This is the Pre-Published Version.

Minimizing Total Completion Time Subject to Job Release Dates and Preemption Penalties

Zhaohui Liu[1,2] and T.C. Edwin Cheng[2,*]

[1]Department of Mathematics, East China University of Science and Technology

Shanghai 200237, People's Republic of China

[2]Department of Management, The Hong Kong Polytechnic University

Kowloon, Hong Kong SAR, People's Republic of China

SUMMARY

Extensive research has been devoted to preemptive scheduling. However, little attention has been paid to problems where a certain time penalty must be incurred if preemption is allowed. In this paper, we consider the single-machine scheduling problem of minimizing the total completion time subject to job release dates and preemption penalties, where each time a job is started, whether initially or after being preempted, a job-independent setup must take place. The problem is proved to be strongly NP-hard even if the setup time is one unit. We also study a natural extension of a greedy algorithm, which is optimal in the absence of preemption penalty. It is proved that the algorithm has a worst-case performance bound of 25/16, even when the maximum completion time, i.e., makespan, criterion is considered simultaneously.

KEY WORDS: preemptive scheduling; preemption penalty; setup time

[*]Corresponding author.

1. INTRODUCTION

Preemptive scheduling problems are those in which the processing of a job can be temporarily interrupted, and restarted at a later time. Conventionally, in the literature on preemptive scheduling, preempted jobs can simply be resumed from the point at which preemption occurred at no cost. However, this situation is not always true in practice. It is likely that in some cases, a certain delay or setup time must be incurred before a preempted job can be resumed, i.e., a certain time penalty must be incurred. Consider the situation in a computer system. In order to execute more urgent or short tasks, the operating system must interrupt current tasks temporarily. Later, when the interrupted tasks are resumed, some extra time must be expended. That might include the time to load relevant compilers into memory, the time to get the information about done and left work, the time to repeat some work, and so on.

Several papers have been devoted to scheduling with preemption penalties. Potts and Van Wassenhove [1] suggested to consider preemption penalties under the lot-sizing model. Then, Monma and Potts [2] and Chen [3] studied the preemptive parallel machine scheduling problem with batch setup times. Zdrzalka [4], Schuurman and Woeginger [5] and Liu and Cheng [6] studied preemptive scheduling problems with job-dependent setup times. Julien, Magazine and Hall [7] proposed more preemption models and applied them to two single-machine scheduling problems. In this paper, we investigate the single-machine problem of minimizing the total completion time subject to job release dates in the *preemption-setup* model, where each time a job is started, whether initially or after having been preempted, a setup must take place.

To state our problem, we are given a set of n jobs $J = \{J_1, J_2, \ldots, J_n\}$, where job J_j is associated with a processing time p_j and a release date r_j, before which it cannot be processed. Also, we are given a machine that can handle only one job at a time. All jobs may be preempted. Whenever a job is to be started, whether initially or after preemption, a job-independent setup is necessary. The setup time is s. The setup can be performed only after the corresponding job is released and the setup is subject to the *preemption-repeat* mode, i.e., a preempted setup must be totally repeated. During the setup time the machine is unavailable for processing. Our objective is to find a schedule that minimizes the total completion time of the n jobs.

It is well-known that if $s = 0$, i.e., no preemption penalty, the above problem is solved by the shortest remaining processing time (SRPT) rule: at any time, process the unfinished job with the shortest remaining processing time among the available jobs. However, little is known about the case of $s \neq 0$. In the next section, we show that the problem is strongly NP-hard even if $s = 1$. Then in Section 3, we present a greedy algorithm, which is a generalization of the SRPT rule. It is proved that the algorithm has a worst-case performance bound of 25/16, even when the maximum completion time, i.e., makespan, criterion is considered simultaneously. Finally, some concluding remarks are made in Section 4.

2. COMPUTATIONAL COMPLEXITY

In this section, we prove that the problem of scheduling subject to job release dates and preemption penalties is strongly NP-hard. This is achieved by a reduction from Numerical Matching with Target Sum (NMTS), which is known to be strongly NP-hard (Garey and Johnson [8]).

NMTS: Given two sets of positive integers $X = \{x'_1, x'_2, \ldots, x'_{2m}\}$ and $B = \{b'_1, b'_2, \ldots, b'_m\}$ with $\sum_{i=1}^{2m} x'_i = \sum_{i=1}^{m} b'_i$, decide if there exists a partition of the index set $I = \{1, 2, \ldots, 2m\}$ into m disjoint 2-element subsets $I_1, I_2, \ldots, I_m$ such that $\sum_{k\in I_j} x'_k = b'_j$ $(j = 1, 2, \ldots, m)$.

Let I be an instance of NMTS and

$$\begin{aligned} b &= 2m^2 \sum_{i=1}^{m} b'_i\,, \\ x_i &= b + x'_i \quad (i = 1, 2 \ldots, 2m)\,, \\ b_i &= 2b + b'_i \quad (i = 1, 2 \ldots, m)\,, \\ L &= 2\sum_{i=1}^{m} b_i + 1\,. \end{aligned}$$

We construct the following instance P of the decision version of the scheduling problem under discussion.

For $i = 1, 2 \ldots, 2m$, let J_i be a job with zero release date and processing time

$$p_i = x_i - 1\,.$$

We call them X-jobs.

For $i = 2m + (j-1)L + 1, \ldots, 2m + jL$ $(1 \le j \le m)$, let J_i be a job with unit processing time and release date

$$r_i = \sum_{k=1}^{j} b_k + 2(j-1)L\,.$$

We call them U-jobs. Note that for given j, $J_{2m+(j-1)L+1}, \ldots, J_{2m+jL}$ have the same release date. We specially call them U_j-jobs and denote their release date by R_j.

The setup time of each job is one unit. Given the threshold value

$$\delta = m^2L^2 + (2m-1)mL + (2+L)\sum_{k=1}^{m}(m-k+1)b_k\,,$$

we are asked to decide if there exists a feasible schedule σ for P such that $TCT(\sigma) \le \delta$, where $TCT(\sigma)$ denotes the total completion time of σ.

Lemma 1 If the answer to I is "Yes", then the answer to P is "Yes", too.

Proof. Suppose that $\{I_1, I_2, \ldots, I_m\}$ is a partition of I such that

$$|I_j| = 2, \quad \sum_{k \in I_j} x'_k = b'_j \quad (j = 1, 2, \ldots, m)\,.$$

Let $I_j = \{\xi(j), \eta(j)\}$, where $\xi(j), \eta(j) \in \{1, 2, \ldots, 2m\}$. Then

$$p_{\xi(j)} + p_{\eta(j)} = x_{\xi(j)} + x_{\eta(j)} - 2 = b_j - 2\,.$$

We construct σ as follows:

$$\sigma = (J_{\xi(1)} J_{\eta(1)} U_1 \cdots U_1 J_{\xi(2)} J_{\eta(2)} U_2 \cdots U_2 \cdots J_{\xi(m)} J_{\eta(m)} U_m \cdots U_m)\,,$$

where no preemption happens. Noticing the completion time of $J_{\eta(j)}$ is equal to $2(j-1)L + \sum_{k=1}^{j}(2 + p_{\xi(k)} + p_{\eta(k)}) = R_j$, we have

$$\begin{aligned} TCT(\sigma) &< (2+L)\sum_{j=1}^{m} R_j + \sum_{j=1}^{m}\sum_{k=1}^{L} 2k \\ &= (2+L)\left(m(m-1)L + \sum_{j=1}^{m}\sum_{k=1}^{j} b_k\right) + mL(L+1) \\ &= \delta\,. \end{aligned}$$

Thus, the answer to P is "Yes". □

In the following, we will show that the converse of Lemma 1 is also true. Let σ be a feasible schedule for P with $TCT(\sigma) \le \delta$. Since all U-jobs have a unit processing time, it is reasonable to require that σ satisfies the following conditions:

(C1) The processing order of U-jobs abides by the earliest release date rule.

(C2) None of the U-jobs is preempted.

(C3) For each $j = 1, 2, \ldots, m$, all U_j-jobs are processed consecutively.

Now we discuss further the form of σ. For each $j = 1, 2, \ldots, m$, let $t_j = R_j + \epsilon_j$ ($\epsilon_j \ge 0$) be the start time of the first U_j-job in σ. Thus, the total completion time of all U-jobs is given by

$$\begin{aligned} \delta_1 &= \sum_{j=1}^{m}\left(L(R_j + \epsilon_j) + \sum_{k=1}^{L} 2k\right) \\ &= L\sum_{j=1}^{m}\epsilon_j + L\sum_{j=1}^{m}\sum_{k=1}^{j} b_k + m(m-1)L^2 + mL(L+1) \\ &= L\sum_{j=1}^{m}\epsilon_j + \delta - 2m(m-1)L - 2\sum_{j=1}^{m}(m-j+1)b_j\,. \end{aligned} \tag{1}$$

Lemma 2 For each $j = 1, 2, \ldots, m$, there are at most $2j$ X-jobs completed by time t_j in schedule σ.

Proof. The conclusion is trivial for $j = m$. Suppose to the contrary that for some j_0 with $1 \le j_0 \le m-1$, there are at least $2j_0 + 1$ X-jobs completed by time t_{j_0}. Note that the total setup and processing requirement of the $2j_0+1$ X-jobs is greater than $(2j_0+1)b$. By condition (C1), all U_1-jobs, U_2-jobs, $\ldots$, U_{j_0-1}-jobs, which have a total setup and processing requirement of $2(j_0-1)L$, should have been finished by time t_{j_0}. Then

$$t_{j_0} = R_{j_0} + \epsilon_{j_0} > (2j_0+1)b + 2(j_0-1)L\,,$$

i.e.,

$$\epsilon_{j_0} > (2j_0+1)b - \sum_{k=1}^{j_0} b_k = b - \sum_{k=1}^{j_0} b'_k \ge 2m^2 \sum_{k=j_0+1}^{m} b'_k\,.$$

Since $j_0 \le m-1$, it holds that $\epsilon_{j_0} > 2m^2$. Combined with (1), the inequality implies that

$$\begin{aligned} TCT(\sigma) &\ge \delta_1 \\ &> 2m^2L + \delta - 2m(m-1)L - 2\sum_{j=1}^{m}(m-j+1)b_j \\ &\ge \delta + 2mL - 2m\sum_{j=1}^{m} b_j > \delta\,, \end{aligned}$$

a contradiction. □

In fact, due to conditions (C2) and (C3), Lemma 2 implies that for each $j = 1, 2, \ldots, m$, there are at most $2j$ X-jobs completed by time $R_j + 2L$, i.e., there are at least $2(m-j)$ X-jobs completed after time $R_j + 2L$. Let θ be the number of X-jobs completed after time $R_m + 2L$, and δ_2 denote the total completion time of all X-jobs. Then

$$\begin{aligned} \delta_2 &\ge 2\sum_{j=1}^{m-1}(R_j + 2L) + \theta(R_m - R_{m-1}) \\ &\ge 2m(m-1)L + 2\sum_{j=1}^{m-1}\sum_{k=1}^{j} b_k + 2\theta L\,. \end{aligned} \tag{2}$$

Lemma 3 $\theta = 0$ and $\epsilon_j = 0$ for each $j = 1, 2, \ldots, m$.

Proof. By (1) and (2), we have

$$TCT(\sigma) = \delta_1 + \delta_2 \ge L\sum_{j=1}^{m}\epsilon_j + \delta - 2\sum_{j=1}^{m} b_j + 2\theta L\,.$$

Since $TCT(\sigma) \leq \delta$, it holds that

$$L\left(2\theta + \sum_{j=1}^{m} \epsilon_j\right) \leq 2\sum_{j=1}^{m} b_j = L - 1\,.$$

Then the desired results follow from the fact that θ and ϵ_j $(j = 1, 2, \ldots, m)$ are integers. □

From Lemma 3, we deduce that all jobs are completed by time $R_m + 2L$ in σ and $t_j = R_j$ for each $j = 1, 2, \ldots, m$. The former implies that σ contains no idle time and no preemption happens in σ. Let I_1 be the index set of X-jobs completed by time t_1, and for $j = 2, 3, \ldots, m$, I_j be the index set of X-jobs processed between $t_{j-1} + 2L$ and t_j. Then

$$\sum_{k \in I_j} (1 + p_k) = t_j - (t_{j-1} + 2L) = b_j \quad (j = 2, 3, \ldots, m)\,,$$

i.e.,

$$\sum_{k \in I_j} x'_k + |I_j| b = 2b + b'_j\,.$$

From the above relation, it is easy to show that $|I_j| = 2$ and $\sum_{k \in I_j} x'_k = b'_j$. Thus, $\{I_1, I_2, \ldots, I_m\}$ is a solution to instance I, i.e., the following lemma is true.

Lemma 4 If the answer to P is "Yes", then the answer to I is "Yes", too.

Combining Lemmas 1 and 4, we obtain the following conclusion.

Theorem 1 The single-machine scheduling problem of minimizing the total completion time subject to job release dates and preemption penalties is strongly NP-hard even if the setup time is one unit.

3. A GREEDY ALGORITHM

The greedy technique is among the fundamental techniques for the design of approximation algorithms. Actually, the SRPT rule is a greedy algorithm for the special case of our problem in which $s = 0$. In the following, we present a greedy algorithm for the general problem, which reduces to the SRPT rule when $s = 0$.

Algorithm H: Whenever a job is completed or a new job is released, schedule the unfinished job that can be completed at the earliest time (preempting when necessary).

To evaluate the performance of algorithm H with respect to the total completion time, we will first analyse its performance with respect to the maximum completion time criterion. Note that minimizing the maximum completion time is solved by scheduling all jobs in order of nondecreasing rlease dates without preemption. But we have two reasons to study the performance of algorithm H regarding the maximum completion time:

(i) the result will serve as a lemma for the analysis of the total completion time criterion;

(ii) a schedule of high quality with respect to more than one criterion is favored in many practical applications.

3.1. The performance with respect to the maximum completion time

Let σ denote the schedule produced by algorithm H. It is reasonable to assume that σ contains no idle time here. Let $C_{[0]} = 0$ and $C_{[k]}$ be the kth earliest completion time in σ for $k = 1, 2, \ldots, n$. Let $J_1^k, J_2^k, \ldots, J_{\lambda(k)}^k$ be all the job-pieces that are performed in that order in the interval $(C_{[k-1]}, C_{[k]})$ according to σ, where $\lambda(k)$ denotes the number of job-pieces in $(C_{[k-1]}, C_{[k]})$. Note that a job-piece is either a whole setup plus a part of a job or only an incomplete setup. For each job-piece J_i^k, we introduce the following notation:

s_i^k – the setup time of J_i^k;

t_i^k – the processing time of J_i^k;

q_i^k – the remaining processing time of the job related to J_i^k after J_i^k is finished;

r_i^k – the release date of the job related to J_i^k;

p_i^k – the total processing time of the job related to J_i^k.

Obviously, each $J_{\lambda(k)}^k$ contains a whole setup and the following lemma holds.

Lemma 5 For each k with $\lambda(k) \geq 2$, it holds that $r_i^k = C_{[k-1]} + \sum_{j=1}^{i-1}(s_j^k + t_j^k)$ and $p_i^k = t_i^k + q_i^k$ $(i = 2, 3, \ldots, \lambda(k))$.

Let l $(0 \leq l < n)$ be the minimum index such that $\lambda(l+1) = \lambda(l+2) = \cdots = \lambda(n) = 1$. Since $\lambda(n) = 1$ always holds, l must exist. If $l = 0$, then $C_{[n]} = \sum_{k=1}^{n}(s + p_k) \leq C_{\max}^*$, where $C_{\max}^*$ denotes the minimum makespan. In the following we assume that $1 \leq l < n$, which implies $\lambda(l) \geq 2$.

Lemma 6 $C_{[l]} \leq C_{\max}^*$.

Proof. Since $\lambda(l) \geq 2$, it follows from Lemma 5 that

$$r^l_{\lambda(l)} = C_{[l-1]} + \sum_{j=1}^{\lambda(l)-1} (s^l_j + t^l_j)\,.$$

Therefore, $C_{[l]} = r^l_{\lambda(l)} + s^l_{\lambda(l)} + t^l_{\lambda(l)} \leq C^*_{\max}$. □

Define

$$\begin{aligned} X &= \sum\{s^k_i \mid \lambda(k) \geq 2,\ 1 \leq i \leq \lambda(k)-1\}\,, \\ Y &= C_{[l]} - X\,, \\ Z &= \sum_{k=l+1}^{n} (s + t^k_1) = C_{[n]} - C_{[l]}\,. \end{aligned}$$

Note that $X + Y + Z = C_{[n]}$ and $Y + Z = ns + \sum_{k=1}^{n} p_k \leq C^*_{\max}$.

Lemma 7 $X \leq Y + \frac{2}{9}Z$.

Proof. See Appendix A. □

Theorem 2 $C_{[n]} \leq \frac{25}{16} C^*_{\max}$.

Proof. Note that $C_{[n]} = X + Y + Z \leq X + C^*_{\max}$. If $X \leq \frac{9}{16} C_{[l]}$, then it follows from Lemma 6 that $X \leq \frac{9}{16} C^*_{\max}$. If $X > \frac{9}{16} C_{[l]}$, then $Y = C_{[l]} - X < \frac{7}{16} C_{[l]} \leq \frac{7}{16} C^*_{\max}$. By Lemma 7, we have

$$X \leq \frac{2}{9}(Y+Z) + \frac{7}{9}Y < \frac{2}{9}C^*_{\max} + \frac{7}{9}\cdot\frac{7}{16}C^*_{\max} = \frac{9}{16}C^*_{\max}\,.$$

Thus, $C_{[n]} \leq X + C^*_{\max} \leq \frac{25}{16} C^*_{\max}$ □

The example with $s = 1$ in Table 1 shows that the bound in Theorem 2 is tight. Obviously, $C^*_{\max} = 16 + 16\epsilon$ is obtained by scheduling jobs in increasing order of their release dates. However, algorithm H produces schedule σ as follows:

$$\begin{array}{ccccccccccccccccccc} r_1 & r_2 & r_3 & r_4 & & r_5 & & r_6 & & r_7 & & r_8 & & r_9 & & r_{10} & & & \\ J_1 & J_2 & J_3 & J_4 & J_3 & J_5 & J_3 & J_6 & J_3 & J_7 & J_3 & J_8 & J_3 & J_9 & J_3 & J_{10} & J_3 & J_2 & J_1 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\ (J^1_1 & J^1_2 & J^1_3 & J^1_4 & J^2_1 & J^2_2 & J^3_1 & J^3_2 & J^4_1 & J^4_2 & J^5_1 & J^5_2 & J^6_1 & J^6_2 & J^7_1 & J^7_2 & J^8_1 & J^9_1 & J^{10}_1) \end{array}$$

where each job-piece contains a whole setup. Thus, $C_{[n]} = 25 + 16\epsilon$. We get

$$C_{[n]}/C^*_{\max} = (25 + 16\epsilon)/(16 + 16\epsilon) \to 25/16\,, \quad \text{as} \quad \epsilon \to 0\,.$$

Table 1

i	1	2	3	4	5	6	7	8	9	10
r_i	0	1	2	3	$5+\epsilon$	$7+2\epsilon$	$9+3\epsilon$	$11+4\epsilon$	$13+5\epsilon$	$15+6\epsilon$
p_i	$3+4\epsilon$	$2+3\epsilon$	$1+2\epsilon$	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ	ϵ

3.2. The performance with respect to the total completion time

In this subsection, we analyse the performance of algorithm H with respect to the total completion time. We will show that by any time $\frac{25}{16}t$, the schedule produced by algorithm H has finished at least as many jobs as an optimal total completion time schedule could have finished by time t. The idea is similar to that used in Phillips, Stein and Wein [9] for studying a parallel machine problem without preemption penalties.

Let $N_H(J,t)$ denote the number of jobs completed by time t when the set of jobs J is scheduled according to algorithm H. We have the following lemma.

Lemma 8 Let I and J be two sets of jobs with $I \subseteq J$. Then for any $t \geq 0$, it holds that $N_H(J,t) \geq N_H(I,t)$.

Proof. See Appendix B. □

Theorem 3 With respect to the total completion time, algorithm H has a performance bound of $\frac{25}{16}$.

Proof. Given an optimal total completion time schedule, we first show that $N_H(J, \frac{25}{16}t) \geq N_{opt}(J,t)$ for any $t \geq 0$. Consider the set of jobs $J_{opt}(t)$ finished in the optimal total completion time schedule by time t. Note that $N_{opt}(J,t) = |J_{opt}(t)|$. Since the performance bound of algorithm H regarding the maximum completion time is $\frac{25}{16}$, we have

$$N_H(J_{opt}(t), \frac{25}{16}t) = |J_{opt}(t)|\,.$$

Since $J_{opt}(t) \subseteq J$, it follows from Lemma 8 that

$$N_H(J, \frac{25}{16}t) \geq N_H(J_{opt}(t), \frac{25}{16}t)\,.$$

Then $N_H(J, \frac{25}{16}t) \geq N_{opt}(J,t)$.

For $k = 1, 2, \ldots, n$, let $C_{[k]}$ and $C_{[k]}^{opt}$ denote the kth earliest completion time in the schedule produced by algorithm H and the optimal total completion time schedule, respectively. From the result above, we obtain that $C_{[k]} \leq \frac{25}{16}C_{[k]}^{opt}$ for each k. This completes the proof. □

4. CONCLUDING REMARKS

In this paper, we have studied the single-machine scheduling problem of minimizing the total completion time subject to job release dates and preemption penalties, where each time a job is started, whether initially or after being preempted, a job-independent setup must take place. The problem is proved to be strongly NP-hard even if the setup time is one unit. Also, a greedy heuristic is presented and its worst-case performance bound with respect to both the total completion time and the maximum completion time is studied. The bound is tight regarding the maximum completion time, but we do not know whether the bound is tight regarding the total completion time.

Scheduling with preemption penalties is a new topic in scheduling research. We hope that more attention can be paid to it. In fact, besides the preemption-setup model, some other preemption models have been presented in [7], such as *preemption-startup* model, where the finished part of a preempted job must be repeated in some proportion.

ACKNOWLEDGMENT

This research was supported in part by The Hong Kong Polytechnic University under grant number G-YW59. The authors were also respectively supported by the National Natural Science Foundation of China under grant number 10101007 and the Research Grants Council of Hong Kong under grant number PolyU 5245/99H.

REFERENCES

1. Potts CN, Van Wassenhove LN. Integrating scheduling with batching and lot-sizing: a review of algorithms and complexity. *Journal of the Operational Research Society* 1992; **43**:395–406.

2. Monma CL, Potts CN. Analysis of heuristics for preemptive parallel machine scheduling with batch setup times. *Operations Research* 1993; **41**:981–993.

3. Chen B. A better heuristic for preemptive parallel machine scheduling with batch setup times. *SIAM Journal on Computing* 1993; **22**:1303–1318.

4. Zdrzalka S. Preemptive scheduling with release dates, delivery times and sequence independent setup times. *European Journal of Operational Research* 1994; **76**:60–71.

5. Schuurman P, Woeginger GJ. Preemptive scheduling with job-dependent setup times. *Proc. 10th ACM-SIAM Symposium on Discrete Algorithms* 1999; 759–767.

6. Liu Z, Cheng TCE. Scheduling with job release dates, delivery times and preemption penalties. *Information Processing Letters* 2002; **82**:107–111.

7. Julien FM, Magazine MJ, Hall NG. Generalized preemption models for single-machine dynamic scheduling problems. *IIE Transactions* 1997; **29**:359–372.

8. Garey MR, Johnson DS. *Computers and Intractability: A Guide to the Theory of NP-completeness.* Freeman: San Francisco, 1979.

9. Phillips C, Stein C, Wein J. Minimizing average completion time in the presence of release dates. *Mathematical Programming* 1998; **82**:199–223.

APPENDIX A. PROOF OF LEMMA 7

We first prove that for any k, i, j with $1 \leq i \leq \lambda(k) - 1$ and $i + 1 \leq j \leq \lambda(k)$, $q_i^k > \sum_{u=i}^{j-1} s_u^k + p_j^k$ holds. By algorithm H, the fact that J_i^k is preempted implies $s - s_i^k + q_i^k > s + p_{i+1}^k$, i.e., $q_i^k > s_i^k + p_{i+1}^k$. Noticing $p_{i+1}^k \geq q_{i+1}^k$, we can successively prove that

$$\begin{aligned} q_i^k &> s_i^k + p_{i+1}^k \\ &> s_i^k + s_{i+1}^k + p_{i+2}^k \\ &\vdots \\ &> s_i^k + s_{i+1}^k + \cdots + s_{j-1}^k + p_j^k \,. \end{aligned} \tag{3}$$

Next we prove Lemma 7. Partition the index set $\{1, 2, \ldots, n\}$ into $K_1, K_2, \ldots, K_m$ by the following two steps:
Step 1. $K_1 := \{1\}$, $m := 1$.
Step 2. For $k := 2$ to n do
If there exist indices i, u, v ($1 \leq i \leq m$, $u \in K_i$, $1 \leq v \leq \lambda(u) - 1$) such that job-pieces J_1^k and J_v^u come from the same job, then $K_i := K_i \cup \{k\}$, else $K_{m+1} := \{k\}, m := m+1$.

For each $K \in \{K_1, K_2, \ldots, K_m\}$, we define

$$\begin{aligned} X(K) &= \sum \{s_i^k \,|\, k \in K,\, \lambda(k) \geq 2,\, 1 \leq i \leq \lambda(k) - 1\}\,, \\ Y(K) &= \sum \{s_{\lambda(k)}^k \,|\, k \in K,\, k \leq l\}\,, \\ Z(K) &= \sum \{s + t_1^k \,|\, k \in K,\, k \geq l + 1\}\,. \end{aligned}$$

Obviously, it holds that $X = \sum_K X(K)$, $Y \geq \sum_K Y(K)$ and $Z = \sum_K Z(K)$. Thus, to show that $X \leq Y + \frac{2}{9}Z$, we need only to show that for each K,

$$X(K) \leq Y(K) + \frac{2}{9} Z(K)\,. \tag{4}$$

Let $k(1) = \min\{k \,|\, k \in K\}$. If $k(1) \geq l + 1$, then $X(K) = 0$. The conclusion certainly holds. In the following we suppose that $k(1) \leq l$. Steps $1' \sim 5'$ choose a subset K^* of K.

Step 1$'$. $g := 1$.

Step 2$'$. Determine the index $h(g)$ $(0 \le h(g) \le \lambda(k(g)) - 1)$ such that

1) the jobs related to $J_1^{k(g)}, \ldots, J_{h(g)}^{k(g)}$ are completed after $C_{[l]}$;

2) the jobs related to $J_{h(g)+1}^{k(g)}, \ldots, J_{\lambda(k(g))}^{k(g)}$ are completed at or before $C_{[l]}$.

Step 3$'$. If $h(g) = 0$ or the job related to $J_{h(g)}^{k(g)}$ does not appear in $(C_{[k(g)]}, C_{[l]})$, then goto Step 5$'$, else perform Step 4$'$.

Step 4$'$. Letting J_1^u be the last job-piece before $C_{[l]}$ that comes from the same job as $J_{h(g)}^{k(g)}$, then $k(g+1) := u$, $g := g+1$ and goto Step 2$'$.

Step 5$'$. $K^* := \{k(1),\, k(2),\, \ldots,\, k(g)\,\}$.

Since for each $i = 1, 2, \ldots, g-1$, $J_1^{k(i+1)}$ and $J_{h(i)}^{k(i)}$ come from the same job, $K^* \subseteq K$ holds. It is easy to verify that

(A1) if $h(1) = 0$, then $g = 1$;

(A2) if $h(1) \ge 1$, then $h(g) \ge 1$ and $h(2), h(3), \ldots, h(g-1) \ge 2$.

Define

$$
\begin{aligned}
X_1(K) &= \sum\{s_1^k \,|\, k \in K \setminus K^*,\, \lambda(k) \ge 2\}\,, \\
Y_1(K) &= \sum\{s_{\lambda(k)}^k \,|\, k \in K \setminus K^*,\, \lambda(k) \ge 2\}\,, \\
X_2(K) &= \sum_{i=1}^{g} \sum_{j=h(i)+1}^{\lambda(k(i))-1} s_j^{k(i)} + \sum\{s_i^k \,|\, k \in K \setminus K^*,\, 2 \le i \le \lambda(k) - 1\}\,, \\
Y_2(K) &= \sum\{s_{\lambda(k)}^k \,|\, k \in K \setminus K^*,\, k < l\,, \lambda(k) = 1\}\,.
\end{aligned}
$$

Obviously, it holds that $X_1(K) \le Y_1(K)$ and

$$
\begin{aligned}
X(K) &= X_1(K) + X_2(K) + \sum_{i=1}^{g} \sum_{j=1}^{h(i)} s_j^{k(i)}\,, \\
Y(K) &= Y_1(K) + Y_2(K) + \sum_{i=1}^{g} s_{\lambda(k(i))}^{k(i)}\,.
\end{aligned}
$$

We now argue that the jobs related to job-pieces J_i^k $(k \in K \setminus K^*, 2 \le i \le \lambda(k) - 1)$ must have been completed by time $C_{[l-1]}$. Otherwise, there exists a smallest index $k_1 \in K \setminus K^*$ such that for some i $(2 \le i \le \lambda(k_1) - 1)$, the job related to job-piece $J_i^{k_1}$ is completed after $C_{[l]}$. Then, the job related to job-piece $J_1^{k_1}$ is also completed after $C_{[l]}$, and it should not appear in $(C_{[k_1]}, C_{[l]})$. Since $k_1 \in K$ and $k_1 \ne k(1)$, there exists a smallest index $k_2 < k_1$ $(k_2 \in K)$ such that $J_1^{k_1}$ comes from the same job as some $J_j^{k_2}$ ($1 \le j \le \lambda(k_2) - 1$). If $k_2 \notin K^*$, then $j = 1$ (according to the definition of k_1). But $j = 1$ implies there exists a smaller index with the property of k_2, a contradiction. Then, $k_2 \in K^*$. Since the job related to $J_j^{k_2}$ appears as $J_1^{k_1}$, the jobs related to $J_{j+1}^{k_2}, \ldots, J_{\lambda(k_2)}^{k_2}$ must have been completed before $J_1^{k_1}$. Since the job related to $J_j^{k_2}$ is completed after $C_{[l]}$, the jobs related to $J_1^{k_2}, \ldots, J_{j-1}^{k_2}$ must be completed after $C_{[l]}$. Thus, $k_1 \in K^*$, a contradiction, too. Now we have proved that the jobs related to job-pieces J_i^k ($k \in K \setminus K^*, 2 \le i \le \lambda(k) - 1$) must have been completed by time $C_{[l-1]}$, so we have $X_2(K) \le Y_2(K)$. To prove (4), it suffices to prove that

$$\sum_{i=1}^{g} \sum_{j=1}^{h(i)} s_j^{k(i)} \le \sum_{i=1}^{g} s_{\lambda(k(i))}^{k(i)} + \frac{2}{9} Z(K) = gs + \frac{2}{9} Z(K) \,. \tag{5}$$

Since the jobs related to $J_1^{k(i)}, J_2^{k(i)}, \ldots, J_{h(i)-1}^{k(i)}$ ($i = 1, 2 \ldots, g$) do not appear in $(C_{[k(i)]}, C_{[l]})$ and the job related to $J_{h(g)}^{k(g)}$ does not appear in $(C_{[k(g)]}, C_{[l]})$, we have

$$Z(K) \ge \sum_{i=1}^{g} \sum_{j=1}^{h(i)-1} (s + q_j^{k(i)}) + s + q_{h(g)}^{k(g)} \,. \tag{6}$$

Now we prove that

$$\sum_{i=1}^{g} \sum_{j=1}^{h(i)-1} q_j^{k(i)} > \sum_{i=1}^{g} \sum_{j=1}^{h(i)-1} \left(j s_j^{k(i)} + s_j^{k(i)} \sum_{u=1}^{i-1} (h(u) - 1) \right) + \sum_{u=1}^{g} (h(u) - 1) p_{h(g)}^{k(g)} \tag{7}$$

by induction on g. When $g = 1$, it follows from (3) that

$$\sum_{j=1}^{h(1)-1} q_j^{k(1)} > \sum_{j=1}^{h(1)-1} \left(\sum_{u=j}^{h(1)-1} s_u^{k(1)} + p_{h(1)}^{k(1)} \right) = \sum_{j=1}^{h(1)-1} j s_j^{k(1)} + (h(1) - 1) p_{h(1)}^{k(1)} \,.$$

Next we consider the case of $g > 1$. By the induction hypothesis, it holds that

$$\sum_{i=1}^{g} \sum_{j=1}^{h(i)-1} q_j^{k(i)} > \sum_{i=1}^{g-1} \sum_{j=1}^{h(i)-1} \left(j s_j^{k(i)} + s_j^{k(i)} \sum_{u=1}^{i-1} (h(u) - 1) \right) + \sum_{u=1}^{g-1} (h(u) - 1) p_{h(g-1)}^{k(g-1)} + \sum_{j=1}^{h(g)-1} q_j^{k(g)} \,.$$

Additionally, noticing $p_{h(g-1)}^{k(g-1)} \ge q_1^{k(g)}$, we obtain from (3) that

$$\begin{aligned}
\sum_{u=1}^{g-1} (h(u) - 1) p_{h(g-1)}^{k(g-1)} \quad + \quad \sum_{j=1}^{h(g)-1} q_j^{k(g)} &> \sum_{u=1}^{g-1} (h(u) - 1) \left(\sum_{j=1}^{h(g)-1} s_j^{k(g)} + p_{h(g)}^{k(g)} \right) \\
&\quad + \sum_{j=1}^{h(g)-1} \left(\sum_{u=j}^{h(g)-1} s_u^{k(g)} + p_{h(g)}^{k(g)} \right) \\
&= \sum_{j=1}^{h(g)-1} \left(j s_j^{k(g)} + s_j^{k(g)} \sum_{u=1}^{g-1} (h(u) - 1) \right) + \sum_{u=1}^{g} (h(u) - 1) p_{h(g)}^{k(g)} \,.
\end{aligned}$$

Thus, (7) is also true for $g > 1$.

Note that $p_{h(g)}^{k(g)} \geq q_{h(g)}^{k(g)} > s_{h(g)}^{k(g)}$. Combining (6) and (7), we have

$$\begin{aligned} Z(K) &> \sum_{i=1}^{g} \sum_{j=1}^{h(i)-1} \left(s + j s_j^{k(i)} + s_j^{k(i)} \sum_{u=1}^{i-1} (h(u)-1) \right) + \sum_{u=1}^{g} (h(u)-1) s_{h(g)}^{k(g)} + s + s_{h(g)}^{k(g)} \\ &= \sum_{i=1}^{g-1} \sum_{j=1}^{h(i)-1} \left(s + j s_j^{k(i)} + s_j^{k(i)} \sum_{u=1}^{i-1} (h(u)-1) \right) \\ &\quad + \sum_{j=1}^{h(g)} \left(s + j s_j^{k(g)} + s_j^{k(g)} \sum_{u=1}^{g-1} (h(u)-1) \right) . \end{aligned}$$

Thus, to show (5), it suffices to show that for $i = 1, 2, \ldots, g$,

$$\sum_{j=1}^{H(i)} s_j^{k(i)} \leq \mu_i s + \frac{2}{9} \sum_{j=1}^{H(i)} \left(s + j s_j^{k(i)} + s_j^{k(i)} \sum_{u=1}^{i-1} H(u) \right) , \tag{8}$$

where

$$\begin{aligned} H(i) &= h(i) - 1 \quad (i = 1, 2, \ldots, g-1) , \\ H(g) &= h(g) , \\ \sum_{i=1}^{g} \mu_i &\leq 1 . \end{aligned}$$

When $g = 1$, it is simple to show that (8) is true by setting $\mu_1 = 1$. Now consider the case of $g \geq 2$. Due to (A2), for $i \geq 5$, (8) is trivially true even if $\mu_i = 0$. By setting μ_i according to Table 2, we can prove (8) for $i = 1, 2, 3, 4$. □

Table 2

$H(1) \geq 3$	$\mu_1 = 1$	$\mu_2 = 0$	$\mu_3 = 0$	$\mu_4 = 0$
$H(1) = 2$	$\mu_1 = 8/9$	$\mu_2 = 1/9$	$\mu_3 = 0$	$\mu_4 = 0$
$H(1) = 1, H(2) \geq 2$	$\mu_1 = 5/9$	$\mu_2 = 4/9$	$\mu_3 = 0$	$\mu_4 = 0$
$H(1) = 1, H(2) = 1$	$\mu_1 = 5/9$	$\mu_2 = 1/3$	$\mu_3 = 1/9$	$\mu_4 = 0$
$H(1) = 0, H(2) \geq 3$	$\mu_1 = 0$	$\mu_2 = 1$	$\mu_3 = 0$	$\mu_4 = 0$
$H(1) = 0, H(2) = 2$	$\mu_1 = 0$	$\mu_2 = 8/9$	$\mu_3 = 1/9$	$\mu_4 = 0$
$H(1) = 0, H(2) = 1, H(3) \geq 2$	$\mu_1 = 0$	$\mu_2 = 5/9$	$\mu_3 = 4/9$	$\mu_4 = 0$
$H(1) = 0, H(2) = 1, H(3) = 1$	$\mu_1 = 0$	$\mu_2 = 5/9$	$\mu_3 = 1/3$	$\mu_4 = 1/9$

APPENDIX B. PROOF OF LEMMA 8

First, we give a lemma. Its proof is trivial.

Lemma 9 Let Q and Q' be two multi-sets of numbers with $Q \preceq Q'$, which means that $|Q| = |Q'|$ and for each i, the ith smallest element of Q is not greater than the ith smallest element of Q'. Let p and p' be two numbers with $p \le p'$. Then $Q \cup \{p\} \preceq Q' \cup \{p'\}$.

Next we prove Lemma 8. Let $J = \{J_1, J_2, \ldots, J_m\}$. We construct a job set $J' = \{J'_1, J'_2, \ldots, J'_m\}$ as follows:

i) if $J_i \in I$, then $J'_i = J_i$;

ii) if $J_i \in J \setminus I$, then J'_i is such that $r'_i = r_i$ and $p'_i = \infty$.

Clearly $N_H(J', t) = N_H(I, t)$ for any $t \ge 0$, since the jobs in $J' \setminus I$ never finish, and they never run if a job with a finite processing time can run instead. Then it suffices to show that for any $t \ge 0$,

$$N_H(J, t) \ge N_H(J', t)\,. \tag{9}$$

Let σ be the schedule produced by algorithm H for J, and $q_i(t)$ be the remaining processing time of J_i at time t in σ. Note that if J_i is finished at time t, then $q_i(t) = 0$. Let $s_i(t)$ be defined as follows. If J_i is running at time t in σ, then $s_i(t)$ is equal to the remaining quantity of the current setup; if J_i is finished, then $s_i(t) = 0$; if J_i is unfinished and not running, then $s_i(t) = s$. Let $s_{[i]}(t) + q_{[i]}(t)$ be the ith smallest element of multi-set

$$Q(t) = \{s_i(t) + q_i(t) \mid 1 \le i \le m,\ r_i \le t\}\,.$$

Also, we make the analogous definitions σ', $q'_i(t)$, $s'_i(t)$ and $Q'(t)$ for J'. We are going to show that for any $t \ge 0$,

$$Q(t) \preceq Q'(t)\,, \tag{10}$$

i.e.,

$$s_{[i]}(t) + q_{[i]}(t) \le s'_{[i]}(t) + q'_{[i]}(t) \quad \text{for each } i\,.$$

Note that (10) implies (9), because if (10) holds, then $Q(t)$ must contain at least as many zeroes as $Q'(t)$, and hence at least as many jobs have been completed by time t in σ as in σ'.

Let $t_0 = 0$ and $t_1 < t_2 < \cdots < t_m$ be all the completion times in σ. We claim that for each k ($0 \le k \le m$), (10) is true over $[0, t_k]$ by induction on k. At time t_0, (10) is trivially true. As the induction hypothesis, (10) is assumed to be true for $t \in [0, t_{k-1}]$, where $k \ge 1$. Then

$$Q(t_{k-1}) \preceq Q'(t_{k-1})\,, \tag{11}$$

and

$$q_{[i]}(t_{k-1}) \le q'_{[i]}(t_{k-1})\,, \quad \forall\ i \ge k \tag{12}$$

where (12) follows from (11) and the fact that $s_{[i]}(t_{k-1}) = s$ $(i \geq k)$. In the following, we consider the case of $t_{k-1} \leq t \leq t_k$.

Note that $s_{[k]}(t)+q_{[k]}(t)$ remains the smallest positive element of $Q(t)$ over $[t_{k-1}, t_k)$ (though it may correspond to different jobs at different time). To prove (10) for $t_{k-1} \leq t \leq t_k$, we need only to show that

$$s_{[k]}(t) + q_{[k]}(t) \leq s'_{[k]}(t) + q'_{[k]}(t)\,,$$

and

$$Q_0(t) \preceq Q'_0(t)\,,$$

where $Q_0(t) = \{q_{[i]}(t) \mid i \geq k+1,\ r_{[i]} \leq t\}$ and $Q'_0(t) = \{q'_{[i]}(t) \mid i \geq k+1,\ r'_{[i]} \leq t\}$. This will be doned by induction on t.

By (11) and (12), the conclusion is true at time t_{k-1}. From $\tau - 1$ to τ $(\tau > t_{k-1})$, we have to perform two steps. First, we complete one unit of setup or processing for $J_{[k]}$ and $J'_{[l]}$, where $J'_{[l]}$ is such that $s'_{[l]}(\tau-1) + q'_{[l]}(\tau-1)$ is the smallest positive element of $Q'(\tau-1)$. Note that $l \leq k$ must hold. Second, we release each pair of jobs $J_{x(\tau)}$ and $J'_{x(\tau)}$ with $r_{x(\tau)} = r'_{x(\tau)} = \tau$.

Let τ^- be referred to as the left limit of τ. After the first step, we obtain $Q(\tau^-)$ and $Q'(\tau^-)$, where

$$\begin{aligned} s_{[k]}(\tau^-) + q_{[k]}(\tau^-) &= s_{[k]}(\tau-1) + q_{[k]}(\tau-1) - 1\,, \\ s'_{[l]}(\tau^-) + q'_{[l]}(\tau^-) &= s'_{[l]}(\tau-1) + q'_{[l]}(\tau-1) - 1\,, \end{aligned}$$

and the other elements are equal to the corresponding elements in $Q(\tau-1)$ and $Q'(\tau-1)$. Since $l \leq k$, $s_{[k]}(\tau^-) + q_{[k]}(\tau^-) \leq s'_{[k]}(\tau^-) + q'_{[k]}(\tau^-)$ and $Q_0(\tau^-) \preceq Q'_0(\tau^-)$ follow from the induction hypothesis on $\tau-1$. Moreover, it is evident that if $q_{[k]}(\tau-1) \leq q'_{[k]}(\tau-1)$, then $q_{[k]}(\tau^-) \leq q'_{[k]}(\tau^-)$.

Now consider the second step. Let $s'_{[j]}(\tau^-)+q'_{[j]}(\tau^-)$ be the smallest positive element of $Q'(\tau^-)$. Obviously, $j \leq k$ holds. We make a case by case analysis. Note that $p_{y(\tau)}$ and $p'_{y(\tau)}$ respectively denote the elements to be added to $Q_0(\tau^-)$ and $Q'_0(\tau^-)$ after $J_{x(\tau)}$ and $J'_{x(\tau)}$ are released.

Case 1. $s + p_{x(\tau)} \geq s_{[k]}(\tau^-) + q_{[k]}(\tau^-)$ and $s + p'_{x(\tau)} \geq s'_{[k]}(\tau^-) + q'_{[k]}(\tau^-)$.

Now we have that $p_{y(\tau)} = p_{x(\tau)}$ and $p'_{y(\tau)} = p'_{x(\tau)}$. Obviously, it holds that

$$s_{[k]}(\tau) + q_{[k]}(\tau) = s_{[k]}(\tau^-) + q_{[k]}(\tau^-) \leq s'_{[k]}(\tau^-) + q'_{[k]}(\tau^-) = s'_{[k]}(\tau) + q'_{[k]}(\tau)\,.$$

Case 2. $s + p_{x(\tau)} \geq s_{[k]}(\tau^-) + q_{[k]}(\tau^-)$ and there exists u $(j \leq u \leq k)$ such that $s'_{[u-1]}(\tau^-) + q'_{[u-1]}(\tau^-) \leq s + p'_{x(\tau)} < s'_{[u]}(\tau^-) + q'_{[u]}(\tau^-)$.

In this case, it holds that $p'_{x(\tau)} < q'_{[u]}(\tau^-)$ and $p'_{x(\tau)} < q'_{[k]}(\tau^-)$. Then,

$$
\begin{aligned}
p_{y(\tau)} &= p_{x(\tau)}\,, \\
p'_{y(\tau)} &= \max\{q'_{[k]}(\tau^-)\,, q'_{[u]}(\tau^-)\}\,, \\
s'_{[k]}(\tau) &= s\,, \\
q'_{[k]}(\tau) &= \begin{cases} \max\{\min\{q'_{[k]}(\tau^-)\,, q'_{[u]}(\tau^-)\}\,, q'_{[k-1]}(\tau^-)\}\,, & u < k\,, \\ p'_{x(\tau)}\,, & u = k\,. \end{cases}
\end{aligned}
$$

Thus,

$$
s_{[k]}(\tau) + q_{[k]}(\tau) = s_{[k]}(\tau^-) + q_{[k]}(\tau^-) \le s + p_{x(\tau)} \le s + p'_{x(\tau)} \le s'_{[k]}(\tau) + q'_{[k]}(\tau)\,.
$$

Case 3. $s + p_{x(\tau)} < s_{[k]}(\tau^-) + q_{[k]}(\tau^-)$ and $s + p'_{x(\tau)} \ge s'_{[k]}(\tau^-) + q'_{[k]}(\tau^-)$.

Now $p'_{x(\tau)} = \infty$ must hold. We have that $p_{y(\tau)} = q_{[k]}(\tau^-)$, $p'_{y(\tau)} = p'_{x(\tau)}$, and it holds that

$$
\begin{aligned}
s_{[k]}(\tau) + q_{[k]}(\tau) &= s + p_{x(\tau)} \\
&< s_{[k]}(\tau^-) + q_{[k]}(\tau^-) \le s'_{[k]}(\tau^-) + q'_{[k]}(\tau^-) = s'_{[k]}(\tau) + q'_{[k]}(\tau)\,.
\end{aligned}
$$

Furthermore, $q_{[k]}(\tau) = p_{x(\tau)} < q'_{[k]}(\tau)$ holds.

Case 4. $s + p_{x(\tau)} < s_{[k]}(\tau^-) + q_{[k]}(\tau^-)$ and there exists u ($j \le u \le k$) such that $s'_{[u-1]}(\tau^-) + q'_{[u-1]}(\tau^-) \le s + p'_{x(\tau)} < s'_{[u]}(\tau^-) + q'_{[u]}(\tau^-)$.

In this case, $p'_{x(\tau)} < q'_{[u]}(\tau^-)$ and $p'_{x(\tau)} < q'_{[k]}(\tau^-)$ hold, too. We have

$$
\begin{aligned}
p_{y(\tau)} &= q_{[k]}(\tau^-)\,, \\
p'_{y(\tau)} &= \max\{q'_{[k]}(\tau^-)\,, q'_{[u]}(\tau^-)\}\,, \\
s_{[k]}(\tau) &= s'_{[k]}(\tau) = s\,, \\
q_{[k]}(\tau) &= p_{x(\tau)}\,, \\
q'_{[k]}(\tau) &= \begin{cases} \max\{\min\{q'_{[k]}(\tau^-)\,, q'_{[u]}(\tau^-)\}\,, q'_{[k-1]}(\tau^-)\}\,, & u < k\,, \\ p'_{x(\tau)}\,, & u = k\,. \end{cases}
\end{aligned}
$$

Obviously, $q_{[k]}(\tau) \le p'_{x(\tau)} \le q'_{[k]}(\tau)$ holds, and hence $s_{[k]}(\tau) + q_{[k]}(\tau) \le s'_{[k]}(\tau) + q'_{[k]}(\tau)$ holds.

Note that in any Case, we have

$$
Q_0(\tau) = Q_0(\tau^-) \cup \{p_{y(\tau)}\} \text{ and } Q'_0(\tau) = Q'_0(\tau^-) \cup \{p'_{y(\tau)}\}\,.
$$

Since $p_{y(\tau)} \le p'_{y(\tau)}$ holds in Cases $1 \sim 3$ and in Case 4 if $q_{[k]}(\tau^-) \le q'_{[k]}(\tau^-)$, $Q_0(\tau) \preceq Q'_0(\tau)$ follows from $Q_0(\tau^-) \preceq Q'_0(\tau^-)$ and Lemma 9 for these cases. In the following, we analyse Case 4 with $q_{[k]}(\tau^-) > q'_{[k]}(\tau^-)$.

By (12), it holds that $q_{[k]}(t_{k-1}) \le q'_{[k]}(t_{k-1})$. We can determine the latest time τ_* such that $q_{[k]}(\tau_*^-) \le q'_{[k]}(\tau_*^-)$ and $q_{[k]}(t) > q'_{[k]}(t)$ ($\forall\ t \in [\tau_*, \tau)$). Then Case 2 must appear at time τ_*, and neither Case 3 nor Case 4 can appear at time $\tau_*+1, \tau_*+2, \ldots, \tau-1$. Suppose that Case 2 appears at time $\tau_* = \tau_1 < \tau_2 < \cdots < \tau_v$ and Case 1 appears at other times in $\{\tau_*, \tau_* + 1, \ldots, \tau - 1\}$. Let $\tau_{v+1} = \tau$. According to Case 1, the jobs corresponding to index $[k]$ do not change from τ_{i-1} to τ_i^- for each $i = 2, 3, \ldots, v + 1$. Since $s_{[k]}(\tau_i^-) + q_{[k]}(\tau_i^-) \le s'_{[k]}(\tau_i^-) + q'_{[k]}(\tau_i^-)$ and $q_{[k]}(\tau_i^-) > q'_{[k]}(\tau_i^-)$ for $i \ge 2$, we have that $s'_{[k]}(\tau_i^-) > 0$, which implies that $q'_{[k]}(\tau_i^-) = q'_{[k]}(\tau_{i-1})$. Then, according to Case 2 or 4, we get that for $i = v + 1, v, \ldots, 2$,

$$p'_{y(\tau_i)} \ge q'_{[k]}(\tau_i^-) = q'_{[k]}(\tau_{i-1}) \ge p'_{x(\tau_{i-1})} \ge p_{x(\tau_{i-1})} = p_{y(\tau_{i-1})} \,.$$

Note that $\tau_1 = \tau_*$ and $\tau_{v+1} = \tau$. We have

$$p'_{y(\tau_1)} \ge q'_{[k]}(\tau_1^-) \ge q_{[k]}(\tau_1^-) \ge q_{[k]}(\tau^-) = p_{y(\tau_{v+1})} \,.$$

Then $Q_0(\tau) \preceq Q'_0(\tau)$ follows from $Q_0(\tau_1^-) \preceq Q'_0(\tau_1^-)$, where Lemma 9 is applied. $\square$