On-line scheduling of parallel jobs on two machines ☆

Wun-Tat Chan [a,*,1], Francis Y.L. Chin [a,2], Deshi Ye [a], Guochuan Zhang [b,3], Yong Zhang [a]

[a] *Department of Computer Science, The University of Hong Kong, Pokfulam Road, Hong Kong*
[b] *Department of Mathematics, Zhejiang University, Hangzhou 310027, China*

Abstract

We study the problem of on-line scheduling of parallel jobs on two machines. The jobs are parallel in the sense that each of them specifies the number of processors, in this case 1 or 2, required for simultaneous processing. The jobs are presented one by one. Upon receiving a job, we must assign the job to a time slot in the schedule before the next job is presented. No re-assignment is allowed. The goal is to minimize the makespan of the final schedule. There is a straightforward algorithm which achieves a competitive ratio of 2. In this paper we show that no on-line algorithm can have a competitive ratio less than $1 + \sqrt{2/3}$ (≈ 1.816). We also study two special cases of the problem: (i) Jobs arrive in a non-decreasing order of processing times where we give an optimal algorithm with competitive ratio $3/2$; (ii) Jobs arrive in a non-increasing order of processing times where we show that no on-line algorithm has a competitive ratio less than $9/7$ and give a greedy algorithm with a competitive ratio $4/3$.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Jobs scheduling; Online algorithms; Competitive Analysis

1. Introduction

In the classical job scheduling, each job requires only a single machine (or processor) for processing. However, this assumption may not apply to modern computer systems, especially in the parallel supercomputers, where some jobs can only be processed on several processors in parallel. The scheduling model for *parallel jobs* has been proposed and studied extensively [1–7] in recent years. The problem we study in this paper can be described as follows. A job $j = (s, p)$ is associated with two parameters p and s, where p is the processing time of the job and s is the number of machines required for simultaneous processing. The jobs are presented one by one. The problem is on-line in the sense that upon receiving a job and before the next job is presented, we need to assign the job to a time slot of the schedule so that the required number of machines is available. No splitting of jobs (i.e., preemption) and no re-assignment is

---

allowed. The goal is to minimize the *makespan* of the final schedule, i.e., the completion time of the last job to finish in the schedule. In this paper we focus on the case of two machines and therefore each job can request for either 1 or 2 machines for processing, which is denoted as the *1-machine job* or *2-machine job*, respectively.

We use the competitive analysis [8] to measure the performance of an on-line algorithm. For any input job sequence I, let $C_A(I)$ denote the makespan of the schedule produced by the on-line algorithm A and $C_{OPT}(I)$ denote the makespan of the optimal schedule. We say that A is *k-competitive* if

$$\sup_I \left\{ \frac{C_A(I)}{C_{OPT}(I)} \right\} \leqslant k.$$

We also say that k is the competitive ratio of A. Notice that for any input job sequence, a k-competitive on-line algorithm can always give a schedule with makespan at most k times that of the optimal schedule. For simplicity, we use C_A and C_{OPT} instead of $C_A(I)$ and $C_{OPT}(I)$ if there is no confusion.

If in our problem all jobs require one machine only, it is the traditional job scheduling problem in multiple machines, first investigated by Graham [9]. A fundamental algorithm called the List Scheduling algorithm [9] is to schedule a newly arrived job to a machine in which the job can start as early as possible. By Graham's analysis, List Scheduling algorithm can be shown to be $(2 - 1/m)$-competitive, where m is the number of machines. Faigle et al. [10] showed that the List Scheduling algorithm is in fact an optimal algorithm for 2 and 3 machines.

In general, if a job can request any number of available machines, Johannes presented an on-line algorithm with competitive ratio 12 [7]. She also proved that no on-line algorithm is better than 2.25-competitive [7]. Ye and Zhang improved the upper bound by giving an 8-competitive on-line algorithm [6], which was then further improved by Ye with a 7-competitive on-line algorithm [11]. There were also some results for the cases where some extra information on the jobs is known in advance. If the jobs arrive in a non-increasing order of processing times, Ye and Zhang gave an on-line algorithm with competitive ratio 2 [6]. If the longest processing time is known, Ye gave an on-line algorithm with competitive ratio 4 [11].

In this paper we investigate the two machines case in scheduling parallel jobs. We prove a lower bound of $1 + \sqrt{2/3}$ on the competitive ratio of any on-line algorithm. We also study two special cases of the problem and give a constant competitive on-line algorithm for each of the cases. (i) When the jobs arrive in a non-decreasing order of processing times, we give an optimal algorithm which is $(3/2)$-competitive. (ii) When the jobs arrive in a non-increasing order of processing times, we show that no on-line algorithm has a competitive ratio less than $9/7$ and we give a $(4/3)$-competitive on-line algorithm.

The rest of the paper is organized as follows. In Section 2 we give an adversary to show the lower bound of $1 + \sqrt{2/3}$. In Section 3 we present a greedy on-line algorithm. For each of two special case inputs where the jobs arrive in either non-decreasing or non-increasing order of the processing times, we show that the greedy algorithm is constant competitive. The greedy algorithm is indeed optimal in the first case.

2. Lower bound

In this section we show that for scheduling parallel jobs on two machines, no on-line algorithm can achieve a competitive ratio less than $1 + \sqrt{2/3}$ (≈ 1.816). Before we prove the lower bound, we point out that, on the other hand, there is a straightforward on-line algorithm that can guarantee a competitive ratio of 2. The algorithm simply schedules all jobs one after the other leaving no idle time between jobs. For the 1-machine jobs, they are all scheduled to one particular machine only. In the worst case, there is no 2-machine job. The optimal schedule has all jobs evenly scheduled in the two machines but it still requires a makespan at least half the makespan of the on-line schedule, which is equal to the total processing time.

We prove the lower bound on the competitive ratios of all on-line algorithms by giving an adversary job sequence such that any on-line algorithm ALG has a competitive ratio at least $1 + \sqrt{2/3}$. The adversary job sequence S consists of at most 5 jobs, j_1, j_2, j_3, j_4, and j_5 arriving in that order. The adversary runs as follows. For $i = 1$ to 5, after ALG schedules j_i, if the ratio of the makespans of the on-line schedule and the optimal schedule for jobs $j_1, \ldots, j_i$ is at least $1 + \sqrt{2/3}$, then the adversary stops.

We let $j_1 = (1, 1)$ where the first '1' denotes the required number of machines and the next '1' denotes the required processing time. The parameters for the subsequent jobs depend on how the on-line algorithm schedules the previous jobs. Without loss of generality, suppose an on-line algorithm schedules j_1 in Machine 1 at some time $h_1 \geqslant 0$ (see
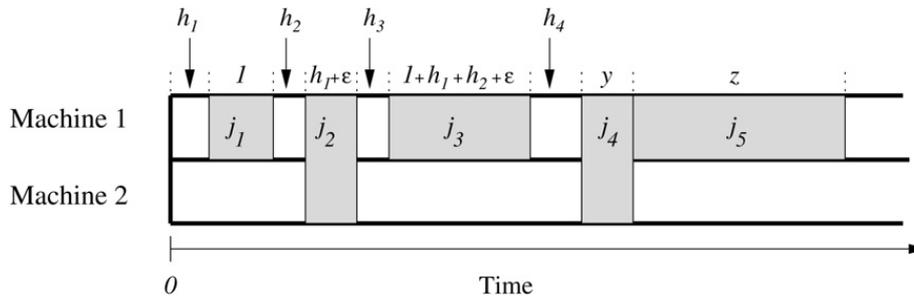
Fig. 1. An on-line schedule for the adversary job sequence S.

Fig. 1). We let $j_2 = (2, h_1 + \epsilon)$ where ϵ is a sufficiently small positive constant (the exactly value of ϵ will be known in the proof of the lower bound). Since the processing time of j_2 is larger than h_1 (the available slot before j_1), j_2 must be scheduled after j_1. Suppose j_2 is scheduled h_2 units after j_1. Next, let $j_3 = (1, 1 + h_1 + h_2 + \epsilon)$. Similarly, since j_3 cannot fit into any available slot in either machine before j_2, j_3 must be scheduled after j_2. Suppose j_3 is scheduled h_3 units after j_2. Since it is indifferent to schedule j_3 in either machine after j_2, we assume without loss of generality that j_3 is scheduled in Machine 1. Next, let $j_4 = (2, y)$ where $y = \max\{h_1, h_2, h_3\} + \epsilon$. Again, since j_4 cannot fit into any available slot in both machines before j_3, we assume j_4 is scheduled h_4 units after j_3. Finally, let $j_5 = (1, z)$ where $z = 1 + h_1 + h_2 + \max\{h_3 + h_4 + \epsilon, 1\} + \epsilon$. We can see that it is also not possible to schedule j_5 before j_4. Thus, the best way is to schedule j_5 immediate after j_4. It is also indifferent to schedule j_5 in either machine, so we assume j_5 is scheduled in Machine 1. See Fig. 1 for the on-line schedule of these jobs.

In the following lemma, we prove that no on-line algorithm can schedule the above adversary job sequence with competitive ratio less than $1 + \sqrt{2/3}$. Note that the analysis is tight because the on-line algorithm can choose $h_1 = \sqrt{2/3}$ and $h_2 = h_3 = h_4 = 0$ and achieve the competitive ratio $1 + \sqrt{2/3}$ in scheduling the adversary job sequence.

Lemma 1. *No on-line algorithm can schedule the adversary job sequence S with competitive ratio less than $1 + \sqrt{2/3}$.*

Proof. Let S_i denote the job subsequence $j_1, \ldots, j_i$. We prove that the maximum among the competitive ratios $\max\{C_{ALG}(S_i)/C_{OPT}(S_i) \mid 1 \leqslant i \leqslant 5\} \geqslant 1 + \sqrt{2/3}$. It is shown by the argument that if both $C_{ALG}(S_1)/C_{OPT}(S_1) \leqslant 1 + \sqrt{2/3} - \delta$ and $C_{ALG}(S_3)/C_{OPT}(S_3) \leqslant 1 + \sqrt{2/3} - \delta$ for some $\delta > 0$, then $C_{ALG}(S_5)/C_{OPT}(S_5) \geqslant 1 + \sqrt{2/3}$.

First, we consider S_1. The optimal schedule for S_1 has j_1 to start at time 0, so we have

$$\frac{C_{ALG}(S_1)}{C_{OPT}(S_1)} = 1 + h_1 \leqslant 1 + \sqrt{\frac{2}{3}} - \delta \Rightarrow h_1 \leqslant \sqrt{\frac{2}{3}} - \delta.$$

For the optimal schedule for S_3, j_1 and j_3 both start at time 0 but on different machines and j_2 starts at time $1 + h_1 + h_2 + \epsilon$ on both machines. Therefore, we have

$$\frac{C_{ALG}(S_3)}{C_{OPT}(S_3)} = \frac{2 + 3h_1 + 2h_2 + h_3 + 2\epsilon}{1 + 2h_1 + h_2 + 2\epsilon} \leqslant 1 + \sqrt{\frac{2}{3}} - \delta$$

$$\Rightarrow h_3 \leqslant \left(2\sqrt{\frac{2}{3}} - 1 - 2\delta\right)h_1 - \left(1 - \sqrt{\frac{2}{3}} + \delta\right)h_2 - \left(1 - \sqrt{\frac{2}{3}}(1 + 2\epsilon) + (1 + 2\epsilon)\delta\right)$$

$$\leqslant \left(2\sqrt{\frac{2}{3}} - 1 - 2\delta\right)h_1 \quad \text{let } \epsilon \leqslant \sqrt{\frac{3}{8}} - \frac{1}{2}$$

$$\leqslant h_1.$$

Since $j_4 = (2, y)$ where $y = \max\{h_1, h_2, h_3\} + \epsilon$, and $h_3 \leqslant h_1$, we can simplify $y = \max\{h_1, h_2\} + \epsilon$.

In the following we consider an offline schedule for S_5 and analyze the ratio of the makespans of the on-line schedule and this offline schedule. Clearly, this ratio will be at most the competitive ratio of the on-line schedule. The offline schedule has j_2 to start at time 0, j_4 at time $h_1 + \epsilon$. Then j_1 and j_5 are scheduled at time $h_1 + y + \epsilon$ in Machine 1 and Machine 2, respectively. Finally, j_3 is scheduled at time $1 + h_1 + y + \epsilon$ in Machine 1. See Fig. 2 for the offline schedule.
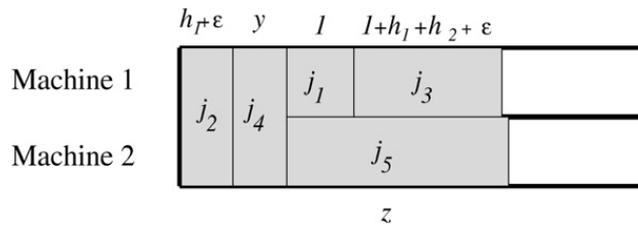
Fig. 2. An offline schedule for the adversary job sequence S.

We present the analysis case by case and show that $C_{ALG}(S_5)/C_{OPT}(S_5) \geqslant 1 + \sqrt{2/3}$.

Case 1. Assume $h_3 + h_4 + \epsilon \geqslant 1$. Thus we have $j_5 = (1, z)$ with $z = 1 + h_1 + h_2 + h_3 + h_4 + 2\epsilon$. The competitive ratio is

$$\frac{C_{ALG}(S_5)}{C_{OPT}(S_5)} \geqslant \frac{3 + 4h_1 + 3h_2 + 2h_3 + 2h_4 + y + 4\epsilon}{1 + 2h_1 + h_2 + h_3 + h_4 + y + 3\epsilon}$$

$$\geqslant \min\left\{2, \frac{1 + h_2 + y + 4\epsilon}{y + 3\epsilon}\right\}$$

$$\geqslant \min\left\{2, \frac{h_1 + h_2 + \max\{h_1, h_2\} + 1 - \sqrt{2/3} + \delta + 5\epsilon}{\max\{h_1, h_2\} + 4\epsilon}\right\} \quad \text{as } h_1 \leqslant \sqrt{\frac{2}{3}} - \delta$$

$$= 2 \quad \text{let } \epsilon \leqslant \frac{1 - \sqrt{2/3} + \delta}{3}$$

$$\geqslant 1 + \sqrt{\frac{2}{3}}.$$

Case 2. Assume $h_3 + h_4 + \epsilon < 1$. Thus we have $j_5 = (1, z)$ with $z = 2 + h_1 + h_2 + \epsilon$. The competitive ratio is

$$\frac{C_{ALG}(S_5)}{C_{OPT}(S_5)} = \frac{4 + 4h_1 + 3h_2 + h_3 + h_4 + y + 3\epsilon}{2 + 2h_1 + h_2 + y + 2\epsilon} \geqslant \frac{4 + 4h_1 + 3h_2 + y + 3\epsilon}{2 + 2h_1 + h_2 + y + 2\epsilon}.$$

Case 2.1. Assume $h_1 \leqslant h_2$. Thus $y = h_2 + \epsilon$, and we have

$$\frac{C_{ALG}(S_5)}{C_{OPT}(S_5)} = \frac{4 + 4h_1 + 4h_2 + 4\epsilon}{2 + 2h_1 + 2h_2 + 3\epsilon} \geqslant 1 + \sqrt{\frac{2}{3}} \quad \text{let } \epsilon \leqslant \frac{2 - 2\sqrt{2/3}}{\sqrt{6} - 1}.$$

Case 2.2. Assume $h_1 > h_2$. Thus $y = h_1 + \epsilon$, and we have

$$\frac{C_{ALG}(S_5)}{C_{OPT}(S_5)} = \frac{4 + 5h_1 + 3h_2 + 4\epsilon}{2 + 3h_1 + h_2 + 3\epsilon}$$

$$\geqslant \frac{4 + 5h_1 + 4\epsilon}{2 + 3h_1 + 3\epsilon} \quad \text{the function is decreasing w.r.t. } h_1 \text{ for } \epsilon \leqslant 2/3$$

$$\geqslant \frac{4 + 5(\sqrt{2/3} - \delta) + 4\epsilon}{2 + 3(\sqrt{2/3} - \delta) + 3\epsilon} \quad \text{as } h_1 \leqslant \sqrt{\frac{2}{3}} - \delta$$

$$\geqslant 1 + \sqrt{\frac{2}{3}} \quad \text{let } \epsilon \leqslant \frac{\delta(\sqrt{6} - 2)}{\sqrt{6} - 1}.$$

In conclusion, we prove that any on-line algorithm has the competitive ratio at least $1 + \sqrt{2/3}$ in scheduling the adversary job sequence S. □

By the above lemma, we have actually proved the following theorem that no on-line algorithm can achieve a competitive ratio less than $1 + \sqrt{2/3}$ in scheduling parallel jobs on two machines.

Theorem 2. *For the problem of scheduling parallel jobs on two machines, no on-line algorithm is better than $(1 + \sqrt{2/3})$-competitive.*

3. Upper bounds on two special cases

In this section we study the problem with special case inputs. Section 3.1 considers the job sequences with jobs arriving in a non-decreasing order of processing times. Section 3.2 considers the job sequences with jobs arriving in a non-increasing order of processing times. In these two sections, we analyse the performance of a *Greedy Algorithm* (defined below) and prove that its competitive ratios are 3/2 and 4/3, respectively. In Section 3.2 we also show that if the jobs arrive in a non-increasing order of processing times, no on-line algorithm can achieve a competitive ratio less than 9/7.

In the following, we define how the Greedy Algorithm schedules a job.

- If the job is $(1, p)$, i.e., a 1-machine job with processing time p, then schedule the job at the earliest time t to a machine where the machine is idle throughout the period from time t to $t + p$. If both machines satisfy the requirement for the same earliest time t, then schedule the job to Machine 1 at time t.
- If the job is $(2, p)$, i.e., a 2-machine job with the processing time p, then schedule the job at the earliest time t when both machines are idle throughout the period from time t to $t + p$.

We observe some properties of the greedy schedule. First, there is no time instance in the schedule (from time 0 to the makespan of the schedule) with both machines being idle. Without loss of generality, we can assume that the last job finished in the greedy schedule is a 1-machine job, which is the worst case scenario. Otherwise, removing that job from the input sequence results in a greedy schedule having a larger competitive ratio.

We need some definitions for further discussion. An *idle slot x* of a machine in the greedy schedule is defined to be a period where the machine is idle and the machine is scheduled to run some jobs immediate before (except for the first idle slot which may start at time 0) and after the period. In fact the job scheduled immediate after the idle slot must be a 2-machine job, otherwise, the job can start earlier which contradicts to the Greedy Algorithm. Let x_i denote the i-th idle slot of the machines in the greedy schedule ordered by the start time of the slot. For simplicity, x_i also denotes the length of the idle slot.

3.1. Jobs arrive in non-decreasing order of processing times

In this section we show that the Greedy Algorithm is $(3/2)$-competitive if the jobs arrive in a non-decreasing order of processing times. In fact, Greedy Algorithm is optimal because no on-line algorithm can achieve a competitive ratio smaller than 3/2 as proved by Faigle et al. [10].

Theorem 3. *Greedy Algorithm is $(3/2)$-competitive in scheduling parallel jobs on two machines if the jobs arrive in a non-decreasing order of processing times.*

Proof. In the greedy schedule, for every idle slot x_i, let y_i denote both the 2-machine job and its processing time that starts immediate after x_i. Let ℓ denote both the last (1-machine) job to finish in the greedy schedule and its processing time. Suppose there are k idle slots in the greedy schedule. Let $X = \sum_{i=1}^{k} x_i$ be the total length of the idle slots and $Y = \sum_{i=1}^{k} y_i$ be the total processing time of the 2-machine jobs starting immediate after the idle slots. Let C_{Greedy} denote the makespan of the greedy schedule.

Define the total weighted processing time to be the sum of the weighted processing time of all jobs where a w-machine job has weight w, for $w = 1$ or 2. Note that any schedule has makespan at least half of the total weighted processing time. By the greedy schedule, the total weighted processing time is at least $2C_{Greedy} - X - \ell$. Let C_{OPT} denote the makespan of the optimal schedule. We have $C_{OPT} \geqslant C_{Greedy} - (X + \ell)/2$. From another point of view, any schedule must have the 1-machine and 2-machine jobs processed in different time slots, in particular consider the jobs y_i and the last job ℓ. Thus the makespan of any schedule, including the optimal schedule, must be at least $Y + \ell$. As a result, we have $C_{OPT} \geqslant \max(C_{Greedy} - (X + \ell)/2, Y + \ell)$.

We show that $C_{OPT} \geqslant \max(C_{Greedy} - (X + \ell)/2, Y + \ell) \geqslant (2/3)C_{Greedy}$ by considering the following two cases.

(1) Assume $(X + \ell)/2 \leqslant C_{Greedy}/3$. We have $C_{OPT} \geqslant C_{Greedy} - (X + \ell)/2 \geqslant (2/3)C_{Greedy}$.

(2) Assume $(X + \ell)/2 > C_{Greedy}/3$. We have $\ell > (2/3)C_{Greedy} - X \geqslant (2/3)C_{Greedy} - Y$ because each job y_i arrives after x_i which implies the processing time $y_i \geqslant x_i$. Therefore $C_{OPT} \geqslant Y + \ell > Y + (2/3)C_{Greedy} - Y = (2/3)C_{Greedy}$.

As a result, we have in any case $C_{OPT} \geqslant (2/3)C_{Greedy}$, i.e., the competitive ratio of the Greedy Algorithm is at most $3/2$. □

3.2. Jobs arrive in non-increasing order of processing times

In this section we show that the Greedy Algorithm is $(4/3)$-competitive if the jobs arrive in a non-increasing order of processing times and we give a job sequence to illustrate that the analysis is tight. Moreover, we also show that no on-line algorithm can achieve a competitive ratio less than $9/7$.

Firstly, we assume that the last job to finish in the greedy schedule does not start at time 0, otherwise, both the greedy and the optimal schedules have the same makespan. For subsequent discussion, we define the *all-busy slot* in the greedy schedule to be a period where both machines are processing some jobs and either of the two machines is idle immediate before and immediate after the period. Note that for ease of explanation we consider both machines are idle before time 0 and after the makespan. Let z_i denote the ith all-busy slot ordered by the start time of the slot. For simplicity, z_i also denotes the length of the slot. The following lemma reveals the arrangement of the all-busy slots and the idle slots. It also implies that the number of all-busy slots is one more than the number of idle slots. See Fig. 3 for an example of a greedy schedule.

Lemma 4. *Every idle slot is immediately preceded and immediately followed by an all-busy slot in the greedy schedule for jobs arriving in a non-increasing order of processing times.*

Proof. Since the greedy schedule does not have any time instance that both machines are idle, every idle slot must be immediately preceded by an all-busy slot, perhaps except the first idle slot which may start at time 0. We argue that if the last job to finish in the greedy schedule is a 1-machine job (which is assumed), then x_1 must be preceded by an all-busy slot. Since the first job arrived is the longest job in this special input, which is scheduled at time 0 by the Greedy Algorithm, x_1 must be at least the value of this processing time. Thus slot x_1 must be able to accommodate any subsequent jobs including the last job, which implies that an all-busy slot must exist preceding x_1. The definition of idle slot implies that the slot must be followed by an all-busy slot.   □

In addition to the idle slots x_i defined before, we define the "open" idle period as follows. Suppose Machine 1 and Machine 2 finish all their jobs at time t_1 and t_2, respectively, and without loss of generality, let $t_1 \geqslant t_2$. The period from time t_2 to t_1, for which Machine 2 is idle, is called the open idle period x'. For simplicity, x' also denotes the length of the period, i.e., $t_1 - t_2$. Similar to the idle slot, the open idle period should be preceded by an all-busy slot, which is the last all-busy slot. Otherwise, the last job must start at time 0 which contradicts to the assumption we made in the beginning of the section. We show the relationship among the all-busy slots, the idles slots, and the open idle period, in the following lemma.

Lemma 5. *Suppose the greedy schedule has k idle slots x_i and $k + 1$ all-busy slots z_i for $k \geqslant 0$, and one open idle period. We have $z_i \geqslant x_i$ for $1 \leqslant i \leqslant k$ and $z_{k+1} \geqslant x'$.*
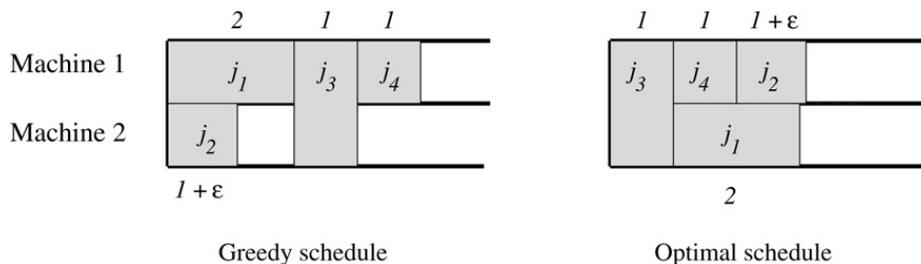
Fig. 3. An example to show that greedy schedule has a competitive ratio arbitrarily close to $4/3$.

Proof. We first prove that $z_i \geqslant x_i$. Consider the slot x_i and for instance suppose it appears in Machine 1. We can see that Machine 2 must be processing a single 1-machine job, say j, throughout the period of slot x_i. Otherwise, the second job processed by Machine 2 during slot x_i can be scheduled by the Greedy Algorithm to start earlier in Machine 1 during slot x_i. Then we have the processing time of j, $p_j \geqslant x_i$. If there is any 2-machine job scheduled in the all-busy slot z_i, the processing time of that job, say p', must be at least p_j because the 2-machine job must arrive before job j. Hence, we have $z_i \geqslant x_i$. If there is no 2-machine job scheduled in the all-busy slot z_i, x_i must be the first idle slot, i.e., $i = 1$. There must be a 1-machine job scheduled and completed in slot z_1 before slot x_1 in Machine 1 and the processing time of this job is at least x_1. It is because this job must have the same or longer processing time than that of the last job which cannot be scheduled in slot x_1. Thus we have proved that $z_i \geqslant x_i$. In a similar approach, we can also prove $z_{k+1} \geqslant x'$. $\quad\square$

Theorem 6. *Greedy Algorithm is (4/3)-competitive in scheduling parallel jobs on two machines if the jobs arrive in a non-increasing order of processing times.*

Proof. Let $X = \sum_{i=1}^{k} x_i$ and $Z = \sum_{i=1}^{k+1} z_i$. By Lemma 5, we have $Z \geqslant X + x'$. Since the greedy schedule does not have any time instance that both machines are idle, we have the makespan of the greedy schedule $C_{Greedy} = Z + X + x'$. Since $Z \geqslant X + x'$, we also have $Z \geqslant C_{Greedy}/2$. On the other hand, the makespan of the optimal schedule C_{OPT} must be at least half the sum of the weighted processing time of all jobs, where the weights of the 1-machine and 2-machine jobs are 1 and 2, respectively. Therefore, we have $C_{OPT} \geqslant Z + (X + x')/2 = C_{Greedy}/2 + Z/2 \geqslant (3/4)C_{Greedy}$. Hence, the competitive ratio of the Greedy Algorithm is at most 4/3. $\quad\square$

We show that the analysis for Theorem 6 is tight by giving a job sequence such that the competitive ratio of the Greedy Algorithm is arbitrarily close to 4/3. Consider a sequence of 4 jobs arriving in the order: $j_1 = (1, 2)$, $j_2 = (1, 1 + \epsilon)$, $j_3 = (2, 1)$ and $j_4 = (1, 1)$, where ϵ is a sufficiently small positive number. The makespan of the greedy schedule is 4, while the makespan of the optimal schedule is $3 + \epsilon$ (see Fig. 3). Thus the competitive ratio is arbitrarily close to 4/3.

In fact, the performance of the Greedy Algorithm is not too bad when compared with the optimal algorithm. In the following theorem, we show that no on-line algorithm is better than (9/7)-competitive.

Theorem 7. *For the problem of scheduling parallel jobs on two machines with the jobs arriving in a non-increasing order of processing times, no on-line algorithm is better than (9/7)-competitive.*

Proof. We give an adversary job sequence to show that no on-line algorithm has a competitive ratio less than $9/7 - \delta$ for any $\delta > 0$. The adversary consists of at most 4 jobs, j_1, j_2, j_3 and j_4 arriving one by one in that order. After each job is scheduled, if the competitive ratio of the on-line schedule is at least $9/7 - \delta$, the adversary stops. We show that if an on-line algorithm has a competitive ratio less than $9/7 - \delta$ both after scheduling j_1 and after scheduling j_2, then the on-line algorithm must have a competitive ratio at least $9/7 - \delta$ after scheduling all 4 jobs.

Let $j_1 = (1, 1)$. The on-line algorithm must schedule j_1 at time $h < 2/7 - \delta$, otherwise, the competitive ratio is at least $9/7 - \delta$. Then let $j_2 = (2, 3/4 + \epsilon)$ where ϵ is a sufficiently small positive number. Job j_2 must be scheduled after j_1 and at time $1 + x$ where $x < 1/2 + 2\epsilon/7$. Otherwise, the competitive ratio is at least $9/7 - \delta$. Let $j_3 = (1, (1 + x)/2 + 6\epsilon/7)$ and $j_4 = (1, (1 + x)/2 + 6\epsilon/7)$. It can be verified that $(1 + x)/2 + 6\epsilon/7 \leqslant 3/4 + \epsilon$ because $x < 1/2 + 2\epsilon/7$. Thus the processing times of all these 4 jobs are in a non-increasing order. Since the sum of the processing times of j_3 and j_4 are greater than the length of the idle slot before j_2, which is $1 + x$, at least one of the two jobs must be scheduled after j_2 (see Fig. 4). Therefore, the competitive ratio of the on-line algorithm after scheduling all 4 jobs is at least

$$\frac{9/4 + 3x/2 + 13\epsilon/7}{7/4 + x + 19\epsilon/7} \geqslant \frac{9/4 - 7\delta/4 + 3x/2 + +7\delta/4 + 13\epsilon/7}{7/4 + x + 19\epsilon/7}$$

$$\geqslant \min\left(9/7 - \delta, 3/2, \frac{7\delta/4 + 13\epsilon/7}{19\epsilon/7}\right)$$

$$= 9/7 - \delta \quad \{\text{let } \epsilon \leqslant 343\delta/320\}.$$
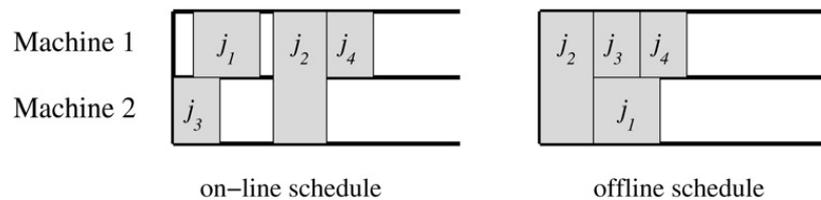
Fig. 4. The on-line and offline schedules for the adversary in the proof of Theorem 7.

By the above argument, if there is an on-line algorithm with a competitive ratio $k < 9/7$, we can show that the algorithm is at least $(9/7 - \delta)$-competitive for $9/7 - \delta > k$, by letting $\delta < 9/7 - k$, which is a contradiction. As a result, we have proved that no on-line algorithm is better than $(9/7)$-competitive. $\square$

References

[1] S. Bischof, E.W. Mayr, On-line scheduling of parallel jobs with runtime restrictions, Theor. Comput. Sci. 268 (1) (2001) 67–90.

[2] M. Drozdowski, Scheduling parallel tasks—algorithms and complexity, in: J.Y-T. Leung (Ed.), Handbook of Scheduling: Algorithms, Models, and Performance Analysis, CRC Press, Boca Raton, FL, 2004 (Chapter 25).

[3] A. Feldmann, M.-Y. Kao, J. Sgall, S.-H. Teng, Optimal on-line scheduling of parallel jobs with dependencies, J. Comb. Optim. 1 (4) (1998) 393–411.

[4] E. Naroska, U. Schwiegelshohn, On an on-line scheduling problem for parallel jobs, Inf. Process. Lett. 81 (6) (2002) 297–304.

[5] D.G. Feitelson, L. Rudolph, U. Schwiegelshohn, Parallel job scheduling—a status report, in: Proceedings of the 10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2004), in: Lecture Notes in Computer Science, vol. 3277, Springer, Berlin, 2004, pp. 1–16.

[6] D. Ye, G. Zhang, On-line scheduling of parallel jobs, in: Proceedings of the 11th International Colloquium of Structural Information and Communication Complexity (SIROCCO 2004), in: Lecture Notes in Computer Science, vol. 3104, Springer, Berlin, 2004, pp. 279–290.

[7] B. Johannes, Scheduling parallel jobs to minimize makespan, Tech. Rep. 723-2001, Technische Universität Berlin, Germany, http://www.math.tu-berlin.de/coga/publications/techreports/2001/Report-723-2001.html, 2001.

[8] A. Borodin, R. El-Yaniv, Online Computation and Competitive Analysis, Cambridge University Press, Cambridge, 1998.

[9] R. Graham, Bounds for certain multiprocessing anomalies, Bell Syst. Tech. J. 45 (1966) 1563–1581.

[10] U. Faigle, W. Kern, G. Turán, On the performance of on-line algorithms for partition problems, Acta Cybern. 9 (2) (1989) 107–119.

[11] D. Ye, On-line and efficient algorithms for scheduling problems in communication networks, PhD thesis, Zhejiang University, Hangzhou, China, 2005.