Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

SEN

Minimizing the total completion time on-line on a single machine, using restarts

R. van Stee, J.A. La Poutré

REPORT SEN-R0211 JUNE 30, 2002

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

Minimizing the Total Completion Time On-line On a Single Machine, Using Restarts

Rob van Stee*

Institut für Informatik, Albert-Ludwigs-Universität,

Georges-Köhler-Allee, 79110 Freiburg, Germany

vanstee@informatik.uni-freiburg.de

Han La Poutré

CWI, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Han.La.Poutre@cwi.nl

ABSTRACT

We give an algorithm to minimize the total completion time on-line on a single machine, using restarts, with a competitive ratio of $3/2$. The optimal competitive ratio without using restarts is 2 for deterministic algorithms and $e/(e-1) \approx 1.582$ for randomized algorithms. This is the first restarting algorithm to minimize the total completion time that is proved to be better than an algorithm that does not restart.

2000 Mathematics Subject Classification: 68Q25, 68W20, 68W25

1998 ACM Computing Classification System: F.2.2.

Keywords and Phrases: on-line scheduling, restarts, total completion time

Note: Work carried out under theme SEN4 "Evolutionary Systems and Applied Algorithmics".

1. Introduction

We examine the scheduling problem of minimizing the total completion time (the sum of completion times) on-line on a single machine, using restarts. Allowing restarts means that the processing of a job may be interrupted, losing all the work done on it. In this case, the job must be started again later (restarted), until it is completed without interruptions. We study the on-line problem, where algorithms must decide how to schedule the existing jobs without any knowledge about the future arrivals of jobs.

We compare the performance of an on-line algorithm $\mathcal{A}$ to that of an optimal off-line algorithm OPT that knows the entire job sequence σ in advance. The total completion time of an input σ given to an algorithm ALG is denoted by ALG(σ). The competitive ratio $\mathcal{R}(\mathcal{A})$ of an on-line algorithm $\mathcal{A}$ is defined as

$$\mathcal{R}(\mathcal{A}) = \sup_{\sigma} \frac{\mathcal{A}(\sigma)}{\text{OPT}(\sigma)} \ .$$

Known results For the case where all jobs are available at time 0, the shortest processing time algorithm SPT [8] has an optimal total completion time. This algorithm runs the jobs in order of

* Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II.

increasing size. Hoogeveen and Vestjens [4] showed that if jobs arrive over time and restarts are not allowed, the optimal competitive ratio is 2, and they gave an algorithm DSPT ('delayed SPT') which maintained that competitive ratio. Two other optimal algorithms were given by Phillips, Stein and Wein [5] and Stougie [9].

Using randomization, it is possible to give an algorithm of competitive ratio $e/(e-1) \approx 1.582$ [1] which is optimal [10]. Vestjens showed a lower bound of 1.112 for deterministic algorithms that can restart jobs [12]. This was recently improved to 1.211 by Epstein and Van Stee [2].

We are aware of three previous instances where restarts were proven to help. Firstly, in [6] it was shown that restarts help to minimize the makespan (the maximum completion time) of jobs with unknown sizes on m related machines. Here each machine has its own speed, which does not depend on the job it is running. The algorithm in [6] obtains a competitive ratio of $O(\log m)$. Without restarts, the lower bound is $\Omega(\sqrt{m})$.

Secondly, [11] shows that restarts help to minimize the maximum delivery time on a single machine, obtaining an (optimal) competitive ratio of 3/2 while without restarts, $(\sqrt{5}+1)/2$ is the best possible. In this problem, each job needs to be delivered after completing, which takes a certain given extra time.

Thirdly, in [3] it is shown that restarts help to minimize the number of *early* jobs (jobs that complete on or before their due date) on a single machine, obtaining an (optimal) competitive ratio of 2 while without restarts, it is not possible to be competitive at all (not even with preemptions).

Our results Until now, it was not known how to use restarts in a deterministic algorithm for minimizing the total completion time on a single machine to get a competitive ratio below 2, whereas a ratio of 2 can be achieved by an algorithm that does not restart. We give an algorithm RSPT ('restarting SPT') of competitive ratio 3/2. This ratio cannot be obtained without restarts, even with the use of randomization.

Our algorithm is arguably the simplest possible algorithm for this problem that uses restarts: it bases the decision about whether or not it will interrupt a running job J for an arriving job J' solely on J and J'. It ignores, for example, all other jobs that are waiting to be run. We show in section 3 that the analysis of our algorithm is tight and that all "RSPT-like" algorithms have a competitive ratio of at least 1.467845. This suggests that a more complicated algorithm would be required to get a substantially better competitive ratio, if possible.

2. Algorithm RSPT

We present our on-line algorithm RSPT for the problem of minimizing the total completion time on a single machine, using restarts. See Figure 1. This algorithm has the following properties (where J, x, s, r and w are defined as in Figure 2.1). OPT is any optimal off-line algorithm (there can be more than one).

R1 RSPT only interrupts a job J for jobs that are smaller and that can finish earlier than J (i.e. $r+w < s+x$).

R2 If RSPT does not interrupt J for a job of size w that arrives at time r, then $r+w > \frac{2}{3}(s+x)$. In this case, if RSPT is still running J at time $r+w$, it runs J until completion.

Proof. Any job J' that arrives after time $r+w$ satisfies $r'+w' \geq r' > r+w > \frac{2}{3}(s+x)$ in this case, and does not cause an interruption. □

R3 Suppose that $s \leq t \leq \frac{2}{3}(s+x)$, and RSPT has been running J continuously from time s until time t. Then at time t, all jobs smaller than J that are completed by OPT are also completed by RSPT.

Proof. The property holds for $t=s$ by definition of RSPT. For $t>s$, a smaller job that OPT

Figure 1: The algorithm RSPT

RSPT maintains a queue Q of unfinished jobs. A job is put into Q when it arrives. A job is removed from Q when it is completed. For any time t, RSPT deals first with all arrivals of jobs at time t before starting or interrupting any job.
At any time t where either RSPT completes a job, or one or more jobs arrive while RSPT is idle, RSPT starts to run the smallest (remaining) job in Q. If $Q = \emptyset$, RSPT is idle (until the next job arrives).
Furthermore, if at time r a job J is running that started (most recently) at time s and has size x, and if at time r a new job J' arrives with size w, then RSPT interrupts J and starts to run J' if and only if

$$r + w \leq \frac{2}{3}(s + x). \tag{2.1}$$

Otherwise, RSPT continues to run J (and J' is put into Q).

completed and RSPT did not, can thus only have arrived after time s. But then it would have caused an interruption of J before time t. □

R4 Suppose that $s < t \leq \frac{2}{3}(s + x)$, and RSPT has been running J continuously from time s until time t. Then at time t, OPT has completed at most one job that RSPT has not completed.
Proof. By R3, the only jobs that OPT can have already completed at time t that RSPT has not, have size at least x. However, we have $t < 2x$, since $t \geq 2x \Rightarrow \frac{3}{2}(s + x) \geq 2x \Rightarrow s \geq 2x \Rightarrow \frac{1}{3}s \geq \frac{2}{3}x \Rightarrow s \geq \frac{2}{3}(s + x) \Rightarrow t > \frac{2}{3}(s + x)$, a contradiction. This proves this property. □

R5 At any time t, RSPT only interrupts jobs that it cannot finish before time $\frac{3}{2}t$. Hence, RSPT does not interrupt any job with a size of at most half its starting time.
Proof. If there is an interruption at time t, then a job arrived at time t for which $t + w \leq \frac{2}{3}(s + x)$, hence without interruptions J would have finished at time $s + x \geq \frac{3}{2}(t + w) \geq \frac{3}{2}t$.□

3. RSPT-like algorithms

It can be seen that the competitive ratio of RSPT is not better than 3/2: consider a job of size 1 that arrives at time 0, and N jobs of size 0 that arrive at time $2/3 + \varepsilon$. RSPT will run these jobs in order of arrival time and have a total completion time of $N + 1$. However, it is possible to obtain a total completion time of $(2/3 + \varepsilon)(N + 1) + 1$ by running the jobs of size 0 first. By letting N grow without bound, the competitive ratio tends to 3/2 for $\varepsilon \to 0$.

We define an algorithm RSPT(α) as follows: RSPT(α) behaves exactly like RSPT, but (2.1) is replaced by

$$r + w \leq \alpha(s + x).$$

It is possible that RSPT(α) outperforms RSPT for some value of α. However, we show that the improvement could only be very small, if any. To keep the analysis manageable, we analyze only RSPT.

Lemma 3.1 *For all* $0 < \alpha < 1$, $\mathcal{R}(\text{RSPT}(\alpha)) \geq 1.467845$.

Proof. Similarly to above, we have $\mathcal{R}(\text{RSPT}(\alpha)) \geq 1/\alpha$.

Consider the following job sequence. A job J_1 of size 1 arrives at time 0, a job J_2 of size α at time ε, a job J_3 of size 0 at time α (causing an interruption).

For $\varepsilon \to 0$, the optimal cost for this sequence tends to $3\alpha+1$ (using the order J_2, J_3, J_1). However, RSPT(α) pays $5\alpha + 1$.

Now consider the same sequence where after job J_3, at time $\alpha+\varepsilon$ one final job J_4 of size $\alpha(2\alpha)-\alpha$ arrives. For this sequence, the optimal cost tends to $4\alpha^2+2\alpha+1$ whereas RSPT(α) pays $4\alpha^2+5\alpha+1$.

This implies that $\mathcal{R}(\text{RSPT}(\alpha)) \geq \max(1/\alpha, \frac{5\alpha+1}{3\alpha+1}, \frac{10\alpha^2+4\alpha+1}{10\alpha^2+1}) \geq \frac{41+5\sqrt{57}}{31+3\sqrt{57}} = 1.467845.$ □

4. Analysis of RSPT

4.1 Outline

Consider an input sequence σ. To analyze RSPT's competitive ratio on such a sequence, we will work with credits and an invariant.

Each job that arrives receives a certain amount of credit, based on its (estimated) completion time in the optimal schedule and in RSPT's schedule. We will show that each time that RSPT starts a job, we can distribute the credits of the jobs so that a certain invariant holds, using an induction. The calculations of the credits at such a time, and in particular of the estimates of the completion times in the two schedules, will be made under the assumption that no more jobs arrive later.

We will first give the invariant. Then we need to show that the invariant holds at the first time that RSPT starts a job. For the induction step, we need to show that the invariant holds again if RSPT starts a job later,

- *given* that it held after the previous start of a job;
- taking into account any jobs which arrived after that (possibly updating calculations for some jobs that had arrived before); and
- assuming no jobs arrive from now on.

Using this structure, the above-mentioned assumption that no jobs arrive after the current start of a job does not invalidate the proof. There will be one special case where the invariant does not hold again immediately. In that case, we will show the invariant is restored at some later time before the completion of σ. This case will be analyzed in Section 11.

Finally, we need to show that if the invariant holds at the last time that RSPT starts a job, then RSPT maintains a competitive ratio of 3/2. We begin by making some definitions and assumptions.

4.2 Definitions and assumptions

Definition 1 *An* event *is the start of a job by* RSPT.

Definition 2 *An event has the property STATIC if no more jobs arrive after this event.*

At the time of an event, RSPT completes a job, interrupts a job, or is idle.

In our analysis, we will use 'global assumptions' and 'event assumptions'. We show that we can restrict our analysis to certain types of input sequences and schedules and formulate these restrictions as Global assumptions. Then, when analyzing an event (from the remaining set of input sequences), we show in several cases that it is sufficient to consider events with certain properties, and make the corresponding Event assumption. The most important one was already mentioned above:

Event assumption 1 *The current event has the property STATIC.*

The optimal off-line algorithm There can be more than one optimal schedule for a given input σ. For the analysis, we fix some optimal schedule and denote the algorithm that makes that schedule by OPT. We use this schedule in the analysis of every event. Hence, OPT takes into account jobs that have not arrived yet in making its schedule, but OPT does not change its schedule between successive events: the schedule is completely determined at time 0. OPT does not interrupt jobs, because it can simply keep the machine idle instead of starting a certain job and interrupting it later, without affecting the total completion time. We can make the following assumption about RSPT and OPT, because the cost of OPT and RSPT for a sequence is unaffected by changing the order of jobs of the same size in their schedules.

Global assumption 1 *If two or more jobs in σ have the same size,* RSPT *and* OPT *complete them in the same order.*

Definition 3 *An input sequence σ has property* SMALL *if, whenever* RSPT *is running a job of some size x from σ, only jobs smaller than x arrive. (Hence, jobs larger than x only arrive at the completion of a job, or when the machine is idle.)*

Lemma 4.1 *For every input sequence σ, it is possible to modify the arrival times of some jobs such that the resulting sequence σ' has the property* SMALL, *the schedule of* RSPT *for σ' is the same as it is for σ, and* $\text{OPT}(\sigma') \leq \text{OPT}(\sigma)$.

Proof. At any time r that a job J arrives that is at least as large as the job that RSPT is running at that time, we modify σ as follows. If there has been an interval before time r in which RSPT was idle, define u as the end of the last such interval before r; otherwise set $u = 0$. Define r' as the last time in the interval (u, r) that a job larger than J was interrupted or completed. If there is no such time, set $r' = u$. We change the release time of J to r'.

When RSPT is run on the resulting sequence σ', it does not consider running J during the interval $[r', r]$: it is running smaller or equal-sized jobs in that entire interval. (For the equal-sized jobs, see Assumption 1.) Hence the schedule of RSPT for σ' is the same as it is for σ, and $\text{OPT}(\sigma') \leq \text{OPT}(\sigma)$ since the optimal cost for a sequence does not increase if the arrival times decrease or remain the same. □

This Lemma implies that if RSPT maintains a competitive ratio of 3/2 on all the sequences that have property *SMALL*, it maintains that competitive ratio overall. Henceforth, we make the following assumption.

Global assumption 2 *The input sequence σ has property SMALL.*

5. Definitions and notations

After these preliminaries, we are ready to state our main definitions. A job J arrives at its release time $r(J)$ and has size (weight) $w(J)$. The size is the time that J needs to be run without interruptions in order to complete. For a job J_i, we will usually abbreviate $r(J_i)$ as r_i and $w(J_i)$ as w_i, and use analogous notation for jobs J', J^* etc. When RSPT is running a job J, we will be interested in *J-large* unfinished jobs, that are at least as large as J, and *J-small* unfinished jobs, that are smaller, separately. To distinguish between these sets of jobs, the unfinished large jobs will be denoted by $J, J^2, J^3, \ldots$ with sizes x, x_2, x_3 while the small jobs will be denoted by $J_1, J_2, \ldots$ with sizes $w_1, w_2, \ldots$

We let $Q(t)$ denote the queue Q of RSPT at time t.

Definition 4 *A* run-interval *is a half-open interval $I = (s(I), t(I)]$, where* RSPT *starts to run a job (denoted by $J(I)$) at time $s(I)$ and runs it continuously until exactly time $t(I)$. At time $t(I)$, $J(I)$ is either completed or interrupted. We denote the size of $J(I)$ by $x(I)$.*

Definition 5 *For a run-interval I, we denote the set of jobs that arrive during I by $ARRIVE(I) = \{J_1(I), \ldots, J_{k(I)}(I)\}$, We write $r_i(I) = r(J_i(I))$ and $w_i(I) = w(J_i(I))$ for $1 \le i \le k(I)$. The jobs are ordered such that $w_1(I) \le w_2(I) \le \cdots \le w_{k(I)}(I)$. We denote the total size of jobs in $ARRIVE(I)$ by $T(I)$, and write $T_i(I) = \sum_{j=1}^{i} w_i(I)$ for $1 \le i \le k(I)$.*

RSPT will run the jobs in $ARRIVE(I)$ in the order $J_1(I), \ldots, J_{k(I)}(I)$ (using Global assumpion 1 if necessary) and we have $w_{k(I)}(I) < x(I)$ using Global assumption 2. Of course it is possible that $ARRIVE(I) = \emptyset$. In that case I ends with the completion of the job RSPT was running ($t(I) = s(I) + x(I)$).

The input sequence σ may contain jobs of size 0. Such jobs are completed instantly when they start and do not have a run-interval associated with them. Thus we can divide the entire execution of RSPT into run-intervals, completions of 0-sized jobs, and intervals where RSPT is idle. The following lemma follows immediately from the definition of RSPT.

Lemma 5.1 *All jobs in σ arrive either in a run-interval or at the end of an interval in which* RSPT *is idle.*

Lemma 5.2 *Suppose* RSPT *interrupts job $J(I)$ at time t. Then $t = r_1(I)$.*

Proof. We have $t \in \{r_1(I), \ldots, r_{k(I)}(I)\}$. Note that $t < r_1(I)$ is not possible since all jobs in $ARRIVE(I)$ arrive on or before time t. Suppose $t = r_i(I) > r_1(I)$ for some $i > 1$, then $r_i(I) + w_i(I) \le \frac{2}{3}(s(I) + x(I))$. By the ordering of the jobs in $ARRIVE(I)$ we have $w_i(I) \ge w_1(I)$ and thus $r_1(I) + w_1(I) < r_i(I) + w_i(I) \le \frac{2}{3}(s(I) + x(I))$. But then RSPT interrupts $J(I)$ no later than at time $r_1(I)$, so $t \le r_1(I)$, a contradiction. □

Definition 6 *For the jobs in $ARRIVE(I)$, we write*

$$r_i(I) + w_i(I) = \frac{2}{3}(s(I) + x(I)) + \tau_i(I) \qquad i = 1, \ldots, k(I). \tag{5.1}$$

We have $\tau_i(I) > 0$ for $i = 2, \ldots, k(I)$, and $\tau_1(I) > 0$ if $J(I)$ completes at time $t(I)$, $\tau_1(I) \le 0$ if it is interrupted at time $t(I)$.

Definition 7 *We define $f_{\text{OPT}}(I)$ as the index of the job that* OPT *completes first from $ARRIVE(I)$.*

Definition 8 *An interruption by* RSPT *at time t is* slow *if* OPT *starts to run a job from $ARRIVE(I)$ strictly before time t; in this case $f_{\text{OPT}}(I) > 1$ and $J_{f_{\text{OPT}}(I)}(I)$ did not cause an interruption when it arrived.*

We call such an interruption slow, because in this case it could have been better for the total completion time of RSPT if it had interrupted $J(I)$ for one of the earlier jobs in $ARRIVE(I)$ (i. e. faster); now, at time t, RSPT still has to run all the jobs in $ARRIVE(I)$, whereas OPT has already partially completed $J_{f_{\text{OPT}}(I)}(I)$. Note that whether an interruption is slow or fast depends entirely on when OPT runs the jobs in $ARRIVE(I)$. It has nothing to do with RSPT.

We now define some variables that can change over time. We will need their values at time t when we are analyzing an event at time t. They give as it were a snapshot of the current situation.

Definition 9 *If job J has arrived but is not completed at time t, $s_t(J)$ is the (next) time at which* RSPT *will start J, based on the jobs that have arrived until time t. For a job J that is completed at time t, $s_t(J)$ is the last time at which J was started (i.e. the time when it was started and not interrupted anymore). (For a job J that has not arrived yet at time t, $s_t(J)$ is undefined.)*

Lemma 5.3 *For every event and every job J, $s_t(J)$ is at least as high as it was during the previous event.*

Proof. Consider an event at time t and a job J. If J completes before or at time t, then $s_t(J)$ is unchanged since the previous event. Any other job J at time t, for which $s_t(J)$ was already defined during the previous event, is larger than the jobs in $ARRIVE(I)$ by definition of RSPT and by Assumption 2. Therefore J will complete after the jobs in $ARRIVE(I)$, i.e. no earlier than previously calculated. □

By this Lemma, for a job J in $Q(t)$, $s_t(J)$ is the earliest possible time that RSPT will start to run J.

Definition 10 *A job J is* interruptable *at time t, if $s_t(J) < 2w(J)$ and $t \leq \frac{2}{3}(s_t(J) + w(J))$.*

I. e. a job J is interruptable if it is still possible that RSPT will interrupt J after time t (cf. Property R5).

Definition 11 *$BEFORE_t(J)$ is the set of jobs that* RSPT *completes before $s_t(J)$ (based on the jobs that have arrived at or before time t). $b_t(J)$ is the total size of jobs in $BEFORE_t(J)$. $\ell_t(J)$ is the size of the largest job in $BEFORE_t(J)$.*

Clearly, $b_t(J)$ and $\ell_t(J)$ can only increase over time, and $\ell_t(J) \leq b_t(J)$ for all times t and jobs J.

During our analysis, we will maintain an *estimate* on the starting time of each job J in the schedule of OPT, denoted by $s_t^{\mathrm{OPT}}(J)$. We describe later how we make and update these estimates. We will maintain the following as part of our invariant (which will be defined in section 7). Denote the actual optimal completion time of a job J by $\mathrm{OPT}(J)$. Then at the time t of an event,

$$\boxed{\sum_{J:r(J)\leq t} \mathrm{OPT}(J) \geq \sum_{J:r(J)\leq t} (s_t^{\mathrm{OPT}}(J) + w(J))} \tag{5.2}$$

This equation implies that at the end of the sequence, $\mathrm{OPT}(\sigma) \geq \sum_J (s_t^{\mathrm{OPT}}(J) + w(J))$. We will use the following Lemma to calculate initial values of $s_t^{\mathrm{OPT}}(J)$ for arriving jobs in such a way that (5.2) holds.

Lemma 5.4 *For a given time t, denote the most recent arrival time of a job by $t' \leq t$. Denote the job that* OPT *is running at time t' by $\Phi(t')$, and its remaining unprocessed jobs by $\Psi(t')$. The total completion time of* OPT *of the jobs in $\Psi(t')$ is at least the total completion time of these jobs in the schedule where those jobs are run consecutively in order of increasing size after $\Phi(t')$ is completed.*

Proof. The schedule described in the lemma is optimal in case no more jobs arrive after time t (Local assumption 1). If other jobs do arrive after time t, it is possible that another order for the jobs in $\Psi(t')$ is better overall. However, since this order is suboptimal for $\Psi(t')$, we must have that the total completion time of the jobs in $\Psi(t')$ is then not smaller. □

The fact that the optimal schedule is not known during the analysis of an event is also the reason that we check that (5.2) is satisfied instead of checking $\mathrm{OPT}(J) \geq s_t^{\mathrm{OPT}}(J) + w(J)$ for each job J separately.

Definition 12 *$D_t(J) = s_t(J) - s_t^{\mathrm{OPT}}(J)$ is the* delay *of job J at time t.*

6. Credits

The credit of job J at time t is denoted by $K_t(J)$. A job will be assigned an initial credit at the first event on or after its arrival. At the end of each run-interval $I = (s,t]$, each job $J_i(I)$ in $ARRIVE(I)$ receives an initial credit of

$$\frac{1}{2}\left(s^{\text{OPT}}(J_i(I)) + w_i(I)\right) - D(J_i(I)) \qquad i = 1,\ldots,k(I). \tag{6.1}$$

If at time t a (non-zero) interval ends in which RSPT is idle, or $t = 0$, then suppose $Q(t) = \{J_1,\ldots,J_k\}$ where $w_1 \le \cdots \le w_k$. The initial credit of job J_i in $Q(t)$ is then

$$\frac{1}{2}t + \frac{1}{2}\sum_{j=1}^{i} w(J_j) \qquad i = 1,\ldots,k(I). \tag{6.2}$$

This is a special case of (6.1): by Lemma 5.4 and Event assumption 1, OPT will run the jobs in $Q(t)$ in order of increasing size, hence $s_t^{\text{OPT}}(J_i) \ge s_t(J_i)$ for $i = 1,\ldots,k$. Therefore $D(J_i) \le 0$ for $i = 1,\ldots,k$. Moreover, by definition of RSPT we have $s_t(J_i) = t + \sum_{j=1}^{i-1} w_i$ for $i = 1,\ldots,k$.

The idea is that the credit of a job indicates how much its execution can still be postponed by RSPT without violating the competitive ratio of 3/2: if a job has δ credit, it can be postponed by δ time.

For the competitive ratio, it does not matter how much credit each individual job has, and we will often transfer credits between jobs as an aid in the analysis. During the analysis of events, apart from transferring credits between jobs, we will also use the following rules.

Rule C1. If $s_t(J)$ increased by δ since the previous event, then $K(J)$ decreases by δ.

Rule C2. If the estimate $s_t^{\text{OPT}}(J)$ increased by δ since the previous event, then $K(J)$ increases by $\frac{3}{2}\delta$.

$s_t(J)$ cannot decrease by Lemma 5.3. We will only adjust (increase) $s_t^{\text{OPT}}(J)$ in a few special cases, where we can show that (5.2) still holds if we increase $s_t^{\text{OPT}}(J)$. Both rules follow directly from (6.1): it can be seen that if $s_t(J)$ or $s_t^{\text{OPT}}(J)$ increases, J should have received more credit initially.

Theorem 1 *Suppose that after* RSPT *completes any input sequence σ, the total amount of credit in the jobs is nonnegative, and (5.2) holds. Then* RSPT *maintains a competitive ratio of* 3/2.

Proof. We can ignore credit transfers between jobs, since they do not affect the total amount of credit. Then each job has at the end credit of

$$K(J) = \frac{1}{2}(s_t^{\text{OPT}}(J) + w(J)) - (s_t(J) - s_t^{\text{OPT}}(J)),$$

where we use the final (highest) value of $s_t^{\text{OPT}}(J)$ for each job J, and the actual starting time $s_t(J)$ of each job. This follows from (6.1) and the rules for increasing job credits mentioned above. Thus if the total credit is nonnegative, we have

$$\begin{aligned}
\sum_J (s(J) - s_t^{\text{OPT}}(J)) &\le \sum_J \frac{1}{2}(s_t^{\text{OPT}}(J) + w(J)) \\
\Rightarrow \sum_J s(J) &\le \frac{3}{2}\sum_J s_t^{\text{OPT}}(J) + \frac{1}{2}\sum_J w(J) \\
\Rightarrow \text{RSPT}(\sigma) = \sum_J (s(J) + w(J)) &\le \frac{3}{2}\sum_J (s_t^{\text{OPT}}(J) + w(J)) \le \frac{3}{2}\text{OPT}(\sigma).
\end{aligned}$$

This proves the lemma. □

Calculating the initial credit The only unknowns in (6.1) are $s_t^{\mathrm{OPT}}(J_i(I))$ $(i = 1, \ldots, k(I))$. If there is an interruption at time t, Lemma 5.4, together with the job that OPT is running at time t, gives us a schedule for OPT that we can use to calculate $s_t^{\mathrm{OPT}}(J_i(I))$ for all i (if OPT uses a different schedule, its overall cost is not lower, so (5.2) still holds). We also use the following Event assumption.

Event assumption 2 *If the run-interval I ends in a completion, all jobs in ARRIVE(I) arrive no later than the time at which* OPT *completes $J_{f_{\mathrm{OPT}}(I)}(I)$.*

It is known that all jobs in $ARRIVE(I)$ arrive no later than at time $t(I)$. Moreover, by the time OPT completes $J_{f_{\mathrm{OPT}}(I)}(I)$, RSPT will not interrupt $J(I)$ anymore by Property R2. Hence Event assumption 2 does not influence RSPT's decisions or its total completion time. It is only used so that we can apply Lemma 5.4 to calculate lower bounds for the completion times of OPT of these jobs. Since the optimal total cost cannot increase when release times are lower, we have that (5.2) holds.

Note that if we were to modify the sequence σ by actually decreasing release times until Event assumption 2 holds, the optimal schedule for the resulting sequence might be quite different. This is the reason we use this assumption only locally, to get some valid lower bounds on the optimal cost.

Note also that both after a completion and after an interruption, the schedule of OPT is not completely known even with these assumptions, because we do not know which job OPT was running at time t. Therefore we still need to consider several off-line schedules in the following analysis.

6.1 Credit requirements

In this section, we describe three situations in which credit is required, and try to clarify some of the intuition behind the invariant defined in Section 7.

Interruptions Suppose a job J of size x is interrupted at time r_1, because job J_1 arrives, after starting at time s. Then $s < 2x$. J_1 will give away credit to J, J^2, J^3 and J^4 as described in Table 1 in the Appendix, and nothing to any other jobs. We briefly describe the intuition behind this. We have the following properties.

INT1 The amount of lost processing time due to this interruption is $r_1 - s$. This is at most $\frac{2}{3}(s+x) - s = \frac{2}{3}x - \frac{s}{3}$, which is monotonically decreasing in s.

INT2 The size of J_1 is w_1. This is at most $\frac{2}{3}(s+x) - r_1 < \frac{2}{3}(s+x) - s = \frac{2}{3}x - \frac{s}{3}$, which is monotonically decreasing in s.

So, in Table 1, J_1 appears to give away more credit if s is larger, but a) it has more (this follows from (6.1); b) it needs less (we will explain this later); and c) $r_1 - s$ is smaller.

From Table 1 we can also see how much credit is still missing. For instance if $s < x$ and $x < r_1$, then J^2 receives $r_1 - x$ from J_1, but it lost $r_1 - s$ because it now starts $r_1 - s$ time later. We will therefore require that in such a case, J^2 has at least $x - s$ of credit itself, so that it still has nonnegative credit after this interruption. In general, any job that does not get all of its lost credit back according to the table above, must have the remaining credit itself. We will formalize this definition in Section 7.

Completions Suppose a job J completes at time $s + x$. We give the following property without proof.

s	$[0,x]$	$[0,x]$	$(x,\frac{3}{2}x]$	$(x,\frac{3}{2}x]$	$(\frac{3}{2}x,2x)$
r_1	$[0,x]$	$(x,\frac{4}{3}x]$	$(x,\frac{3}{2}x]$	$(\frac{3}{2}x,\frac{5}{3}x]$	$(\frac{3}{2}x,2x)$
To J	r_1-s	r_1+w_1-s	r_1+w_1-s	r_1+w_1-s	r_1+w_1-s
To J^2	0	r_1-x	r_1-s	r_1-s	r_1-s
To J^3	0	0	0	$r_1-\frac{3}{2}x$	r_1-s
To J^4	0	0	0	$r_1-\frac{3}{2}x$	r_1-s
Total	r_1-s	$2r_1+w_1$ $-(s+x)$	$2(r_1-s)+w_1$	$4r_1+w_1$ $-2s-3x$	$4(r_1-s)+w_1$

Table 1: Credit given by J_1 to other jobs

COM1 The jobs in $ARRIVE(I)$ (where $I=(s,s+x]$) need to get at most $\frac{1}{2}(x-b_s(J))$ of credit from J.

By this ("needing" credit) we mean that the amount of credit those jobs receive initially, together with at most $\frac{1}{2}(x-b_s(J))$, is sufficient for these jobs to satisfy the conditions that we will specify in the next section.

Small jobs As long as a job J has not been completed yet, it is possible that smaller jobs than J arrive that are completed before J by RSPT. If OPT completes them after J, then $D_t(J)$ increases.

7. The invariant

From the previous section we see that for a job, sometimes credit is required to pay for interruptions of jobs that are run before it, (e.g. on page 9 below, J^2 pays $x-s$ for an interruption of J), and sometimes to make sure that jobs that arrive during its final run have sufficient credit (COM1). We will make sure that each job has enough credit to pay both for interruptions of jobs before it and for its own completion (i.e. for jobs that arrive during its final run).

For a job J, we define the *interrupt-delay* associated with an interruption as the amount of increase of $D_t(J)$ compared to the previous event. This amount is at most $t-s$ at the end of a run-interval $(s,t]$. (It is less for a job J if $s_{\mathrm{OPT}}(J)$ also increases).

Credit can also be required because the situation marked "Small jobs" in Section 6.1 occurs. The *small job-delay* of J associated with an event at time t is the total size of jobs smaller than J in $ARRIVE(I)$ that are completed before J by RSPT and after J by OPT.

Interrupt-credit and completion-credit When considering the credit of a job J, we will make a distinction between *interrupt-credit* $K_{INT}(J)$, which is used to pay for interrupt-delays whenever they occur, and *completion-credit* $K_{COM}(J)$, which is used to "pay for the completion" of J (see above). (We do not reserve credit for small job-delays since they will be paid for by the small jobs that cause it.) Accordingly, we now make two important definitions.

Definition 13 *$N_{INT}(J,t)$ is the maximum amount of credit that J may need to pay for (all the) interruptions of jobs that* RSPT *completes before it, as it is known (for the on-line algorithm* RSPT*) at time t.*

I.e. this is the amount of credit needed if all the jobs before J *that have arrived before time t* are interrupted as often as possible. If other jobs arrive later, $N_{INT}(J,t)$ can change. For any job J that is already completed at time t by RSPT, $N_{INT}(J,t)=0$.

Definition 14 *$N_{COM}(J,t)$ is the maximum amount of credit that J may need to pay for its own completion, as known (for the on-line algorithm* RSPT*) at time t.*

This amount can also change over time, namely if $s(J)$ increases. Again, if J is completed before time t, then $N_{COM}(J,t)=0$.

Consider an event at time t. Suppose $Q(t)=\{J_1,\ldots,J_k\}$, where $w_1 \le w_2 \le \cdots \le w_k$. We write $s_i=s_t(J_i)=t+\sum_{j=1}^{i-1} w_j$. From Table 1 it can be seen that

$$N_{INT}(J_i,t)=\max(0,w_{i-1}-s_{i-1})+\max(0,\frac{3}{2}w_{i-2}-s_{i-2})+\max(0,2w_{i-4}-t), \tag{7.1}$$

where each maximum only appears if the corresponding job exists. For the third maximum in this equation, note that the total interrupt-delay of J_i caused by interruptions of the jobs $J_1,\ldots,J_{i-4}$ is at most $2w_{i-4}-t$ after time t, since RSPT starts to run J_1 at time t and does not interrupt any of the jobs $J_1,\ldots,J_{i-4}$ after time $2w_{i-4}$ by Property 4.

The following table gives upper bounds for $N_{INT}(J_i,t)$ in all possible cases.

$t\le 2w_{i-4}$	$s_{i-2}\le \frac{3}{2}w_{i-2}$	$s_{i-1}\le w_{i-1}$	$N_{INT}(J,t)$
•	•	•	$\max(0,\ w_{i-1}+\frac{1}{2}w_{i-2}-2s_{i-4}-t)$
•	•		$\max(0,\ \frac{3}{2}w_{i-2}-s_{i-4}-t)$
•		•	$\max(0,\ w_{i-1}-s_{i-3}-t)$
•			$\max(0,\ 2w_{i-4}-t)$
	•	•	$\max(0,\ w_{i-1}+\frac{1}{2}w_{i-2}-2s_{i-2})$
	•		$\max(0,\ \frac{3}{2}w_{i-2}-s_{i-2})$
		•	$\max(0,\ w_{i-1}-s_{i-1})$
			0

Note that in all cases

$$N_{INT}(J_i,t)\le \max(0,w_{i-1}+\frac{1}{2}w_{i-2}+\frac{1}{2}w_{i-4}-t). \tag{7.2}$$

Using property COM1, we have

$$N_{COM}(J_i,t)=\max\left(0,\frac{1}{2}(w_i-b_t(J_i))\right). \tag{7.3}$$

$N_{COM}(J_i,t)$ can only decrease over time (since s_i and $b_t(J_i)$ only increase).
For all jobs J that have arrived at time t, we wish to maintain

$$\boxed{K_t(J)\ge N_{COM}(J,t)+N_{INT}(J,t).} \tag{7.4}$$

This means that each job will be able to pay for the specified parts of its interrupt-delay and for its completion. I. e. the total credit of each job will be sufficient to pay for both these things.

Invariant We now define our invariant, that will hold at specific times t in the execution, and in particular when a sequence is completed:

Invariant: At time t, for all jobs that have arrived, (7.4) holds; furthermore, (5.2) holds.

Theorem 2 $\mathcal{R}(\text{RSPT})\le 3/2$.

Proof outline. The proof consists of a case analysis, which makes up the rest of this paper. In the rest of this section we show that (5.2) can be maintained and that (7.4) holds for large jobs. In section 9 and beyond, we consider all possible interruptions and completions. "All possible" refers to both the times at which these events occur, and the possible schedules of the off-line algorithm.

For every possible event, we will give a time at which the above invariant holds again, assuming that it held after the previous event. This will be no later than at the completion of the last job in σ. At that time, the invariant implies that all completed jobs have nonnegative credit, since for completed jobs we have $N_{COM}(J,t) = N_{INT}(J,t) = 0$. Also, (5.2) holds. We can then apply Theorem 1. $\square$

For almost all events, it will be the case that the invariant holds again immediately after the current event. However, there is one event for which it takes slightly longer: this is a slow interruption of a job J, where $s(J) \leq \frac{2}{3}w(J)$. If such an event occurs at some time t, we will show that the invariant is restored no later than when RSPT has completed the second-smallest job in $ARRIVE(I)$.

In order to ensure that the invariant holds again after an event, we will often transfer credits between jobs. Also, we will use the credit that some jobs must have because the invariant was true previously, to pay for their interrupt-delay or for their completion. We need to take into account that $N_{INT}(J,t)$, $s_t^{\mathrm{OPT}}(J)$ etc. of some jobs that arrived before or at the previous event can change as a result of the arrival of new jobs, compared to the calculations in that event (that were made under the assumption that STATIC held).

By the discussion following Theorem 1, (5.2) holds at each event if it held at the previous event and if $s_t^{\mathrm{OPT}}(J)$ is not changed for any job J that arrived at or before the previous event. We also have the following lemma.

Lemma 7.1 *Suppose $Q(t) = \{J_1, \ldots, J_k\}$, where $w_1 \leq w_2 \leq \cdots \leq w_k$, and* RSPT *starts J_1 at time t. A job J_i satisfies (7.4) in any of the following situations.*

1. $K_t(J_i) \geq \max(0, \frac{1}{2}\sum_{j=1}^{i} w_j - t)$.
2. $K_t(J_i) \geq \frac{1}{2}(w_i - w_{i-1}) + \frac{1}{2}\sum_{j=1}^{i-2} w_j$ *and* $t \geq w_{i-1}$
3. $K_t(J_i) \geq \frac{1}{2}(w_i - w_{i-1}) + \frac{1}{2}\sum_{j=1}^{i-2} w_j + (w_{i-1} - t)$ *and* $t < w_{i-1}$
4. $K_{t'}(J_i) \geq \frac{1}{2}(w_i - w_{i-1})$ *and J_i starts to run at time t'*

Proof. Note first of all that $w_{i-1} \leq \ell_t(J_i)$ because RSPT runs the jobs in order of increasing size.

1. We have $\frac{1}{2}\sum_{j=1}^{i} w_j - t = \frac{1}{2}(w_i - w_{i-1}) + w_{i-1} + \frac{1}{2}\sum_{j=1}^{i-2} w_j - t \geq \frac{1}{2}(w_i - \ell_t(J)) + N_{INT}(J_i,t) \geq N_{COM}(J_i,t) + N_{INT}(J_i,t)$.

2. Here we have $N_{INT}(J_i,t) \leq \frac{1}{2}(w_{i-2} + w_{i-4})$, since $t \geq w_{i-1} \geq w_{i-2} \geq w_{i-4}$. Hence $K_t(J_i) \geq \frac{1}{2}(w_i - w_{i-1}) + \frac{1}{2}\sum_{j=1}^{i-2} w_j \geq N_{COM}(J_i,t) + N_{INT}(J_i,t)$.

3. Now $N_{INT}(J_i,t) \leq \frac{1}{2}(w_{i-2} + w_{i-4}) + (w_{i-1} - t)$ and we are done similarly.

4. This can only happen if $J_1, \ldots, J_{i-1}$ are completed, because RSPT runs jobs in order of size. We have $N_{INT}(J_i,t') = 0$ by (7.1): note that J_i is job J_1 in (7.1) at time t', since it is the smallest job available at time t' by definition of RSPT. Furthermore, since $w_{i-1} \leq \ell_{t'}(J_i)$, $K_{t'}(J_i) \geq \frac{1}{2}(w_i - \ell_{t'}(J_i)) \geq N_{COM}(J_i,t)$. $\square$

Corollary 7.1 *Suppose* RSPT *starts to run jobs at time t, where $t = 0$ or t is the end of a (nonzero) interval in which* RSPT *was idle. Then (7.4) and (5.2) hold for the jobs that arrive at time t.*

Proof. This follows directly from (6.2) and Lemma 7.1, Case 1. $\square$

We can apply this corollary at the arrival time of the first job in σ, and anytime after RSPT has been idle.

Notation	Definition	Long notation
t	time of the current event	
s	start of the most recent run-interval $I = (s,t]$	
J	job that RSPT was running in $I = (s,t]$	$J(I)$
$ARRIVE$	jobs that arrive in I	$ARRIVE(I)$
J_1	smallest job that arrives in I	$J_t(I)$
r_1	its arrival time	$r_1(I)$
w_1	its size	$w_1(I)$
f	index of job in $ARRIVE$ that OPT runs first	$f_{\text{OPT}}(I)$

Table 2: Notations

Proof overview We will show that for all events, it is possible to restore the invariant before the sequence completes. This proves that RSPT maintains a competitive ratio of 3/2. We divide the analysis into the following cases.

1. An interruption of a job J (Lemmas 9.1, 9.2, 9.3) in all but one case, Case 3 below
2. Completion of a job J (Lemmas 10.1, 10.2, 10.3, 10.4)
3. A slow interruption of a job J of size x in the case that RSPT started it before time $2x/3$ (Section 11), and OPT does not run any J-large jobs before $ARRIVE(I)$.

8. Analysis of an event

As described in the previous section, for the analysis of RSPT we need to analyze every possible event that can occur during its execution, i. e. show that the invariant holds after the event, if it holds after the previous event.

For each event, we will only be interested in the credits of jobs at the time of the current event, denoted by t. Hence, we will drop the subscript t and write $K(J_i)$ for each job J_i. Furthermore, the job that was interrupted or completed at the time of the event will be denoted by J, and the set of smaller jobs that arrived during the most recent run of J will be denoted by $ARRIVE = \{J_1, \ldots, J_k\}$. The most recent starting time of J will be denoted by s. We will call J-large jobs *large*, and others *small*. Remember that f_{OPT} is the index of the job in $ARRIVE$ that OPT completes first. Our notation is summarized in Table 2.

Lemma 8.1 *After* OPT *completes J_f, then if STATIC holds,* OPT *does not run any J-large job until all jobs in $ARRIVE$ are completed.*

Proof. This is a direct consequence of Lemma 5.4 and Event assumptions 1 and 2: after J_f, OPT will complete first the remaining jobs in $ARRIVE$ in order of increasing size, and then the remaining large jobs - as long as no new jobs arrive. □

Using this lemma, we only need to consider when OPT is running large jobs in relation to the set $ARRIVE$ (as a whole), as long as we assume STATIC holds. We have the following lemma.

Lemma 8.2 *Suppose that there is a job L that* RSPT *completes on or before time t, whereas* OPT *completes it after a job J_i in $ARRIVE$. Then there is $\frac{3}{2}w_i$ of credit available that can be freely assigned to uncompleted jobs.*

Proof. In the analyses of previous events, $ARRIVE$ was not taken into account when considering the credit of job L. Compared to that analysis, we have that $s^{\text{OPT}}(L)$ increases by at least w_i. Rule C2 implies that L now has $\frac{3}{2}w_i$ more credit than calculated at the previous events. But

$N_{COM}(L,t) = N_{INT}(L,t) = 0$ since L is completed. Hence we can give $\frac{3}{2}w_i$ to other jobs, while L still satisfies (7.4). $\square$

Suppose that OPT runs at least one large job J' before *ARRIVE*. In this case we will not use any lower bound on the optimal starting time of J' (except that it is at least 0). This enables us to make the following assumption.

Event assumption 3 *If* OPT *runs at least one large job before ARRIVE,* OPT *runs J before ARRIVE.*

Suppose OPT runs $J' \neq J$ before *ARRIVE*, but not J. By Assumption 1, J' is then larger than J. This can only increase the optimal starting time of jobs in *ARRIVE* compared to the situation where OPT does run J before *ARRIVE* (J' could not start earlier than J because we will only use that J starts on or after time 0.)

If OPT runs exactly one large job $J' \neq J$ before *ARRIVE*, then $s^{\text{OPT}}(J)$ increases by T by the arrival of the jobs in *ARRIVE*, and therefore $K(J)$ increases by $\frac{3}{2}T$ by Rule C2, that we do not take into account if we assume OPT runs J before *ARRIVE*. On the other hand, RSPT starts to run J' an additional T later by the arrival of *ARRIVE* (aside from the interrupt-delay, in the case that J_1 from *ARRIVE* causes a restart), hence the credit of J' decreases by T, in contrast to the case where OPT runs J' after *ARRIVE* as well. In addition, consider $N_{INT}(J', r_1)$. Since $x' > x$, RSPT runs (after *ARRIVE*) first J and later J'. Hence $N_{INT}(J', r_1) \le N_{INT}(J', s) + \frac{1}{2}T$. J' would receive the additional $\frac{1}{2}T$ automatically if OPT completed J' after *ARRIVE*, because then $s_t^{\text{OPT}}(J')$ would increase by T as well.

Putting these facts together, it is sufficient to analyze the case where OPT runs J before *ARRIVE* (not using any lower bounds on J's starting time), switch J and J', and transfer an additional $\frac{3}{2}T$ worth of credit (that we do not take into account in this analysis) from J to J'.

The case where OPT runs two jobs before *ARRIVE*, none of which is J, can be treated similarly. We will therefore always make Event assumption 3.

Lemma 8.3 *The condition (5.2) can be maintained throughout the execution of σ.*

Proof. Using Lemma 5.4 we can calculate valid initial values $s_t^{\text{OPT}}(J)$ for any arriving job J. The only modification we will make in later events, is that for any $J(I)$-large job J' that OPT completes after *ARRIVE*(I), we increase $s_t^{\text{OPT}}(J')$ by $T_k(I)$. (Here we will use Event assumption 3.) Since the set *ARRIVE*(I) was not taken into consideration when $s_t^{\text{OPT}}(J')$ was originally determined, the resulting bound is still valid, so (5.2) holds. $\square$

Lemma 8.4 *The large jobs that* OPT *and* RSPT *complete after ARRIVE besides J satisfy (7.4).*

Proof. Consider such a job J^i. It receives extra credit of $\frac{1}{2}T$ by Rules C1 and C2, since both $s(J^i)$ and $s^{\text{OPT}}(J^i)$ increase by T. It possibly loses some credit if there was an interruption.

To see that $J^i \neq J$ (still) satisfies (7.4), note that $N_{COM}(J^i, t) \le N_{COM}(J^i, s)$ for all $s \ge t$. Regarding $N_{INT}(J^i, t)$, we use the bound (7.1). This bound refers to three different jobs. For J^i, some of these jobs may be in *ARRIVE* and some of them can be the (previously arrived) jobs $J, J^2, \ldots, J^{i-1}$.

The jobs in *ARRIVE* can only occur as jobs "J_{i-2}" and "J_{i-4}", since after *ARRIVE* at least J is also completed before J^i. Hence the gained credit of $\frac{1}{2}T$ is sufficient to cover this.

For the remaining jobs, if there is no interruption, the maximums in (7.1) can only decrease since $s(J^i)$ increases by T_k. In this case we are done. If there is an interruption, we either have that this part of $N_{INT}(J^i, t)$ decreases by at least the amount of credit that is lost, or J^i receives the remaining lost credit back from J_1. This follows from Table 1 and (7.1).

As an example, J is interrupted at time $2x/3$ after starting at time 0, for a job J_1 of size 0. Job J^2 loses $2x/3$ of credit because of this ($2x/3$ processing time is wasted). But $N_{INT}(J^2, 2x/3) = x/3$ and $N_{INT}(J^2, 0) = x$, so J^2 still satisfies (7.4).

As a second example, J is interrupted at time $10x/9$ after starting at time $2x/3$, for a job J_1 of size 0. Job J^2 loses $4x/9$ of credit because of this ($4x/9$ processing time is wasted). It receives $x/9$ of credit from J_1 due to Table 1. Also, $N_{INT}(J^2, 2x/3) = x/3$ and $N_{INT}(J^2, 10x/9) = 0$, a decrease of $x/3$. Since $4x/9 = x/9 + x/3$, J^2 still satisfies (7.4).

It can be seen that Table 1 and (7.1) complement each other in all cases: the credit from J_1 together with the decrease in (7.4) is sufficient for any large job J^i to keep satisfying (7.4). □

By the results in this section, in the remainder of the paper it is sufficient to check (or ensure) that J and the jobs in $ARRIVE$ satisfy (7.4). We already have that (5.2) holds and that (7.4) holds for the J-large jobs besides J. All other jobs have either already been completed by RSPT, or have not arrived yet.

9. Interruptions

Consider an interruption at time r_1 of a job J that started at time s. The interrupt-delay associated with this interruption is $r_1 - s$ for all jobs that were already in Q at time s. The small job-delay of this event of the jobs that OPT completes before $ARRIVE$, and RSPT does not, is $T = \sum_{i=1}^k w_i$.

The last event before this one was the start of J, at time s. No jobs were completed since then by RSPT.

We divide the interruptions into three types, based on when OPT runs large jobs relative to the set $ARRIVE$:

Large jobs before $ARRIVE$ by OPT	0	1	2
Lemma	9.1	9.2	9.3

We need to distinguish between the cases $f = 1$ and $f > 1$. First suppose $f = 1$, i. e. OPT runs the jobs from $ARRIVE$ in the same order as RSPT. The credit reassignments in this case take place in four steps.

1. On arrival of J_1, the jobs $J, J_2, \ldots, J_k$ are shifted. However, for the moment we keep the order of those jobs the same (as in the situation where J_1 does not arrive). We reassign credit from J_1 to J so that its credit remains constant. (Some jobs can have negative credit in this step.)
2. We reorder the jobs so that the order is now $J_1, \ldots, J_k, J$. However, the credits of the jobs stay "in the same place", so that e. g. J_2 now has the credit that J had in Step 1.
3. We calculate the extra credit that this reordering generates. If a completion is now δ time earlier, there is δ more credit available by Rule 1.
4. We reassign credits to make sure all waiting jobs satisfy (7.4). (This step is not always required.)

A graphical representation of the first three steps of this procedure can be seen in Figure 2.

In case $f > 1$, we proceed similarly. However, in the first step we consider different orders for the jobs (which will be described at the time), instead of the order described above. We then put the jobs in the order they will be executed by RSPT in Step 2 and continue as above.

Lemma 9.1 *If* RSPT *interrupts a job J at time r_1, and* OPT *runs no large jobs before $ARRIVE$, and $s \geq 2x/3$, and STATIC holds, and the invariant held at time s, then it holds at time r_1.*

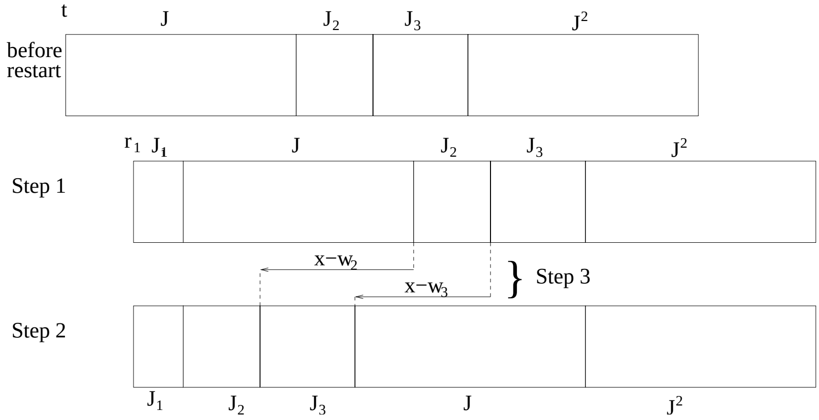

Figure 2: Credit transfers

Proof. Case 1. $f = 1$. Only in this very first case will we not use the procedure outlined above, and instead calculate the credits for the proper order directly. We begin by showing that J_1 still satisfies (7.4) after giving credit to J and J-large jobs as described in Table 1. We have initially $K(J_1) = \frac{1}{2}(r_1 + w_1)$. Also, $N_{INT}(J_1, r_1) = 0$ and $N_{COM}(J_1, r_1) \le \frac{1}{2}w_1$ by definition.

Suppose $s < r_1 \le x$. Since $s \ge 2x/3$, we have $r_1 \le \frac{2}{3}(s+x) \le \frac{5}{3}s < 2s$. Then $\frac{1}{2}(r_1+w_1)-(r_1-s) = \frac{1}{2}w_1 + s - \frac{1}{2}r_1 \ge \frac{1}{2}w_1$.

Suppose $s \le x < r_1$. Since $r_1 + w_1 \le \frac{2}{3}(s + x)$, then using Table 1, we have that J_1 is left with credit of $\frac{1}{2}(r_1+w_1)-(2r_1+w_1-s-x) = s+x-\frac{3}{2}r_1-\frac{1}{2}w_1 \ge s+x-\frac{3}{2}(\frac{2}{3}(s+x)-w_1)-\frac{1}{2}w_1 = w_1$.

Suppose $x < s < r_1 \le \frac{3}{2}x$. J_1 is left with $\frac{1}{2}(r_1 + w_1) - 2(r_1 - s) - w_1 = 2s - \frac{3}{2}r_1 - \frac{1}{2}w_1 \ge w_1$, since $r_1 + w_1 \le \frac{4}{3}s$.

Suppose $x < s \le \frac{3}{2}x < r_1$. We take $2(r_1 - s) + w_1 + 2(r_1 - \frac{3}{2}x)$ out of J_1. J_1 still has $\frac{1}{2}(r_1+w_1)-4r_1+2s+3x-w_1 \ge -\frac{7}{2}\cdot\frac{2}{3}(s+x)+3w_1+2s+3x = -\frac{1}{3}s+\frac{2}{3}x+3w_1 > 3w_1$ of credit.

Suppose $\frac{3}{2}x < s$. We take $4(r_1 - s) + w_1$ out of J_1, leaving it with $\frac{1}{2}(r_1 + w_1) - 4r_1 + 4s - w_1 \ge 4s - \frac{7}{3}(s + x) + 3w_1 = \frac{5}{3}s - \frac{7}{3}x + 3w_1 > 3w_1$.

In all cases, $K(J_1)$ satisfies (7.4). For $2 \le i \le k$, we have $K(J_i) = \frac{1}{2}(r_1 + T_i)$ so that we are done by Lemma 7.1, Case 1. For J, we have that $s^{\mathrm{OPT}}(J)$ increases by $\frac{1}{2}T$ (see proof of Lemma 8.3). Note that $\frac{1}{2}T \ge w_k + \frac{1}{2}w_{k-1} + \frac{1}{2}w_{k-3} - r_1 \ge N_{INT}(J, r_1)$, since $r_1 > s \ge \frac{2}{3}x > \frac{2}{3}w_k$. We also have $N_{COM}(J, r_1) \le N_{COM}(J, s)$, so J still satisfies (7.4).

Case 2. $f > 1$. We define $D_1 = r_1 - r_f > 0$ (if $r_1 = r_f$, then by Lemma 5.4 OPT runs the jobs in order of increasing size after time r_1 and hence $f = 1$) and $D_2 = (r_f + w_f) - (r_1 + w_1) > 0$ (if $D_2 = 0$, either J_f would have already caused a restart before time r_1, or J_1 would not have caused a restart).

As in Case 1, we begin by checking the credit of J_1. In this case, initially, $K(J_1) = \frac{3}{2}(r_f + w_f) + \frac{1}{2}w_1 - r_1$ since $s^{\mathrm{OPT}}(J_1) = r_f + w_f$ and $s(J_1) = r_1$. We take an extra w_1 out of $K(J_1)$ for J_2.

Suppose $r_1 \le x$. Then $\frac{3}{2}(r_f + w_f) - \frac{1}{2}w_1 - 2r_1 + s = \frac{3}{2}D_2 + w_1 - \frac{1}{2}r_1 + s > w_1$. We have used $r_1 \le \frac{2}{3}(s + x) < 2s$ which holds since $s \ge \frac{2}{3}x$.

Suppose $s \le x < r_1$. In this case we take $2r_1+2w_1-s-x$ of credit out of J_1. Since $r_1+w_1 \le \frac{2}{3}(s+x)$, we have that J_1 is left with credit of $\frac{3}{2}(r_f+w_f-w_1)-3r_1+s+x = \frac{3}{2}D_2-\frac{3}{2}r_1+s+x \ge \frac{3}{2}D_2+\frac{3}{2}w_1$.

Suppose $x < s < r_1 \le \frac{3}{2}x$. We take $2(r_1 - s) + 2w_1$ of credit out of J_1. Since $r_1 + w_1 \le \frac{4}{3}s$ in this case, again J_1 is left with at least $\frac{3}{2}w_1 + \frac{3}{2}D_2$.

Suppose $x < s \le \frac{3}{2}x < r_1$. We take $2(r_1 - s) + 2w_1 + 2(r_1 - \frac{3}{2}x)$ out of J_1. J_1 still has

	1	2	3	final
J_1	$\frac{1}{2}w_1$	J_1	0	$\frac{1}{2}w_1$
J_f	$\frac{3}{2}r_f + \frac{1}{2}w_f - r_1$	J_2	0	$\frac{1}{2}w_f$
$i = 2, \dots, f-1:$				
J_i	$\frac{1}{2}(r_f + w_f + T_i) - D_1$	J_{i+1}	$w_f - w_i$	$\frac{r_f+w_f+T_{i-1}-w_i}{2} + w_f - D_1$
$i = f+1, \dots, k:$				
J_i	$\frac{1}{2}(r_f + T_i) - D_1$	J_i	0	$\frac{1}{2}(r_f + T_i) - D_1$
J	$N_{COM}(J,s) + \frac{1}{2}T$	J	0	$N_{COM}(J,r_1) + N_{INT}(J,r_1)$

Table 3: Credits in Lemma 9.1

$\frac{3}{2}(r_f + w_f - w_1) - 5r_1 + 2s + 3x \geq 3x + 2s - \frac{7}{2}r_1 + \frac{3}{2}D_2 \geq x - \frac{1}{2}r_1 + \frac{3}{2}D_2 + 3w_1 \geq 3\frac{1}{2}w_1 + \frac{3}{2}D_2$ of credit, using $3(r_1 + w_1) \leq 2(s + x)$.

Suppose $\frac{3}{2}x < s$. We take $4(r_1 - s) + 2w_1$ out of J_1, leaving it with $\frac{3}{2}(r_f + w_f - w_1) - 5r_1 + 4s \geq 4s - 3\frac{1}{2}r_1 + \frac{3}{2}D_2 \geq 4s - \frac{7}{2}(\frac{2}{3}(s + x) - w_1) + \frac{3}{2}D_2 \geq \frac{5}{3}s - \frac{7}{3}x + \frac{3}{2}D_2 + \frac{7}{2}w_1 > \frac{7}{2}w_1 + \frac{3}{2}D_2$ since $\frac{5}{3}s \geq \frac{5}{2}x > \frac{7}{3}x$.

In all cases, (7.4) holds for J_1. For the other jobs, we use Table 3.

In this table, the column marked 1 contains the credits of the jobs assuming the order $J_1, J_f, J_2, \dots, J_{f-1}, J_{f+1}, \dots, J_k, J$, and after J_1 has given away credit to J-large jobs as described in Table 1. Column 2 shows the new order of the jobs. The credits stay in the same place, hence e. g. J_2 now has a credit of $\frac{3}{2}r_f + \frac{1}{2}w_f - r_1$. Column 3 shows how much credit is gained by the reordering of the jobs. Column 4 shows credit transfers; in this case, $\frac{1}{2}r_f$ is transferred from J to J_2. The last column contains the final credit of each job. The numbers above the columns refer to the steps in the procedure described at the start of this section.

We now show that the credit in the last column is sufficient so that all jobs satisfy (7.4). We have $w_1 = (r_1 + w_1) - r_1 \leq \frac{3}{2}(s + x) - s = \frac{3}{2}x - \frac{1}{3}s \leq \frac{4}{9}x \leq \frac{4}{9}x$ and $r_1 > \frac{2}{3}x$, so J_1 cannot start before time w_1. Hence $N_{INT}(J_2, r_1) = 0$. Also $N_{COM}(J_2, r_1) \leq \frac{1}{2}(w_2 - w_1)$. As in Case 1, we use that $s^{\mathrm{OPT}}(J)$ has increased by $\frac{1}{2}T$.

Suppose $r_1 + w_1 \geq w_k$. Then $N_{INT}(J, r_1) \leq \frac{1}{2}(w_{k-1} + w_{k-3})$. Therefore J can give $\frac{1}{2}w_k$ of credit to J_2, and still satisfy (7.4). Then we find $K(J_2) \geq D_2 + \frac{1}{2}r_f$. We are done if $r_f \geq w_f$; otherwise, since $r_f \geq \frac{2}{3}x$, $D_2 = (r_f + w_f) - (r_1 + w_1) \geq r_f + \frac{2}{3}x - \frac{2}{3}(s + x) \geq \frac{1}{3}r_f$. Thus $K(J_2) = D_2 + \frac{1}{2}r_f \geq \frac{5}{6}r_f \geq \frac{5}{9}x > \frac{1}{2}w_2$.

Suppose $r_1 + w_1 < w_k < x$. Then $\frac{3}{2}r_f - r_1 \geq 0$, so $K(J_2) \geq \frac{1}{2}w_f \geq \frac{1}{2}w_2$.

For jobs $J_3, \dots, J_f$, we have $D_1 = w_f - w_1 - D_2 < w_f$ so we are done by Lemma 7.1, Case 2 if $r_1 \geq w_{i-1}$. If $r_1 < w_{i-1}$, then $\frac{r_f - w_{i-1}}{2} + w_f - D_1 \geq \frac{3}{2}r_f + \frac{1}{2}w_f - r_1 \geq w_{i-1} - r_1$ since $r_f > \frac{2}{3}x$. We are done (Lemma 7.1, Case 3).

For jobs $J_{f+1}, \dots, J_k$, we are done if $r_1 \geq w_{i-1}$ since then $K(J_i) \geq \frac{1}{2}(r_f + T_i) - D_1 \geq \frac{1}{2}(w_i - w_{i-1} + T_{i-2})$ (using $D_1 < w_f$). And if $r_1 < w_{i-1}$, then $\frac{1}{2}T_i - D_1 \geq \frac{1}{2}(w_i - w_{i-1} + T_{i-2}) + (w_{i-1} - r_1)$. □

Lemma 9.2 *If* RSPT *interrupts a job J at time r_1, and* OPT *runs one large job before ARRIVE, and STATIC holds, and the invariant held at time s, then it holds at time r_1.*

Proof. We distinguish between three cases depending on s and f. We ignore that OPT has to run the jobs in $BEFORE_s(J)$ too at some point; this can only decrease the optimal cost on the other jobs.

Case 1. $s \le x/2$. We have $r_1 + w_1 \le \frac{2}{3}(s+x) \le x$, so $f = 1$ by Lemma 5.4. Also $b_s(J) \le s < x$.

We have $s^{\mathrm{OPT}}(J_1) \ge x$ and $D(J_1) \le r_1 - x$, so $K(J_1) \ge \frac{3}{2}x + \frac{1}{2}w_1 - r_1$. On arrival of J_1, job J is in Step 1 shifted by RSPT by $r_1 + w_1 - s$ time. We take $r_1 + 2w_1 - s$ of credit out of J_1 for J, so that J_1 is left with $\frac{3}{2}x + \frac{1}{2}w_1 - 2(r_1 + w_1) + s \ge \frac{1}{2}w_1 + \frac{1}{6}x - \frac{1}{3}s \ge \frac{1}{2}w_1$, so J_1 satisfies (7.4).

Credit transfers For the Step 1-column, we use (6.1).

	1	**2**	**3**	final
J_1	$\frac{1}{2}w_1$	J_1	0	$\frac{1}{2}w_1$
J	$N_{COM}(J,s) + w_1$	J_2	0	$N_{COM}(J,s) + w_1$
J_2	$\frac{1}{2}(x + T_2) - r_1$	J_3	$x - w_2$	$\frac{1}{2}(x + T_1 - w_2) + (x - r_1)$
	$\vdots$	$\vdots$		
J_k	$\frac{1}{2}(x + T_k) - r_1$	J	$x - w_k$	$\frac{1}{2}(x + T_{k-1} - w_k) + (x - r_1)$

J_1 satisfies (7.4) by Lemma 7.1, Case 1; J_2 as well, using that $x > w_2$ and $b_s(J) \le b_{r_1}(J_2)$; the other jobs too by Lemma 7.1, Case 3.

Case 2. $s > x/2$ and $f = 1$. Since $f = 1$, we have in Step 1 of our calculations that $D(J_i) \le \min(x, r_1)$ for $2 \le i \le k$: after time $r_1 + w_1$, RSPT first runs J (of size x) whereas OPT runs $J_2, \ldots, J_k$ immediately after J and J_1, at most $\min(x, r_1)$ time earlier. We also have $w_1 \le \frac{2}{3}(s+x) - r_1 \le \frac{2}{3}x - \frac{1}{3}s \le \frac{1}{2}x < s$, so $N_{INT}(J_2, r_1) = 0$.

If $r_1 \le x$, we are done exactly as in Case 1.

Now suppose $r_1 > x$. Since in this case $K(J_1) = \frac{1}{2}(r_1 + w_1)$ initially, we have already checked in the proof of Lemma 9.1, Case 1, that J_1 satisfies (7.4) after giving away credit as in Table 1.

Credit transfers ($r_1 > x$) See the following table.

	1	**2**	**3**	final
J_1	$\frac{1}{2}w_1$	J_1	0	$\frac{1}{2}w_1$
J	$N_{COM}(J,s)$	J_2	0	$N_{COM}(J,s)$
J_2	$\frac{1}{2}(x + T_2) - x$	J_3	$x - w_2$	$\frac{1}{2}(x + T_1 - w_2)$
	$\vdots$	$\vdots$		
J_k	$\frac{1}{2}(x + T_k) - x$	J	$x - w_k$	$\frac{1}{2}(x + T_{k-1} - w_k)$

Since $r_1 > x > w_i$ for $1 \le i \le k$, it follows immediately from Lemma 7.1, Case 2, that all jobs satisfy (7.4).

Case 3. $s > x/2$ and $f > 1$. Write $\tilde{r} = \max(x, r_f)$. Suppose $r_1 \le x$. Then at time $\tilde{r} \ge r_1$, OPT will run the remaining jobs in order of increasing size by Lemma 5.4. But then $f = 1$, a contradiction. Therefore $r_1 > x$ and $D(J_1) \le r_1 - (\tilde{r} + w_f) < 0$ using Property R3.

There can be only two jobs in *ARRIVE* that are interruptable, because the first one must have size at least $p = (r_1 + w_1)/2 > x/2$ by Property R5, the second one size at least $p' = (r_1 + w_1 + p)/2 > \frac{3}{4}x$, so the third should have size $(r_1 + w_1 + p + p')/2 > x$, which is not possible for a job in *ARRIVE*. It follows similarly that if J_i is interruptable, then from *ARRIVE* only J_{i-1} or J_{i+1} can be interruptable as well. Moreover, since all jobs in *ARRIVE* start after time $r_1 > x$, for every job J_i in *ARRIVE* we have $w_{i-1} - s_{i-1} < 0$. Therefore if a job J_i is interruptable, then $N_{INT}(J_i, r_1) = 0$. Moreover, in this case $T_{i-1} < w_i$, because all jobs in *ARRIVE* start after time

	J_1	J	J_f
1	$\frac{3}{2}w_1+\frac{3}{2}D_2$	$N_{COM}(J,s)$	$\frac{\tilde{r}+w_f}{2}-D_1-x-w_1$
2a	J_1	J	J_2
3a	0	0	0
2b	J_1	J_2	J_3
3b	0	0	$x-w_2$
4	$-\frac{1}{2}D_2$	0	$\frac{1}{2}D_2$
final	$\frac{3}{2}w_1+D_2$	$N_{COM}(J,s)$	$\frac{1}{2}(w_3-w_2)$

	$J_i\ (2\le i\le f-1)$	$J_i(f+1\le i\le k-1)$	J_k
1	$\frac{\tilde{r}+w_f+T_i}{2}-D_1-x$	$\frac{\tilde{r}+T_i}{2}-D_1-x$	$\frac{1}{2}(\tilde{r}+T_k)-D_1-x$
2a	J_{i+1}	J_i	J_k
3a	w_f-w_i	0	0
2b	J_{i+2}	J_{i+1}	J
3b	$x-w_{i+1}$	$x-w_i$	$x-w_k$
4	0	0	0
final	$\frac{\tilde{r}+w_f+T_{i-3}-w_{i-2}}{2}-D_1$	$\frac{\tilde{r}+T_{i-3}-w_i}{2}+D_2+w_1$	$\frac{\tilde{r}+T_{k-2}-w_k}{2}+D_2+w_1$

Table 4: Credit transfers in Lemma 9.2, Case 3

$r_1 > x$. Thus in such a case we have

$$N_{COM}(J_i,r_1)=\frac{1}{2}(w_i-T_{i-1}) \qquad \text{and} \qquad N_{INT}(J_i,r_1)=0 \tag{9.1}$$

If J_i is not interruptable, and J_{i-2} is, then $w_{i-2}+w_{i-1}>x>w_i$ so

$$N_{COM}(J_i,r_1)=0 \qquad \text{and} \qquad N_{INT}(J_i,r_1)=\max(\frac{3}{2}w_{i-2}-s_{i-2},0).$$

Since $s^{\mathrm{OPT}}(J_1)\ge\tilde{r}+w_f$, we have $K(J_1)\ge\frac{3}{2}(\tilde{r}+w_f)+\frac{1}{2}w_1-r_1$. Define $D_2=(\tilde{r}+w_f)-(r_1+w_1)>0$, and $D_1=r_1-\tilde{r}>0$. We will make much use of the following property:

$$w_f-D_2=(r_1+w_1)-\tilde{r}\le\frac{2}{3}(s+x)-\tilde{r}\le\frac{2}{3}x-\frac{1}{3}\tilde{r}\le\frac{1}{3}x. \tag{9.2}$$

This property implies $D_1=w_f-w_1-D_2\le w_f-D_2\le\frac{1}{3}x$ and $D_1\le w_f$.

We consider the various possibilities for s and r_1 and take credit out of J_1 as described in Table 1. The calculations are identical to the ones in Lemma 9.1, Case 2, except that r_f is replaced by $\tilde{r}$ and D_2 is defined as above. It follows that J_1 ends up with credit of at least $\frac{3}{2}(D_2+w_1)$ and satisfies (7.4).

Credit transfers Since $r_1>x$, we can again use Case 2 of Lemma 7.1 to check if jobs satisfy (7.4). We transfer credits as in Table 4. We now have columns for jobs in stead of rows as before, due to space constraints. Note that in this case, there are two reorderings of the jobs (a and b). Also, there is an additional row 4, indicating credit transfers between jobs. Note that the entries in this row add up to 0. The entries in the last row will be explained below.

J_1 is not interruptable because $w_1\le\frac{2}{3}(s+x)-s=\frac{2}{3}x-\frac{1}{3}s<\frac{1}{2}x<\frac{1}{2}r_1$. Moreover, $r_1>x>w_2$, so $N_{INT}(J_2,r_1)=N_{INT}(J_3,r_1)=0$. For J_2, we have $N_{COM}(J,s)=\frac{1}{2}(x-b_s(J))$, $x>w_2$ and $b_s(J)\le b_{r_1}(J_2)$. Therefore J_2 satisfies (7.4).

For J_3, we have $K(J_3) = \frac{1}{2}(\tilde{r}+w_f) - D_1 - x - w_1 + x - w_2 + \frac{1}{2}D_2 = \frac{1}{2}(\tilde{r}+w_f) + \frac{3}{2}D_2 - w_2 - w_f \geq \frac{1}{2}(\tilde{r} - 2w_2 - w_f) + \frac{3}{2}D_2 = \frac{1}{2}(w_3 - w_2) + \frac{1}{2}(\tilde{r} - w_2 - w_3 - w_f + 3D_2) \geq \frac{1}{2}(w_3 - w_2)$ using (9.2).

For $i = 4, \ldots, f$ we find

$$\begin{aligned}
K(J_i) &= \frac{1}{2}(\tilde{r} + w_f + T_{i-2}) - D_1 - x + x - w_{i-1} + w_f - w_{i-2} \\
&\geq \frac{1}{2}(\tilde{r} + w_f + T_{i-3} - w_{i-2}) - D_1 && (9.3) \\
&= \frac{1}{2}(\tilde{r} + T_{i-3} - w_f - w_{i-2}) + D_2 + w_1 && (9.4) \\
&= \frac{(w_i + T_{i-3} - w_{i-2}) + (\tilde{r} - w_f - w_i + 2D_2)}{2} \geq \frac{w_i + T_{i-3} - w_{i-1}}{2}.
\end{aligned}$$

If J_i is interruptable, J_{i-2} and earlier jobs are not and we are done. Suppose J_i is not interruptable and $s_{i-2} < \frac{3}{2}w_{i-2}$. (If $s_{i-2} \geq \frac{3}{2}w_{i-2}$, we are done.) Then we use that $s_{i-2} > r_1$ and we have from (9.3)

$$K(J_i) + s_{i-2} > \frac{1}{2}(\tilde{r} + w_f + T_{i-3} - w_{i-2}) + \tilde{r} \geq \frac{1}{2}(\tilde{r} - w_{i-2} + T_{i-3}) + \frac{3}{2}w_{i-2}.$$

For J_{f+1}, the calculations are similar, but from (9.4) we now derive $K(J_{f+1}) \geq \frac{1}{2}T_{f-1} + \frac{1}{2}(w_{f+1} - T_f)$ and $K(J_i) \geq \frac{1}{2}T_{f-1} + 0$ (using (9.2)). Thus there is sufficient completion-credit, and the case $s_{f-1} < \frac{3}{2}w_{f-1}$ is handled as above.

For $i = f+2, \ldots, k$ we have $K(J_i) \geq \frac{1}{2}(\tilde{r} + T_{i-1}) - D_1 - x + x - w_{i-1} \geq \frac{1}{2}(\tilde{r} + T_{i-3} - w_f - w_{i-1}) + D_2 + w_1$.

Suppose $r_1 < \frac{3}{2}w_{i-2}$. Then $K(J_i) + r_1 \geq \frac{1}{2}(\tilde{r} + T_{i-2} - w_{i-1}) + \tilde{r} \geq \frac{1}{2}(\tilde{r} + T_{i-3} - w_{i-1}) + \frac{3}{2}w_{i-2}$, and we are done.

Otherwise, $s_{i-2} \geq r_1 \geq \frac{3}{2}w_{i-2}$ so $N_{INT}(J_i, r_1) \leq 2w_{i-4} - r_1 \leq \frac{1}{2}w_{i-4}$.

Suppose $w_f \geq \frac{2}{3}x$. Then $D_2 \geq \frac{1}{3}\tilde{r}$ and thus $K(J_i) \geq \frac{1}{2}(w_i + T_{i-3} - w_{i-1}) + \frac{1}{2}(\tilde{r} - (w_f - D_2)) - \frac{1}{2}(w_i - D_2) \geq \frac{1}{2}(w_i + T_{i-3} - w_{i-1})$.

If $w_f < \frac{2}{3}x$, then also $w_2 < \frac{2}{3}x$. For this particular case only, we consider when OPT runs the jobs in $BEFORE_s(J)$. If any job in $BEFORE_s(J)$ is completed after J_f, then by Lemma 8.2 there is extra credit available of $\frac{3}{2}w_f$. We give this to J_i which then has credit of at least $\frac{1}{2}(w_i + T_{i-3} - w_{i-1}) + \frac{1}{2}(\tilde{r} - w_i) \geq \frac{1}{2}(w_i + T_{i-3} - w_{i-1})$. If all jobs in $BEFORE_s(J)$ are completed before J_f, then the credit of J_f is $\frac{3}{2}b_s(J)$ larger than previously calculated. We give this to J_2. Then $K(J_2) \geq \frac{1}{2}x$. However, $w_2 < \frac{2}{3}x$, so we take $\frac{1}{6}x$ out of the credit of J_2 again and give it to J_i. Then $K(J_i) \geq \frac{1}{2}(\tilde{r} + T_{i-3} - w_{i-1}) + \frac{1}{6}x + \frac{1}{2}w_{i-2} - D_1 \geq \frac{1}{2}(\tilde{r} + T_{i-3} - w_{i-1})$ since $D_1 \leq \frac{1}{3}x$ and $D_1 \leq \frac{1}{2}w_f \leq \frac{1}{2}w_{i-2}$.

For J, we calculate as for J_{f+1} in case $k = f$ and as for J_{f+2} and higher in case $k > f+1$. (The calculations for $J_{f+1}, \ldots, J_k$ hold for any job size at most x, so they also hold when applied to J.) □

Lemma 9.3 *If* RSPT *interrupts a job J at time r_1, and* OPT *runs at least two large jobs before ARRIVE, and STATIC holds, and the invariant held at time s, then it holds at time r_1.*

Proof. Since RSPT only interrupts J before time $2x$, we have $f = 1$: OPT starts to run the jobs in *ARRIVE* after RSPT does, and will use the optimal order for them by Lemma 5.4. That Lemma also implies that OPT runs not more than two large jobs before *ARRIVE*. If $s \leq x/2$, we have $r_i \leq x$ for all jobs $J_i \in ARRIVE$. Then, again by Lemma 5.4, we may assume OPT runs only one large job before *ARRIVE*: a contradiction. Hence $x/2 < s < 2x$ and $r_1 > x$.

	1	2	3	4	final
J_1	$\frac{1}{2}w_1 + K_1$	J_1	0	$-K_1$	$\frac{1}{2}w_1$
J	$N_{COM}(J, s)$	J_2	0	0	$N_{COM}(J, s)$
J_2	$\frac{1}{2}T_2 + D_1$	J_3	$x - w_2$	$-\frac{x+w_1}{2} - D_1$	$\frac{1}{2}(x - w_2)$
$i = 3, \ldots, k-1$:					
J_i	$\frac{1}{2}T_i + D_1$	J_{i+1}	$x - w_i$	$-\frac{1}{2}(x + w_{i-1})$	$\frac{x+T_{i-2}-w_i}{2} + D_1$
J_k	$\frac{1}{2}T_k + D_1$	J	$x - w_k$	$-\frac{1}{2}(x + w_{k-1})$	$\frac{x+T_{k-2}-w_k}{2} + D_1$
J^2	$N_{COM}(J^2, s) - T_k + w_1$	J^2	0	$+T_k + D_1$	$N_{COM}(J^2, s) + D_1$

Table 5: Credit transfers in Lemma 9.3 ($r_1 \leq \frac{3}{2}x$, $k \geq 3$)

Suppose $\ell_s(J) < x$. In this case, we ignore that OPT has to run the job of this size too at some time. This can only help OPT. The interrupt-delay caused by the current interruption is $r_1 - s$ for each large job that RSPT has not completed yet.

Credit of J_1 and large jobs We define $D_1 = -D(J_1) = x + x_2 - r_1 \geq \frac{4}{3}x - \frac{2}{3}s + w_1$. We have $K(J_1) = \frac{3}{2}(x + x_2) + \frac{1}{2}w_1 - r_1$. We consider the various possibilities for s and r_1 and take credit out of J_1 as described in Table 1, and an extra w_1 for the small-job delay of J^2. By these reassignments, and because J and J^2 satisfied (7.4), we have in Step 1 that $K(J) \geq \frac{1}{2}(x - b_s(J))$ and $K(J^2) \geq \frac{1}{2}(x_2 - x) - T_k + w_1$.

Suppose $s \leq x$. Then $r_1 \leq \frac{2}{3}(s + x) \leq \frac{4}{3}x$. We take $2(r_1 + w_1) - s - x$ of credit out of J_1 to give to J and J^2, and we let it keep $\frac{1}{2}w_1$ for itself. We denote the remainder, which will be given to J^2, by K_1. We have $r_1 \leq \frac{2}{3}(x + s) - w_1$, so in this case $K_1 = \frac{3}{2}(x + x_2) - 3r_1 - 2w_1 + s + x \geq \frac{1}{2}x + \frac{3}{2}x_2 - s + w_1 \geq x_2 + w_1$.

Now suppose $x < s < r_1 \leq \frac{3}{2}x$. In this case we need $2(r_1 + w_1) - 2s$ for J and J^2. Hence $K_1 = \frac{3}{2}(x + x_2) - 3r_1 - 2w_1 + 2s \geq \frac{3}{2}(x + x_2) - 2x + w_1 \geq x_2 + w_1$.

Thirdly, suppose $x < s \leq \frac{3}{2}x < r_1$. Now the required credit is in total $2(r_1 - s + w_1) + 2(r_1 - \frac{3}{2}x)$. Hence $K_1 \geq \frac{3}{2}(x + x_2) - \frac{4}{3}s - \frac{1}{3}x + 3w_1 \geq \frac{2}{3}x + 3w_1$.

Finally, if $\frac{3}{2}x < s < r_1$ the jobs require $4(r_1 - s) + w_1$, and again $K_1 \geq \frac{2}{3}x + 3w_1$.

Credit transfers We transfer credits as in Table 5.

For $2 \leq i \leq k$, we have $D(J_i) \leq (r_1 + x + T_{i-1}) - (x + x_2 + T_{i-1}) = r_1 - x_2$ in Step 1. Using (6.1) we have $K(J_i) \geq \frac{1}{2}(x + x_2 + T_i) - (r_1 - x_2) \geq \frac{1}{2}T_i + D_1$ for $2 \leq i \leq k$.

J_1 is not interruptable by Property R5 since $r_1 \geq \max(x, s)$ and $r_1 + w_1 \leq \frac{2}{3}(s + x) \leq \frac{4}{3}\max(s, x)$, hence $N_{INT}(J_i, r_1) = 0$ for $1 \leq i \leq 3$. Furthermore, $D_1 = x + x_2 - r_1$ is an upper bound for the total amount of future interrupt-delays caused by interruptions of jobs before J^2 that have arrived so far.

Suppose $k \leq 2$. In this case we do not use the table. If $k = 1$, we are done immediately. If $k = 2$, then $N_{INT}(J, r_1) = 0$ and $K(J) = \frac{1}{2}T_2 + D_1 + x - w_2 = \frac{1}{2}(x - w_2 + w_1) + \frac{1}{2}x + D_1$. Thus J can give $\frac{1}{2}x + D_1$ to J^2, and J_1 can give an additional $K_1 \geq \frac{1}{2}x$ to J^2. Then J^2 receives in total $x + D_1 \geq w_2 + D_1$, and it received w_1 already from J_1 at the start. Thus J^2 satisfies (7.4). For job J_2, we have $K(J_2) = N_{COM}(J, s) = \frac{1}{2}(x - b_s(J))$, $x \geq w_2$ and $b_{r_1}(J_2) \geq b_s(J)$, so $K(J_2) \geq N_{COM}(J_2, r_1)$.

Suppose $k \geq 3$. In this case we use Table 5. J_3 can give D_1 away because $N_{INT}(J_3, r_1) = 0$. We distinguish between two cases: $r_1 \leq \frac{3}{2}x$ and $r_1 > \frac{3}{2}x$.

Case 1. $r_1 \leq \frac{3}{2}x$: The jobs $J_4, \ldots, J_k, J, J^2$ have sufficient interrupt-credit because they have D_1. Since $x > w_i$ for $i = 4, \ldots, k$, all these jobs also have sufficient completion-credit. J_2 satisfies (7.4) as above.

Hence we only need to show that the total transferred credit in Column 4 is at most 0, i. e. we do not in total give more credit to jobs than we remove from jobs. In other words, all the credit given to J^2 is actually available. To see this, note that $\frac{1}{2}(x + w_{i-1}) \geq w_{i-1}$ for $2 \leq i \leq k$. Furthermore, if $r_1 \leq \frac{3}{2}x$ (shown in the table), we have $K_1 \geq x_2 \geq w_k$, and the additional D_1 from J_3 completes the credit given to J^2.

Case 2. $r_1 > \frac{3}{2}x$: Denote the jobs that RSPT starts to run at time r_1 alternatively by $J'_1, \ldots, J'_{k'}$, then by (7.1) we have $N_{INT}(J'_i, r_1) \leq \frac{1}{2}w'_{i-4}$. We take D_1 from J_3 as before and also D_1 from the very next job J'_4 ($J'_4 \in \{J_4, J\}$). We can do that because $N_{INT}(J'_4, r_1) = 0$. This makes in total again at least $T_k + D_1$ to give to J_2, since in this case $K_1 + D_1 \geq x_2 \geq w_k$. Giving it to J^2 again ensures that J^2 satisfies (7.4).

Now suppose $\ell_s(J) \geq x$. In this case OPT runs only one more large job than RSPT before *ARRIVE*. Hence in this case, J^2 does not have small-job delay and we are done after Step 3 in the table: all jobs already satisfy (7.4). □

10. Job completions

We divide the job completions into cases based on how many large jobs OPT runs before *ARRIVE*. The case where no jobs arrived at all while RSPT was running J is treated separately in Lemma 10.1. An overview can be found in the following table.

Large jobs before *ARRIVE* by OPT	0	1	2	More than 2
Lemma	10.2	10.3	10.4	10.5, 10.10

Remember that to calculate the initial credit of jobs, we will use Event assumption 2. Since all jobs in *ARRIVE* are smaller than x, and complete after J, we have immediately $N_{COM}(J_i, t) = 0$ for $J_1, \ldots, J_k$.

We use again Event assumption 3.

Lemma 10.1 *If* RSPT *completes a job J and no jobs arrived while it was running J, and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. RSPT now starts the run the smallest available job at the time that was calculated in the analysis of the most recent event. Hence, for the remaining jobs the situation (and the credit) does not change. The completed job has nonnegative credit. □

Lemma 10.2 *If* RSPT *completes a job J without interruptions and* OPT *does not run any large jobs before ARRIVE, and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. We use similar tables as in Section 9, starting with a job order for which it is easy to calculate the credits and then reordering the jobs. Here we ignore that RSPT starts to run the jobs already at time s and not only at time r_f. This gives us a lower bound for the amount of credit that is actually available.

Applying Lemma 8.2 repeatedly, we have that there is $\frac{3}{2}T$ of credit available. We give this to J. We consider first the credit of the jobs if RSPT would run the jobs in the same order as OPT, and starting at time r_f. Then J starts T time later than calculated at the previous event, and thus only has $\frac{1}{2}T$ left of the $\frac{3}{2}T$ that it just received. In Step 1, we have that RSPT starts each job at the same time as OPT starts it. See Table 6. We use (6.1).

	1	2	3	final
J_f	$\frac{1}{2}(r_f + w_f)$	J	$-(x - w_f)$	0
J_i $(i = 1, \dots, f-1)$	$\frac{1}{2}(r_f + w_f + T_i)$	J_i	$-(x - w_f)$	$\frac{1}{2}T_i$
J_i $(i = f+1, \dots, k)$	$\frac{1}{2}(r_f + T_i)$	J_{i-1}	$-(x - w_i)$	$\frac{1}{2}(T_i - w_f)$
J	$N_{COM}(J, s) + \frac{1}{2}T$	J_k	0	$N_{COM}(J, s) + \frac{1}{2}T$

Table 6: Credits in Lemma 10.2

We use in this table that $\frac{1}{2}(r_f + w_f) - (x - w_f) = \frac{3}{2}w_f - x + \frac{1}{2}r_f \geq s - r_f \geq 0$, which holds since $\frac{3}{2}w_f \geq s + x - \frac{3}{2}r_f$. For $J_f, \dots, J_{k-1}$ we also use that $x - w_i \leq x - w_f$. Note that $T_i - w_f \geq T_{i-1}$ for $i > f$. Hence all the jobs in Table 6 satisfy (7.4) by Lemma 7.1, Case 1. Note that this proof also holds for $f = 1$. □

Lemma 10.3 *If* RSPT *completes J without interruptions and* OPT *runs one large job before ARRIVE, and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. If $s \leq x/2$, the jobs in *ARRIVE* start within a factor of $\frac{3}{2}$ of their optimal starting time (and run in the best possible order), so that the credit of J_i is at least $\frac{1}{2}\sum_{j=1}^{i} w_i$ by (6.1): all jobs in *ARRIVE* satisfy (7.4) by Lemma 7.1. The same thing holds if $s \geq 2x$.

Suppose $x/2 < s < 2x$. This implies $N_{INT}(J_i, t) \leq 2w_{i-4} - t \leq \frac{1}{2}w_{i-4}$ for $4 \leq i \leq k$ and $N_{INT}(J_i, t) = 0$ for $1 \leq i \leq 3$. Recall that $N_{COM}(J_i, t) = 0$ for all $J_i \in ARRIVE$.

Suppose $1 \leq f < k$. We define $v_f = s^{\text{OPT}}(J_1) - \frac{2}{3}(s + x) > 0$ and $\tilde{s} = \max(s, x + b_s(J))$. Suppose first that OPT runs the jobs in $BEFORE_s(J)$ before *ARRIVE*, then $s^{\text{OPT}}(J_1) \geq \tilde{s} + w_f$. We have

$$K(J_f) \geq \frac{1}{2}(\tilde{s} + w_f) - (s + x - \tilde{s}) = \frac{3}{2}\tilde{s} + \frac{1}{2}w_f - (s + x) = \frac{3}{2}v_f - w_f.$$

If $s \leq x + b_s(J)$, then $w_f - v_f \leq \frac{2}{3}(s + x) - \tilde{s} \leq \frac{2}{3}(x + b_s(J) + x) - (x + b_s(J) = \frac{1}{3}(x - b_s(J))$. If $s \geq x + b_s(J)$, then $w_f - v_f \leq \frac{2}{3}x - \frac{1}{3}s \leq \frac{1}{3}(x - b_s(J))$ as well. Using this bound, we transfer credits as in Table 7. It can be seen that the entries in Column 4 add up to at most 0, and that all jobs satisfy (7.4).

	1	2	3	4	final
J	$\frac{1}{2}(x - b_s(J))$	J	0	$-\frac{1}{2}(x - b_s(J))$	0
J_f	$\frac{3}{2}v_f - w_f$	J_1	0	$\frac{1}{3}(x - b_s(J))$	$\frac{1}{2}v_f$
J_1	$\frac{3}{2}v_f - w_f + \frac{1}{2}w_1$	J_2	$w_f - w_1$	$\frac{1}{2}w_1$	$\frac{3}{2}v_f$
$i = 2, \dots, f:$					
J_i	$\frac{3}{2}v_f - w_f + \frac{1}{2}T_i$	J_{i+1}	$w_f - w_i$	$\frac{1}{2}(w_i - w_{i-1})$	$\frac{3}{2}v_f + \frac{1}{2}T_{i-2}$
J_{f+1}	$\frac{3}{2}v_f - w_f + \frac{T_{f+1} - w_f}{2}$	J_{f+1}	0	$\frac{x - b_s(J)}{6} - \frac{1}{2}w_{f-1}$	$v_f + \frac{1}{2}T_{f-2}$
$i = f+2, \dots, k:$					
J_i	$\frac{3}{2}v_f - w_f + \frac{T_i - w_f}{2}$	J_i	0	0	$\frac{3}{2}v_f + \frac{1}{2}T_{i-3}$

Table 7: Credit transfers in Lemma 10.3 $(1 \leq f < k)$

If $f = k$, then we cannot take $\frac{1}{2}w_{f-1}$ of credit out of J_{f+1} because there is no such job. However, we can now give $\frac{x - b_s(J)}{6}$ from J to J_f instead of to J_{f+1}, and $\frac{x - b_s(J)}{6} \geq \frac{1}{2}(w_f - v_f) \geq \frac{1}{2}(w_{f-1} - v_f)$.

Finally, consider the case where OPT does not run all jobs in $BEFORE_s(J)$ before all the jobs in *ARRIVE*. Then OPT runs one job in $BEFORE_s(J)$ in particular after job J_f, which implies that there is an additional $\frac{3}{2}w_f$ of credit available by Lemma 8.2. We can give w_f to J_1 and $\frac{1}{2}w_f$ to J_{f+1} (instead of giving those jobs credit from J). For the other jobs, we can still transfer credits as in Table 7. If $f = k$, we give $\frac{3}{2}w_f = \frac{3}{2}w_k$ to J_1. □

Corollary 10.1 *If* RSPT *completes J without interruptions and* OPT *runs one large job before ARRIVE, and STATIC holds, the jobs in ARRIVE need to receive at most an additional $\frac{3}{2}(w_f - v_f) \le \frac{1}{2}(x - b_s(J))$ of credit in total in order to satisfy (7.4), where $v_f = s^{\text{OPT}}(J_1) - \frac{2}{3}(s + x)$.*

Lemma 10.4 *If* RSPT *completes J without interruptions and* OPT *runs two large jobs before ARRIVE, and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. If $\ell_s(J) \ge x$, there is nothing to prove since the same number of jobs is delayed by RSPT and by OPT, and RSPT starts the jobs in *ARRIVE* at most a factor of 3/2 after OPT starts them since $s \ge x$. Hence, the jobs in *ARRIVE* satisfy (7.4) and the remaining large jobs gain credit by Rule C2 and still satisfy (7.4). (We can assume OPT runs the same two large jobs before *ARRIVE* as RSPT, similarly to Event assumption 3.)

Suppose $\ell_s(J) < x$. Denote the largest of the two jobs that OPT completes before *ARRIVE* by J^2. We distinguish between the cases $s \le x_2$ and $s > x_2$.

Case 1. $s \le x_2$. Then $K(J_i) \ge \frac{1}{2}(x + x_2 + T_i)$. Note that $N_{COM}(J_i, t) = 0$, and $N_{INT}(J_i, t) \le \frac{1}{2}T_{i-2}$. We let each job J_i keep $\frac{1}{2}T_{i-1}$ and give $\frac{1}{2}(x + x_2 + w_i) > \frac{3}{2}w_i$ to J^2. This is sufficient to both pay for the small job-delay of J^2, which is T, and to add $\frac{1}{2}T$ for $K_{INT}(J^2)$, which ensures $K(J^2) \ge N_{COM}(J^2, t) + N_{INT}(J^2, t)$.

Case 2. $s > x_2$. Then the jobs in *ARRIVE* are not interruptable, hence $N_{INT}(J_i, t) = N_{COM}(J_i, t) = 0$ for all jobs $J_i \in$ *ARRIVE*. Therefore, $N_{INT}(J_2, t) = 0$. Consider the set $BEFORE_s(J)$ of jobs that RSPT already completed, and suppose OPT completes all these jobs before J_k.

Then we have $K(J_i) \ge \frac{3}{2}(\max(s, x + x_2)) + \frac{1}{2}T_i - (s + x) \ge \frac{1}{2}T_i$ for $i = 1, \ldots, k - 1$ and $K(J_k) \ge \frac{3}{2}(\max(s, x + x_2) + b_s(J)) + \frac{1}{2}T_k - (s + x) \ge \frac{3}{2}b_s(J) + \frac{1}{2}T_k$. All the credit of these jobs can go to J^2. The sum is at least $\frac{3}{2}b_s(J) + T_{k-1} + \frac{1}{2}w_k$. Furthermore, J^2 receives $\frac{1}{2}(x - b_s(J))$ from J, and it loses at most T_k because it is delayed by RSPT. Hence in total J^2 does not lose credit and still satisfies (7.4).

Finally, suppose there is a job in $BEFORE_s(J)$ that OPT does not complete before the final job J_k in *ARRIVE*. By Lemma 8.2 there is $\frac{3}{2}w_k$ of credit available that we can give to J^2, in addition to the $T_{k-1} + \frac{1}{2}w_k$ that it gets from the jobs in *ARRIVE*. This is sufficient for J^2 to satisfy (7.4) again. □

10.1 OPT runs at least three jobs before ARRIVE

Define $\delta(t)$ as the number of jobs OPT has completed minus the number of jobs RSPT has completed at time t. Property R4 implies that if RSPT is running a job of size x at time $t < 2x$, then $\delta(t) \le 1$. In other words, $\delta(t) \ge 2 \Rightarrow t \ge 2x$. Thus as long as $\delta(t) \ge 2$, no jobs are ever interrupted by RSPT by Property R5.

Lemma 10.5 *If $\delta(s) \le 1$, and a job J is completed by* RSPT, *and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. Because of Lemma 10.1, 10.2, 10.3 and 10.4 we only need to consider the case where OPT runs $a \ge 3$ large jobs before *ARRIVE*. Since $\delta(s) \le 1$, after time s OPT still starts $a - 2 \ge 1$ J-large job before it runs the jobs in $ARRIVE$. Therefore, RSPT completes the jobs in $ARRIVE$ no later than OPT does. Moreover, $a = 3$ since the jobs in $ARRIVE$ arrive before OPT completes the third J-large job. We have $K(J_i) \geq \frac{1}{2}(3x + T_i) \geq 2w_i$. We give w_i to any jobs that RSPT completes after $ARRIVE$ and OPT before $ARRIVE$, since the jobs in $ARRIVE$ are not interruptable and do not need any credit themselves: we have $s > x$, else $a \leq 2$. There are at most two such jobs, and their credit decreased by T because of the jobs in $ARRIVE$, using Rule C1. Therefore they get all the lost credit back from the jobs in $ARRIVE$, and again satisfy (7.4). □

Suppose $\delta(s) \geq 2$. It can only happen during the final run of a job that $\delta(s)$ increases from at most 1 to above 1, because we can apply Property R4 whenever a job is interrupted. For any maximal interval $[a, b)$ in which $\delta(s) \geq 2$ and where RSPT completes a job at time a, denote the job that it completes at time a by $J(a)$.

Lemma 10.6 *Suppose* OPT *starts its next $J(a)$-large job after time a at time s_2. Then there is a time $t \in (a, s_2 + w(J(a))]$ such that $\delta(t) \leq 1$.*

Proof. At time a, RSPT starts to run the $J(a)$-small jobs that arrived while it was running $J(a)$. Suppose $\delta(t) \geq 2$ in the entire interval $(a, s_2 + w(J(a))]$, then RSPT does not interrupt any job in this interval. Then in this interval, it certainly completes at least as many jobs as OPT starts and completes in the interval $(a - w(J(a)), s_2]$ (it is possible that OPT decides not to run some small jobs that have arrived yet, but then it can only complete less jobs in $(a - w(J(a)), s_2]$ than RSPT does in $(a, s_2 + w(J(a))]$). Thus $\delta(s_2 + w(J(a))) \leq \delta(a) \leq 1$. □

Lemma 10.7 *Suppose $\delta(s) \geq 2$ for a job J. Then $J(a)$ is J-large.*

Proof. At time $a - w(J(a))$, OPT has completed at most one job that RSPT has not completed, and such a job can only be $J(a)$-large. At time a, RSPT completes a $J(a)$-large job, namely $J(a)$ itself. OPT completes at most one $J(a)$-large job in the interval $(a - w(J(a)), a]$. Thus at time a, OPT has (still) completed at most one $J(a)$-large job more than RSPT.

By Lemma 10.6, OPT does not complete any other $J(a)$-large job within the current interval where $\delta(t) \geq 2$. On the other hand, at time s RSPT has completed all J-small jobs that have arrived, so OPT must have completed at least two J-large jobs that RSPT has not completed. Then J must be $J(a)$-small. □

Lemma 10.8 *Suppose $\delta(s) \geq 2$ for a job J. Then $s \geq 3x$.*

Proof. $J(a)$ is J-large by Lemma 10.7. At time s, RSPT has completed all J-small jobs that have arrived. Also it has completed $J(a)$. There are two cases.

If OPT completes $J(a)$ before time s, and at least two other J-large jobs, then $s \geq w(J(a)) + 2x \geq 3x$. If OPT does not complete $J(a)$ before time s, there must be at least three other J-large jobs that OPT has completed at time s since $\delta(s) \geq 2$, and thus $s \geq 3x$. □

Lemma 10.9 *If $\delta(s) \geq 2$, then at time s the total size of the smallest $\delta(s) - 1$ jobs in* RSPT*'s queue is at most* $\min(w(J(a)), a/3)$.

Proof. We begin by showing it holds at time a. At that time, we have that $\delta(a) - 1$ jobs that OPT has completed and RSPT has not, were started and completed by OPT in $(a - w(J(a)), a]$. Thus their total size is at most $w(J(a))$. Then the total size of the $\delta(a) - 1$ *smallest* such jobs is certainly at most $w(J(a))$.

If $a \geq 3w(J(a))$, the other bound also follows immediately. Otherwise, if OPT completes two $J(a)$-large jobs before time a, we use $a - 2w(J(a)) \leq a/3$. Finally, if it completes only one, then the first $J(a)$-small job that OPT completes in $(a - w(J(a)), a]$ must complete after time $\frac{2}{3}a$, since it does not cause an interruption. Thus the $\delta(a) - 1$ smallest jobs can start and complete in an interval of size $a/3$. (Note that if OPT completes a $J(a)$-large job in $(a - w(J(a)), a]$ in this case, then $\delta(a - w(J(a))) \leq 0$.)

Now we show that it holds later, by induction. Consider a time s for which (still) $\delta(s) \geq 2$. Denote the number of jobs that RSPT and OPT complete in $(a, s]$ by c_1 and c_2, respectively. From these c_2 jobs and the last $\delta(a) - 1$ jobs that OPT starts and completes in $(a - w(J(a)), a]$, there are then exactly $\delta(a) - 1 + c_2 - c_1 = \delta(s) - 1$ jobs that RSPT must still run. The total size of the c_1 jobs that RSPT completes in $(a, s]$ is exactly $s - a$, so the total size of these $\delta(s) - 1$ jobs is again bounded by $\min(w(J(a)), a/3)$. Then this certainly holds for the smallest $\delta(s) - 1$ jobs that RSPT must still complete. □

Lemma 10.10 *If $\delta(s) \geq 2$, and a job J is completed, and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. Since $\delta(s) \geq 2 \Rightarrow s \geq 3x$ by Lemma 10.8, the jobs in $ARRIVE(I)$ require 0 credit because they cannot be interrupted and a larger job completes before them. Because of Lemma 10.9, the $\delta(s) - 1$ smallest waiting jobs also require just 0 credit for the same reason (using $s \geq a$).

We first consider an alternative schedule, where RSPT runs the jobs in $ARRIVE(I)$ not just after J, but after the $\delta(s) - 1 \geq 1$ smallest waiting jobs in RSPT's queue. (If $\delta(s) = 2$, this is only J.) The jobs in $ARRIVE(I)$ then cause only small-job delay for at most one job J', and they are executed within $\frac{1}{3}s$ of their optimal starting time. Therefore each such job J_i has credit of at least $\frac{1}{2}(s + T_i) - \frac{s}{3} \geq \frac{1}{2}T_i + \frac{1}{6}s \geq \frac{1}{2}T_i + \frac{1}{2}x(I) \geq w_i$, which it can give to J' to make up for its small-job delay. By finally putting the jobs in the correct order (but keeping the credits in the same locations as usual), the total amount of credit does not decrease. This proves the lemma. □

Lemma 10.11 *If RSPT completes a job J, and STATIC holds, and the invariant held at time s, then it holds at time t.*

Proof. This follows from Lemmas 10.1, 10.2, 10.3, 10.4, 10.5 and 10.10. □

11. Interruptions, $s < 2x/3$

In Section 9, it was shown for several situations that the invariant keeps holding if an interruption occurs. There is only one case left of the situation where OPT does not complete any large jobs before $ARRIVE$, and this is the complement of Lemma 9.1: $s < 2x/3$.

We will use the following Lemma. We do **not** use Assumption 2 at time s or r_1.

Lemma 11.1 *For any input sequence σ, where some run-interval $I = (s(J), r_1]$ ends with an interruption, and where $s(J) \leq \frac{2}{3}w(J)$, and where OPT does not run any J-large job before $ARRIVE(I)$, it is possible to modify the release times of some jobs so that σ can be divided into two sequences σ_1, σ_2 so that the schedule of RSPT for σ_1 is unchanged and the schedule for σ_2 is unchanged starting from time $s(J)$; no job in σ_2 arrives before time $s(J)$; $J \in \sigma_2$ arrives at time $s(J)$; all other J-large jobs arrive at time r_1 or later; and $\mathrm{OPT}(\sigma_1) + \mathrm{OPT}(\sigma_2) \leq \mathrm{OPT}(\sigma)$.*

Proof. We divide σ into two parts: we let σ_1 contain the jobs from σ that RSPT completes before time $s(J)$, and σ_2' all the other jobs.

All jobs in σ_1 are finished by RSPT before time $s(J) \le \frac{2}{3}w(J)$. The jobs in σ_2' either have size at least $w(J)$ or arrive after time $s(J)$ by definition of RSPT. Therefore, when processing σ, the total completion time of RSPT of the jobs in σ_1 is the same as it would have been if the jobs in σ_2' all arrived after time $s(J)$: any job in σ_2' that is running before time $s(J)$, is interrupted immediately whenever a job in σ_1 arrives, since such a job from σ_2' has size at least $w(J)$.

Moreover, OPT does not start any J-large job in σ_2' before time r_1, since OPT runs $ARRIVE(I)$ before any J-large job and OPT does not complete any job in $ARRIVE(I)$ before time r_1 by Property R3. Therefore, the optimal total completion time of the jobs in σ_2' is unaffected if we constuct σ_2 by changing the release time of J to $s(J)$ and the release time of all other J-large jobs in σ_2' that arrive before time r_1, to r_1. Clearly, this cannot affect the optimal total completion time of the jobs in σ_1. (Note that it is possible that OPT still runs some jobs in σ_1 after time s.)

Thus $\text{RSPT}(\sigma_1) + \text{RSPT}(\sigma_2) = \text{RSPT}(\sigma)$, $\text{OPT}(\sigma_1)$ is the cost of σ_1 in σ, $\text{OPT}(\sigma_2)$ is at most the cost of σ_2' in σ and we are done. □

Thus if there is an interruption at time r_1, and $s \le \frac{2}{3}x$, we can consider all the jobs completed earlier as a separate job sequence and make the following assumption:

Global assumption 3 *The interrupted job was the first job in the input sequence.*

Consider such an interruption and make assumption 3. If $f = 1$, we can in fact assume **all** jobs in σ arrive at time r_1, since both OPT and RSPT start and complete all jobs in σ after time r_1. Then we are in the case where r_1 is the end of an interval in which RSPT was idle, and we apply Lemma 7.1.

In the remainder of this section, we only need to consider the case $f > 1$. The important thing about Assumption 3 is that it implies that the first event of this sequence occurred at time s, and the job that started then still has all of its original credit. This is much more credit than could be deduced from the invariant.

Lemma 11.2 *If* RSPT *interrupts a job J at time r_1, and* OPT *runs no large jobs before ARRIVE, and $x/3 \le s < 2x/3$, and $f > 1$, and STATIC holds, then the invariant holds at time r_1.*

Proof. We use Assumption 3 and consider the credits of the jobs, assuming J arrived at time s. Again we can assume all J-large jobs besides J arrived at time r_1 (thus not using Assumption 2 in this case). Define $D_1 = r_1 - r_f > 0$ and $D_2 = (r_f + w_f) - (r_1 + w_1)$, as before. See Table 8.

	1	**2**	**3**	**4**	final
J_f	$\frac{r_f+w_f}{2} - D_1$	J_1	0	$\frac{w_f-r_f-w_1}{2}$	$\frac{1}{2}w_1 + D_2$
J_1	$\frac{r_f+w_f+w_1}{2} - D_1$	J_2	$w_f - w_1$	$\frac{w_1+2r_f-w_f}{2}$	$\frac{3}{2}r_f + w_1 + D_2$
$i = 2, \ldots, f-1:$					
J_i	$\frac{r_f+w_f+T_i}{2} - D_1$	J_{i+1}	$w_f - w_i$	0	$\frac{r_f+w_f+T_{i-1}-w_i}{2} + D_2 + w_1$
$i = f+1, \ldots, k:$					
J_i	$\frac{r_f+T_i}{2} - D_1$	J_i	0	0	$\frac{r_f+T_i}{2} - D_1$
J	$\frac{r_f+T_k+x}{2} - D_1$	J	0	$-\frac{1}{2}r_f$	$\frac{T_k+x}{2} - D_1$

Table 8: Credit transfers in Lemma 11.2

J_1 satisfies (7.4) by Lemma 7.1, Case 1.

For J_2, note that $\frac{3}{2}r_f \ge \frac{3}{2}s \ge \frac{1}{2}x \ge \frac{1}{2}w_2$, and use the same lemma.

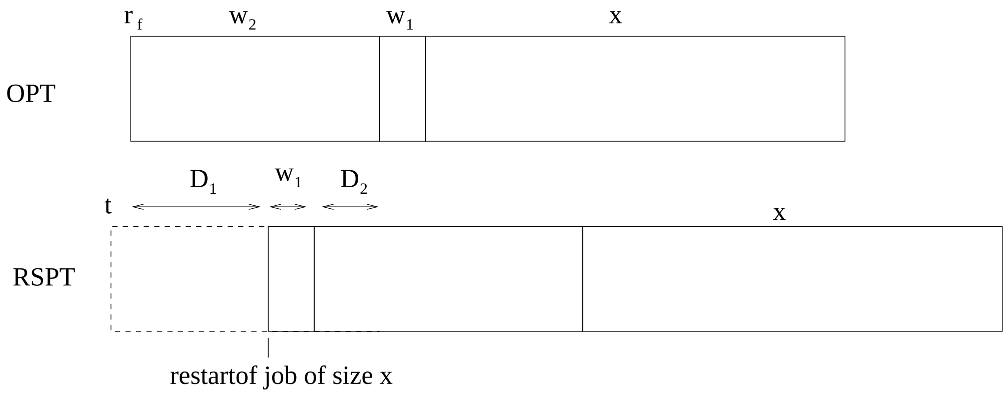

Figure 3: An example of a late interruption

For $i = 3, \ldots, f$, we use $w_f \geq w_i$, $D_2 + w_1 = (r_f + w_f) - r_1 \geq w_f - r_1 \geq w_{i-1} - r_1$ and $D_2 + w_1 \geq 0$. This shows these jobs satisfy (7.4) by Lemma 7.1, Cases 2 and 3.

For $i = f+1, \ldots, k$ we use $K(J_i) \geq \frac{1}{2}T_i - D_1 = \frac{1}{2}(w_i - w_{i-1} + T_{i-2}) + (w_{i-1} - D_1)$ and note that $w_{i-1} - D_1 \geq w_{i-1} - r_1$ and $w_{i-1} - D_1 = w_{i-1} + D_2 - w_f + w_1 \geq D_2 + w_1 \geq 0$, and we use the same Lemma. We can reason analogously for J (implying that we can indeed take $\frac{1}{2}r_f$ out of the credit of J) and for J-large jobs. □

Note that the proof of Lemma 11.2 works as long as $r_f \geq x/3$. From now on, we assume $r_f < x/3$. For this case, we use the same credit transfers as described in Table 8. However, in this case, this may not be enough for job J_2 to satisfy (7.4). We make one additional transfer apart from the ones mentioned in Table 8: we give D_2 from J_1 to J_2. By (5.1), we have $D_2 = \tau_f - \tau_1$ where $\tau_f > 0$ and $\tau_1 \leq 0$. See Figure 3.

Lemma 11.3 *Consider the jobs involved in a slow interruption as above. If J_2 does not satisfy (7.4), then*

1. $s(J_1) \geq w_1$
2. $w_f > \frac{1}{2}x$
3. $r_1 + w_1 > \frac{1}{2}x$
4. $f \in \{k-1, k\}$

Proof. After the above transfers, we have $K(J_2) = \frac{3}{2}r_f + w_1 + 2D_2$.

1. Suppose $r_1 < w_1$. Then $w_1 + D_2 = w_1 + r_f + w_f - r_1 - w_1 \geq w_f - r_1 \geq \frac{1}{2}(w_1 + w_2) - r_1$, and $w_1 + D_2 \geq 0$, so J_2 satisfies (7.4).
2. Suppose $w_f \leq \frac{1}{2}x$. Then also $w_2 \leq \frac{1}{2}x$. Moreover, since $r_f + w_f > \frac{2}{3}(s+x) \geq \frac{2}{3}x$ we have $r_f > \frac{1}{6}x$. If J_2 does not satisfy (7.4), then (using item 1) $\frac{3}{2}r_f + w_1 + 2D_2 \leq \frac{1}{2}(w_2 - w_1)$ and thus $\frac{3}{2}w_1 + 2D_2 < \frac{x}{4} - \frac{x}{4} = 0$, a contradiction.
3. Suppose $r_1 + w_1 \leq \frac{1}{2}x$. We have $D_2 \geq r_f + w_2 - (r_1 + w_1) \geq r_f + w_2 - \frac{1}{2}x$. If $w_2 \geq \frac{2}{3}x$, then $2D_2 \geq 2w_2 - x \geq \frac{1}{2}w_2$. If $w_2 < \frac{2}{3}x$, then $D_2 \geq r_f + w_f - \frac{1}{2}x \geq \frac{1}{6}x$ and $2D_2 \geq \frac{1}{2}w_2$. In both cases, we find that J_2 satisfies (7.4).
4. If $f = k-2$ or $f \leq k-4$, we can take $\frac{1}{2}w_f$ out of the credit of J and still satisfy (7.4). If $f = k-3$, we can take $\frac{1}{2}w_{f+1}$ out of $K(J)$. □

J_2 may not have enough credit to pay for its completion (it does not need to pay for interruptions of J_1, or of smaller jobs that arrive before J_2 starts). The credit to pay for the completion of J_2

will have to come from another job. We will show that we can use some of the credit of J to pay for this. To begin with, we transfer $2D_2$ of credit from J_2 to J. Then, we divide the credit of J as follows. First suppose $f = k$.

$$\begin{aligned} K_{COM}(J) &= \frac{x - w_f + 4D_2}{2} \geq \frac{x + r_f - (r_1 + w_1) + 3D_2}{2} \geq \frac{1}{6}(x + r_f) + \frac{3}{2}D_2; && (11.1) \\ K_{INT}(J) &= \frac{1}{2}T_{f-1} + w_f - D_1 \geq \frac{1}{2}T_{f-1} + (w_f - r_1). && (11.2) \end{aligned}$$

If $f = k - 1$, note that J_k cannot start before time $x > w_k$ since $r_1 + w_1 + w_f > \frac{x}{2} + \frac{x}{2} = x$ by Lemma 11.3. Hence $N_{INT}(J, r_1) \leq \frac{1}{2}T_{k-2} + \frac{3}{2}w_f - r_1$. We have

$$\begin{aligned} K_{COM}(J) &= \frac{1}{2}(x - w_k) + 2D_2 \\ K_{INT}(J) &= \frac{1}{2}T_{k-1} + (w_k - D_1) \geq \frac{1}{2}T_{k-2} + (\frac{3}{2}w_k - D_1) \end{aligned}$$

so J can certainly give $\frac{1}{2}(w_k - w_f)$ to its own completion-credit, so that we again have (11.1).

We now consider the events after time r_1, using the analysis from the previous sections. Note that in those analyses, in some cases a job J' transfers its completion-credit $N_{COM}(J', s)$ to another job: this happens if J' is interrupted. However, the target job can only be a smaller job than J'. We need to keep track of the job that does not have enough completion-credit. This job will be called *red* and be denoted by J_R. Job J above, that was slowly interrupted, will be called *green* and be denoted by J_G. It satisfies (11.1) at time r_1.

Lemma 11.4 *Suppose there exists a red job J_R and a green job J_G. Until J_R completes, all jobs besides J_R and J_G satisfy (7.4), and (5.2) holds. Moreover, there will appear no further red jobs until J_R completes. J_G is the job that was slowly interrupted at time r_1, and satisfies (11.1). $K_{INT}(J_G) \geq N_{INT}(J_G, t)$ holds for all times t where an event occurs, up to and including the completion of J_R.*

Proof. We consider the events in the sequence from the first event after time r_1 until the last event before J_R completes, and use induction. At time r_1 (the base case), all the statements of the lemma hold.

Since OPT starts J_f before time r_1, it completes J_f before any job that arrives later. Since $w_R \leq w_f$ by induction, we are always in the case where OPT completes at least one $J(I)$-large job before $ARRIVE(I)$. This implies in particular that as long as J_R is not completed, there can occur no further slow interruptions where OPT does not run any large jobs before $ARRIVE$, so no further red jobs can appear.

Consider a later event. If it is an interruption (either of J_R, or of smaller jobs), consider the credit of the jobs in Q. As can be seen from the credit reassignments in lemmas 9.2 and 9.3, if J_R is interrupted, it is possible that another job in stead of J_R becomes red as a result. However, this can only be a job smaller than J_R. Furthermore, job J_G keeps satisfying (11.1) throughout such interruptions, since of J_R-large jobs only the amount of interrupt-credit can be affected. Also, the credit of J_G is not transferred to another job, so it is the same job that remains green. If there is an interruption of a job smaller than J_R, then J_R remains red for the same reason. In both cases, all other jobs satisfy (7.4) by the analyses in those lemmas, including the job that was J_R if another job is red now.

Now consider a completion of a job J' before J_R completes. Then J' and any smaller jobs that arrived while J' was running are all small relative to J_R and J_G. In Lemma 10.3, J_R and J_G are then among the large jobs whose credit increases by $\frac{1}{2}T(I')$, and remain red and green respectively. Their completion credit is unaffected. In Lemma 10.4 and Lemma 10.5, the same holds. In Lemma 10.10, the credit of J_R can be moved to another job (that thus becomes red), but then that is again a smaller job. □

At some point, the red job J_R will complete. By (7.3), as long as we maintain $K_{COM}(J) \geq \frac{1}{2}(x - b_s(J))$, we can take credit out of J to pay for the completion of J_R.

Lemma 11.5 *Suppose there is a red job. When it completes, credits can be transferred so that all jobs satisfy (7.4).*

Proof. Denote the set of jobs that arrive during J_R's final execution by $ARRIVE'$, and their total size by T'. Note that OPT completes $J_f \geq J_R$ before $ARRIVE'$. There are thus three cases.

Case 1. OPT completes exactly one J_R-large job before $ARRIVE'$ (i.e. job J_f).

By Corollary 10.1, the jobs in $ARRIVE'$ need at most $\frac{3}{2}(w'_{f'} - v'_{f'}) \leq \frac{1}{2}w_R \leq \frac{1}{2}w_f$ of credit to satisfy (7.4). We have that the credit of J increases by $\frac{1}{2}T'$.

Claim: $K(J) \geq \frac{1}{6}(x + r_f) + \frac{3}{2}D_2 + N_{INT}(J, t') + \frac{1}{2}T'$.

Proof: At the previous event, J satisfied (11.1) and $K_{INT}(J) \geq N_{INT}(J, t)$. If J_R started after time w_R, none of the jobs in $ARRIVE'$ are interruptable and the claim follows. Otherwise, note that J had enough credit to pay for interruptions of J_R until time w_R, because it was green. (By Table 1, any job following a job J^* that is interrupted after it started before time $w(J^*)$, needs to pay for this itself until time $w(J^*)$.) This credit can now instead be used for any interruptions of jobs in $ARRIVE'$ until time $2w_R$. □

Thus $\frac{1}{2}T'$ can go to the jobs in $ARRIVE'$ that need it. Moreover, since $w'_{f'} - v'_{f'} \leq \frac{1}{3}w_f \leq \frac{1}{3}x$ using Corollary 10.1, we can also take $\frac{1}{2}(w'_{f'} - v'_{f'}) \leq \frac{1}{6}x$ out of the credit of J and still have $K_{COM}(J) \geq N_{COM}(J, t)$, because we take at most half the size of a job that completes before J out of J's credit, and J actually has at least this amount of credit by (11.1).

If $f' < k'$, we have $\frac{1}{2}T' \geq w'_{f'}$ and hence $\frac{3}{2}(w'_{f'} - v'_{f'}) \leq \frac{1}{2}T' + \frac{1}{2}(w'_{f'} - v'_{f'})$, which is the amount we could take from J.

If $f' = k' > 1$, we have $\frac{1}{2}T' \geq \frac{1}{2}(w'_{f'} + w'_{f'-1})$. We can give $\frac{1}{2}w'_{f'-1}$ to $J'_{f'}$ and $\frac{1}{2}w'_{f'}$ to J'_1. Also, we give $\frac{1}{2}(w'_{f'} - v'_{f'}) \leq \frac{1}{2}w'_{f'}$ from J to J'_1. It can be seen from the proof of Lemma 10.3 that this is sufficient.

If $f' = k' = 1$, then $\frac{1}{2}T' = \frac{1}{2}w'_1$. Giving this and an additional $\frac{1}{2}(w'_1 - v'_1) \leq \min(\frac{1}{6}x, \frac{1}{2}w'_1)$ from J to J'_1 is sufficient as in the proof of Lemma 10.3.

Case 2. OPT completes two J_R-large jobs before $ARRIVE'$.

If OPT runs two large jobs other than J before $ARRIVE'$, we again have that the credit of J increases by $\frac{1}{2}T'$ and we can reason as above.

Suppose OPT completes J and J_R before $ARRIVE'$. Following the proof of Lemma 10.4, we are done immediately if $s \leq x$, so suppose $s > x$. This implies the jobs in $ARRIVE'$ are not interruptable, so $N_{INT}(J, t) \leq N_{INT}(J, s)$. Then $K(J'_i) = \frac{3}{2}(w_R + x) + \frac{1}{2}T'_i - (s + w_R) = \frac{1}{2}(T'_i + w_R) + \frac{3}{2}x - s$. This implies that as long as $s \leq \frac{3}{2}x$, we can give $\frac{1}{2}(w'_i + w_f) \geq w'_i$ to J from each job J'_i, which is sufficient: J receives in total at least T', which it lost because $s_t(J)$ increased.

Otherwise, note that we only need to find an extra $\frac{1}{2}w'_{k'}$ of credit to give to J, since J gets at least $T_{k'-1} + \frac{1}{2}w'_{k'}$ from the jobs in $ARRIVE'$.

Suppose $w_R \leq \frac{2}{3}x$. If $s \leq x + w_R$ then $K(J'_{k'}) \geq \frac{1}{2}(w_R + x) + \frac{1}{2}T' - w_R \geq \frac{1}{2}(x - w_R) + \frac{1}{2}w'_{k'} \geq \frac{1}{6}x + \frac{1}{2}w'_{k'} \geq \frac{3}{4}w'_{k'}$, and if $s > x + w_R$ then $K(J'_{k'}) \geq \frac{1}{2}s + \frac{1}{2}T' - w_R \geq \frac{1}{2}(x - w_R) + \frac{1}{2}w'_{k'} \geq \frac{3}{4}w'_{k'}$.

We can take the last $\frac{1}{4}w'_{k'} < \frac{1}{4}w_R \leq \frac{1}{6}x$ out of the credit of J, and then all the small job-delay is paid for; J still satisfies $K_{COM}(J) \geq N_{COM}(J,t)$.

Finally, if $w_R = \frac{2}{3}x + a$ for some $a > 0$, then $D_2 = (r_f + w_f) - (r_1 + w_1) \geq s + \frac{2}{3}x + a - \frac{2}{3}(s+x) \geq a$ and we can take $\frac{1}{4}w'_{k'} + \frac{3}{4}D_2$ out of $K_{COM}(J)$ itself, since we then still have $K_{COM}(J) \geq \frac{1}{2}(x - w_f) + 2D_2 - \frac{1}{4}w_f - \frac{3}{4}D_2 \geq \frac{1}{2}(x - \frac{3}{2}(w_f - D_2)) + \frac{1}{2}D_2 \geq 0$. Here we use

$$w_f - D_2 = r_1 + w_1 - r_f \leq \frac{2}{3}(s+x) - r_f \leq \frac{2}{3}x.$$

Since we take only less than half of the size of a job that completes before J out of the credit of J, we also still have $K_{COM}(J) \geq N_{COM}(J,s)$ as before. To complete the missing credit, we can take $\frac{3}{4}(w'_{k'} - a)$ out of $K(J'_{k'})$; the calculations are similar to above.

Case 3. OPT completes three or more J_R-large jobs before $ARRIVE'$. Note from the proofs of Lemmas 10.5 and 10.10 that in this case, no completion credit from J_R is required to pay for any small job-delay. Hence we are done immediately. □

References

1. C. Chekuri, R. Motwani, B. Natarajan, and C. Stein. Approximation techniques for average completion time scheduling. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*, pages 609–618. SIAM, Philadelphia, PA, 1997.
2. L. Epstein and R. van Stee. Lower bounds for single-machine on-line scheduling. In *Proc. 26th International Symposium on Mathematical Foundations of Computer Science*, pages 338–350, 2001.
3. H. Hoogeveen, C.N. Potts, and G.J. Woeginger. On-line scheduling on a single machine: maximizing the number of early jobs. *Operations Research Letters*, 27:193–197, 2000.
4. J.A. Hoogeveen and A.P.A. Vestjens. Optimal on-line algorithms for single-machine scheduling. In W. H. Cunningham, S. T. McCormick, and M. Queyranne, editors, *Integer Programming and Combinatorial Optimization, 5th International IPCO Conference, Proceedings*, volume 1084 of *Lecture Notes in Computer Science*, pages 404–414. Springer, 1996.
5. C.A. Phillips, C. Stein, and J. Wein. Scheduling jobs that arrive over time. In *Proceedings of the 4th Workshop on Algorithms and Data Structures (WADS'95)*, volume 955 of *Lecture Notes in Computer Science*, pages 86–97. Springer, 1995.
6. David B. Shmoys, Joel Wein, and David P. Williamson. Scheduling parallel machines on-line. *SIAM Journal on Computing*, 24(6):1313–1331, December 1995.
7. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.
8. W.E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3:59–66, 1956.
9. L. Stougie. Unpublished manuscript, 1995.
10. L. Stougie and A.P.A. Vestjens. Randomized on-line scheduling: How low can't you go? Unpublished manuscript.
11. Marjan van den Akker, Han Hoogeveen, and Nodari Vakhania. Restarts can help in the on-line minimization of the maximum delivery time on a single machine. *Journal of Scheduling*, 3:333–341, 2000.
12. A.P.A. Vestjens. *On-line Machine Scheduling*. PhD thesis, Technical University Eindhoven, The Netherlands, 1997.