Networks and Heterogeneous Media

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

NHM, 18(3): 1378–1392.
DOI: 10.3934/nhm.2023060
Received: 05 January 2023
Revised: 07 May 2023
Accepted: 10 May 2023
Published: 23 May 2023

Research article

Bicriteria multi-machine scheduling with equal processing times subject to release dates

Zhimeng Liu[1], Shuguang Li[1,*], Muhammad Ijaz Khan[2,3], Shaimaa A. M. Abdelmohsen[4] and Sayed M. Eldin[5]

[1] School of Computer Science and Technology, Shandong Technology and Business University, Yantai 264005, China
[2] Department of Mechanical Engineering, Lebanese American University, Beirut 362060, Lebanon
[3] Department of Mechanics and Engineering Science, Peking University, Beijing 100871, China
[4] Department of Physics, College of Science, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[5] Center of Research, Faculty of Engineering, Future University in Egypt, New Cairo 11835, Egypt

* **Correspondence:** Email: sgliytu@hotmail.com; Tel: +86-18753509226.

Abstract: This paper addresses the problem of scheduling n equal-processing-time jobs with release dates non-preemptively on identical machines to optimize two criteria simultaneously or hierarchically. For simultaneous optimization of total completion time (and makespan) and maximum cost, an algorithm is presented which can produce all Pareto-optimal points together with the corresponding schedules. The algorithm is then adapted to solve the hierarchical optimization of two min-max criteria, and the final schedule has a minimum total completion time and minimum makespan among the hierarchical optimal schedules. The two algorithms provided in this paper run in $O(n^3)$ time.

Keywords: scheduling; Pareto optimization; lexicographical optimization; identical machines; release dates

1. Introduction

Parallel machine scheduling has received extensive attention since 1950, given the wide diversity of real-world systems it represents. A variety of criteria has been considered. Among the most studied criteria are makespan (maximum completion time) and total completion time, which can measure the effective utilization of the machines. A second set of criteria are related to meeting due dates and thus considering the system's customers. If the criteria are not specified, we can consider two types of general objective functions: min-sum and min-max.

In real production, decision makers may need to consider a number of criteria simultaneously before arriving at a decision. However, it is often the case that different criteria are in conflict. A solution which is optimal with respect to a given criterion might be a poor candidate for some other criterion. Thus, in the last two decades, multicriteria optimization approaches and techniques have been increasingly applied to provide solutions where the criteria are balanced in an acceptable and profitable way [1, 2].

An important subclass in multicriteria optimization is bicriteria optimization where only two criteria, say γ^1 and γ^2, are considered. There are four popular approaches in the literature: (a) Positive combination optimization: find a schedule to minimize the positive linear combination of γ^1 and γ^2. (b) Constrained optimization: find a schedule to minimize γ^2 under an upper bound on γ^1. (c) Pareto optimization (also called simultaneous optimization): find all Pareto-optimal solutions for γ^1 and γ^2. A feasible schedule σ is *(strict) Pareto-optimal* for γ^1 and γ^2 if there is no feasible schedule σ' such that $\gamma^1(\sigma') \leq \gamma^1(\sigma)$ and $\gamma^2(\sigma') \leq \gamma^2(\sigma)$, where at least one of the inequalities is strict. The objective vector $(\gamma^1(\sigma), \gamma^2(\sigma))$ of a Pareto-optimal schedule σ is called a *Pareto-optimal point* [1]. A feasible schedule σ is *weak Pareto-optimal* for γ^1 and γ^2 if there is no feasible schedule σ' such that $\gamma^1(\sigma') < \gamma^1(\sigma)$ and $\gamma^2(\sigma') < \gamma^2(\sigma)$. (d) Hierarchical optimization (also called lexicographical optimization): find a schedule to minimize γ^2 among the set of optimal schedules minimizing γ^1. In hierarchical bicriteria scheduling problems, the two criteria have different levels of importance thus they are optimized in a lexicographic fashion. Such problems appear naturally in situations where there are several optimal solutions with respect to a specific objective and the decision maker needs to select from among these solutions the one with the best second objective.

In this paper, we consider the bicriteria problem of scheduling equal-processing-time jobs with release dates non-preemptively on identical machines. We apply a Pareto optimization approach to minimize the total completion time (and makespan) and maximum cost simultaneously, and we apply a hierarchical optimization approach to minimize two general min-max criteria hierarchically.

Formally speaking, we are given a set of n jobs, $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$, to be processed on m identical machines. The machines run in parallel and each machine can process at most one job at a time. All jobs have the same processing time $p > 0$. Each job, $J_j \in \mathcal{J}$, has a release date $r_j \geq 0$ before which it cannot be processed, as well as two cost functions $f_j(t)$ and $g_j(t)$ which denote the costs incurred if the job is completed at time t. We assume that all f_j and g_j are *regular*, i.e., f_j and g_j are non-decreasing in the job completion times [3].

A schedule assigns each job J_j to exactly one machine and specifies its completion time C_j on the machine. Given a schedule σ, let $f_j(C_j(\sigma))$ and $g_j(C_j(\sigma))$ be two scheduling costs of J_j. Then $f_{\max}(\sigma) = \max_j f_j(C_j(\sigma))$ and $g_{\max}(\sigma) = \max_j g_j(C_j(\sigma))$ are two maximum costs of σ. Two important special cases of maximum cost are the makespan $C_{\max}(\sigma) = \max_j\{C_j(\sigma)\}$ and the maximum lateness $L_{\max}(\sigma) = \max_j\{C_j(\sigma) - d_j\}$, where d_j denotes the due date of job J_j. We omit the argument σ when it is clear to which schedule we are referring.

The first bicriteria problem we consider is to determine Pareto-optimal schedules which simultaneously minimize the total completion time $\sum_{j=1}^n C_j$ (and makespan $C_{\max}$) and maximum cost $f_{\max}$. Following the notation schemes of [1–3], it can be denoted by $P|r_j, p_j = p|(\sum_{j=1}^n C_j, f_{\max})$ (and $P|r_j, p_j = p|(C_{\max}, f_{\max})$).

The second bicriteria problem we consider is to determine a lexicographical optimal schedule such that the secondary criterion $g_{\max}$ is minimized under the constraint that the primary criterion $f_{\max}$ is

minimized. Following the notation schemes of [1–3], it can be denoted by $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$, where the criterion mentioned first in the argument of Lex is the more important one.

For parallel machine scheduling that considers multiple criteria, please refer to [4–6] for the surveys. Examples of Pareto optimization and hierarchical optimization scheduling on parallel machines can be found in [7–11] and [12–14], respectively.

Bruno et al. [15] proved that problem $P||Lex(\sum_{j=1}^{n} C_j, C_{\max})$ (the jobs have unequal processing times and equal release dates) is NP-hard. Gupta et al. [16] further gave a complexity result: they showed that $P||Lex(\sum_{j=1}^{n} C_j, C_{\max})$ is strongly NP-hard. Hence, Pareto optimization problem $P||(\sum_{j=1}^{n} C_j, f_{\max})$ is also strongly NP-hard. Since the single criterion problem $P||C_{\max}$ is strongly NP-hard [17], the lexicographical optimization problem $P||Lex(f_{\max}, g_{\max})$ is strongly NP-hard, too. Also, problem $1|r_j|(\sum_{j=1}^{n} C_j, f_{\max})$ (the single machine case where the jobs have arbitrary processing times and release dates) is strongly NP-hard, due to the strong NP-hardness results by Lenstra et al. [18] for problems $1|r_j|\sum_{j=1}^{n} C_j$ and $1|r_j|L_{\max}$. Thus, we are interested in the special case where all jobs have equal processing times.

Although the problem setting of equal-processing-time jobs appears simple, it captures important aspects of a wide range of applications. For example, in standardized systems in practice, the products consistently have the same processing times. In networking and information systems, transmission packets also often have a constant length [19]. Since products and data packets usually arrive dynamically, it is reasonable to consider the jobs with release dates.

For single criterion scheduling, Kravchenko and Werner [19, 20] surveyed the approaches and exposed the problems with an open complexity status for scheduling jobs with equal processing times on parallel machines. Brucker and Shakhlevich [21] characterized optimal schedules for scheduling jobs with unit processing times on parallel machines by providing necessary and sufficient conditions of optimality. Hong et al. [22] studied the problem of scheduling jobs with equal processing times and eligibility restrictions on identical machines to minimize total completion time. For the problem with a fixed number of machines, they provided a polynomial time dynamic programming algorithm. For the problem with an arbitrary number of machines, they provided two polynomial time approximation algorithms with approximation ratios of 3/5 and 1.4. Vakhania [23] studied the problem of scheduling jobs with equal processing times on identical machines to minimize the maximum delivery completion time, which is defined to be the time by which all jobs are delivered. He presented an algorithm which can be considered as either pseudo-polynomial with time complexity $O(q_{\max} mn \log n)$ or as polynomial with time complexity $O(m\kappa n)$), where $q_{\max}$ denotes the maximum delivery time of all jobs and $\kappa < n$ is a parameter which is known only after the termination of the algorithm. The maximum cost minimization problem $P|r_j, p_j = p, \bar{d}_j|f_{\max}$ can be solved by the polynomial time algorithm developed in [19] by Kravchenko and Werner, where $\bar{d}_j$ denotes the deadline of job J_j before which J_j must be completed in any feasible schedule. Vakhania and Werner [24] studied the problem of scheduling jobs with equal processing times on uniform machines (processing jobs at different speeds) to minimize the maximum delivery completion time. For this problem whose complexity status remains open for a long time, they presented an $O(\lambda m^2 n \log n)$-time algorithm which is optimal under an explicitly stated special condition, where λ can be any of the magnitudes n or $q_{\max}$.

Tuzikov et al. [25] studied the bicriteria problems of scheduling jobs with equal processing times on uniform machines, denoted as $Q|p_j = p|(f_{\max}, g_{\max})$ and $Q|p_j = p|(\sum_{j=1}^{n} f_j, g_{\max})$, where $f_{\max}$ and

$g_{\max}$ are two min-max criteria, and $\sum_{j=1}^{n} f_j$ is a min-sum criterion. They showed that problems $Q|p_j = p|(f_{\max}, g_{\max})$ and $Q|p_j = p|(\sum_{j=1}^{n} f_j, g_{\max})$ can be solved iteratively in $O(n^4)$ and $O(n^5)$ time, respectively. Note that they discussed only the case of equal release dates. In this paper, we apply the framework used in [25] and extend the results in [25] to deal with release dates. In fact, the main contribution of this paper is the incorporation of the release dates and general maximum costs into the problem.

Sarin and Prakash [26] studied the lexicographical optimization problem of scheduling jobs with equal processing times and equal release dates on identical machines for various pairwise combinations of primary and secondary criteria f and g, where $f, g \in \{T_{\max}, \sum_j T_j, \sum_j U_j, \sum_j C_j, \sum_j w_j C_j\}$. (Please refer to [3] for the definitions.) Apart from $P|p_j = p|Lex(\sum_j U_j, \sum_j w_j C_j)$ whose computational complexity was left open in [26], all other problems $P|p_j = p|Lex(f, g)$ studied in [26] are solvable in polynomial time. Zhao and Yuan [27] revisited the bicriteria problems of scheduling jobs with equal processing times on uniform machines. They presented a comprehensive study on the problems with respect to various regular criteria. Particularly, they obtained an $O(n^3)$-time algorithm for $P|p_j = p|Lex(\sum_j U_j, \sum_j w_j C_j)$, solving the open problem posed in [26].

As for the parallel machines case with release dates and equal processing times, Simons [28] proposed the first polynomial algorithm running in $O(n^3 \log\log n)$ time for $P|r_j, p_j = p, \bar{d}_j| \sum_{j=1}^{n} C_j$. (The algorithm also solves the feasibility problem $P|r_j, p_j = p, \bar{d}_j|-$). Simons and Warmuth [29] further improved this bound to $O(mn^2)$. For the same problem, Dürr and Hurand [30], López-Ortiz and Quimper [31] gave algorithms that run in $O(n^3)$ and $O(\min\{1, p/m\}n^2)$ time, respectively. These schedules all minimize both the objectives $\sum_{j=1}^{n} C_j$ and $C_{\max}$. Since the maximum lateness $L_{\max}$ is upper-bounded by $\lceil np/m \rceil$, Fahimi and Quimper [32] remarked that problems $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, L_{\max})$ and $P|r_j, p_j = p|(C_{\max}, L_{\max})$ can be solved in polynomial time with time complexity $O(\log(np/m) \min\{1, p/m\}n^2)$ and using the binary search that calls the algorithm in [31] at most $\log(np/m)$ times. They also extended the algorithm presented in [31] for $P|r_j, p_j = p, \bar{d}_j| \sum_{j=1}^{n} C_j$ to solve a variation of the problem where the number of machines fluctuates over time. They further proved that minimizing the total cost of the jobs, i.e., $\sum_{j=1}^{n} f_j(S_j)$, for arbitrary functions $f_j(t)$ is NP-hard, where S_j denotes the start time of job J_j. They then specialized this objective function to the case that it is merely contingent on the time and showed that although this case is pseudo-polynomial solvable, one can derive polynomial time algorithms for either a monotonic or periodic cost function.

To the best of our knowledge, problems $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$ and $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$ have not been studied to date. Note that here $f_{\max}$ and $g_{\max}$ are two general min-max criteria. The above-mentioned results [28–32] discussed only $\sum_{j=1}^{n} C_j$, $C_{\max}$ or $L_{\max}$.

In Section 2, we present an $O(n^3)$-time algorithm for $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$. The algorithm also solves problem $P|r_j, p_j = p|(C_{\max}, f_{\max})$. Consequently, problem $P|r_j, p_j = p|f_{\max}$ can be solved in $O(n^3)$ time, which has its own independent interest. In Section 3, we adapt the algorithm to solve $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$ in $O(n^3)$ time. The final generated schedule also has the minimum total completion time and minimum makespan among the lexicographical optimal schedules for $f_{\max}$ and $g_{\max}$. Finally, we draw some concluding remarks in the last section.

2. An algorithm for $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$

In this section we will present an $O(n^3)$-time algorithm for $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$. As a by-product, the last schedule constructed by the algorithm is optimal for $P|r_j, p_j = p|f_{\max}$.

For ease of discussion, throughout the paper, we always represent a feasible schedule σ by a sequence $J_{\sigma(1)}, J_{\sigma(2)}, \cdots, J_{\sigma(n)}$ of jobs, where $J_{\sigma(i)}$ is the job scheduled at the i-th position in σ, $i = 1, 2, \ldots, n$. The positions in σ are indexed from 1 to n in non-decreasing order of their start times in σ (ties broken in favor of the job on the machine with the smallest index).

Intuitively, if a job has a large cost when it completes late, then we need to move it to the left (i.e., start it earlier) to decrease its cost, even if it has a large release date. Therefore, it is quite often that in a feasible schedule, some jobs with larger release dates may start earlier than some jobs with smaller release dates.

To recover a schedule from its job sequence representation, we need the following lemma whose proof can be found in [28]. Though simple, this lemma plays a non-negligible role in our algorithms. It allows us to focus on the positions of the jobs in a schedule; we do not worry about their release dates.

Let $S_{\sigma(i)}$ denote the start time of $J_{\sigma(i)}$ in $\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \cdots, J_{\sigma(n)})$, $i = 1, 2, \ldots, n$.

Lemma 2.1. *([28]) For any feasible schedule, a solution σ identical except in machine assignment exists and is cyclic, i.e., for any i, $J_{\sigma(i)}, J_{\sigma(i+m)}, \ldots$ are scheduled on the same machine. Moreover, $S_{\sigma(1)} = r_{\sigma(1)}$, $S_{\sigma(i)} = \max\{S_{\sigma(i-1)}, r_{\sigma(i)}\}$ $(i = 2, 3, \ldots, m)$ and $S_{\sigma(i)} = \max\{S_{\sigma(i-1)}, r_{\sigma(i)}, S_{\sigma(i-m)} + p\}$ $(i = m+1, m+2, \ldots, n)$.*

The ε-constraint method (see, e.g., [1, 2]) provides a general way to find Pareto-optimal points: let y be the optimal value of constrained optimization problem $\alpha|f \le \hat{x}|g$, and let x be the optimal value of constrained optimization problem $\alpha|g \le y|f$. Then (x, y) is a Pareto-optimal point for problem $\alpha||(f, g)$.

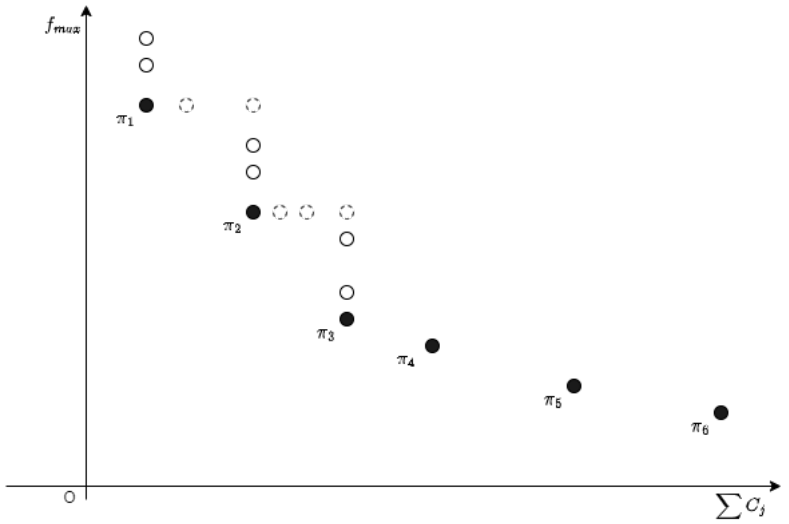

Figure 1. Illustration of Algorithm M_1.

The algorithm follows the framework used in [25] which repeatedly uses the ε-constraint method to construct the Pareto-optimal schedules. Please see Figure 1 for an illustration. Similar figures and

illustrations can be found, e.g., in [25, 33]. All circles (white solid, black solid and white dashed) in Figure 1 represent weak Pareto-optimal points (schedules), but only the black solid circles represent Pareto-optimal points (schedules). The weak Pareto-optimal schedules ($\sigma_1, \sigma_2, \sigma_3, \ldots$, which are generated in turn in Algorithm M_1) are constructed in strictly decreasing order of their $f_{\max}$-values, and within that order in non-decreasing order of their $\sum_{j=1}^{n} C_j$-values. Since the constraint $f_{\max} < y$ is used instead of $f_{\max} \leq y$, all white dashed circles will be ignored by Algorithm M_1. The Pareto-optimal schedules output by Algorithm M_1 are $\pi_1, \pi_2, \pi_3, \ldots$. The last Pareto-optimal schedule has the minimum $f_{\max}$-value.

Let $\Omega(\mathcal{J})$ denote the *Pareto set* which consists of all Pareto-optimal points together with their corresponding Pareto-optimal schedules. Let $\Pi(\mathcal{J})$ denote the set of all feasible schedules for $\mathcal{J}$. Let $\Pi(\mathcal{J}, y) \subseteq \Pi(\mathcal{J})$ denote the set of the schedules with maximum cost ($f_{\max}$-value) less than y, where y denotes a given threshold value. Obviously, we have that $\Pi(\mathcal{J}, +\infty) = \Pi(\mathcal{J})$.

Below is the algorithm called Algorithm M_1 for constructing the Pareto set $\Omega(\mathcal{J})$ for $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$. It first assigns the unassigned job with the largest release date to the i-th position ($i = n, n-1, \ldots, 1$), ignoring the scheduling cost $f_{\max}$ (see the initial schedule $\hat{\sigma}$ below). It then repeatedly decreases the $f_{\max}$-value of the current schedule until the cost cannot be further improved. During the process, all Pareto-optimal schedules are constructed one by one.

The initial schedule is $\hat{\sigma} = \{J_{\hat{\sigma}(1)}, J_{\hat{\sigma}(2)}, \ldots, J_{\hat{\sigma}(n)}\}$, where the jobs $J_{\hat{\sigma}(1)}, J_{\hat{\sigma}(2)}, \ldots, J_{\hat{\sigma}(n)}$ are in non-decreasing order of their release dates (ties broken arbitrarily). Note that this order is also the non-decreasing order of their start times in $\hat{\sigma}$ (ties broken in favor of the job on the machine with the smallest index). It is easy to see that $\hat{\sigma}$ is optimal for $P|r_j, p_j = p|\sum_{j=1}^{n} C_j$ and $P|r_j, p_j = p|C_{\max}$. (Set the start times of the jobs in $\hat{\sigma}$ by Lemma 2.1.)

The basic idea of our algorithms is as follows: schedule the jobs backwardly (from the right to the left) and at each decision point always select the job with the largest release date from among the candidate jobs (check the choice of $\hat{\sigma}$ in Step 1 of Algorithm M_1 and Step 3 of Procedure $A_1(\ldots)$, as well as the choice of π^* in Step 1 of Algorithm M_2 and Step 3 of Procedure $A_2(\ldots)$). To coincide with this idea, we treat the initial schedule $\hat{\sigma} = \{J_{\hat{\sigma}(1)}, J_{\hat{\sigma}(2)}, \cdots, J_{\hat{\sigma}(n)}\}$ as a schedule in which the jobs $J_{\hat{\sigma}(n)}, J_{\hat{\sigma}(n-1)}, \ldots, J_{\hat{\sigma}(1)}$ are in non-increasing order of their release dates.

Algorithm M_1**:** Input: An instance of $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$.

Output: The Pareto set $\Omega(\mathcal{J})$.

Step 1. Initially, set $s = 1$. Let $\sigma_s = \{J_{\sigma_s(1)}, J_{\sigma_s(2)}, \cdots, J_{\sigma_s(n)}\} = \hat{\sigma}$, $y_s = f_{\max}(\sigma_s)$. Let $\Omega(\mathcal{J}) = \emptyset$, $k = 0$.

Step 2. Run Procedure $A_1(y_s)$ to get a schedule σ_{s+1}, using σ_s as the input schedule.

Step 3. If $\sigma_{s+1} \neq \emptyset$, then do the following:

(i) Set $y_{s+1} = f_{\max}(\sigma_{s+1})$.

(ii) If $\sum_{j=1}^{n} C_j(\sigma_s) < \sum_{j=1}^{n} C_j(\sigma_{s+1})$, then set $k = k + 1$ and $\pi_k = \sigma_s$. Incorporate $(\sum_{j=1}^{n} C_j(\pi_k), f_{\max}(\pi_k), \pi_k)$ into $\Omega(\mathcal{J})$.

(iii) Set $s = s + 1$. Go to Step 2.

Step 4. If $\sigma_{s+1} = \emptyset$, then set $k = k + 1$ and $\pi_k = \sigma_s$. Incorporate $(\sum_{j=1}^{n} C_j(\pi_k), f_{\max}(\pi_k), \pi_k)$ into $\Omega(\mathcal{J})$ and return $\Omega(\mathcal{J})$.

Procedure $A_1(y_s)$**:**

Input: Schedule $\sigma_s = \{J_{\sigma_s(1)}, J_{\sigma_s(2)}, \cdots, J_{\sigma_s(n)}\}$ with $y_s = f_{\max}(\sigma_s)$.

Output: Schedule $\sigma_{s+1} = \{J_{\sigma_{s+1}(1)}, J_{\sigma_{s+1}(2)}, \cdots, J_{\sigma_{s+1}(n)}\}$ which has the minimum total completion time (and minimum makespan) among all schedules in $\Pi(\mathcal{J}, y_s)$.

Step 1. Initially, set $h = 0$. Let $\sigma^h = \{J_{\sigma^h(1)}, J_{\sigma^h(2)}, \cdots, J_{\sigma^h(n)}\}$, where $\sigma^h(i) = \sigma_s(i)$, $i = 1, 2, \ldots, n$.

Step 2. Set the start times of the jobs in σ^h (by Lemma 2.1): $S_{\sigma^h(1)} = r_{\sigma^h(1)}$; for $i = 2, 3, \ldots, m$, let $S_{\sigma^h(i)} = \max\{S_{\sigma^h(i-1)}, r_{\sigma^h(i)}\}$; for $i = m+1, m+2, \ldots, n$, let $S_{\sigma^h(i)} = \max\{S_{\sigma^h(i-1)}, r_{\sigma^h(i)}, S_{\sigma^h(i-m)}+p\}$. Update the cost $f_j(C_j(\sigma^h))$ of job j in σ^h, $J_j \in \mathcal{J}$.

Step 3. Adjust σ^h:

IF for all i, the inequality $f_{\sigma^h(i)}(C_{\sigma^h(i)}) < y_s$ holds, THEN Return $\sigma_{s+1} = \sigma^h$.

ELSE Pick a job $J_{\sigma^h(i)}$ such that $f_{\sigma^h(i)}(C_{\sigma^h(i)}) \geq y_s$. Let $E(\sigma^h(i)) = \{l | 1 \leq l \leq i \wedge f_{\sigma^h(l)}(C_{\sigma^h(i)}) < y_s\}$ denote the set of the candidate jobs at time $C_{\sigma^h(i)}$.

IF $E(\sigma^h(i)) = \emptyset$, THEN Return $\sigma_{s+1} = \emptyset$.

ELSE Find the job with the largest release date in $E(\sigma^h(i))$, say $J_{\sigma^h(e)}$. Let $J_{\sigma^h(e)}$ be scheduled at the i-th position instead of $J_{\sigma^h(i)}$. Set $J_x = J_{\sigma^h(i)}$. For $q = i-1, i-2, \ldots, e+1$ (this ordering is used crucially), let $J_{\sigma^h(q)}$ be the job in $\{J_{\sigma^h(q)}, J_x\}$ with the larger release date, and let J_x be the other job. Finally, let J_x be scheduled at the e-th position.

Let $\sigma^{h+1} = \sigma^h$ and then set $h = h + 1$. Go to Step 2.

Remark 1. Let us illustrate Step 3 of Procedure $A_1(y_s)$ in more detail. Suppose that we find a job $J_{\sigma^h(i)}$ violating its inequality. We remove $J_{\sigma^h(i)}$ from position i and select the candidate job $J_{\sigma^h(e)}$ from $E(\sigma^h(i))$ which has the largest release date. Let $J_{\sigma^h(e)}$ be scheduled at position i. Treat $J_{\sigma^h(i)}$ as *the job in hand*, denoted by J_x, for which we need to find a suitable position. We compare J_x and $J_{\sigma^h(i-1)}$, and let the one with the larger release date be scheduled at position $i-1$. Let J_x be the other job. As will be seen in Lemma 2.2 below, the job at position $i-1$ now has the release date that is not less than the largest release date among the candidate jobs in $E(\sigma^h(i-1))$. Its cost may not less than y_s at time $C_{\sigma^h(i-1)}$. However, we do not worry about this possibility, since the job can be moved further to the left in the next iterations. We continue to compare J_x and $J_{\sigma^h(i-2)}$, and so on. In this way, we can find suitable positions for jobs $J_{\sigma^h(i)}, J_{\sigma^h(i-1)}, \ldots, J_{\sigma^h(e+1)}$. Thus, we accomplish the idea mentioned before: schedule the jobs backwardly and at each decision point always select the job with the largest release date from among the candidate jobs.

Example: We now demonstrate an example to illustrate Algorithm M_1. There are three machines and six jobs $J_1, J_2, \ldots, J_6$ with processing four times , where $r_1 = r_2 = 0$, $r_3 = 1$, $r_4 = 2$, $r_5 = r_6 = 3$; $d_1 = d_2 = 4$, $d_3 = 3$, $d_4 = 2$, $d_5 = d_6 = 1$. Algorithm M_1 works as follows:

(1) $\sigma_1 = \hat{\sigma} = \{J_1, J_2, J_3, J_4, J_5, J_6\}$, jobs $J_1, J_2, \ldots, J_6$ are in non-decreasing order of their release dates. The schedule is recovered by Lemma 2.1 with $\sum C_j(\sigma_1) = 38$ and $L_{\max}(\sigma_1) = 8$.

(2) Run Procedure $A_1(y_1)$, where $y_1 = 8$.

Initially, $\sigma^0 = \sigma_1 = \{J_1, J_2, J_3, J_4, J_5, J_6\}$. The job violating the inequality is J_6. The set of the candidate jobs at time C_6 is $E(\sigma^0(6)) = \{J_1, J_2, J_3, J_4\}$. Since J_4 has the largest release date among

the jobs in $E(\sigma^0(6))$, it is scheduled at the sixth position instead of J_6. Job J_6 becomes the job in hand. We compare J_6 and J_5. Since $r_5 \geq r_6$, J_5 stays at the fifth position. We continue to consider the fourth position. The fourth position is not occupied because J_4 has been moved from this position to the sixth position. Therefore, J_6 is scheduled at the fourth position. Set the start times of the jobs by Lemma 2.1. We get: $\sigma^1 = \{J_1, J_2, J_3, J_6, J_5, J_4\}$ with $\sum C_j(\sigma^1) = 38$ and $L_{\max}(\sigma^1) = 7$. Since $L_{\max}(\sigma_1) = 8 > 7 = L_{\max}(\sigma^1)$, Procedure $A_1(y_1)$ returns $\sigma_2 = \sigma^1 = \{J_1, J_2, J_3, J_6, J_5, J_4\}$ with $\sum C_j(\sigma_2) = 38$ and $L_{\max}(\sigma_2) = 7$. Since $\sum C_j(\sigma_1) = 38 = \sum C_j(\sigma_2)$, by Step 3 of Algorithm M_1, we get rid of σ_1 since it cannot be a Pareto-optimal schedule.

(3) Run Procedure $A_1(y_2)$, where $y_2 = 7$.

(i) Initially, $\sigma^0 = \sigma_2 = \{J_1, J_2, J_3, J_6, J_5, J_4\}$. The jobs violating the inequalities are J_4, J_5, J_6. The set of the candidate jobs at time C_4 is $E(\sigma^0(6)) = \{J_1, J_2, J_3\}$. Since J_3 has the largest release date among the jobs in $E(\sigma^0(6))$, it is scheduled at the sixth position instead of J_4. Job J_4 becomes the job in hand. We compare J_4 and J_5. Next, we compare J_4 and J_6, and then decide to schedule J_4 at the third position. Set the start times of the jobs by Lemma 2.1. We get $\sigma^1 = \{J_1, J_2, J_4, J_6, J_5, J_3\}$ with $\sum C_j(\sigma^1) = 40$ and $L_{\max}(\sigma^1) = 7$.

(ii) Now, the jobs violating the inequalities are J_3, J_5, J_6. We select J_3 as the job in hand and adjust σ^1. We get $\sigma^2 = \{J_1, J_3, J_4, J_6, J_5, J_2\}$ with $\sum C_j(\sigma^2) = 42$ and $L_{\max}(\sigma^2) = 8$.

(iii) Since $y_2 = 7$, the jobs violating the inequalities are J_5, J_6. We select J_5 as the job in hand and adjust σ^2. We get: $\sigma^3 = \{J_1, J_4, J_5, J_6, J_3, J_2\}$ with $\sum C_j(\sigma^3) = 47$ and $L_{\max}(\sigma^3) = 8$.

(iv) Since $y_2 = 7$, the jobs violating the inequalities are J_2, J_3, J_6. We select J_2 as the job in hand and adjust σ^3. Since $E(\sigma^3(6)) = \emptyset$, Procedure $A_1(y_2)$ returns $\sigma_3 = \emptyset$, implying that $\Pi(\mathcal{J}, y_2) = \emptyset$.

By Step 4 of Algorithm M_1, we get: $\pi_1 = \sigma_2 = \{J_1, J_2, J_3, J_6, J_5, J_4\}$ with $\sum C_j(\pi_1) = 38$ and $L_{\max}(\pi_1) = 7$. Finally, Algorithm M_1 returns the Pareto set $\Omega(\mathcal{J}) = \{(38, 7, \pi_1)\}$.

Step 1 of Procedure $A_1(y_s)$ can be implemented in $O(n)$ time. Steps 2 and 3 can be implemented in $O(n)$ time in each iteration. Here, an iteration refers to a job inequality violation adjustment. In each iteration, there is a job which has to be moved to the left because of the inequality violation. Later, by Lemma 3.1 we will know that this job cannot be moved back again. Hence, since there are n jobs and each job goes through at most $n-1$ positions (from the rightmost to the leftmost), the total number of iterations is $O(n^2)$. The running time of Procedure $A_1(y_s)$ is $O(n^3)$.

The running time of Algorithm M_1 is $O(n^3)$, since although it is not clear how many times Algorithm M_1 calls for Procedure $A_1(...)$, the total number of iterations in all calls for Procedure $A_1(...)$ is still $O(n^2)$, and each iteration can be done in $O(n)$ time.

In Lemma 2.2 below, we will prove the following: (1) Algorithm M_1 schedules the jobs backwardly and always selects the job with the largest release date from among the candidate jobs; (2) in the course of the algorithm, no job moved to the left can be moved back again; (3) at each iteration, Procedure $A_1(y_s)$ constructs a schedule in which the i-th job, counting from the left, starts no later than the i-th job in any feasible schedule in $\Pi(\mathcal{J}, y_s)$, $i = 1, 2, \ldots, n$.

Therefore, the schedule obtained at each iteration of Procedure $A_1(y_s)$ is no worse than any feasible schedule in $\Pi(\mathcal{J}, y_s)$ for the objectives $\sum_{j=1}^{n} C_j$ and $C_{\max}$. Therefore, the final schedule (i.e., σ_{s+1}, if it exists) obtained by Procedure $A_1(y_s)$ at the last iteration is in $\Pi(\mathcal{J}, y_s)$ and optimal for $\sum_{j=1}^{n} C_j$ and $C_{\max}$.

Lemma 2.2. *Let $\sigma^h = (J_{\sigma^h(1)}, J_{\sigma^h(2)}, \cdots, J_{\sigma^h(n)})$ be the schedule obtained at iteration h ($h = 0, 1, \ldots$) of Procedure $A_1(y_s)$. Let $\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \cdots, J_{\sigma(n)})$ be any schedule in $\Pi(\mathcal{J}, y_s)$. Then for $i = 1, 2, \ldots, n$, we have the following: (1) $r_{\sigma^h(i)} \geq \max\{r_j | j \in E(\sigma^h(i))\}$; (2) $S_{\sigma^h(i)} \leq S_{\sigma(i)}$ (and thus $C_{\sigma^h(i)} \leq C_{\sigma(i)}$); (3) $C_{\sigma^h(i)} \leq C_{\sigma^{h+1}(i)}$.*

Proof. We prove the lemma by induction on s and h.

First, we consider the input schedule for Procedure $A_1(y_1)$, which is the initial schedule $\hat{\sigma}$. Property (1) of the lemma clearly holds. We are going to prove property (2) for the base case $h = 0$ of the call for Procedure $A_1(y_1)$. Let σ be any schedule in $\Pi(\mathcal{J}, y_1)$. We compare $\hat{\sigma}$ and σ backwardly (from the right to the left) looking for a difference between the jobs. Suppose that the first difference occurs at the k-th position, which is occupied by jobs J_a and J_b in $\hat{\sigma}$ and σ, respectively. By the construction of $\hat{\sigma}$, we know that $r_a \geq r_b$. Since J_a is processed earlier than J_b in σ, we can safely interchange J_a and J_b in σ (regardless of whether J_a can be scheduled at the k-th position in σ), without increasing the job start or completion time at any position. Repetition of this argument shows that σ can be safely transformed into $\hat{\sigma}$. Thus, for $i = 1, 2, \ldots, n$, we have that $S_{\hat{\sigma}(i)} \leq S_{\sigma(i)}$, proving property (2) for the base case $h = 0$ of the call for Procedure $A_1(y_1)$. Hence, the lemma holds for the 0-th iteration of Procedure $A_1(y_1)$.

Assume that the lemma holds for the first h iterations of Procedure $A_1(y_1)$. We now consider the $(h+1)$-th iteration. More precisely, we observe σ^{h+1} at the moment that it is being constructed to adjust σ^h during Step 3 of Procedure $A_1(y_1)$, but the next round of Step 2 has not been executed yet. That is, σ^{h+1} is obtained from σ^h by performing an inequality violation adjustment, but the completion times and the costs of the jobs in σ^{h+1} have not been updated yet.

As described in Step 3 of Procedure $A_1(y_1)$, for adjusting σ^h, we pick a job $J_{\sigma^h(i)}$ in σ^h such that $f_{\sigma^h(i)}(C_{\sigma^h(i)}) \geq y_s$, and find a job $J_{\sigma^h(e)}$ which has the largest release date in $E(\sigma^h(i))$. Let $J_{\sigma^h(e)}$ be scheduled at the i-th position instead of $J_{\sigma^h(i)}$. Clearly, $r_{\sigma^h(e)} = \max\{r_j | j \in E(\sigma^h(i))\}$. By the inductive assumption, if we do not consider $J_{\sigma^h(i)}$, then we have that $r_{\sigma^h(i-1)} \geq \max\{r_j | j \in E(\sigma^h(i-1))\}$. By comparing $J_x = J_{\sigma^h(i)}$ and $J_{\sigma^h(i-1)}$, letting the one with the larger release date be scheduled at position $i-1$, and letting J_x be the other job, we can ensure that $r_{\sigma^h(i-1)} \geq \max\{r_j | j \in E(\sigma^h(i-1))\}$ after $J_{\sigma^h(i)}$ has been taken into consideration. We continue to deal with J_x and $J_{\sigma^h(i-2)}$, and so on, as described in Step 3. Therefore, we prove property (1) for the $(h+1)$-th iteration of Procedure $A_1(y_1)$.

To prove property (2) for the $(h+1)$-th iteration of Procedure $A_1(y_1)$, we compare σ^{h+1} and σ backwardly looking for a difference between the jobs. Suppose that the first difference occurs at the k-th position, which is occupied by jobs J_a and J_b in σ^{h+1} and σ, respectively. By the inductive assumption, $C_{\sigma^h(k)} \leq C_{\sigma(k)}$ and thus $f_b(C_{\sigma^h(k)}) \leq f_b(C_{\sigma(k)}) < y_1$ (test the feasibility of job J_b when it is completed at $C_{\sigma^h(k)}$), which means that job J_b is also a candidate job at time $C_{\sigma^h(k)}$. By the rule of selecting a candidate job in favor of the largest release date (which has just been proved), we know that $r_a \geq r_b$. Since J_a is processed earlier than J_b in σ, we can safely interchange J_a and J_b in σ, without increasing the job start or completion time at any position. Repetition of this argument shows that σ can be safely transformed into σ^{h+1}, without increasing the job start or completion time at any position. Thus, for $i = 1, 2, \ldots, n$ we have that $S_{\sigma^{h+1}(i)} \leq S_{\sigma(i)}$, proving property (2) of the lemma for the $(h+1)$-th iteration of Procedure $A_1(y_1)$.

Finally, we observe σ^{h+1} at the moment that it has been obtained and the next round of Step 2 has been executed already, which is, the moment when the completion times and the costs of the jobs in σ^{h+1} have been updated. Clearly, for $e+1 \leq l \leq i$ we have that $r_{\sigma^h(l)} \geq r_{\sigma^h(e)}$. (Otherwise, since $J_{\sigma^h(e)}$ is a candidate job at time $C_{\sigma^h(l)}$, by the rule of selecting a candidate job in favor of the

largest release date, $J_{\sigma^h(e)}$ should have been scheduled at the l-th position instead of $J_{\sigma^h(l)}$.) After scheduling jobs $J_{\sigma^h(i)}, J_{\sigma^h(i-1)}, \ldots, J_{\sigma^h(e)}$ at the suitable positions, only the release date at i-th position may become smaller. The release dates at all the other positions remain unchanged or become larger. Since $J_{\sigma^h(1)}, J_{\sigma^h(2)}, \cdots, J_{\sigma^h(n)}$ are processed in non-decreasing order of their start times in σ^h, by Lemma 2.1, during the adjustment of σ^h, none among $C_{\sigma^h(1)}, C_{\sigma^h(2)}, \cdots, C_{\sigma^h(n)}$ can decrease. It follows that for all i $C_{\sigma^h(i)} \leq C_{\sigma^{h+1}(i)}$. Therefore, we prove property (3) for the $(h+1)$-th iteration of Procedure $A_1(y_1)$. Note that property (1) still holds, since the increasing completion times can only reduce the set of candidate jobs, and thus only makes property (1) easier to satisfy. Property (3) also holds, since scheduling the jobs in a given sequence as described in Lemma 2.1 is optimal.

Summarizing the above, we have proved the lemma for Procedure $A_1(y_1)$. Assume that the lemma holds for Procedures $A_1(y_1)$, $A_1(y_2), \ldots,$ $A_1(y_s)$. We now consider Procedure $A_1(y_{s+1})$. Let $\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \cdots, J_{\sigma(n)})$ be any schedule in $\Pi(\mathcal{J}, y_{s+1})$. Since $y_{s+1} < y_s$, we have that $\Pi(\mathcal{J}, y_{s+1}) \subseteq \Pi(\mathcal{J}, y_s)$. The input schedule for Procedure $A_1(y_{s+1})$ is just the output schedule of Procedure $A_1(y_s)$. By the inductive assumption, the lemma holds for this schedule and σ. Assume that the lemma holds for the first h iterations of Procedure $A_1(y_{s+1})$. We now consider σ^{h+1} and σ. In almost the same manner as described above, we can prove that the lemma holds for the $(h+1)$-th iteration of Procedure $A_1(y_{s+1})$.

By the principle of induction, we complete the proof. □

We get the following:

Lemma 2.3. *Let σ^{last} (i.e., σ_{s+1}) be the schedule obtained at the last iteration of Procedure $A_1(y_s)$. If $\sigma^{last} = \emptyset$, then $\Pi(\mathcal{J}, y_s) = \emptyset$; Otherwise σ^{last} is a schedule which has minimum total completion time (and minimum makespan) among all schedules in $\Pi(\mathcal{J}, y_s)$.*

Proof. If $\sigma^{last} = \emptyset$, then by Step 3 of Procedure $A_1(y_s)$, at the last iteration, there is a job $J_{\sigma^h(i)}$ such that $f_{\sigma^h(i)}(C_{\sigma^h(i)}) \geq y_s$ and $E(\sigma^h(i)) = \emptyset$, where $E(\sigma^h(i)) = \{l | 1 \leq l \leq i \wedge f_{\sigma^h(l)}(C_{\sigma^h(i)}) < y_s\}$ denotes the set of the candidate jobs at time $C_{\sigma^h(i)}$. Therefore, the first i jobs in σ^{last-1} can only be scheduled at the first i positions in any schedule in $\Pi(\mathcal{J}, y_s)$, but none of them can be scheduled at the i-th position. This contradiction tells us that $\Pi(\mathcal{J}, y_s) = \emptyset$.

If $\sigma^{last} \neq \emptyset$, then by Lemma 2.2, we have the following: $C_{\sigma^{last}(i)} \leq C_{\sigma(i)}$, $i = 1, 2, \ldots, n$, where $\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \cdots, J_{\sigma(n)})$ denotes any schedule in $\Pi(\mathcal{J}, y_s)$. Hence, σ^{last} is a schedule which has the minimum total completion time (and minimum makespan) among all schedules in $\Pi(\mathcal{J}, y_s)$.

□

Algorithm M_1 applies the ε-constraint method of Pareto optimization. The following theorem shows its correctness, the proof of which is based on Lemma 2.3. We omit the proof since it is standard and implied in Figure 1 and, e.g., [1, 25].

Theorem 2.4. *Algorithm M_1 constructs all Pareto-optimal points together with the corresponding Pareto-optimal schedules for $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$ in $O(n^3)$ time. Consequently, problem $P|r_j, p_j = p|f_{\max}$ can also be solved in $O(n^3)$ time.*

Moreover, by Lemma 2.3, Algorithm M_1 also solves $P|r_j, p_j = p|(C_{\max}, f_{\max})$ in $O(n^3)$ time. We only need to change the obtained Pareto-optimal points. The Pareto-optimal schedules for $C_{\max}$ and $f_{\max}$ are the same as those for $\sum_{j=1}^{n} C_j$ and $f_{\max}$.

3. An algorithm for $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$

In this section we will adapt Algorithm M_1 to solve $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$ in $O(n^3)$ time. Note that Lemma 2.1 still holds for this problem.

During the run of Algorithm M_1, we only care about the criteria $\sum_{j=1}^{n} C_j$ and $f_{\max}$, totally ignoring $g_{\max}$. To solve $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$, we need to incorporate $g_{\max}$ into the framework.

Let schedule π^* be the last schedule obtained upon the completion of Algorithm M_1. Let $f^* = f_{\max}(\pi^*)$. By Theorem 2.4, π^* is an optimal schedule for $P|r_j, p_j = p|f_{\max}$, i.e., $f^* = \min_{\sigma\in\Pi(\mathcal{J})} f_{\max}(\sigma)$. Let σ be any schedule in $\Pi(\mathcal{J})$ with $f_{\max}(\sigma) = f^*$. By Lemma 2.2, the i-th job in π^*, counting from the left, starts no later than the i-th job in σ, $i = 1, 2, \ldots, n$. As we saw in the last section, this property plays a key role in solving $P|r_j, p_j = p|(\sum_{j=1}^{n} C_j, f_{\max})$. We will maintain a similar property for solving $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$.

Let $\Pi(\mathcal{J}, f^*, y)$ denote the set of the schedules in $\Pi(\mathcal{J})$ whose $f_{\max}$-values are equal to f^* and $g_{\max}$-values are less than y.

Below is the algorithm (Algorithm M_2) for $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$. (The basic idea of the algorithm has been illustrated in the preceding section before the description of Algorithm M_1.) The initial schedule is π^*, which is the optimal schedule for $P|r_j, p_j = p|f_{\max}$ obtained by Algorithm M_1.

Algorithm M_2:

Input: An instance of $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$.

Output: A lexicographical optimal schedule such that $g_{\max}$ is minimized under the constraint that the $f_{\max}$-value is equal to f^*.

Step 1. Initially, set $s = 1$. Let $\sigma_s = \pi^*$, $y_s = g_{\max}(\sigma_s)$.

Step 2. Run Procedure $A_2(y_s)$ to get a schedule σ_{s+1} in $\Pi(\mathcal{J}, f^*, y_s)$, using σ_s as the input schedule.

Step 3. If $\sigma_{s+1} \neq \emptyset$, then set $y_{s+1} = g_{\max}(\sigma_{s+1})$, $s = s + 1$. Go to Step 2. Otherwise, return σ_s.

Procedure $A_2(y_s)$:

Input: Schedule $\sigma_s = \{J_{\sigma_s(1)}, J_{\sigma_s(2)}, \cdots, J_{\sigma_s(n)}\}$ with $f_{\max}(\sigma_s) = f^*$ and $y_s = g_{\max}(\sigma_s)$.

Output: Schedule $\sigma_{s+1} = \{J_{\sigma_{s+1}(1)}, J_{\sigma_{s+1}(2)}, \cdots, J_{\sigma_{s+1}(n)}\}$ which has the minimum total completion time (and minimum makespan) among all schedules in $\Pi(\mathcal{J}, f^*, y_s)$.

Step 1. Initially, set $h = 0$. Let $\sigma^h = \{J_{\sigma^h(1)}, J_{\sigma^h(2)}, \cdots, J_{\sigma^h(n)}\}$, where $\sigma^h(i) = \sigma_s(i)$, $i = 1, 2, \ldots, n$.

Step 2. Set the start times of the jobs in σ^h (by Lemma 2.1): $S_{\sigma^h(1)} = r_{\sigma^h(1)}$; for $i = 2, 3, \ldots, m$, let $S_{\sigma^h(i)} = \max\{S_{\sigma^h(i-1)}, r_{\sigma^h(i)}\}$; for $i = m+1, m+2, \ldots, n$, let $S_{\sigma^h(i)} = \max\{S_{\sigma^h(i-1)}, r_{\sigma^h(i)}, S_{\sigma^h(i-m)}+p\}$.

Step 3. Adjust σ^h:

IF for all i, the inequalities $f_{\sigma^h(i)}(C_{\sigma^h(i)}) \leq f^*$ and $g_{\sigma^h(i)}(C_{\sigma^h(i)}) < y_s$ hold,
THEN Return $\sigma_{s+1} = \sigma^h$.

ELSE Pick a job $J_{\sigma^h(i)}$ such that $f_{\sigma^h(i)}(C_{\sigma^h(i)}) > f^*$ or $g_{\sigma^h(i)}(C_{\sigma^h(i)}) \geq y_s$. Let $E(\sigma^h(i)) = \{l | 1 \leq l \leq i \wedge f_{\sigma^h(l)}(C_{\sigma^h(i)}) \leq f^* \wedge g_{\sigma^h(l)}(C_{\sigma^h(i)}) < y_s\}$ denote the set of the candidate jobs at time $C_{\sigma^h(i)}$.

IF $E(i) = \emptyset$, THEN Return $\sigma_{s+1} = \emptyset$.

ELSE Find the job with the largest release date in $E(\sigma^h(i))$, say $J_{\sigma^h(e)}$. Let $J_{\sigma^h(e)}$ be scheduled

at the i-th position instead of $J_{\sigma^h(i)}$. Set $J_x = J_{\sigma^h(i)}$. For $q = i-1, i-2, \ldots, e+1$ (this ordering is used crucially), let $J_{\sigma^h(q)}$ be the job in $\{J_{\sigma^h(q)}, J_x\}$ with the larger release date, and let J_x be the other job. Finally, let J_x be scheduled at the e-th position.

Let $\sigma^{h+1} = \sigma^h$ and then set $h = h+1$. Go to Step 2.

We omit the proof of Lemma 3.1 because it is very similar to that of Lemma 2.2.

Lemma 3.1. *Let $\sigma^h = (J_{\sigma^h(1)}, J_{\sigma^h(2)}, \cdots, J_{\sigma^h(n)})$ be the schedule obtained at iteration h ($h = 0, 1, \ldots$) of Procedure $A_2(y_s)$. Let $\sigma = (J_{\sigma(1)}, J_{\sigma(2)}, \cdots, J_{\sigma(n)})$ be any schedule in $\Pi(\mathcal{J}, f^*, y_s)$. Then for $i = 1, 2, \ldots, n$, we have the following: (1) $r_{\sigma^h(i)} \geq \max\{r_j | j \in E(\sigma^h(i))\}$; (2) $S_{\sigma^h(i)} \leq S_{\sigma(i)}$ (and thus $C_{\sigma^h(i)} \leq C_{\sigma(i)}$); (3) $C_{\sigma^h(i)} \leq C_{\sigma^{h+1}(i)}$.*

We get the following:

Lemma 3.2. *Let σ^{last} (i.e., σ_{s+1}) be the schedule obtained at the last iteration of Procedure $A_2(y_s)$. If $\sigma^{last} = \emptyset$, then $\Pi(\mathcal{J}, f^*, y_s) = \emptyset$; otherwise σ^{last} is a schedule which has the minimum total completion time and minimum makespan among all schedules in $\Pi(\mathcal{J}, f^*, y_s)$.*

Proof. If $\sigma^{last} = \emptyset$, then by Step 3 of Procedure $A_2(y_s)$, at the last iteration, there is a job $J_{\sigma^h(i)}$ such that $f_{\sigma^h(i)}(C_{\sigma^h(i)}) > f^*$ or $g_{\sigma^h(i)}(C_{\sigma^h(i)}) \geq y_s$ and $E(\sigma^h(i)) = \emptyset$, where $E(\sigma^h(i)) = \{l | 1 \leq l \leq i \wedge f_{\sigma^h(l)}(C_{\sigma^h(i)}) \leq f^* \wedge g_{\sigma^h(l)}(C_{\sigma^h(i)}) < y_s\}$ denotes the set of candidate jobs at time $C_{\sigma^h(i)}$. Therefore, the first i jobs in σ^{last-1} can only be scheduled at the first i positions in any schedule in $\Pi(\mathcal{J}, f^*, y_s)$, but none of them can be scheduled at the i-th position. This contradiction tells us that $\Pi(\mathcal{J}, f^*, y_s) = \emptyset$.

If $\sigma^{last} \neq \emptyset$, then by Lemma 3.1, σ^{last} is a schedule which has the minimum total completion time and minimum makespan among all schedules in $\Pi(\mathcal{J}, f^*, y_s)$.

□

Based on Lemma 3.2, we get the following:

Theorem 3.3. *Algorithm M_2 solves $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$ in $O(n^3)$ time.*

Moreover, by Lemma 3.2, the last schedule generated by Algorithm M_2 has the minimum total completion time and minimum makespan among the lexicographical optimal schedules for $P|r_j, p_j = p|Lex(f_{\max}, g_{\max})$.

4. Conclusions

In this paper we studied the bicriteria problem of scheduling equal-processing-time jobs with release dates on identical machines to minimize total completion time (and makespan) and maximum cost simultaneously, or to minimize the two general min-max criteria hierarchically. We presented $O(n^3)$-time algorithms for the two problems. For future research, for Pareto optimization it is interesting to consider more general min-sum objective functions instead of total the completion time, such as the total weighted completion time, in combination with a maximum cost or another min-sum objective function. For lexicographical optimization, we can try to extend the results in [26, 27] to the case of unequal release dates.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

We thank the editor and reviewers for their helpful suggestions.

This work is supported by Natural Science Foundation of Shandong Province China (No. ZR2020MA030), and Shandong Soft Science Project (No. 2021RKY02040).

The authors express their gratitude to Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R61), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia.

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. H. Hoogeveen, Multicriteria scheduling, *Eur. J. Oper.*, **167** (2005), 592–623. https://doi.org/10.1016/j.ejor.2004.07.011
2. V. T'kindt, J. C. Billaut, *Multicriteria Scheduling: Theory, Models and Algorithms*, Berlin: Springer Verlag, 2006.
3. P Brucker, Scheduling Algorithms, *J Oper Res Soc*, **50** (1999), 774–774. https://doi.org/10.2307/3010332
4. A. Nagar, J. Haddock, S. Heragu, Multiple and bicriteria scheduling: A literature survey, *Eur. J. Oper.*, **81** (1995), 88–104. https://doi.org/10.1016/0022-4049(94)00117-2
5. M. Pfund, J. W. Fowler, J. N. D Gupta, A survey of algorithms for single and multi-objective unrelated parallel-machine deterministic scheduling problems, *Journal of the Chinese Institute of Industrial Engineers*, **21** (2004), 230–241. https://doi.org/10.1080/10170660409509404
6. D. M. Lei, Multi-objective production scheduling: a survey, *Int. J. Adv. Manuf.*, **43** (2009), 926–938. https://doi.org/10.1007/s00170-008-1770-4
7. F. S. Erenay, I. Sabuncuoglu, A. Toptal, M. K. Tiwari, New solution methods for single machine bicriteria scheduling problem: Minimization of average flowtime and number of tardy jobs, *Eur. J. Oper.*, **201** (2010), 89–98. https://doi.org/10.1057/fsp.2010.5
8. Y. K. Lin, J. W. Fowler, M. E. Pfund, Multiple-objective heuristics for scheduling unrelated parallel machines, *Eur. J. Oper.*, **227** (2013), 239–253. https://doi.org/10.1016/j.ejor.2012.10.008
9. A. J. Ruiz-Torres, J. H. Ablanedo-Rosas, S. Mukhopadhyay, G. Paletta, Scheduling workers: A multi-criteria model considering their satisfaction, *Comput Ind Eng*, **128** (2019), 747–754. https://doi.org/10.1016/j.cie.2018.12.070

10. J. C. Yepes-Borrero, F. Perea, R. Ruiz, F. Villa, Bi-objective parallel machine scheduling with additional resources during setups, *Eur. J. Oper.*, **292** (2021), 443–455. https://doi.org/10.1016/j.ejor.2020.10.052
11. J. Mar-Ortiz, A. J. Ruiz Torres, B. Adenso-Díaz, Scheduling in parallel machines with two objectives: analysis of factors that influence the pareto frontier, *Oper. Res.*, **22** (2022), 4585–4605. https://doi.org/10.1007/s12351-021-00684-9
12. J. ND Gupta, A. J. Ruiz-Torres, Minimizing makespan subject to minimum total flow-time on identical parallel machines, *Eur. J. Oper.*, **125** (2000), 370–380. https://doi.org/10.1016/S0377-2217(99)00386-0
13. C. H. Lin, C. J. Liao, Makespan minimization subject to flowtime optimality on identical parallel machines, *Comput. Oper. Res.*, **31** (2004), 1655–1666. https://doi.org/10.1016/S0305-0548(03)00113-8
14. L. H. Su, Minimizing earliness and tardiness subject to total completion time in an identical parallel machine system, *Comput. Oper. Res.*, **36** (2009), 461–471. https://doi.org/10.1016/j.cor.2007.09.013
15. J. L. Bruno, E. G. Coffman Jr, R. Sethi, Algorithms for minimizing mean flow time, *Proceedings of the IFIP Congress*, **74** (1974), 504–510.
16. J. ND Gupta, A. J. Ruiz-Torres, S. Webster, Minimizing maximum tardiness and number of tardy jobs on parallel machines subject to minimum flow-time, *J. Oper. Res. Soc.* , **54** (2003), 1263–1274. https://doi.org/10.1057/palgrave.jors.2601638
17. E. L. Lawler, J. K. Lenstra, A. H. G. R. Kan, D. B. Shmoys, Sequencing and scheduling: Algorithms and complexity, *Handbooks in operations research and management science*, **4** (1993), 445–522.
18. J. K. Lenstra, A. H. G. R. Kan, P. Brucker, Complexity of machine scheduling problems, *Annals of discrete mathematics*, **1** (1977), 343–362.
19. S. A. Kravchenko, F. Werner, Scheduling jobs with equal processing times, *Proceedings of the 13th IFAC Symposium on Information Control Problems in Manufacturing*, **42** (2009), 1262–1267. https://doi.org/10.3182/20090603-3-RU-2001.0042
20. S. A. Kravchenko, F. Werner, Parallel machine problems with equal processing times: a survey, *J. Sched.*, **14** (2011), 435–444. https://doi.org/10.1007/s10951-011-0231-3
21. P. Brucker, N. V. Shakhlevich, Necessary and sufficient optimality conditions for scheduling unit time jobs on identical parallel machines, *J. Sched.*, **19** (2016), 659–685. https://doi.org/10.1007/s10951-016-0471-3
22. 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
23. N. Vakhania, A better algorithm for sequencing with release and delivery times on identical machines, *J. Algorithm*, **48** (2003), 273–293. https://doi.org/10.1016/S0196-6774(03)00072-5
24. N. Vakhania, F. Werner, A polynomial algorithm for sequencing jobs with release and delivery times on uniform machines, 2021010142, [Preprint], (2021) [cited 2023 May 23]. Available from: https://www.preprints.org/manuscript/202101.0142/v2

25. A. Tuzikov, M. Makhaniok, R. Männer, Bicriterion scheduling of identical processing time jobs by uniform processors, *Comput. Oper. Res.*, **25** (1998), 31–35. https://doi.org/10.1016/S0305-0548(98)80005-1

26. S. C. Sarin, D. Prakash, Equal processing time bicriteria scheduling on parallel machines, *J Comb Optim*, **8** (2004), 227–240.

27. Q. L. Zhao, J. J. Yuan, Bicriteria scheduling of equal length jobs on uniform parallel machines, *J Comb Optim*, **39** (2020), 637–661. https://doi.org/10.1007/s10878-019-00507-w

28. B. Simons, Multiprocessor scheduling of unit-time jobs with arbitrary release times and deadlines, *SIAM J. Comput.*, **12** (1983), 294–299. https://doi.org/10.1137/0212018

29. B. B. Simons, M. K. Warmuth, A fast algorithm for multiprocessor scheduling of unit-length jobs, *SIAM J. Comput.*, **18** (1989), 690–710. https://doi.org/10.1137/0218048

30. C. Dürr, M. Hurand, Finding total unimodularity in optimization problems solved by linear programs, *Algorithmica*, **59** (2011), 256–268. https://doi.org/10.1007/s00453-009-9310-7

31. A. López-Ortiz, C. G. Quimper, A fast algorithm for multi-machine scheduling problems with jobs of equal processing times, *Symposium on Theoretical Aspects of Computer Science*, **9** (2011), 380–391.

32. H. Fahimi, C. G. Quimper, Variants of multi-resource scheduling problems with equal processing times, In *Combinatorial Optimization and Applications: 9th International Conference*, Houston: Springer International Publishing, 2015.

33. D. Elvikis, V.Tkindt, Two-agent scheduling on uniform parallel machines with min-max criteria, *Ann. Oper. Res.*, **213** (2014), 79–94.

AIMS Press