Improved Semi-Online Makespan Scheduling with a Reordering Buffer

Hongyang Sun, Rui Fan

School of Computer Engineering, Nanyang Technological University, Singapore
{sunh0007, fanrui}@ntu.edu.sg

Abstract: We study semi-online scheduling with a reordering buffer for identical parallel machines. In this problem, jobs arrive one by one and the online algorithm is equipped with a buffer of limited size, which can be used to reorder the jobs when making scheduling decisions. When the buffer is full, one of the jobs in the buffer must be irrevocably assigned to a machine before the next job can be revealed. No preemption is allowed and the objective is to minimize the makespan. We propose an optimal online algorithm using a buffer size of $2.068m$ and a 1.5-competitive algorithm using a buffer size of $1.477m$, where m is the number of machines. Both results improve upon the best existing buffer sizes for the corresponding competitive ratios by a constant fraction of m.

Keywords: Semi-online scheduling; Reordering buffer; Competitive analysis

1 Introduction

In a classical online scheduling problem, a sequence of n jobs arrive one by one, and they need to be scheduled on m identical parallel machines. Each job must be assigned irrevocably to a machine before the next job can be revealed. No preemption is allowed, and the objective is to minimize the makespan, i.e., the maximum completion time of any job. For this classical online problem, the well-known *list scheduling* algorithm by Graham [6], which always assigns each arriving job to the machine with the least load, is $(2-1/m)$-competitive. Several improved deterministic algorithms [2, 7, 1, 5] have been obtained over the years for large m. The current best algorithm gives a competitive ratio of 1.9201 [5], and the current best lower bound is 1.88 [10].

In this paper, we consider a semi-online variant of the above scheduling problem. In this variant, jobs still arrive one by one, but an online algorithm is equipped with a buffer of limited size, say k, which can be used to reorder the jobs when making scheduling decisions. At any time when the buffer is not full, the algorithm can choose to admit the arriving job into the buffer without having to assign any job to a machine. However, when the capacity of the buffer is reached, i.e., there are already k jobs in the buffer, one of the jobs stored in the buffer or the arriving job must be irrevocably assigned to a machine before the next job can be revealed. Hence, the classical online scheduling problem can be considered as a special case of this problem with a reordering buffer of size 0.

For this semi-online problem, Kellerer et al. [8] and Zhang [11] independently showed that when $m = 2$ a competitive ratio of $4/3$ can be achieved with a reordering buffer of size $k = 1$, which is the minimum buffer size that allows reordering. The ratio $4/3$ was also shown to be a lower bound in this case even when a larger buffer is allowed [8]. Englert et al. [4] considered this problem for general m, and presented an online algorithm with competitive ratio r_m, whose value monotonically increases with m and is the solution to a certain equation. Specifically, for $m = 2$, $r_2 = 4/3$ and $\lim_{m\to\infty} r_m \approx 1.4659$. Their algorithm uses a reordering buffer of size $k = \lceil (1 + 2/r_m) \cdot m \rceil + 1$ for any $m \geq 2$. They also showed that any semi-online algorithm cannot achieve a better competitive ratio using a reordering buffer whose size does not depend on the input size, i.e., the number of jobs n.

In contrast to the classical online scheduling problem, where there is still a gap between the best competitive ratio (1.9201) and the best lower bound (1.88), the competitive ratio r_m for the semi-online problem is completely settled and is considerably below the best result in the classical setting. The challenge for this problem, therefore, has become finding the minimum buffer size k in order to admit the optimal competitive ratio [3]. It was shown in [4] that no online algorithm with a buffer size smaller than $\lfloor m/2 \rfloor$ can achieve a competitive ratio less than $3/2$. This suggests that any online algorithm will require a buffer size in the range of $[0.5m, 2.364m]$ in order to achieve the optimal competitive ratio for large m. In this paper, we take a step forward in closing this gap by presenting an online algorithm that requires a smaller buffer size. The following theorem states our main result.

Theorem 1 *We present an online algorithm that achieves the optimal competitive ratio r_m using a reordering buffer of size $k = \lceil (5 - 2r_m) \cdot m \rceil + 1$ for any $m \geq 2$.*

For large m, our algorithm uses a buffer size of $2.068m$ while the algorithm by Englert et al. needs a buffer size of $2.364m$. This improves upon the existing result by nearly $0.3m$ in the buffer size.

In addition, we consider the tradeoff between the buffer size and the competitive ratio. Specifically, to get a competitive ratio of 3/2, Englert et al. [4] presented an algorithm, which requires a reordering buffer of size $\lceil (2/3 + 2/(1 + \ln 3)) \cdot m \rceil \geq 1.619m$. Recently, Lan et al. [9] improved the buffer size to $1.5m$ while maintaining the same competitive ratio. In this paper, we present a 3/2-competitive algorithm that further improves the buffer size given in [9] by $0.023m$ for large m. The following theorem states our result in this case.

Theorem 2 *We present a 3/2-competitive online algorithm using a reordering buffer of size $k = \lceil (1 + 1/(1 + \ln 3)) \cdot m \rceil + 1 \leq 1.477m + 1$ for any $m \geq 2$.*

2 Preliminaries

In this section, we present some preliminary concepts and notations. Let $\mathcal{M} = \{M_0, M_1, \cdots, M_{m-1}\}$ denote a set of m identical parallel machines, and let $\mathcal{J} = \{J_1, J_2, \cdots, J_n\}$ denote a sequence of n jobs, which arrive one by one into the system. For convenience, let t denote the time right after job J_t has arrived but the algorithm has not done any job assignment in response to the arrival of J_t. For each machine M_i, where $0 \leq i \leq m-1$, let $L_i(t)$ denote its *load* at time t, i.e., the sum of the sizes of all jobs assigned to it after job J_t arrives. Let $T(t) = \sum_{i=0}^{m-1} L_i(t)$ denote the *total load* of all machines at time t. The *weight* w_i of machine M_i, which was first defined in [4] and will be heavily used in the analysis of the optimal online algorithm, is given by

$$w_i := \begin{cases} \frac{r_m}{m} & \text{if } 0 \leq i < \frac{r_m - 1}{r_m} \cdot m \\ \frac{r_m - 1}{i} & \text{if } \frac{r_m - 1}{r_m} \cdot m \leq i \leq m - 1 \end{cases},$$

where r_m is the optimal competitive ratio, and is the solution to the equation $\sum_{i=0}^{m-1} w_i = \lceil (r_m - 1)m/r_m \rceil \cdot r_m/m + (r_m - 1) \cdot \sum_{i=\lceil (r_m-1)m/r_m \rceil}^{m-1} 1/i = 1$. Note that in the above equation we ensure the weights of all machines sum up to 1. Solving r_m numerically shows that its value monotonically increases with m from $r_2 = 4/3$ to $\lim_{m\to\infty} r_m \approx 1.4659$ [4]. For the convenience of our analysis, we also define index i' to be

$$i' := \begin{cases} \left\lfloor \frac{r_m - 1}{r_m} \cdot m \right\rfloor & \text{if } 0 \leq i < \frac{r_m - 1}{r_m} \cdot m \\ i & \text{if } \frac{r_m - 1}{r_m} \cdot m \leq i \leq m - 1 \end{cases}.$$

The following lemma provides a useful lower bound on the sum of $w_i \cdot i'$.

Lemma 1 *For $m \geq 2$, $\sum_{i=0}^{m-1} w_i \cdot i' \geq (r_m - 1) \cdot m - 1$.*

Proof. According to definitions, $w_i \cdot i' \geq r_m - 1 - r_m/m$ if $0 \leq i < (r_m - 1) \cdot m/r_m$ and $w_i \cdot i' = r_m - 1$ if $(r_m - 1) \cdot m/r_m \leq i \leq m - 1$. Therefore, we have

$$\begin{aligned} \sum_{i=0}^{m-1} w_i \cdot i' &\geq (r_m - 1) \cdot m - r_m/m \cdot ((r_m - 1) \cdot m/r_m + 1) \\ &= (r_m - 1) \cdot m - (r_m - 1 + r_m/m) \\ &\geq (r_m - 1) \cdot m - 1. \end{aligned} \tag{1}$$

The last inequality is because $r_m - 1 + r_m/m \leq 1$, since $r_2 - 1 + r_2/2 = 1$ and $r_m - 1 + r_m/m < 1.5 - 1 + 1.5/3 = 1$ for all $m \geq 3$. □

Both algorithms we present in this paper follow the basic three-phase procedure [4]: An algorithm first starts with the *initial phase*, in which it admits k jobs into the reordering buffer without assigning any job to any machine. Then, it enters the *iterative phase*. At each step of this phase, when a new job arrives, the smallest job among the ones in the buffer and the arriving job is selected and assigned to a machine, the choice of which depends on the target competitive ratio. After all jobs have arrived, the algorithm enters the *final phase* by scheduling the remaining jobs in the buffer. The assignment scheme for this phase again depends on the target competitive ratio. Based on this general three-phase framework, the following two sections present the details of our r_m-competitive algorithm and 3/2-competitive algorithm, respectively.

Algorithm 1

Require: Buffer size $k = \lceil (5 - 2r_m) \cdot m \rceil + 1$
Ensure: Optimal competitive ratio r_m

1: *Initial phase*: Admit the first k jobs, i.e., $J_1, \cdots, J_k$, into the reordering buffer without assigning any job to any machine.
2: *Iterative phase*: At any time t right after a new job J_t arrives, where $k+1 \le t \le n$, find a job J of smallest size p among all the jobs in the buffer and J_t. Assign J to a machine M_i whose load at time t satisfies

$$L_i(t) \le w_i \cdot (T(t) + (k + 1 - 2(m - i')) \cdot p) - p,$$

where $T(t)$ denotes the total load of all machines at time t after job J_t has arrived but before job J is assigned.
3: *Final phase*: At the end of the iterative phase, k jobs remain the reordering buffer. There are two steps in the final phase.

- In the first step, the k remaining jobs are first scheduled virtually on a set $M' = \{M'_0, M'_1, \cdots, M'_{m-1}\}$ of m empty machines using the Longest Processing Time (LPT) algorithm, which assigns the jobs in descending order of size to a machine with minimum load. However, the schedule is aborted when a new job assignment will make one of the virtual machines have more than two jobs on it. Let L'_i denote the load of virtual machine M'_i at the end of this step, and assume that $L'_0 \le L'_1 \le \ldots \le L'_{m-1}$. Then for each $0 \le i \le m-1$, schedule the jobs of virtual machine M'_i on the actual machine M_i.
- In the second step, schedule the remaining jobs in the reordering buffer using the greedy algorithm, which assigns the jobs in any order to a machine with minimum load.

3 An Optimal r_m-competitive Algorithm

Our optimal competitive algorithm uses a reordering buffer of size $k = \lceil (5 - 2r_m) \cdot m \rceil + 1$, and it is described in detail in Algorithm 1. Compared to the algorithm proposed in [4], the novelty of our algorithm lies in machine selection when assigning jobs in the iterative phase. In particular, instead of using a uniform total load for all machines, our algorithm uses non-uniform loads for different machines. Based on an important observation, we show that such nonuniformity leads to the optimal competitive ratio while requiring a smaller buffer size.

Before proving the optimality of the algorithm, we first show in the following lemma that the iterative phase is always feasible.

Lemma 2 *At any time t in the iterative phase after job J_t has arrived but before job J with smallest size p is assigned, there always exists a machine M_i whose load satisfies $L_i(t) \le w_i \cdot (T(t) + (k + 1 - 2(m - i')) \cdot p) - p$, where $k = \lceil (5 - 2r_m) \cdot m \rceil + 1$.*

Proof. Suppose that the load of machine M_i is strictly greater than $w_i \cdot (T(t) + (k + 1 - 2(m - i')) \cdot p) - p$ for all $0 \le i \le m-1$. Since $\sum_{i=0}^{m-1} w_i = 1$ by definition and $\sum_{i=0}^{m-1} w_i \cdot i' \ge (r_m - 1) \cdot m - 1$ from Lemma 1, the total load of all machines at time t satisfies

$$\begin{aligned}
T(t) &> \sum_{i=0}^{m-1} (w_i \cdot (T(t) + (k + 1 - 2(m - i')) \cdot p) - p) \\
&= T(t) + \sum_{i=0}^{m-1} (w_i \cdot (k + 1 - 2(m - i')) \cdot p) - m \cdot p \\
&= T(t) + \left(k + 1 - 2m + 2 \sum_{i=0}^{m-1} w_i \cdot i' \right) \cdot p - m \cdot p \\
&\ge T(t) + ((3 - 2r_m) \cdot m + 2(r_m - 1) \cdot m) \cdot p - m \cdot p \\
&= T(t),
\end{aligned}$$

which yields a contradiction. Hence, at least one machine satisfies the target load at any time, so the iterative phase is always feasible. □

Now, we will prove a sequence of lemmas in the following, which will bound the load of any machine M_i at the end of each step of the online algorithm. The last lemma of the section (Lemma 5) will then imply

Theorem 1 and hence the optimality of the algorithm. For convenience, let $n+1$ denote the time after the iterative phase has finished, and let t_1 and t_2 denote the time at the end of the first step and the second step of the final phase, respectively.

Lemma 3 *At time $n+1$ after the iterative phase has ended, the load of each machine M_i satisfies $L_i(n+1) \leq w_i \cdot (T(n+1) + (k-2(m-i')) \cdot p_{\min})$, where $p_{\min}$ is the smallest size of all jobs remaining in the buffer.*

Proof. Let $t \leq n+1$ denote the time right before the last job J is scheduled on machine M_i, and let p denote the size of J. Apparently, we have $T(t) + p \leq T(n+1)$, and since the algorithm always assigns the smallest job during the iterative phase, we also have $p \leq p_{\min}$. Because job J is assigned to machine M_i, according to the choice of the algorithm and Lemma 2, the load of machine M_i at time t must satisfy $L_i(t) \leq w_i \cdot (T(t) + (k+1-2(m-i')) \cdot p) - p$. Also because $k - 2(m-i') \geq 0$ for all i' and $m \geq 2$, the load of the machine at the end of the iterative phase satisfies $L_i(n+1) = L_i(t) + p \leq w_i \cdot (T(t) + (k+1-2(m-i')) \cdot p) \leq w_i \cdot (T(n+1) + (k-2(m-i')) \cdot p_{\min})$. □

Lemma 4 *At time t_1 after the first step of the final phase has ended, the load of each machine M_i satisfies $L_i(t_1) \leq r_m \cdot OPT$, where OPT denotes the minimum makespan achieved by an optimal offline algorithm.*

Proof. Recall that in the first step of the final phase, the k remaining jobs in the reordering buffer are first scheduled by the LPT algorithm on m virtual machines, and this process is aborted with each virtual machine having at most 2 jobs on it. Since it is well known that LPT produces an optimal schedule if at most two jobs are assigned to each machine, we have $L'_i \leq OPT$ for each $0 \leq i \leq m-1$. Let $\mathcal{J}_i$ denote the set of jobs in the reordering buffer which are not scheduled on virtual machines M'_i to M'_{m-1}. Then, we have $|\mathcal{J}_i| \geq k - 2(m-i)$. Let p_j denote the size of any job $J_j \in \mathcal{J}_i$, and we have that p_j is at least the smallest size $p_{\min}$ of all jobs in the buffer. Therefore, the makespan of the optimal schedule can be bounded by

$$\begin{aligned} OPT &\geq \frac{T(n+1) + \sum_{l=i}^{m-1} L'_l + \sum_{J_j \in \mathcal{J}_i} p_j}{m} \\ &\geq \frac{T(n+1) + (m-i) \cdot L'_i + (k - 2(m-i)) \cdot p_{\min}}{m}, \end{aligned} \tag{2}$$

which holds for each $0 \leq i \leq m-1$. We now bound the loads of the machines at the end of the first step for the following two cases.

Case (1): For each $\frac{r_m - 1}{r_m} \cdot m \leq i \leq m-1$, we have $w_i = \frac{r_m - 1}{i}$ and $i' = i$ according to the definitions of w_i and i'. Based on Lemma 3 and Inequality (2), the load of machine M_i at time t_1 is given by

$$\begin{aligned} L_i(t_1) &= L_i(n+1) + L'_i \\ &\leq w_i \cdot (T(n+1) + (k - 2(m-i)) \cdot p_{\min}) + L'_i \\ &\leq \frac{r_m - 1}{i} \cdot (m \cdot OPT - (m-i) \cdot L'_i) + L'_i \\ &= \frac{(r_m - 1) \cdot m}{i} \cdot (OPT - L'_i) + r_m \cdot L'_i \\ &\leq r_m \cdot (OPT - L'_i) + r_m \cdot L'_i \\ &= r_m \cdot OPT. \end{aligned}$$

Case (2): For each $0 \leq i < \frac{r_m - 1}{r_m} \cdot m$, we have $w_i = \frac{r_m}{m}$ and $i' = \left\lfloor \frac{r_m - 1}{r_m} \cdot m \right\rfloor$. Define $l = \left\lfloor \frac{r_m - 1}{r_m} \cdot m \right\rfloor \leq \frac{r_m - 1}{r_m} \cdot m$, so we have $L'_i \leq L'_l$. Also from Lemma 3 and Inequality (2), the load of machine M_i at time t_1 is given by

$$\begin{aligned} L_i(t_1) &= L_i(n+1) + L'_i \\ &\leq w_i \cdot (T(n+1) + (k - 2\,(m-l)) \cdot p_{\min}) + L'_i \\ &\leq \frac{r_m}{m} \cdot (m \cdot OPT - (m-l) \cdot L'_l) + L'_l \\ &\leq \frac{r_m}{m} \cdot \left(m \cdot OPT - \frac{1}{r_m} \cdot m \cdot L'_l \right) + L'_l \\ &= r_m \cdot OPT. \end{aligned}$$

□

Lemma 5 *At time t_2 after the second step of the final phase has ended, the load of each machine M_i satisfies $L_i(t_2) \leq r_m \cdot OPT$, where OPT denotes the minimum makespan achieved by an optimal offline algorithm.*

Proof. Suppose that l largest jobs out of the k remaining jobs in the buffer are scheduled during the first step of the final phase by the LPT algorithm. Now, consider the $l+1$ largest jobs in the reordering buffer before the final phase begins. An optimal schedule for these $l+1$ jobs must have at least three jobs on at least one of the machines, otherwise the LPT algorithm would not have stopped at the $(l+1)$-th job. Let p_{l+1} denote the size of the $(l+1)$-th largest job. Because all jobs in this optimal schedule have size at least p_{l+1}, the optimal makespan satisfies $OPT \geq 3p_{l+1}$.

In the second step of the final phase, the $k-l$ remaining jobs in the reordering buffer are scheduled by the greedy algorithm. If a machine M_i does not have any job scheduled on it during this step, its final load according to Lemma 4 satisfies $L_i(t_2) = L_i(t_1) \leq r_m \cdot OPT$. Otherwise, let t denote the time when the last job J of size p is about to be scheduled on machine M_i, where $t_1 < t \leq t_2$. According to the greedy algorithm, machine M_i must have the minimum load $L_i(t)$ among all machines at time t. Therefore, we must have $L_i(t) \leq OPT$, since the average load at any time is always a lower bound on the optimal makespan. Also, because the size of job J satisfies $p \leq p_{l+1} \leq OPT/3 \leq (r_m - 1) \cdot OPT$, the load of machine M_i at the end of the second step satisfies $L_i(t_2) = L_i(t) + p \leq OPT + (r_m - 1) \cdot OPT = r_m \cdot OPT$. □

4 A 3/2-competitive Algorithm

In this section, we present a 3/2-competitive algorithm, which uses a reordering buffer of size $k = \lceil (1 + 1/(2a)) \cdot m \rceil + 1 \leq 1.477m + 1$, where $a = (1 + \ln 3)/2 \approx 1.05$. The algorithm adopts of the idea of the optimal competitive algorithm by selecting machines in the iterative phase according to nonuniform total loads. A similar idea has been used in [9] to show a 3/2-competitive algorithm with a reordering buffer of size $1.5m$.

Algorithm 2 shows the details of our algorithm. For presentation purpose, we redefine the *weight* w_i of machine M_i to be

$$w_i := \begin{cases} \frac{3}{2m} & \text{if } 0 \leq i < \frac{m}{3} \\ \frac{1}{2i} & \text{if } \frac{m}{3} \leq i \leq m-1 \end{cases}.$$

The sum of the weights of all machines is then given by $\sum_{i=0}^{m-1} w_i = \lceil m/3 \rceil \cdot 3/(2m) + \sum_{i=\lceil m/3 \rceil}^{m-1} 1/(2i)$, which monotonically decreases with m according to [4]. Therefore, given that $\lim_{m\to\infty} \sum_{i=0}^{m-1} w_i = 1/2 - \ln(1/3)/2 = (1 + \ln 3)/2 = a$, we have $\sum_{i=0}^{m-1} w_i \geq a$ for all $m \geq 2$. Also, we redefine the index i' to be

$$i' := \begin{cases} \lfloor \frac{m}{3} \rfloor & \text{if } 0 \leq i < \frac{m}{3} \\ i & \text{if } \frac{m}{3} \leq i \leq m-1 \end{cases}.$$

Similarly to Lemma 1, the sum of $w_i \cdot i'$ can be shown to satisfy the following inequality for any $m \geq 2$,

$$\begin{aligned} \sum_{i=0}^{m-1} w_i \cdot i' &\geq m/2 - 3/(2m) \cdot (m/3 + 1) \\ &= m/2 - (1/2 + 3/(2m)) \\ &\geq m/2 - 5/4. \end{aligned} \tag{3}$$

First, the following lemma shows that the iterative phase of our algorithm is always feasible.

Lemma 6 *At any time t in the iterative phase after job J_t has arrived but before job J with smallest size p is assigned, there always exists a machine M_i whose load satisfies $L_i(t) \leq w_i \cdot (T(t) + (k + 1 - (m - i')) \cdot p) - p$, where $k = \lceil (1 + 1/(2a)) \cdot m \rceil + 1$ and $a = (1 + \ln 3)/2$.*

Proof. Suppose that the load of machine M_i is strictly greater than $w_i \cdot (T(t) + (k + 1 - (m - i')) \cdot p) - p$ for

Algorithm 2

Require: Buffer size $k = \lceil (1 + 1/(2a)) \cdot m \rceil + 1$ where $a = (1 + \ln 3)/2$
Ensure: Competitive ratio 3/2

1: *Initial phase*: Admit the first k jobs, i.e., $J_1, \cdots, J_k$, into the reordering buffer without assigning any job to any machine.
2: *Iterative phase*: At any time t right after a new job J_t arrives, where $k+1 \le t \le n$, find a job J of smallest size p among all the jobs in the buffer and J_t. Assign J to a machine M_i whose load at time t satisfies

$$L_i(t) \le w_i \cdot (T(t) + (k + 1 - (m - i')) \cdot p) - p,$$

where $T(t)$ denotes the total load of all machines at time t after job J_t has arrived but before job J is assigned.
3: *Final phase*: At the end of the iterative phase, k jobs remain in the reordering buffer. There are two steps in the final phase.

- In the first step, m largest jobs are selected from the k remaining jobs in the reordering buffer. Let $\{J'_0, J'_1, \cdots, J'_{m-1}\}$ denote these jobs, and we assume that they are sorted in ascending order of size, i.e., $p'_0 \le p'_1 \le \cdots \le p'_{m-1}$. Assign job J'_i to machine M_i for each $0 \le i \le m-1$.
- In the second step, schedule the remaining $k - m$ jobs in the reordering buffer using the greedy algorithm, which assigns the jobs in any order to a machine with minimum load.

all $0 \le i \le m-1$. Since $\sum_{i=0}^{m-1} w_i \ge a > 1$ and $\sum_{i=0}^{m-1} w_i \cdot i' \ge m/2 - 5/4$ from Inequality (3), we have

$$\begin{aligned}
T(t) &> \sum_{i=0}^{m-1} (w_i \cdot (T(t) + (k + 1 - (m - i')) \cdot p) - p) \\
&> T(t) + \sum_{i=0}^{m-1} (w_i \cdot (k + 1 - (m - i')) \cdot p) - m \cdot p \\
&= T(t) + \left((k + 1 - m) \cdot a + \sum_{i=0}^{m-1} w_i \cdot i' \right) \cdot p - m \cdot p \\
&\ge T(t) + ((m/(2a) + 2) \cdot a + m/2 - 5/4) \cdot p - m \cdot p \\
&> T(t),
\end{aligned}$$

which yields a contradiction. □

As with the analysis of the optimal online algorithm, we bound the load of any machine M_i at the end of each step in the following lemmas. In particular, Lemma 9 implies Theorem 2 and shows that our algorithm is 3/2-competitive. Again, let $n+1$ denote the time after the iterative phase has finished, and let t_1 and t_2 denote the time at the end of the first step and the second step of the final phase, respectively.

Lemma 7 *At time $n+1$ after the iterative phase has ended, the load of each machine M_i satisfies $L_i(n+1) \le w_i \cdot (T(n+1) + (k - (m - i')) \cdot p_{\min})$, where $p_{\min}$ is the smallest size of all jobs remaining in the buffer.*

Proof. Let $t \le n+1$ denote the time right before the last job J is scheduled on machine M_i, and let p denote the size of J. We have $T(t) + p \le T(n+1)$ and $p \le p_{\min}$. According to the algorithm and Lemma 6, the load of machine M_i at time t satisfies $L_i(t) \le w_i \cdot (T(t) + (k + 1 - (m - i')) \cdot p) - p$. Because $k - (m - i') \ge 0$, the load of the machine at the end of the iterative phase satisfies $L_i(n+1) = L_i(t) + p \le w_i \cdot (T(t) + (k + 1 - (m - i')) \cdot p) \le w_i \cdot (T(n+1) + (k - (m - i')) \cdot p_{\min})$. □

Lemma 8 *At time t_1 after the first step of the final phase has ended, the load of each machine M_i satisfies $L_i(t_1) \le 3/2 \cdot OPT$, where OPT denotes the minimum makespan achieved by an optimal offline algorithm.*

Proof. In the first step of the final phase, m largest jobs out of the k remaining jobs are scheduled on the m machines by assigning job J'_i to machine M_i for each $0 \le i \le m-1$. Let $p_{\min}$ denote the smallest size of all jobs that remain in the reordering buffer. The makespan of the optimal schedule can be bounded by

$$OPT \ge \frac{T(n+1) + (m - i) \cdot p'_i + (k - (m - i)) \cdot p_{\min}}{m}, \tag{4}$$

which holds for each $0 \leq i \leq m-1$. We again consider two cases.

Case (1): For each $m/3 \leq i \leq m-1$, we have $w_i = 1/(2i)$ and $i' = i$. Based on Lemma 7 and Inequality (4), the load of machine M_i at time t_1 is given by

$$\begin{aligned} L_i(t_1) &= L_i(n+1) + p'_i \\ &\leq w_i \cdot (T(n+1) + (k - (m-i)) \cdot p_{\min}) + p'_i \\ &\leq \frac{1}{2i} \cdot (m \cdot OPT - (m-i) \cdot p'_i) + p'_i \\ &= \frac{m}{2i} \cdot (OPT - p'_i) + \frac{3}{2} \cdot p'_i \\ &\leq \frac{3}{2} \cdot (OPT - p'_i) + \frac{3}{2} \cdot p'_i \\ &= \frac{3}{2} \cdot OPT. \end{aligned}$$

Case (2): For each $0 \leq i < m/3$, we have $w_i = 3/(2m)$ and $i' = \lfloor m/3 \rfloor \leq m/3$. The load of machine M_i at time t_1 is given by

$$\begin{aligned} L_i(t_1) &= L_i(n+1) + p'_i \\ &\leq w_i \cdot (T(n+1) + (k - (m - \lfloor m/3 \rfloor)) \cdot p_{\min}) + p'_i \\ &\leq \frac{3}{2m} \cdot \left(m \cdot OPT - (m - \lfloor m/3 \rfloor) \cdot p'_{\lfloor m/3 \rfloor}\right) + p'_{\lfloor m/3 \rfloor} \\ &\leq \frac{3}{2m} \cdot \left(m \cdot OPT - \frac{2}{3} \cdot m \cdot p'_{\lfloor m/3 \rfloor}\right) + p'_{\lfloor m/3 \rfloor} \\ &= \frac{3}{2} \cdot OPT. \end{aligned}$$

□

Lemma 9 *At time t_2 after the second step of the final phase has ended, the load of each machine M_i satisfies $L_i(t_2) \leq 3/2 \cdot OPT$, where OPT denotes the minimum makespan achieved by an optimal offline algorithm.*

Proof. Consider the $m+1$ largest jobs that remain in the reordering buffer before the final phase begins, and let p_{m+1} denote the size of the smallest job in this set. An optimal schedule for this set of jobs must have exactly two jobs on one of the machines. Since all jobs in this set have size at least p_{m+1}, the optimal makespan satisfies $OPT \geq 2p_{m+1}$.

In the second step of the final phase, the $k-m$ remaining jobs in the reordering buffer are scheduled by the greedy algorithm. If a machine M_i does not have any job scheduled on it during this step, its final load according to Lemma 8 satisfies $L_i(t_2) = L_i(t_1) \leq 3/2 \cdot OPT$. Otherwise, let t denote the time when the last job J of size p is about to be scheduled on machine M_i, where $t_1 < t \leq t_2$. Then, we must have $L_i(t) \leq OPT$, and the size of job J satisfies $p \leq p_{m+1} \leq OPT/2$. The load of machine M_i at the end of the second step satisfies $L_i(t_2) = L_i(t) + p \leq OPT + 1/2 \cdot OPT = 3/2 \cdot OPT$. □

References

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29(2):459–473, 1999.

[2] Y. Bartal, A. Fiat, H. J. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *Journal of Computer and System Sciences*, 51(3):359–366, 1995.

[3] M. Englert. An overview of some results for reordering buffers. *Computer Science - Research and Development*, 27(3):217-223, 2012.

[4] M. Englert, D. Özmen, and M. Westermann. The power of reordering for online minimum makespan scheduling. In *FOCS*, pages 603–612, Philadelphia, PA, USA, 2008.

[5] R. Fleischer and M. Wahl. On-line scheduling revisited. *Journal of Scheduling*, 16(3):554–560, 2000.

[6] R. L. Graham. Bounds on multiprocessing anomalies. *SIAM Journal on Applied Mathematics*, 17(2):416–429, 1969.

[7] D. R. Karger, S. J. Phillips, and E. Torng. A better algorithm for an ancient scheduling problem. *Journal of Algorithms*, 21(5):235–242, 1997.

[8] H. Kellerer, V. Kotov, M. G. Speranza, and Z. Tuza. Semi on-line algorithms for the partition problem. *Operations Research Letters*, 21(5):235–242, 1997.

[9] Y. Lan, X. Chen, N. Ding, G. Dósa, X. Han. Online Minimum Makespan Scheduling with a Buffer. *FAW-AAIM*, pages 161–171, Beijing, China, 2012.

[10] J. F. Rudin III. *Improved Bound for the Online Scheduling Problem*. PhD thesis, University of Texas at Dallas, 2001.

[11] G. Zhang. A simple semi on-line algorithm for $P2||C_{max}$ with a buffer. *Information Processing Letter*, 61(3):145–148, 1997.