared. According to this literature, the job number n was selected from $\{10, 12, 14, 16, 18\}$, while the processing time p_{ji} and the job weight w_j were randomly taken from $[1, 10]$ and $[1, 5]$, respectively. Finally, the common due date d was set to be 30% of the mean machine load, i.e., $d = 30\% \times (\sum_{j=1}^{n} (p_{j1} + p_{j2})/2)$.

We use the same language and platform as in Section 3.4, i.e., C++ and a PC with Intel Core i7-8550U 1.8 GHz CPU and 8 GB DDR4 RAM. For each value of n, we generate 25 random instances and report the experiment results in Table 2. For each procedure, its average running time (in milliseconds, ms) is shown in the columns of "time (ms)". Moreover, the improvement ratio from Algo_DP$_f$ to Algo_DP$_f'$, calculated as $(1 - (t'/t)) \times 100\%$ (where t' and t are the running time of Algo_DP$_f'$ and Algo_DP$_f$, respectively), is displayed in column "Imp (%)".

Now, we see the power of the two properties in Section 4.2, at least from the implementation point of view. For all the

(1) Reorder the jobs by nonincreasing order of their weights, i.e., $w_1 \geq w_2 \geq \cdots \geq w_n$;
(2) Enumerate all the possible partitions of the job set;
(3) for each partition
(4) do Keep the nonincreasing order of the jobs on each machine;
(5) Calculate the criterion value for this schedule;
(6) Output the best solution with the maximal criterion value;

Algorithm 2: Algo_EM.

Table 1: Running time (ms) of Algo_DP$_p$ and Algo_EM.

m	n	Algo_DP$_p$	Algo_EM
2	15	4.78	75.18
	20	8.86	2431.07
	25	18.69	94018.37
	30	44.00	—
	35	93.98	—
	40	169.97	—
	45	380.40	—
	50	891.51	—
3	5	7.59	4.41
	10	21.01	326.03
	15	55.04	27998.25
	20	166.15	—
	25	442.99	—
	30	901.01	—
	35	1977.48	—
	40	4807.40	—

Table 2: Comparison between Algo_DP$_f$ and Algo_DP$'_f$.

n	Algo_DP$_f$	Algo_DP$'_f$	
	Time (ms)	Time (ms)	Imp (%)
10	200.18	132.42	33.85
12	474.83	342.77	27.81
14	966.04	738.26	23.58
16	2044.31	1623.05	20.61
18	3710.85	2993.86	19.32
avg.	1479.24	1166.07	21.17

settings of value n, Algo_DP$'_f$ beats Algo_DP$_f$, and, on average, the former procedure saves 21.17% of the execution time of the latter one. The reason is that Algo_DP$'_f$ can avoid several unnecessary calculations if the two properties are violated.

5. Conclusions

In this paper, we considered two problems of scheduling on multimachine with weighted early work maximization (or equivalently, late work minimization):

(1) For identical machines version, i.e., problem $Pm|d_j = d|\max(X_w)$, we proposed a dynamic programming approach (Algo_DP$_p$) running in pseudopolynomial time and, therefore, showed that the problem is NP-hard in the weak sense. We also designed an enumeration algorithm (Algo_EM), to reveal the correctness and efficiency of Algo_DP$_p$ through computational experiments.
(2) For two-machine flow shop setting ($F2|d_j = d|\max(X_w)$), we studied a previous dynamic programming approach by precise analyses. By introducing two properties in the state function, some unnecessary calculations could be avoided during processing. Finally, we again used the computational experiments to show that these improvements are efficient at least from the implementation point of view.

For the future work, one can consider the more general cases, e.g., the individual due date models, and design exact or heuristic algorithms to solve them. Moreover, the approximation analyses for some special cases, especially for the flow or job shop setting, could be considered as another research direction.

Data Availability

The authors use the benchmarks from their references.

Conflicts of Interest

The authors declare that there are no conflicts of interest in this paper.

Acknowledgments

This research was funded by the Korea-China Young Researchers Exchange Program (2020), the Science Foundation of Shenyang University of Chemical Technology (no. LQ2020020), the Program for Liaoning Innovation Talents in University (no. LR2019034), and the Key Lab for OCME (School of Mathematical Sciences in Chongqing Normal University).

References

[1] V. Gordon, J.-M. Proth, and C. Chu, "A survey of the state-of-the-art of common due date assignment and scheduling research," *European Journal of Operational Research*, vol. 139, no. 1, pp. 1–25, 2002.
[2] C. Low, R.-K. Li, and G.-H. Wu, "Minimizing total earliness and tardiness for common due date single-machine scheduling with an unavailability interval," *Mathematical Problems in Engineering*, vol. 2016, Article ID 6124734, 12 pages, 2016.
[3] H. Emmons, "One-machine sequencing to minimize certain functions of job tardiness," *Operations Research*, vol. 17, no. 4, pp. 701–715, 1969.

[4] G. McMahon and M. Florian, "On scheduling with ready times and due dates to minimize maximum lateness," *Operations Research*, vol. 23, no. 3, pp. 475–482, 1975.
[5] J. M. Moore, "AnnJob, one machine sequencing algorithm for minimizing the number of late jobs," *Management Science*, vol. 15, no. 1, pp. 102–109, 1968.
[6] M. Sterna, "A survey of scheduling problems with late work criteria," *Omega*, vol. 39, no. 2, pp. 120–129, 2011.
[7] M. Sterna and K. Czerniachowska, "Polynomial time approximation scheme for two parallel machines scheduling with a common due date to maximize early work," *Journal of Optimization Theory and Applications*, vol. 174, no. 3, pp. 927–944, 2017.
[8] J. Błażewicz, "Scheduling preemptible tasks on parallel processors with information loss, Technique et," *Science Informatiques*, vol. 3, no. 6, pp. 415–420, 1984.
[9] C. N. Potts and L. N. Van Wassenhove, "Single machine scheduling to minimize total late work," *Operations Research*, vol. 40, no. 3, pp. 586–595, 1992.
[10] J. Błażewicz, E. Pesch, M. Sterna, and F. Werner, "Open shop scheduling problems with late work criteria," *Discrete Applied Mathematics*, vol. 134, no. 1, pp. 1–24, 2004.
[11] J. Blazewicz, E. Pesch, M. Sterna, and F. Werner, "A note on the two machine job shop with the weighted late work criterion," *Journal of Scheduling*, vol. 10, no. 2, pp. 87–95, 2007.
[12] X. Chen, M. Sterna, X. Han, and J. Blazewicz, "Scheduling on parallel identical machines with late work criterion: offline and online cases," *Journal of Scheduling*, vol. 19, no. 6, pp. 729–736, 2016.
[13] X. Chen, Y. Liang, M. Sterna, W. Wang, and J. Błażewicz, "Fully polynomial time approximation scheme to maximize early work on parallel machines with common due date," *European Journal of Operational Research*, vol. 284, no. 1, pp. 67–74, 2020.
[14] X. Chen, W. Wang, P. Xie, X. Zhang, M. Sterna, and J. Błażewicz, "Exact and heuristic algorithms for scheduling on two identical machines with early work maximization," *Computers & Industrial Engineering*, vol. 144, Article ID 106449, 2020.
[15] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.
[16] J. Błażewicz, E. Pesch, M. Sterna, and F. Werner, "The two-machine flow-shop problem with weighted late work criterion and common due date," *European Journal of Operational Research*, vol. 165, no. 2, pp. 408–415, 2005.
[17] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.
[18] J. Blazewicz, E. Pesch, M. Sterna, and F. Werner, "A comparison of solution procedures for two-machine flow shop scheduling with late work criterion," *Computers & Industrial Engineering*, vol. 49, no. 4, pp. 611–624, 2005.