IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 15 May 2023, accepted 18 June 2023, date of publication 26 June 2023, date of current version 30 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3289288

RESEARCH ARTICLE

Sensitivity Analysis for the Single-Machine Preemptive Scheduling Problem of Minimizing Flow Time

XIAOXI LI[1] AND SHANLIN LI[2]

[1]Department of Arthroscopic Surgery, Shanghai Sixth People's Hospital, Shanghai Jiao Tong University School of Medicine (SJTUSM), Shanghai 200233, China
[2]School of Electronics and Information Engineering, Taizhou University, Taizhou, Zhejiang 317000, China

Corresponding author: Shanlin Li (lishanlin56@hotmail.com)

ABSTRACT Sensitivity analysis for the single-machine preemptive scheduling problem of minimizing flow time is discussed herein. Note that an optimal solution of the scheduling problem contains not just a combinatorial structure but also a temporal structure. The combinatorial structure specifies the sequence in which jobs or job pieces are processed. The temporal structure provides the start and completion time of every job or job piece. It is certain that a change in parameters causes a change in temporal structure. In this paper, we focus on the robust performance in the combinatorial structure, or the optimal sequence, after a single release data changes. By observing the effects of a parameter change on the optimal sequence, we found that the parameter change results only in the shifting forward (or backward) of some job blocks, and the amount of shifting forward (or backward) is equal to the amount of the first job piece size decrease (or increase) or the amount of the last job piece size increase (or decrease) for any shifting job block. Furthermore,the first job piece and the last job piece are, respectively, the first piece of the same job being processed and the last piece of the same job being processed. Therefore, to answer the question of focus in this paper, we first rewrite the optimal sequence in a job block form, and show some of its important properties. Then, a necessary and sufficient condition for the optimal sequence to remain optimal after a single release data changes is generated by four $O(nlogn)$ iterative time algorithms in the order of the job blocks.

INDEX TERMS Job block, optimization production scheduling, sensitivity analysis, single-machine deterministic sequencing.

I. INTRODUCTION

Sensitivity analysis consists of checking how the values of chosen parameters can vary so that the obtained solution remains optimal. Such analysis is an important part of optimization. It is a well-established topic in linear programming [1], [2]. However, in mixed integer programming and combinatorial optimization, it is a much less developed research area. Some general results on the complexity of post-optimality analysis of 0-1 programs are presented by [3]. Sensitivity analysis has been applied to several specific problems, such as the traveling salesperson problem [4], minimum spanning tree and shortest path problems [5], the generalized assignment problem [6], and strictly periodic tasks in multi-core real-time systems [7]. Some general results on stability analysis related to the situation in which several problem parameters may vary simultaneously are presented by [8] and [9].

There are several works on sensitivity analysis for scheduling problems. For ease of presentation, we use the widely adopted three-field classification scheme $\alpha|\beta|\gamma$ (see, e.g., [10]) to represent every scheduling problem we are concerned with in this paper, where α, β, and γ represent, respectively, the machine configuration, model restrictions and conditions, and the objective function to be minimized. In a single-machine environment, Mahadev et al. [11] examine the sensitivity of a schedule to earliness and tardiness penalties in the objective function for problems $1||\sum(\alpha_j E_j + \beta_j T_j)$ and $1||\max\{\alpha_j E_j, \beta_j T_j\}$. Moreover, Penz and Rapine [12] investigate the sensitivity of problem $1||\sum C_j$ with

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Asaduzzaman.

independent tasks that can be guaranteed not to exceed the square root of the magnitude of the perturbation. Chanas and Kasperski [13] consider sensitivity analysis for the problem $1|prec|\max\{w_jL_j\}$ by checking how the values of given parameters can vary so that a given optimal sequence remains optimal. Jiang et al. [14] conduct sensitivity analysis for several scheduling problems, i.e., $1||\sum C_j^2$, $1|prec|\sum w_jC_j$, and $1|r_j, p_j = 1|L_{\max}$, by giving limits to a parameter change such that the solution remains optimal. In parallel-machine environments, Tovey [15] considers the scheduling problem $Pm||C_{\max}$ on a changing number of identical processors. Kolen et al. [16] study a class of heuristic algorithms, known as list scheduling rules, for the problem $Pm||C_{\max}$. For several heuristics, they derive bounds on the number of possible assignments of jobs for machines. Penz and Rapine [12] investigate the sensitivity of the problem $Pm||\sum C_j$ with independent tasks that can be guaranteed not to exceed the square root of the magnitude of the perturbation. Jiang et al. [14] conduct sensitivity analysis for the flowshop problem $F2||C_{\max}$, and for the flowshop with a series of dominating machines $Fm||C_{\max}$, by giving the limits to a parameter change such that the solution remains optimal. Moreover, Maqsood et al. [17] present a detailed heuristic-based genetic algorithm parameter analysis for the jobshop scheduling problem $Jm||C_{\max}$ by finding the best possible parameter combination. Hall and Posner [19] are the first to attempt a systematic study on sensitivity analysis for scheduling problems. For three classes of scheduling problems, i.e., list scheduling problems, polynomial solvable problems, and NP-hard problems, they reveal the issues that occur in sensitivity analysis and introduce some basic methods to address them.

In this paper, a sensitivity analysis for the problem $1|r_j, pmtn|\sum C_j$ is discussed. That is to determine the condition of a parameter change such that the optimal sequence of the problem remains optimal. The sensitivity analysis enables the identification of critical jobs in a schedule, in the sense that such jobs have the highest impact on the quality of a schedule, and is of great importance for scheduling practitioners, which could help bridge the gap between productions scheduling theory and practice. Here, we employ a different research strategy than that used in the existing literature in the text. They typically give a condition and then prove that it is sufficient. We first examine the necessity conditions, and then determine the sufficient conditions that make the necessity conditions true. Specifically, we first focus on the effects of a parameter change on the optimal sequence. We find that the effect of the parameter change is closely related to the movement of job blocks. Thus, we rewrite the optimal sequence in job-block form, and its basic characteristics are investigated. This is conducive to our sensitivity analysis. Then, we provide a necessary and sufficient condition for the optimal sequence to remain optimal after a single release data changes. This condition is generated by four $O(nlogn)$ iterative time algorithms in the order of the job blocks, and the corresponding proofs are given.

The paper is organized as follows. In Section II, we define our notations and present the structure of the optimal schedule. Section III provides a sensitivity analysis for the problem. Finally, Section IV contains a conclusion and discussion of some possible extensions.

II. PRELIMINARIES

The single-machine preemptive scheduling problem of flow time minimization is a classical scheduling problem. There is a set of n jobs, i.e., $J = \{1, \cdots, n\}$, to be processed on a single machine. Each job $j \in J$ has a nonnegative release time r_j, and a positive processing time p_j. Processing may be interrupted and resumed later. In a given preemptive schedule, for each job $j \in J$, we denote C_j as its completion time. The objective is to find a preemptive schedule that minimizes $\sum_{j=1}^{n} C_j$. The problem is referred to as $1|r_j, pmtn|\sum C_j$.

Schrage [18] shows that an optimal schedule for $1|r_j, pmtn|\sum C_j$ can be found in $O(nlogn)$ time by processing the available job with the shortest remaining processing time at each point in time. This is called the shortest remaining processing time (SRPT) rule. Furthermore, this schedule contains no more than $n-1$ preemptions.

In this paper, we discuss the sensitivity analysis issue for $1|r_j, pmtn|\sum C_j$. This analysis considers the effects of a single release time change on the optimal sequence. For ease of presentation, we use the following definitions hereafter. The jobs are indexed such that $r_1 \le r_2 \le \cdots \le r_n$. Moreover, if $r_j = r_{j+1}$, then $p_j \le p_{j+1}$. To break ties when using the SPRT rule, we assume that the smallest index is selected for processing. Let π denote a job *sequence*. A *schedule* σ provides the start and completion time of every job or job piece. $C_j(\sigma)$ denotes the completion time of job j in schedule σ. The start time of job j in schedule σ is specified by $S_j(\sigma)$. Parameters and variables of the modified problem are identified with an added "′"; for example, r_j, S_j, and C_j become r_j', S_j' and C_j'. Let $\triangle$ denote the amount of a parameter change. When there is a single parameter change of the form $r_k' = r_k + \triangle$, σ^* denotes the optimal SRPT schedule before the parameter change occurs, and $\sigma^{*'}$ denotes the optimal SRPT schedule after the parameter change occurs. Similarly, π^* and $\pi^{*'}$ represent the optimal job sequences before and after there is a parameter change, respectively. When obvious, $C_j(\sigma^*)$, $C_j(\sigma^{*'})$, $S_j(\sigma^*)$, and $S_j(\sigma^{*'})$ are replaced by C_j^*, $C_j^{*'}$, S_j^* and $S_j^{*'}$, respectively. Here, we define some additional notations as follows. Let A be a subsequences of sequence π^*.

- r_A^*: the release time of the first job of A in σ^*.
- S_A^*: the start time of the first job of A in σ^*.
- C_A^*: the completion time of the last job of A in σ^*.
- $R_j(t)$: the remaining processing time of job j at time t in σ^*.
- $I(t_0, t_1)$: the total idle time in the interval $[t_0, t_1]$ in σ^*.

To answer the question in this paper, we present a example. In several cases where a parameter changes we will observe the effects of the parameter change on the optimal sequence so that we might be able to find a way to solve the question.

Example 1: Consider an instance of $1|r_j, pmtn|\sum C_j$ with six jobs $J = \{1, 2, 3, 4, 5, 6\}$ and the following parameters.

TABLE 1. Parameter data in Example 1.

job	1	2	3	4	5	6
p_i	1	6	1	4	4	1
r_i	1	5	6	11	15	17

Solving this instance using the SPRT rule yields the optimal sequence $\pi^* = (1, 2, 3, 2, 4, 5, 6, 5)$, and the optimal schedule σ^* :

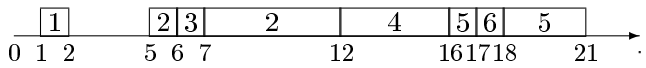

FIGURE 1. Optimal schedule (Example 1).

Let $r_2' = r_2 + \triangle = 5 - 2$ be a parameter change for the instance. We have the optimal sequence $\pi^{*'} = (1, 2, 3, 2, 4, 5, 6, 5) = \pi^*$, and the optimal schedule $\sigma^{*'}$:

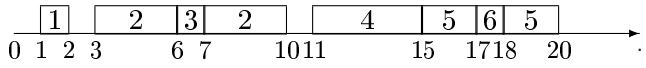

FIGURE 2. Optimal schedule (replace $r_2 = 5$ with $r_2 = 3$ in Example 1).

Let $r_2' = r_2 + \triangle = 5 - 4$. We have the optimal sequence $\pi^{*'} = (1, 2, 3, 2, 4, 5, 6, 5) = \pi^*$, and the optimal schedule $\sigma^{*'}$:

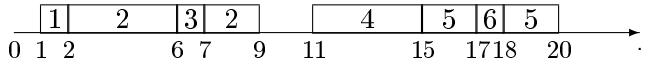

FIGURE 3. Optimal schedule (replace $r_2 = 5$ with $r_2 = 1$ in Example 1).

Let $r_2' = r_2 + \triangle = 5 - 5$. We have the optimal sequence $\pi^{*'} = (2, 1, 2, 3, 4, 5, 6, 5) \neq \pi^*$.

Let $r_2' = r_2 + \triangle = 5 + 0.5 < 6$. We have the optimal sequence $\pi^{*'} = (1, 2, 3, 2, 4, 5, 6, 5) = \pi^*$, and the optimal schedule $\sigma^{*'}$:

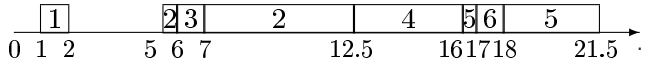

FIGURE 4. Optimal schedule (replace $r_2 = 5$ with $r_2 = 5.5$ in Example 1).

Let $r_2' = r_2 + \triangle = 5 + 2 \geq 6$. We have the optimal sequence $\pi^{*'} = (1, 3, 2, 4, 6, 5) \neq \pi^*$.

From observing the instances above, we made some interesting observations under $\pi^{*'} = \pi^*$.

(1) The start time and completion time of some jobs that are located in front of the parameter variable or between two pieces of the same job are identical in σ^*, such as job 1, job 3, and job 6 in Example 1.

(2) The parameter change results only in the shifting forward (or backward) of a job block $j^1 \Phi j^2$, such as job block (2, 3, 2) and job block (5, 6, 5) of π^* in Example 1; moreover, the amount of shifting forward (or backward) is equal to the decrease (or increase) in the size of job piece j^1 or the increase (or decrease) in the size of job piece j^2, where job piece j^1 is the first piece of job j, job piece j^2 the last piece of job j, and Φ the subsequences of sequence π^* located between j^1 and j^2.

(3) The amount of movement of different job blocks and the amount of parameter change may be different.

These observations suggest that it is more convenient and intuitive to rewrite π^* as a block form for the problem we are studying. The invariants shown in (1) should be some important constraints. (2) suggests that we should find the conditions of maximum shift of each job block. (3) suggests that the solution process should adopt a recursive form, because the amount of movement of a job block depends not only on the start time and release time of its first job, but also on the completion time of the previous job block. Now we rewrite the optimal sequence π^* in job block form, and show some of its important properties.

For some $k \in J$, let $r_k' = r_k + \triangle$ be a parameter change for $1|r_j, pmtn|\sum C_j$. To determine the limits to the parameter change such that optimal sequence π^* remains optimal, there are two cases to consider:

(i) job k does not preempt any job in σ^*.

(ii) job k preempts some job, say i, at r_k in σ^*.

In case (i), we rewrite optimal sequence π^* in job block form as follows:

$$\pi^* = (A_0, B_1, B_2, \cdots, B_m),$$

where $B_t = (j_t^1, \Phi_t, j_t^2)$ for $t = 1, 2, \cdots, m$; j_1^1 is the first piece of job $j_1 = k$ and j_1^2 the last piece of job $j_1 = k$ in σ^*; j_t^1 is the first piece of job j_t that is processed after $C^*_{j_{t-1}}$; j_t^2 the last piece of job j_t in σ^* for $t = 2, \cdots, m$; A_0 and $\Phi_1, \cdots, \Phi_m$ are all subsequences of sequence π^*, and can be empty. In Example 1, $m = 3$, $A_0 = (1)$, $B_1 = (2, 3, 2)$, $\Phi_1 = (3)$, $B_2 = (4)$, $\Phi_2 = \emptyset$, $B_3 = (5, 6, 5)$, $\Phi_3 = (6)$, and $\pi^* = (1, 2, 3, 2, 4, 5, 6, 5) = (A_0, B_1, B_2, B_3)$.

In case (ii), we rewrite the optimal sequence π^* in job block form as follows.

$$\pi^* = (A_0, i^1, B_1, B_2, \cdots, B_m, i^2, A_1),$$

where i^1 is the last piece of job i that is processed before r_k in σ^*; i^2 is the first piece of job i that is processed after r_k in σ^*; $B_t = (j_t^1, \Phi_t, j_t^2)$ for $t = 1, 2, \cdots, m$; j_1^1 is the first piece of job k in σ^*; j_1^2 is the last piece of job k in σ^*; j_t^1 is the first piece of job j_t in σ^* for $t = 2, \cdots, m$; j_t^2 is the last piece of job j_t in σ^* for $t = 2, \cdots, m$; and A_0, A_1, and $\Phi_1, \cdots, \Phi_m$ are the subsequences of sequence π^*, and can be empty. Now we give a preliminary result that shows the structural properties of the optimal schedules σ^* and $\sigma^{*'}$ in case (i) and case (ii), respectively. This is key for generating a correct condition such that the optimal sequence undergoes no change.

Lemma 1: For some $k \in J$, let $r_k' = r_k + \triangle$ be a parameter change for $1|r_j, pmtn|\sum C_j$, and job k does not preempt any

job in σ^*. Suppose that $\pi^{*'} = \pi^*$. Then all the following conditions hold.

(a) There is no idle time between the jobs or/and the job pieces of B_t, for $t = 1, 2, \cdots, m$, in $\sigma^{*'}$ and σ^*, respectively.

(b) For $t = 1, 2, \cdots, m$, the release time and completion time of every job in $\Phi_t/\{j_t\}$ must be on closed interval $[S^*_{\Phi_t}, C^*_{\Phi_t}]$.

(c) The start time and completion time of each job or job piece in A_0 and $\Phi_t/\{j_t\}$, for $t = 1, 2, \cdots, m$, are identical in $\sigma^{*'}$ and σ^*.

Proof. Proof of (a). Result (a) follows from job j_t being available at each point between the start time and completion time of B_t, for $t = 1, 2, \cdots, m$, in $\sigma^{*'}$ and in σ^*, respectively.

Proof of (b). For each job $j \in \Phi_t/\{j_t\}$, it is clear that $R_j(S^*_{\Phi_t}) < R_{j_t}(S^*_{\Phi_t})$. This implies that $C^*_j \in [S^*_{\Phi_t}, C^*_{\Phi_t}]$, and that $r_j \in [S^*_{\Phi_t}, C^*_{\Phi_t}]$. Job j will finish processing before $S^*_{\Phi_t}$ otherwise, which contradicts the optimality of π^*.

Proof of (c). Because $\pi^{*'} = \pi^*$, selections at each point on interval $[0, C^*_{A_0}]$ are not effected by the change of parameter r_k. Thus, result (c) holds for each job or job piece in A_0. For any $t \in \{1, 2, \cdots .m\}$, the first job of Φ_t preempts job j_t at r_{Φ_t} in $\sigma^{*'}$ and σ^*, therefore, the start time of the first job of Φ_t is does not change across $\sigma^{*'}$ and σ^*. Furthermore, since the processing order follows from $\pi^{*'} = \pi^*$, there is no idle time as follows from (a), and the size of each job is as follows from (b), the start and completion times of each job or job piece in Φ_t are identical in $\sigma^{*'}$ and σ^*.

Lemma 2: For some $k \in J$, let $r'_k = r_k + \triangle$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Job k preempts some job, say i, at r_k in σ^*. Suppose that $\pi' = \pi^*$. Then, all of the following conditions hold.

(a) There is no idle time between jobs or/and job pieces in interval $[S^*_{i^1}, C^*_{i^2}]$ in $\sigma^{*'}$ and σ^*, respectively.

(b) For $t = 1, 2, \cdots, m$, the release time and completion time of every job in $\Phi_t/\{j_t\}$ must be in closed interval $[r^*_{\Phi_t}, C^*_{\Phi_t}]$.

(c) The start time and completion time of every job or job piece in A_0, A_1, and $\Phi_t/\{j_t\}$ for $t = 1, 2, \cdots, m$ are identical in $\sigma^{*'}$ and σ^*.

Proof. Similar to that of Lemma 1.

III. SENSITIVITY ANALYSIS

In this section, we give a description of the correct condition, together with the corresponding proofs and analysis.

In Case (i), job k does not preempt any job in σ^*.

Before giving the result, we first briefly describe the condition-forming process. We adopt an iterative method according to the job blocks to generate the condition.

In the case $\triangle \le 0$, the parameter change possibly results in a gap between two adjacent job blocks that there is no a gap between them in σ^*. We use Algorithm 1, given later in this section, to compute the lower bound of $\triangle$ such that optimal sequence π^* remains optimal. When running the tth iteration, we need to compute three quantities: a_t, b_t, and c_t, where a_t denotes the maximum shifting backward quantity of job block B_t, b_t is the maximum shifting backward quantity of job block B_t that ensures the processing order of jobs in job block B_t undergo no change, and c_t is used to check whether a_t is valid. If $\min\{a_t, b_t, \delta_2(t-1)\} = a_t < \delta_2(t-1)$ and $c_t < r_{j_t}$, and a_t is valid. a_t is invalid otherwise. Then, we output two quantities, $\delta_1(t)$ and $\delta_2(t)$, where $\delta_1(t)$ describes the maximum lead time to start processing job k that ensures the job sequence before $c^*_{j_t} - \delta_2(t)$ undergoes no change, and $\delta_2(t)$ is the shifting backward quantity of job block B_t and the available shifting room provided for the next job block when $S'_k = S^*_k - \delta_1(t)$. After the iteration finishes, if $S^*_k - \delta_1 > C^*_{A_0}$, then $-\delta_1$ is the lower bound of $\triangle$. Otherwise, we need to further compute the maximum lead time of the release time for job k that ensures the job sequence before job k undergoes no change; this value then becomes the lower bound of $\triangle$.

In the case $\triangle \ge 0$, the parameter change results in the gap between $C^*_{A_0}$ and S^*_k increasing while gaps between two adjacent shiftable job blocks disappear. We use Algorithm 2 to compute the upper bound of $\triangle$ such that the optimal sequence π^* remains optimal. When running the tth iteration, we need to compute two quantities, i.e., a_t and b_t, where a_t denotes the maximum shifting forward quantity of job block B_t that ensures that the processing order of jobs j_t and j_{t+1} undergoes no change, and b_t is the maximum shifting forward quantity of job block B_t that prevents job piece j^1_t from disappearing. We also compute c_1, which prevents some job from appearing in the new gap between $C^*_{A_0}$ and S^*_k. We output the quantity $\delta(t)$ that describes the maximum delay time to start processing job k that ensures the job sequence before $\min\{c^*_{j_t} + \delta(t), r_{\Phi_{t+1}}\}$ undergoes no change.

Theorem 1: For some $k \in J$, let $r'_k = r_k + \triangle$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Assume that job k does not preempt any job in σ^*. Then, π^* is optimal for the modified scheduling problem if and only if $\triangle^1 \le \triangle \le \triangle^2$, where $\triangle^1$ and $\triangle^2$ are obtained by following Algorithm 1 and Algorithm 2, respectively. The specifications of strict or weak inequalities in $\triangle^1 \le \triangle \le \triangle^2$ will be pointed out in the following algorithms. The computational complexity of algorithms 1 and 2 is $O(nlogn)$.

Algorithm 1: In following statement of the algorithm, we prescribe that $C^*_{A_0} = 0$ if $A_0 = \emptyset$; $b_t = +\infty$ if $\Phi_t = \emptyset$; and $c_m = +\infty$.

1. Let $\delta_1(1) = \min\{a_1, b_1\}$, $\delta_2(1) = \min\{a_1, b_1\}$, and $t = 2$, where $a_1 = S^*_{j_1} - C^*_{A_0}$, and $b_1 = \min\{R_{j_1}(r_j) - p_j | j \in \Phi_1, j \ne j_1\}$.
2. If $r_{j_t} > C^*_{j_{t-1}}$ or $t = m+1$, then let $\delta = \delta_1(t-1)$, and go to step 5.
3. Compute $a_t = S^*_{j_t} - r_{j_t}$, $b_t = \min\{R_{j_t}(r_j) - p_j | j \in \Phi_t, j \ne j_t\}$, $c_t = \min\{r_{j_{t+1}}, \cdots, r_{j_m}\}$ and $d_t = \min\{a_t, b_t, \delta_2(t-1)\}$.
4. If $d_t = b_t$ or $d_t = a_t < \delta_2(t-1)$, and $c_t < r_{j_t}$, then let $\delta_1(t) = d_t$, $\delta_2(t) = d_t$, and $t = t+1$, and return to step 2. Otherwise, let $\delta_1(t) = \delta_1(t-1)$, $\delta_2(t) = d_t$, and $t = t+1$, and return to step 2.
5. If $S^*_k - \delta = C^*_{A_0}$ and $\delta \ne b_t$ for $t = 1, 2, \ldots, m$, then output $\triangle^1 = \max\{\min\{t | t \le C^*_{A_0}, I(t, C^*_{A_0}) =$

$0\}, \max\{t|R_j(t) \geq p_k \text{ for some } j \in A_0\ \}\} - r_k$, and $\triangle^1 \leq \triangle$. If $S_k^* - \delta > C_{A_0}^*$ and $\delta \neq b_t$ for $t = 1, 2, \ldots, m$, then output $\triangle^1 = -\delta$ and $\triangle^1 \leq \triangle$. If $S_k^* - \delta \geq C_{A_0}^*$ and δ equals some b_t, then output $\triangle^1 = -\delta$ and $\triangle^1 < \triangle$.

Algorithm 2: We use the below notation in Algorithm 2.

$$a_t = \begin{cases} r_{j_{t+1}} - C_{j_t}^* + p_{j_{t+1}} & \text{if } r_{j_{t+1}} > C_{j_t}^*,\ p_{j_{t+1}} < p_{j_t}; \\ p_{j_{t+1}} - R_{j_t}(r_{j_{t+1}}) & \text{if } r_{j_{t+1}} \leq C_{j_t}^*,\ p_{j_{t+1}} < p_{j_t}; \\ +\infty, & \text{otherwise.} \end{cases}$$

We note that $b_t = +\infty$ if $\Phi_t = \emptyset$; and $c_1 = +\infty$ if $m = 1$; $a_m = +\infty$.

1. Let $\delta(1) = \min\{a_1, b_1, c_1\}$, and $t = 2$, where $b_1 = r_{\Phi_1} - S_{j_1}^*$, and $c_1 = \min\{\max\{0, r_j - S_k^*\}|j \in \{j_2, \cdots, j_m\}\}$.
2. If $r_{j_t} \geq C_{j_{t-1}}^* + \delta(t-1)$ or $t = m+1$, then let $\delta = \delta(t-1)$, and go to step 5.
3. Compute a_t, $b_t = r_{\Phi_t} - S_{j_t}^*$.
4. Let $\delta(t) = \min\{I(r_k, r_t) + a_t, I(r_k, r_t) + b_t, \delta(t-1)\}$, and $t = t+1$, and then return to step 2.
5. If $\delta \neq I(r_k, r_t) + b_t$ for $t = 1, 2, \ldots, m$, then output $\triangle^2 = S_k^* - r_k + \delta$ and $\triangle \leq \triangle^2$. Otherwise, output $\triangle^2 = S_k^* - r_k + \delta$ and $\triangle < \triangle^2$.

Proof. ($\Leftarrow$). We will check the correctness of the condition step by step in time order. There are two cases to consider: (ia1) $\triangle \leq 0$, and (ib1) $\triangle \geq 0$.

Case (ia1) $\triangle \leq 0$. According to the output of Algorithm 1, there are three cases.

First, the output is $\triangle^1 = \max\{\min\{t|t \leq C_{A_0}^*, I(t, C_{A_0}^*) = 0\}, \max\{t|R_j(t) \geq p_k \text{ for some } j \in A_0\ \}\} - r_k$ and $\triangle^1 \leq \triangle$ on condition that $S_k^* - \delta = C_{A_0}^*$ and $\delta \neq b_t$. It is clear that $\pi' = \pi^*$ under $\triangle^1 = \triangle \leq 0$ implies $\pi' = \pi^*$ under $\triangle^1 \leq \triangle \leq 0$. We assume that $\triangle^1 = \triangle$, $r_k' = r_k^* + \triangle^1$. In interval $[0, r_k')$, the selections by the SRPT rule are not affected by the change of the parameter. Thus, $\sigma^{*'}$ and σ^* are identical in the interval. In interval $[r_k', C_{A_0}^*)$, the available jobs and the remaining processing time of each available job at each point in time in the interval are the same as those for original parameter data for jobs other than k. Since $r_k' \geq \min\{t|t \leq C_{A_0}^*, I(t, C_{A_0}^*) = 0\}$, there is no idle time in interval $[r_k', C_{A_0}^*)$ in σ^*. From $r_k' \geq \max\{t|R_j(t) \geq p_k \text{ for some } j \in A_0\}$, we have $p_k \geq R_j(t)$ for $j \in A_0$ and $t \in [r_k', C_{A_0}^*)$. Thus $\sigma^{*'}$ and σ^* are identical in the interval, which follows from the SRPT rule and our assumptions above. Noting that interval $(C_{A_0}^*, S_k^*)$ is idle in σ^*, we have job k processed in $[C_{A_0}^*, S_k^*)$ in $\sigma^{*'}$. In interval $[S_k^*, C_k^* - \delta)$, the available jobs and the remaining processing time of each available job at each point in time in the interval are the same as those for original parameter data for jobs other than k, and the remaining processing time of job k at each point in time in the interval is decreased by δ compared with those for original parameter data. The condition $\delta \leq \delta_1(1) < b_1 = \min\{R_{j_1}(r_j) - p_j|j \in \Phi_1, j \neq j_1\}$ ensures that $\sigma^{*'}$ and σ^* are identical in the interval. We see that the job sequence before $C_k^* - \delta$ in $\sigma^{*'}$ does not destroy π^*.

Now, we examine job sequence $(B_2, \cdots, B_m)$ in $\sigma^{*'}$. If $r_{j_2} > C_k^*$, we have $\pi' = \pi^*$, which follows from job j_2 starting at the same time r_{j_2} in both $\sigma^{*'}$ and σ^* and from the results obtained earlier. If $c_2 \geq r_{j_2}$, and noting that if $r_{j_2} \leq r_j \leq S_{j_2}^*$ for some $j \in \{j_3, \cdots, j_m\}$, then either $R_j(r_{j_2}) > p_{j_2}$ or $j_2 < j$ when $R_j(r_{j_2}) = p_{j_2}$, the interval $[C_k^* - \delta, r_{j_2}]$ is idle. Job j_2 is processed in interval $[r_{j_2}, S_{j_2}^*]$ in $\sigma^{*'}$ if $C_k^* - \delta < r_{j_2}$, but is processed in interval $[C_k^* - \delta, S_{j_2}^*]$ in $\sigma^{*'}$ otherwise. This means that the job sequence in $[C_k^* - \delta, S_{j_2}^*]$ in $\sigma^{*'}$ does not destroy π^*. When job block B_1 shifts δ backward, we note that the actual shifting backward quantity of job block B_2, say $\triangle_1$, is less than or equal to $\delta_2(2)$. Furthermore, the condition $\delta_2(2) < b_2 = \min\{R_{j_2}(r_j) - p_j|j \in \Phi_2, j \neq j_2\}$, along with $\triangle_1 \leq \delta_2(2)$ and the remaining processing time of job j_2 at $S_{j_2}^*$ decreases by $\triangle_1$, and ensures that $\sigma^{*'}$ and σ^* are identical in the interval $[S_{j_2}^*, C_{j_2}^* - \min\{\delta, \delta_2(2)\})$. We see that the job sequence before $C_{j_2}^* - \min\{\delta, \delta_2(2)\}$ in $\sigma^{*'}$ does not destroy π^*. If $c_2 < r_{j_2}$, by Algorithm 1, we have $\delta \leq \delta_1(2) = \delta_2(2)$. This means that there is no gap between job blocks B_1 and B_2 in $\sigma^{*'}$. Following a similar argument in case $c_2 \geq r_{j_2}$, the job sequence before $C_{j_2}^* - \min\{\delta, \delta_2(2)\}$ in $\sigma^{*'}$ does not destroy π^*. By applying a similar argument to sequence $(B_3, \cdots, B_m)$ in $\sigma^{*'}$, we conclude that $\pi' = \pi^*$.

The proofs for two other cases are similar to that of the first case.

Case (ib1) $\triangle \geq 0$. There are two cases for consideration according to the output of Algorithm 2.

First, the output is $\triangle^2 = S_k^* - r_k + \delta$ and $\triangle \leq \triangle^2$ on condition that $\delta \neq b_t$. Without loss of generality, we assume that $\triangle = \triangle^2 = S_k^* - r_k + \delta$. Clearly, $\sigma^{*'}$ and σ^* are identical in interval $[0, C_{A_0}^*]$. From $\delta \leq c_1 = \min\{\max\{0, r_j - S_k^*\}|j \in \{j_2, \cdots, j_m\}\}$, it follows that interval $(C_{A_0}^*, S_k^* + \delta)$ is idle in $\sigma^{*'}$. By $\delta < b_1 = r_{\Phi_1} - S_k^*$, we have $S_k^* + \delta < r_{\Phi_1}$. Thus, in interval $[S_k^* + \delta, C_k^*)$, the available jobs and remaining processing time of each available job at each point in time in the interval are the same as those for original parameter data for jobs other than k, and the remaining processing time of job k at each point in time in the interval is increased by δ compared with those for original parameter data. The condition $\delta \leq a_1$ ensures that $\sigma^{*'}$ and σ^* are identical in the interval. For interval $[C_k^*, C_k^* + \delta]$, from $\delta \leq \delta(2) \leq I(r_k, r_{j_2}) + b_2 = I(r_k, r_t) + r_{\Phi_2} - S_{j_2}^*$, it follows that $C_k^* + \delta < r_{\Phi_2}$. If $r_{j_2} \geq C_k^* + \delta$, and since $[C_k^*, C_k^* + \delta]$ is idle in σ^*, job k is processed in the interval. We conclude that the $\pi' = \pi^*$ that follows from job j_2 starts at the same time r_{j_2} in $\sigma^{*'}$ and σ^*, and from the results obtained earlier. If $C_k^* < r_{j_2} < C_k^* + \delta$ or $r_{j_2} \leq C_k^*$, the condition $\delta \leq I(r_k, r_{j_2}) + a_1$, along with the remaining processing time of job k at C_k^* increased by δ, ensure that job k is processed in interval $[C_k^*, C_k^* + \delta]$. We see that the job sequence before $C_k^* + \delta$ in $\sigma^{*'}$ does not destroy π^*. By applying a similar argument to the sequence $(B_2, \cdots, B_m)$ in $\sigma^{*'}$, we conclude that $\pi' = \pi^*$.

The proof of the second case is similar to that of the first case.

($\Rightarrow$). There are also two cases for consideration: (ia2) $\triangle \leq 0$ and (ib2) $\triangle \geq 0$.

Case (ia2) $\triangle \leq 0$. We show this by contradiction. Suppose that $\triangle$ is different from the output by Algorithm 1. Consider the following three situations. First, the output is $\triangle^1 = \max\{\min\{t|t \leq C^*_{A_0}, I(t, C^*_{A_0}) = 0\}, \max\{t|R_j(t) \geq p_k \text{ for some } j \in A_0\}\} - r_k$ and $\triangle^1 \leq \triangle$. We suppose that $\triangle < \triangle^1$. By the SRPT rule, job k either starts before $\min\{t|t \leq C^*_{A_0}, I(t, C^*_{A_0}) = 0\}$ or preempts some job in A_0 in $\sigma^{*'}$. This contradicts that $\pi' = \pi^*$. Second, the output is $\triangle^1 = -\delta$ and $\triangle^1 \leq \triangle$ on condition that $S^*_k - \delta > C^*_{A_0}$ and $\delta \neq b_t$. We suppose that $\triangle^1 > \triangle$. When $\delta \neq b_t$, δ equals to some a_t. Since $S^*_k - \delta > C^*_{A_0}$, $\delta \neq \delta_1(1)$. Let t_0 be the lowest index such that $\delta = \delta_1(t)$. Due to $\delta \neq b_t$ and $\delta_1(t_0 - 1) > \delta_1(t_0)$, $\delta_1(t_0) = a_{t_0} < \delta_2(t_0 - 1)$ and $c_{t_0} < r_{j_{t_0}}$. It is obvious that $\pi' \neq \pi^*$ under $\triangle = g < 0$ implies $\pi' \neq \pi^*$ under $\triangle \leq g < 0$ for some real number g. Now we assume that $-\delta_2(t_0 - 1) < \triangle < -\delta$, and apply the SRPT rule to the modified data $r'_k = r_k + \triangle$. From $\delta_2(1) \geq \cdots \geq \delta_2(t_0 - 1) > -\triangle$ and $\delta_1(1) \geq \cdots \geq \delta_1(t_0 - 1) \geq \delta_2(t_0 - 1) > -\triangle$, there is no idle time, and the job sequence by the SRPT rule does not destroy π^* in interval $[r'_k, C^*_{t_0-1}+\triangle]$; moreover, job j_{t_0-1} completes at time $C^*_{t_0-1} + \triangle$. Following from $-\triangle > \delta = a_t = S^*_{j_{t_0}} - r_{j_{t_0}}$ and $c_{t_0} < r_{j_{t_0}}$, there is some job j_t with $t > t_0$ that starts in interval $(C^*_{t_0-1}+\triangle, r_{j_{t_0}})$. This contradicts $\pi' = \pi^*$. Finally, the output is $\triangle^1 = -\delta$ and $\triangle^1 < \triangle$ on condition that $S^*_k - \delta \geq C^*_{A_0}$, and δ equals some b_t. We suppose that $\triangle = \triangle^1$. Let t_0 be the lowest index such that $\delta = b_{t_0}$ and j_0 is the first job in Φ_{t_0}, and such that $b_{t_0} = R_{j_{t_0}}(r_{j_0}) - p_{j_0}$. Job j_0 will not start at $S^*_{j_0}$, which follows from $r_{j_0} > r_{j_{t_0}}$. This contradicts that $\pi' = \pi^*$.

Case (ib2) $\triangle \geq 0$. We show this by contradiction. Suppose that $\triangle$ is different from the output by Algorithm 2. There are two cases to consider. First, the output is $\triangle^2 = S^*_k - r_k + \delta$ and $\triangle \leq \triangle^2$ on condition that $\delta \neq b_t$. Let t_0 be the lowest index such that $\delta = \delta(t)$. If $t_0 = 1$ and $\delta(1) = c_1$, we assume that $\triangle > S^*_k - r_k + \delta$. From $S^*_k \leq S^*_k + c_1 < \triangle + r_k = r'_k$, it follows that there is a piece of some job j_t with $t > 1$ that starts before r'_k. This contradicts that $\pi^{*'} = \pi^*$. If $t_0 = 1$ and $\delta(1) = a_1 < c_1$, we assume that $S^*_k - r_k + a_1 < \triangle < \min\{S^*_k - r_k + b_1, S^*_k - r_k + c_1\}$. From the SRPT rule, job j_2 starts before the completion of job k. This contradicts that $\pi^{*'} = \pi^*$. If $t_0 > 1$, we assume that $S^*_k - r_k + a_{t_0} < \triangle < \min\{S^*_k - r_k + b_{t_0}, \delta(t_0 - 1)\}$. Similarly, we can find a contradiction. Second, the output is $\triangle^2 = S^*_k - r_k + \delta$ and $\triangle < \triangle^2$ on condition that δ equals some b_t. We only assume that $\triangle = S^*_k - r_k + \delta$, and then can find a contradiction.

We now look at the computational complexity of Algorithms 1 and 2. Because σ^* contains no more than $n - 1$ preemptions, π^* is composed of at most $2n - 1$ jobs or job pieces. And then it takes $O(nlogn)$ time to sort the jobs or job pieces in π^* (or a subsequence of π^*) according to their corresponding values.

In Algorithm 1, it takes constant time to compute $a_1 = S^*_{j_1} - C^*_{A_0}$. It takes $O(nlogn)$ time to compute $b_1 = \min\{R_{j_1}(r_j) - p_j | j \in \Phi_1, j \neq j_1\}$, which is the time to sort the jobs or job pieces in $\Phi_1 \setminus j_1$ according to their corresponding values. Given a_1 and b_1, $\delta_1(1)$ and $\delta_2(1)$ takes constant time to compute. Thus, Step 1 takes $O(nlogn)$ time. In Step 5, it takes $O(n)$ time to verify $S^*_k - \delta = C^*_{A_0}$ and $\delta \neq b_t$. It takes $O(n)$ time to compute $\min\{t|t \leq C^*_{A_0}, I(t, C^*_{A_0}) = 0\}$ and $\max\{t|R_j(t) \geq p_k \text{ for some } j \in A_0\}$. Thus, Step 5 takes $O(n)$ time. Steps 2 to 4 is iterated $m - 1$ times. Inside the iteration loop, the most time-consuming calculation is for $b_t = \min\{R_{j_t}(r_j) - p_j | j \in \Phi_t, j \neq j_t\}$ and $c_t = \min\{r_{j_{t+1}}, \cdots, r_{j_m}\}$. Given b_t and c_t, the other variables takes constant time to compute. Since it takes $O(nlogn)$ time to compute $\min\{R_{j_t}(r_j) - p_j | j \in \cup_{t=2}^m \Phi_t \setminus \{j_t\}\}$, it takes $O(nlogn)$ time to compute $b_2, b_3, \ldots, b_m$. It takes $O(nlogn)$ time to compute $c_2, c_3, \ldots, c_m$ because it takes $O(nlogn)$ time to compute $\min\{r_{j_2}, r_{j_3}, \ldots, r_{j_m}\}$. In summary, it can be seen that the computational complexity of Algorithm 1 is $O(nlogn)$.

Similarly, we can show that the computational complexity of Algorithm 2 is $O(nlogn)$.

Case (ii) job k preempts some job, say i, at r_k in σ^*.

Before we present our main result for case (ii), we first briefly describe the condition-forming process. We adopt an iterative method according to the job blocks to generate the condition.

In case $\triangle \leq 0$, there is no gap between two adjacent job blocks in σ^*, and the parameter change does not result in a gap between them. All the shifting backward quantities of job blocks are the same. We use Algorithm 3 to compute the lower bound of $\triangle$ such that the optimal sequence π^* remains optimal. When running the tth iteration, we need to compute two quantities, i.e., a_t and b_t, where a_t denotes the maximum shifting backward quantity of job block B_t, b_t is the maximal shifting backward quantity of job block B_t that ensures the processing order of jobs in job block B_t undergoes no change, and we output quantity $\delta(t)$ that describes the maximum lead time to start processing job k that ensures the job sequence before $C^*_{j_t} - \delta(t)$ undergoes no change. We also compute the α value that prevents some job in A_1 from appearing before i^2.

In case $\triangle \geq 0$, there is no a gap between two adjacent job blocks in σ^*, and the parameter change does not result in a gap between them. All the shifting forward quantities of job blocks are the same. We use Algorithm 4 to compute the upper bound of $\triangle$ such that the optimal sequence π^* remains optimal. When running the tth iteration, we need to compute two quantities, i.e., a_t and b_t, where a_t denotes the maximal shifting forward quantity of job block B_t that ensures the processing order of job j_t and j_{t+1} undergoes no change; b_t is both the maximum shifting forward quantity of job block B_t that prevents job piece j^1_t from disappearing and the maximal shifting forward quantity of job block B_t that prevents job piece i^2 from disappearing. We also compute the c_1 value that prevents some job in $\{j_2, \cdots, j_m\}$ from appearing before j_1. We output $\delta(t)$, which describes the maximum delay time to start processing job k that ensures the job sequence before $\min\{C^*_{j_t} + \delta(t), r_{\Phi_{t+1}}\}$ undergoes no change.

Theorem 2: For some $k \in J$, let $r'_k = r_k + \triangle$ be a parameter change for $1|r_j, pmtn|\sum C_j$. Suppose that job k preempts some job, say i, at r_k in σ^*. Then, π^* is optimal for the modified scheduling problem if and only if $\triangle^3 \leq \triangle \leq \triangle^4$, where $\triangle^3$ and $\triangle^4$ are provided by Algorithm 3 and Algorithm 4, respectively. The specifications of strict or weak inequalities in $\triangle^3 \leq \triangle \leq \triangle^4$ will be pointed out in the following algorithms. The computational complexity of algorithms 3 and 4 is $O(nlogn)$.

Algorithm 3: In the following algorithm description, we use the notation

$$\alpha = \begin{cases} \min\{p_j - R_i(r_j) | j \in A_1\} & \text{if } r_j < S^*_{i^2}, \\ & \text{and } p_j < R_i(S^*_{i^1}); \\ +\infty, & \text{otherwise.} \end{cases}$$

Moreover, $b_t = +\infty$ if $\Phi_t = \emptyset$.

1. Let $\delta(1) = \min\{a_1, b_1\}$, and $t = 2$, where $a_1 = S^*_{j_1} - S^*_{i^1}$ and $b_1 = \min\{R_{j_1}(r_j) - p_j | j \in \Phi_1, j \neq j_1\}$.
2. If $t = m + 1$, then go to step 5.
3. Compute $a_t = S^*_{j_t} - r_{j_t}$, and $b_t = \min\{R_{j_t}(r_j) - p_j | j \in \Phi_t, j \neq j_t\}$.
4. Let $\delta(t) = \min\{a_t, b_t, \delta(t-1)\}$, and $t = t + 1$, and then return to step 2.
5. Output $\triangle^3 = -\min\{\delta(m), \alpha\}$ and $\triangle^3 \leq \triangle$ if $\triangle^3 \neq b_t$ for $t = 1, 2, \ldots, m$, and $\triangle^3 \neq a_1$. Otherwise, output $\triangle^3 = -\min\{\delta(m), \alpha\}$ and $\triangle^3 < \triangle$.

Algorithm 4: In the following algorithm description, we use the notation for $t = 1, 2, \ldots, m-1$,

$$a_t = \begin{cases} p_{j_{t+1}} - R_{j_t}(r_{j_{t+1}}) & \text{if } p_{j_{t+1}} < p_{j_t}; \\ +\infty, & \text{otherwise.} \end{cases}$$

Furthermore, $r_{\Phi_t} - S^*_{j_t} = +\infty$ if $\Phi_t = \emptyset$; $c_1 = +\infty$ if $m = 1$; and $a_m = C^*_{i^2} - S^*_{i^2}$.

1. Let $\delta(1) = \min\{a_1, b_1, c_1\}$, and $t = 2$, where $b_1 = \min\{r_{\Phi_1} - S^*_{j_1}, R_i(r_{j_1}) - p_{j_1}\}$, and $c_1 = \min\{r_j - S^*_{j_1} | j \in \{j_2, \cdots, j_m\}\}$.
2. If $t = m + 1$, then let $\delta = \delta(t-1)$, and go to step 5.
3. Compute $a_t, b_t = \min\{r_{\Phi_t} - S^*_{j_t}, R_i(j_t) - p_{j_t}\}$.
4. Let $\delta(t) = \min\{a_t, b_t, \delta(t-1)\}$, and $t = t + 1$, and then return to step 2.
5. If $\delta \neq b_t$ for $t = 1, 2, \ldots, m$, then output $\triangle^4 = \delta$ and $\triangle \leq \triangle^4$. Otherwise, output $\triangle^4 = \delta$ and $\triangle < \triangle^4$.

Proof. ($\Leftarrow$). We will check the correctness of the condition step by step in time order. There are two cases to consider: (iia1) $\triangle \leq 0$, and (iib1) $\triangle \geq 0$.

Case (iia1) $\triangle \leq 0$. According to the output of Algorithm 3, there are two cases to consider.

First, the output is $\triangle^3 = -\min\{\delta(m), \alpha\}$ and $\triangle^3 \leq \triangle$ on condition that $\triangle^3 \neq b_t$ for $t = 1, 2, \ldots, m$, and $\triangle^3 \neq a_1$. Without loss of generality, we assume that $\triangle = \triangle^3 = -\min\{\delta(m), \alpha\}$. Clearly, $\sigma^{*'}$ and σ^* are identical in interval $[0, r'_k)$. Since $-\triangle < a_1 = S^*_k - S^*_{i^1}$, and $R_i(S^*_k) > p_k$, and job i being processed in interval $[r'_k, S^*_k)$ in σ^*, job k is processed in interval $[r'_k, S^*_k)$ in $\sigma^{*'}$, and job i is processed in interval $[S^*_{i^1}, r'_k)$ in $\sigma^{*'}$. In interval $[S^*_k, C^*_k + \triangle)$, the available jobs and the remaining processing time of each available job at each point in time in the interval are the same as those for original parameter data other than the remaining processing time of job k decreased by $-\triangle$ and the remaining processing time of job i increased by $-\triangle$. The condition $-\triangle \leq \delta(1) < b_1$ ensures that $\sigma^{*'}$ and σ^* are identical in the interval. Note that if $r_j \leq S^*_{j_2}$ with $j \in \{j_3, \cdots, j_m\} \cup A_1$, then either $R_j(S^*_{j_2}) > p_{j_2}$ or $j_2 < j$ when $R_j(S^*_{j_2}) = p_{j_2}$. Then, job j_2 is processed in interval $[C^*_k + \triangle, S^*_{j_2}]$ in $\sigma^{*'}$, which follows from condition $-\triangle \leq \delta(2) \leq a_2$. Similarly, the job sequence before $C^*_{j_m} + \triangle$ in $\sigma^{*'}$ does not destroy π^*. From condition $-\triangle \leq \alpha$, it follows that job i is processed in interval $[C^*_m + \triangle, S^*_{i^2}]$ in $\sigma^{*'}$. Because the sets of the unscheduled jobs are the same, and each job in the set has the same remaining processing time at time $S^*_{i^2}$ and the same release time in $\sigma^{*'}$ and σ^*, respectively, $\sigma^{*'}$ and σ^* are identical in interval $[S^*_{i^2}, C^*_{A_1}]$. This and the results obtained earlier imply that $\pi^{*'} = \pi^*$.

The proof of the second case is similar to that of the first case.

Case (iib1) $\triangle \geq 0$. According to the output of Algorithm 4, there are two cases to consider.

First, the output is $\triangle^4 = \delta$ and $\triangle \leq \triangle^4$ on condition that $\delta \neq b_t$ for $t = 1, 2, \ldots, m$. Without loss of generality, we assume that $\triangle = \triangle^4 = \delta$. Clearly, $\sigma^{*'}$ and σ^* are identical in interval $[0, S^*_k]$. Note that $p_j > R_i(S^*_{i^1})$ if $r_j \leq r_i$ with $j \in A_1$ and $p_j \geq R_i(S^*_k)$ if $S^*_{i^1} < r_j \leq S^*_k + \delta$ with $j \in A_1$. Job i is processed in interval $[S^*_k, S^*_k + \triangle]$, which follows from the condition $\triangle \leq c_1 = \min\{r_j - S^*_{j_1} | j \in \{j_2, \cdots, j_m\}\}$. By $\triangle < b_1 \leq r_{\Phi_1} - S^*_k$, we have $S^*_k + \triangle < r_{\Phi_1}$. From condition $\triangle \leq \min\{a_1, R_i(r_{j_1}) - p_{j_1}\}$ and the fact that if $r_{j_t} \leq S^*_{j_2}$ with $t > 2$, then $p_{j_t} \geq p_{j_2}$, and it follows that job k is processed in interval $[S^*_k + \delta, r_{\Phi_1})$. In interval $[r_{\Phi_1}, C^*_k)$ the available jobs and the remaining processing time of each available job at each point in time in the interval are the same as those for original parameter data other than the remaining processing time of job k increased by $\triangle$ and the remaining processing time of job i decreased by $\triangle$. $\sigma^{*'}$ and σ^* are identical in interval $[r_{\Phi_1}, C^*_k)$, which follows from $\triangle < R_i(r_{j_1}) - p_{j_1} < R_i(r_j) - p_j$ for $j \in \Phi_1, j \neq j_1$. From conditions $\triangle < b_2 \leq r_{\Phi_2} - S^*_{j_2}$ and $\triangle \leq a_2$, $C^*_k + \triangle < r_{\Phi_2}$, and job k is processed in interval $[C^*_k, C^*_k + \triangle)$. Similarly, we can verify that the job sequence before $C^*_{j_m} + \triangle$ in $\sigma^{*'}$ does not destroy π^*. Since $\triangle < a_m = C^*_{i^2} - S^*_{i^2}$, we have $C^*_{j_m} + \triangle < C^*_{i^2}$. Then, since the sets of the unscheduled jobs are the same, and each job in the set has the same remaining processing time at time $C^*_{j_m} + \triangle$ and the same release time in $\sigma^{*'}$ and σ^*, respectively, $\sigma^{*'}$ and σ^* are identical in interval $[S^*_{i^2}, C^*_{A_1}]$. This and the results obtained earlier imply that $\pi^{*'} = \pi^*$.

The proof of the second case is similar to that of the first case.

($\Rightarrow$). We show this by contradiction. There are also two cases to consider here: case (iia2) $\triangle \leq 0$ and case (iib2) $\triangle \geq 0$.

Case (iia2) $\triangle \leq 0$. Suppose that $\triangle$ is different from the output of Algorithm 3. If $\triangle^3 = -\alpha > -\delta(m)$, we assume

that $-\delta(m) < \triangle < -\alpha$. If $\triangle^3 = -a_{t_0}$ and $\triangle^3 \neq b_t$ for $t = 1, 2, \ldots, m$, where $t_0 > 1$ is the lowest index such that $\triangle^3 = -a_t$, we assume that $-a_{t_0-1} < \triangle < -a_{t_0}$. If $\triangle^3 = -b_{t_0}$, where t_0 is the lowest index such that $\triangle^3 = -b_t$, we assume that $\triangle = -b_{t_0}$. If $\triangle^3 = -a_1$, we assume that $\triangle = -a_1$. Similar to the argument in case (ia2), each of the four assumptions above will result in a contradiction.

Case (iib2) $\triangle \geq 0$. Suppose that $\triangle$ is different from the output of Algorithm 4. If $\triangle^4 = c_1$, we assume that $c_1 < \triangle < c_1 + \epsilon$, where ϵ is an arbitrarily small positive number. If $\triangle^4 = a_{t_0}$ and $\triangle^4 \neq b_t$ for $t = 1, 2, \ldots, m$, where t_0 is the lowest index such that $\triangle^4 = a_t$, we assume that $a_{t_0} < \triangle < a_{t_0} + \epsilon$, where ϵ is an arbitrarily small positive number. If $\triangle^4 = b_{t_0}$, where t_0 is the lowest index such that $\triangle^4 = b_t$, we assume that $\triangle = b_{t_0}$. Similar to the argument in case (ib2), each of the three assumptions above will result in a contradiction.

We now look at the computational complexity of Algorithms 3 and 4. In Algorithm 3, since it takes $O(nlogn)$ time to compute $\min\{p_j - R_i(r_j) | j \in A_1, r_j < S^*_{i^2}, p_j < R_i(S^*_{i^1})\}$, the preprocessing step takes $O(nlogn)$ time. In Step 1, it takes constant time to compute $a_1 = S^*_{j_1} - S^*_{i^1}$. It takes $O(nlogn)$ time to compute $b_1 = \min\{R_{j_1}(r_j) - p_j | j \in \Phi_1, j \neq j_1\}$. Given a_1 and b_1, $\delta(1)$ takes constant time to compute. Thus, Step 1 takes $O(nlogn)$ time. In Step 5, it takes $O(n)$ time to verify $\triangle^3 \neq b_t$ for $t = 1, 2, \ldots, m$, and $\triangle^3 \neq a_1$. Thus, Step 5 takes $O(n)$ time. Steps 2 to 4 is iterated $m - 1$ times. Inside the iteration loop, the most time-consuming calculation is for $b_t = \min\{R_{j_t}(r_j) - p_j | j \in \Phi_t, j \neq j_t\}$. Given b_t, the other variables takes constant time to compute. Since it takes $O(nlogn)$ time to compute $\min\{R_{j_t}(r_j) - p_j | j \in \cup_{t=2}^{m} \Phi_t \setminus \{j_t\}\}$, it takes $O(nlogn)$ time to compute $b_2, b_3, \ldots, b_m$. In summary, it can be seen that the computational complexity of Algorithm 3 is $O(nlogn)$.

Similarly, we can show that the computational complexity of Algorithm 4 is $O(nlogn)$.

IV. CONCLUSION

We present in this paper a sensitivity analysis for the single-machine preemptive scheduling problem of minimizing flow time. First, we focus on the effects of a parameter change on the optimal sequence. We find that the result of the parameter change is closely related to the movement of job blocks. Thus, we rewrite the optimal sequence in job block form, and its basic characteristics are investigated bit by bit. The results show that a correct necessary and sufficient condition is provided for the optimal sequence to remain optimal after a single release data changes. This is generated by four $O(nlogn)$ iterative time algorithms in the order of the job blocks.

Our future work will explore how to apply the results and analysis methods of this paper to practice, and investigate the robust performance in the optimal sequence after multiple release data change, and further study and develop the theory and methods of sensitivity analysis for scheduling problems.

REFERENCES

[1] T. Gal, *Postoptimal Analysis, Parametric Programming and Related Topics*. New York, NY, USA: McGraw-Hill, 1979.
[2] T. Gal and H. J. Greenberg, *Advances in Sensitivity Analysis and Parametric Programming*. Boston, MA, USA: Kluwer Academic, 1993.
[3] S. Van Hoesel and A. Wagelmans, "On the complexity of postoptimality analysis of 01 programs," *Discrete Appl. Math.*, vol. 91, nos. 1–3, pp. 251–263, Jan. 1999.
[4] M. Libura, "Sensitivity analysis for minimum Hamiltonian path and traveling salesman problems," *Discrete Appl. Math.*, vol. 30, nos. 2–3, pp. 197–211, Feb. 1991.
[5] R. E. Tarjan, "Sensitivity analysis of minimum spanning trees and shortest path trees," *Inf. Process. Lett.*, vol. 14, no. 1, pp. 30–33, Mar. 1982.
[6] J. Schulte and V. Nissen, "Sensitivity analysis on constraints of combinatorial optimization problems," in *Learning and Intelligent Optimization* (Lecture Notes in Computer Science) vol. 12931. Cham, Switzerland: Springer, 2021, pp. 394–408.
[7] J. Chen, C. Du, P. Han, and Y. Zhang, "Sensitivity analysis of strictly periodic tasks in multi-core real-time systems," *IEEE Access*, vol. 7, pp. 135005–135022, 2019.
[8] Y. N. Sotskov, V. K. Leontev, and E. N. Gordeev, "Some concepts of stability analysis in combinatorial optimization," *Discrete Appl. Math.*, vol. 58, no. 2, pp. 169–190, Mar. 1995.
[9] E. Michael, T. A. Wood, C. Manzie, and I. Shames, "Sensitivity analysis for bottleneck assignment problems," *Eur. J. Oper. Res.*, vol. 303, no. 1, pp. 159–167, Nov. 2022.
[10] M. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 4th ed. Englewood Cliffs, NJ, USA: Prentice Hall, 2012.
[11] N. V. R. Mahadev, A. Pekec, and F. S. Robert, "On the meaningfulness of optimal solutions to scheduling problems: Can an optimal solution be nonoptimal?" *Operation Res.*, vol. 46, no. 3, pp. 120–134, 1998.
[12] B. Penz, C. Rapine, and D. Trystram, "Sensitivity analysis of scheduling algorithms," *Eur. J. Oper. Res.*, vol. 134, no. 3, pp. 606–615, Nov. 2001.
[13] S. Chanas and A. Kasperski, "Sensitivity analysis in the single-machine scheduling problem with max-min criterion," *Int. Trans. Oper. Res.*, vol. 12, no. 3, pp. 287–298, May 2005.
[14] Z.-D. Jiang, S.-J. Sun, and Z.-G. Wu, "Sensitivity analysis for some scheduling problems," *J. Shanghai Univ.*, vol. 12, no. 1, pp. 20–25, Feb. 2008.
[15] C. A. Tovey, "Rescheduling to minimize makespan on a changing number of identical processors," *Nav. Res. Logistics Quart.*, vol. 33, no. 4, pp. 717–724, Nov. 1986.
[16] A. W. J. Kolen, A. H. G. R. Kan, C. P. M. van Hoesel, and A. P. M. Wagelmans, "Sensitivity analysis of list scheduling heuristics," *Discrete Appl. Math.*, vol. 55, no. 2, pp. 145–162, Nov. 1994.
[17] S. Maqsood, S. Noor, M. K. Khan, and A. Wood, "Hybrid Genetic Algorithm (GA) for job shop scheduling problems and its sensitivity analysis," *Int. J. Intell. Syst. Technol. Appl.*, vol. 11, nos. 1–2, pp. 49–62, 2012.
[18] G. M. Schrage, "A proof of the optimility of the shortest remaining processing time discipline," *Operation Res.*, vol. 16, pp. 687–690, Jun. 1968.
[19] N. G. Hall and M. E. Posner, "Sensitivity analysis for scheduling problems," *J. Scheduling*, vol. 7, no. 1, pp. 49–83, Jan. 2004.

XIAOXI LI received the Ph.D. degree in surgery from Shanghai Jiao Tong University School of Medicine (SJTUSM), Shanghai, China, in 2018. She is currently an Orthopedic Surgical Resident with the Shanghai Sixth People's Hospital. Aside from clinical research, she is also interested in healthcare management and scheduling theory.

SHANLIN LI received the M.S. degree in applied mathematics from Chongqing University, Chongqing, China, in 1988. He is currently an Associate Professor with the School of Electronics and Information Engineering, Taizhou University. His research interests include scheduling theory, operations research, and healthcare management.