Online scheduling with a buffer on related machines

György Dósa[*] Leah Epstein[†]

Abstract

Online scheduling with a buffer is a semi-online problem which is strongly related to the basic online scheduling problem. Jobs arrive one by one and are to be assigned to parallel machines. A buffer of a fixed capacity K is available for storing at most K input jobs. An arriving job must be either assigned to a machine immediately upon arrival, or it can be stored in the buffer for unlimited time. A stored job which is removed from the buffer (possibly, in order to allocate a space in the buffer for a new job) must be assigned immediately as well. We study the case of two uniformly related machines of speed ratio $s \geq 1$, with the goal of makespan minimization.

Two natural questions can be asked. The first question is whether this model is different from standard online scheduling, that is, is any size of buffer $K > 0$ already helpful to the algorithm, compared to the case $K = 0$. The second question is whether there exists a constant K, so that a larger buffer is no longer beneficial to an algorithm, that is, increasing the size of the buffer above this threshold would not change the best competitive ratio further. Previous work [15, 19, 5] shows that in the case $s = 1$, already $K = 1$ allows to design a $\frac{4}{3}$-competitive algorithm, which is best possible for any $K \geq 1$, whereas the best possible ratio for $K = 0$ is $\frac{3}{2}$. Similar results have been show for multiple identical machines [5].

We answer both questions affirmatively, and show that a buffer of size $K = 2$ is sufficient to achieve the a competitive ratio which matches the lower bound for $K \to \infty$ for any $s > 1$. In fact, we show that a buffer of size $K = 1$ can evidently be exploited by the algorithm for any $s > 1$, but for a range of values of s, it is still weaker than a buffer of size 2. On the other hand, in the case $s \geq 2$, a buffer of size $K = 1$ already allows to achieve optimal bounds.

1 Introduction

Scheduling of jobs arriving one by one (also called *over list*) is a basic model in online scheduling [18]. The system consists of a set of processors that can process a sequence of arriving jobs. Each job j, which has a processing time p_j associated with it (also called size), needs to be assigned to a processor upon arrival. The processors are *uniformly related*, in the sense that a processor i had a speed s_i, and the time to process job j on machine i is $\frac{p_j}{s_i}$. The completion time, or load, of a machine is the total time needed to process the jobs assigned to it (i.e., the total processing time, divided by the speed), and the goal is to minimize the maximum load of any machine, also called the *makespan*.

We consider online algorithms. For an algorithm $\mathcal{A}$, we denote its cost by $\mathcal{A}$ as well. An optimal offline algorithm that knows the complete sequence of jobs in advance, as well as its cost are denoted by $\textsc{opt}$. In this paper we consider the (absolute) competitive ratio. The competitive ratio of $\mathcal{A}$ is the infimum $\mathcal{R}$ such that for any input, $\mathcal{A} \leq \mathcal{R} \cdot \textsc{opt}$. If the competitive ratio of an online algorithm is at most $\mathcal{C}$, then we say that it is $\mathcal{C}$-competitive.

This model is strict in the sense that decisions need to be done without sufficient information on the future. Lower bounds on the competitive ratio already for identical speed machines [10, 1, 11] often make use of one large job that arrives when the assignment is very balanced, for that, the exact assignment of all

[*]Department of Mathematics, University of Pannonia, Veszprém, Hungary, `dosagy@almos.vein.hu`.
[†]Department of Mathematics, University of Haifa, 31905 Haifa, Israel. `lea@math.haifa.ac.il`.

jobs which arrived so far must be fixed. A natural way to possibly overcome this is the usage of lookahead. This means that the scheduler needs to assign a job while seeing a prefix of the future jobs in the sequence, where the prefix is of some fixed length $\ell \geq 1$. It is not difficult to see, however, that for any constant ℓ, lookahead is not useful, since it is possible to augment the sequence by many jobs of size zero.

Several semi-online models, which allow the usage of various types of information on future jobs, have been introduced. Some relevant models are as follows. Zhang and Ye [20] studied a model, in which it is known in advance that the largest job is last, and the scheduler is notified when this last job arrives (see also [9]). Some models assume a priori knowledge on the input already in the beginning of the input sequence, such as the total size of all jobs [15, 2, 3], bounds on possible job sizes [14], the cost of an optimal schedule [4, 6], or other properties of the sizes, e.g., an arrival in a sorted order [13, 17, 7].

The variant we study in this paper is one where the scheduler is given a constant length buffer, that can be used for temporary storage of jobs. Specifically, let $K \geq 1$ be an integer. The buffer may contain up to K jobs at any time. Upon arrival of a job, the scheduler can do one of the following actions. The first one is assignment of the new job to a machine. The second one is removal of a job from the buffer, and the assignment of this removed job to some machine. This latter action can be repeated several times without additional arrivals of jobs. The last action is the insertion of the new job into the buffer, if it was not assigned to a machine. This can be done if the buffer contains at most $K-1$ jobs at this time. Thus, at every time there may be at most $K+1$ jobs which arrived already, but not were assigned yet.

This model was first studied for the case of two identical speed machines by Kellerer et al. [15], and by Zhang [19]. In both papers, an algorithm of competitive ratio $\frac{4}{3}$, which uses $K=1$ was designed. It was shown in [15] that this competitive ratio is best possible for any $K > 1$, that is, using a larger buffer cannot be beneficial. Note that for $K=0$, and identical machines, the best possible competitive ratio is $\frac{3}{2}$ [12, 10]. We compare our results with the case $K=0$ and $s>1$ (see [8]). In this case, the tight competitive ratio is $\frac{2s+1}{s+1}$ if $s \leq \frac{\sqrt{5}+1}{2} \approx 1.618$, and $\frac{s+1}{s}$ if $s \geq \frac{\sqrt{5}+1}{2}$.

The case of m related machines was studied by Englert, Özmen and Westermann [5]. They designed a $2+\varepsilon$-competitive algorithm with a buffer size of m. In the same paper, [5], the case of m identical speed machines was studied extensively, and tight bounds on the competitive ratio for every value of m were found. These bounds are achievable with a buffer of size $\Theta(m)$ (see also [16], for previous results on the case of multiple identical speed machines).

In this paper, we consider two related machines. We denote the speed ratio between the speeds of the two machines by $s \geq 1$, and assume without loss of generality that the speed of the first machine (also called the *slow* machine) is 1, and the speed of the second machine (also called the *fast* machine) is s. The number of jobs (which is unknown until all jobs have arrived) is denoted by n.

We introduce some notations. Time t is considered to be the time at which the t-th jobs has arrived, but was not considered yet. We define P_t to be the total size of all jobs that have arrived by that time, that is, of the first t jobs $P_t = \sum\limits_{j=1}^{t} p_j$. We denote the cost (i.e., makespan) of an optimal schedule for the subsequence of the first i jobs (that is, of the first i jobs, without leaving any jobs in the buffer) by OPT_i. We have $\text{OPT} = \text{OPT}_n$. Let $M_i = \max\limits_{1 \leq j \leq i} p_j$.

We use the following lower bounds on OPT_i. The standard lower bounds are $\text{OPT}_i \geq \frac{M_i}{s}$ and $\text{OPT}_i \geq \frac{P_i}{s+1}$. We let $LB_i^1 = \frac{M_i}{s}$ and $LB_i^2 = \frac{P_i}{s+1}$.

A third lower bound which is useful in some cases is developed as follows. Let $1 \leq m(i) \leq i$ be such that $p_{m(i)} = M_i$. Let $M_i' = \max\limits_{1 \leq j \leq i, j \neq m(i)} p_j$. If both jobs of sizes M_i and M_i' (which are the two largest jobs in the sequence) are assigned to the same machine, then $\text{OPT}_i \geq \frac{M_i + M_i'}{s}$. Otherwise, at least one of them is assigned to the first machine and so $\text{OPT}_i \geq \min\{M_i, M_i'\} = M_i'$. Therefore, we let $LB_i^3 = \min\{M_i', \frac{M_i+M_i'}{s}\}$. Since in most cases the first two lower bounds are sufficient, we let $LB_i =$

$\max\{LB_i^1, LB_i^2\}$.

We let L_1^i and L_2^i denote the total processing time (on the relevant machine) of jobs assigned to the first machine and second machine, respectively, at time i, that is, after i jobs have arrived, but before the i-th job was dealt with. The completion times of these machines at that time are L_1^i and L_2^i. Note that the final cost of an algorithm is not computed using just the loads at the $n+1$-th time, L_1^{n+1} and $\frac{L_2^{n+1}}{s}$, but we need to take into account the assignment to machines of the jobs stored in the buffer.

We show that the best possible competitive ratio for an arbitrary value of K is $\frac{(s+1)^2}{s^2+s+1}$ for $1 < s \le \frac{\sqrt{5}+1}{2}$, $\frac{s^2}{s^2-s+1}$ for $\frac{\sqrt{5}+1}{2} \le s \le 2$ and $\frac{s+2}{s+1}$ for $s \ge 2$. These competitive ratios are achievable already for $K = 2$, and in the case $s \ge 2$, even for $K = 1$. We shed some light of the case $K = 1$ for $s < 2$, in particular, we give tight bounds of $\frac{s+2}{s+1}$ on the competitive ratio for $\sqrt{2} \le s < 2$ and $K = 1$, and relatively close bounds for $1 < s < \sqrt{2}$. Thus we show that a buffer of size $K = 1$ allows to get reduced competitive ratio compared to the case $K = 0$, for any $s > 1$.

2 Algorithms for the case $K = 1$

We start with the case where the buffer has a single slot. Note that if an algorithm can use a buffer of size $K > 0$, the algorithm is forced to assign some job only after $K + 1$ jobs have arrives.

2.1 A simple algorithm

The first arriving job can be stored in the buffer until a second job arrives. Upon the arrival of a second job, either the job in the buffer or the new job should be assigned. The algorithm keeps one job in the buffer at all times (possibly replacing this job with the new arriving job), as long as jobs keeps arriving. At the time when it is known that no further jobs will arrive, the job in the buffer must be assigned. The following algorithm always stores a job of largest size in the current input in the buffer, that is, at time $i + 1$, a job of size M_i is in the buffer. The algorithm uses a parameter $C(s) > 1$.

Algorithm *Largest-Last* **(***LL***)**

1. Store the first job in the buffer. Let $L_2^1 = L_2^2 = 0$, $P_1 = p_1$ and $M_1 = p_1$.
2. For each arriving job of index t act as follows.
   - 2.1. Let $P_t = P_{t-1} + p_t$, $M_t = \max\{M_{t-1}, p_t\}$. Consider the new job of index t and the job in the buffer. Let $X_t \ge Y_t$ be their sizes. Store the job of size X_t in the buffer.
   - 2.2. If $L_t^1 + Y_t \le C(s) \cdot LB_t$ then assign the job of size Y_t to the first machine and let $L_{t+1}^1 = L_t^1 + Y_t$, $L_{t+1}^2 = L_t^2$. Otherwise assign it to the second machine and let $L_{t+1}^1 = L_t^1$, $L_{t+1}^2 = L_t^2 + \frac{Y_t}{s}$.
3. The last job which remains in the buffer is assigned to the second machine.

We analyze the algorithm for $1 < s < \sqrt{2}$. Note that in this range, $\frac{(s+1)^2}{s^2+s+1} < \frac{2(s+1)}{s+2} < \frac{s+2}{s+1}$.

Theorem 1 *Algorithm LL with the parameter $C(s) = \frac{2(s+1)}{s+2}$ has a competitive ratio of $C(s)$ for any $1 < s \le \sqrt{2}$. No other choice of parameter $C(s)$ can lead to a smaller competitive ratio.*

Proof. We prove the upper bound first. Assume that the statement is not true. We consider a minimal counter example (in terms of the number of jobs), and assume by scaling that $\text{OPT} = 1$. Note that by the definition of the algorithm, assigning a job to the first machine can never cause the algorithm to violate the competitiveness, thus we assume that the final load of the second machine exceeds $\frac{2(s+1)}{s+2}$, i.e., the total size of jobs assigned to it exceeds $\frac{2s(s+1)}{s+2}$. No job has size of more than s, therefore, at least two jobs were assigned to the second machine. Let Z be the last job ever assigned to the second machine by Step 2. We consider two cases.

If Z is assigned to the second machine at a time when the job of size X_n, which is the last job assigned to the second machine, did not arrive yet. Since Z is assigned by step 2, there exists a time at which a job of size X_t is stored in the buffer, and Z is the job of size Y_t which is going to be assigned. At this time, $L_t^1 + Y_t > \frac{2(s+1)}{s+2} LB_t$, and $LB_t \geq \frac{P_t}{s+1}$, where $P_t = L_t^1 + sL_t^2 + Y_t + X_t$. Since the job of size X_n arrives later, we have $P_n \geq P_t + X_n$.

The total size of jobs assigned to the second machine is $sL_t^2 + Y_t + X_n = P_t - L_t^1 - X_t + X_n < P_t - X_t + X_n - \frac{2(s+1)}{s+2} LB_t + Y_t \leq P_t + X_n - \frac{2P_t}{s+2} = \frac{s}{s+2} P_t + X_n \leq \frac{s}{s+2} P_n + \frac{2}{s+2} X_n \leq \frac{s(s+1)}{s+2} LB_n + \frac{2s}{s+2} LB_n = \frac{s^2+3s}{s+2} \text{OPT} < \frac{2s(s+1)}{s+2}$.

If Z is assigned to the second machine at a time when X_n is already present in the buffer. We assume that Z is the job of size Y_n. Otherwise, if Z is the job of size Y_t for some $t < n$, then we get that after the assignment of this job, at least one job is assigned to the first machine, but the final load of the second machine is $\frac{sL_t^2+Y_t+X_n}{s}$. Therefore, removing the jobs of sizes $Y_{t+1}, \ldots, Y_{n-1}$ results in a smaller example in which the second machine has the same load, i.e., a smaller counter example. Since Z is assigned to the second machine, we have $L_n^1 + Y_n > \frac{2(s+1)}{s+2} LB_n$ and since this is a counter example, then $sL_n^2 + Y_n + X_n > \frac{2s(s+1)}{s+2} \geq \frac{2s}{s+2} P_n$. Taking the sum and using $LB_n \geq \frac{P_n}{s+1}$, we get $P_n + Y_n > \frac{2s+2}{s+2} P_n$ and $X_n \geq Y_n > \frac{s}{s+2} P_n$. On the other hand, $P_n \geq L_n^1 + Y_n + X_n > \frac{2}{s+2} P_n + \frac{s}{s+2} P_n = P_n$, which leads to a contradiction.

We next prove that the analysis of the performance is tight, and that using a different parameter cannot reduce the competitive ratio. Using $C(s) < 1$ implies a competitive ratio of at least $\frac{s+1}{s}$, as follows. Consider a sequence which contains two jobs of sizes 1 and s. $LB_2 = 1$, and therefore *LL* assigns both jobs to the second machine. If $C(s) \geq 1$, the sequence consists of four jobs, of sizes $2 - s^2$, s, $s^2 + s$, $s^2 + s$. An optimal assignment of these jobs is to assign the first job to the first machine, the second job to the second machine, and one additional job to each machine. This gives a makespan of $s + 2$. Note that $LB_2 \geq 1$ and $LB_3 \geq 2$. Since $2 - s^2 < 1 < s$, and $2 - s^2 + s < 2$, both the first job and the second job are assigned to the first machine. If at least one additional job is assigned to the first machine, it achieves the load $2s + 2$. Otherwise, the second machine achieves a load of $2s(s + 1)$. ■

Note that it is possible to prove that this algorithm has a competitive ratio of at most $\frac{s+2}{s+1}$ for any $\sqrt{2} \leq s \leq 2$, using $C(s) = \frac{s+2}{s+1}$. We omit the proof since we present an additional algorithm later, whose competitive ratio is $\frac{s+2}{s+1}$ for any $s \geq 1$.

We have showed that a buffer of size $K = 1$ reduces the competitive ratio compared with the best bound for the case $K = 0$, for which the best competitive ratio is $\frac{2s+1}{s+1}$.

2.2 A second algorithm

We let $C_1(s) = \frac{s+2}{s+1}$. We define an additional algorithm, which is optimal in some cases stated below.

Algorithm *Small-Large* **(***SL***)**

1. Store the first job in the buffer. Let $L_2^1 = L_2^2 = 0$, $P_1 = p_1$ and $M_1 = p_1$.
2. For each arriving job of index t act as follows.

    2.1 Let $P_t = P_{t-1} + p_t$, $M_t = \max\{M_{t-1}, p_t\}$. Consider the new job of index t and the job in the buffer. Let $X_t \geq Y_t$ be their sizes.

    2.2 If $L_t^1 + Y_t \leq C_1(s) \cdot LB_t$ then assign the job of size Y_t to the first machine and store the job of size X_t in the buffer, and let $L_{t+1}^1 = L_t^1 + Y_t$, $L_{t+1}^2 = L_t^2$. Otherwise assign the job of size X_t it to the second machine and store the job of size Y_t in the buffer, and let $L_{t+1}^1 = L_t^1$, $L_{t+1}^2 = L_t^2 + \frac{X_t}{s}$.

3. The last job which remains in the buffer is assigned as follows. Let n be the total number of jobs. Assign the job which is still in the buffer to the first machine if the resulting load would not exceed $C_1(s) \cdot LB_n$, and otherwise assign it to the second machine.

Theorem 2 *Algorithm SL has a competitive ratio of at most $C_1(s)$ for all $s \geq 1$. This is best possible for any $s \geq 2$ and any $K \geq 1$, and best possible for $\sqrt{2} \leq s < 2$ and $K = 1$.*

Proof. We prove the upper bound first. Assume that the statement is not true. We consider a counter example, and scale it so that the cost of an optimal solution for this sequence is $s+1$. Clearly, in the optimal solution the load of the first machine is at most $s+1$, and the load of the second machine is at most $s(s+1)$, and the total size of jobs in the sequence is at most $(s+1)^2$. Let A denote the job which is assigned last, i.e. the job remaining in the buffer after all jobs have arrived. We consider a specific optimal schedule.

Note that by the definition of the algorithm, assigning a job to the first machine would never cause the algorithm to violate the competitiveness, thus we assume that the final load of the second machine, after all jobs have been assigned, is more than $C_1(s)(s+1) = s+2$. Therefore, the final load of the first machine is less than $(s+1)^2 - s(s+2) = 1$. Let Z denote the job which is assigned last to the second machine (and its size), Z can either be the job A, or a different job. We next prove that $Z > s+1$. Let a be the load of the first machine at the time in which Z is assigned to the second machine. Then $a < 1$ since it cannot exceed the final load of the first machine. Since Z is assigned to the second machine, $a + Z > \frac{s+2}{s+1}L$, where L is the value of the lower bound at the time of assignment. Using the second lower bound, since the total size of jobs assigned to the second machine exceeds $s(s+2)$, $L \geq \frac{a+s(s+2)}{s+1}$ at this time, it follows that $a + Z > \frac{s+2}{s+1}\frac{a+s(s+2)}{s+1}$, from which we get (by using $a < 1$), $(s+1)^2 Z > (s+2)\,a - (s+1)^2 a + s\,(s+2)^2 = s\,(s+2)^2 - \left(s^2+s-1\right)a > s\,(s+2)^2 - \left(s^2+s-1\right) = (s+1)^3$, which implies $Z > s+1$.

Let U be the job, which is assigned to the slow machine in the optimal schedule, and is assigned **last** to the second machine by the algorithm. If there is not such job, the total size of jobs assigned to the second machine cannot exceed $s(s+1)$, which would be a contradiction. Moreover, U and Z are different jobs, since $U \leq s+1$ and $Z > s+1$, furthermore, U and A are different jobs, since if A is assigned to the second machine, then A and Z are the same job. It follows that U is assigned to the second machine at some step t, while an additional job is present in the buffer. Since U is assigned to the second machine, U is the job of size X_t, and there exists a job of size $Y_t \leq X_t$ which is stored in the buffer at this time. At the termination time of the algorithm, the total size of jobs assigned to the second machine is more than $s(s+2)$, but for the optimal solution it is at most $s(s+1)$. Therefore, a total size of more than s must be caused by jobs which are assigned to the first machine in the optimal solution, and therefore, just after the assignment of U to the second machine, its load L^2_{t+1}, satisfies $L^2_{t+1} > 1$. The load of the first machine at the same time satisfies $L^1_{t+1} < 1$, since the final load of the first machine is less than 1. Note that $L^1_{t+1} + Y_t = L^1_t + Y_t > \frac{s+2}{s+1}LB_t \geq \frac{s+2}{s+1}\frac{L^1_t + Y_t + s}{s+1}$ must hold, otherwise the job of size Y_t would be assigned to the first machine at the time of arrival of the t-th job, instead of assigning the job of size X_t to the second machine. From this inequality we get that $L^1_t + Y_t > \frac{s(s+2)}{s^2+s-1} > 1$.

We next prove that starting this time, at each time $t' > t$ (the time just after the $t'-1$-th job has been assigned), the buffer contains a job of size $b_{t'} \leq s+1$, which satisfies $L^1_{t'} + b_{t'} > 1$. We prove this by induction. Consider the t'-th job, which has size $p_{t'}$. If $p_{t'} < b_{t'} \leq s+1$ then there are two options. If the smaller job is assigned to the first machine, the job in the buffer is not replaced and $L^1_{t'+1} + b_{t'} > L^1_{t'} + b_{t'} > 1$, and otherwise, the job of size p_t is stored in the buffer, and similarly to the above argument, $L^1_{t'} + p_{t'} > \frac{s+2}{s+1}LB_{t'} > \frac{s+2}{(s+1)^2}(L^1_{t'} + p_{t'} + s)$, and $L^1_{t'} + p_{t'} > 1$. If $p_{t'} \geq b_{t'}$, then we claim that the job in the buffer is not replaced. If it is replaced, then this means that it was assigned to the first machine, but this would result in a load larger than 1, and we assume that this never happens. We have proved that the last job which is stored in the buffer has size of at most $s + 1$, but it is not assigned to the first machine, since its assignment to the first machine would result in a load larger than 1. Therefore this last job is Z. However, we showed that this job has size larger than $s + 1$, which is a contradiction.

We next prove lower bounds for cases where $s \geq \sqrt{2}$. Note that $\frac{s+2}{s+1} \leq s$ holds for $s \geq \sqrt{2}$.

To prove a lower bound for $s \geq 2$, let $K \geq 1$ be an arbitrary integer size of buffer. The sequence starts with many very small jobs of size $\varepsilon > 0$, of total size $1+K\varepsilon$. Denote the loads of first and second machines, respectively, after the arrival of these jobs by α and β, where $\alpha + \beta \geq 1$. The cost of an optimal solution at this moment is $\frac{1+K\varepsilon}{s+1}$.

If $\beta \geq \frac{s}{s+1}$, one further job of size s arrives. The cost of an optimal solution is no larger than $1 + K\varepsilon$. The cost of the resulting solution got is at least $\min\left\{\alpha + s, \frac{\beta+s}{s}\right\} \geq \min\left\{s, \frac{s+2}{s+1}\right\} = \frac{s+2}{s+1}$. Letting ε tend to zero implies the lower bound in this case.

Consider next the case where $\alpha > \frac{1}{s+1}$. Two further jobs arrive, with the sizes $Y = (s + 1)\alpha$, and $X = sY - 1 = s(s+1)\alpha - 1$. Note, that $X - Y = (s - 1)\,Y - 1 = (s-1)(s+1)\alpha - 1 > s - 2 \geq 0$, i.e., $X > Y$. The cost of an optimal solution is at most $Y + K\varepsilon$, since we can create a solution which assigns the job of size Y and a total size of very small jobs of $K\varepsilon$ to the first machine, and all other jobs to the second machine. If at least one of the two big jobs is assigned to the first machine, then the makespan is at least $\alpha + (s + 1)\alpha = (s + 2)\alpha$, otherwise both of them are assigned to the second machine, and its load will be $\beta + X + Y \geq 1 - \alpha + (s+1)\alpha + s(s+1)\alpha - 1 = s\,(s+2)\,\alpha$, thus in both cases, the makespan is at least $(s + 2)\alpha$, and the statement follows by letting ε tend to zero.

We next prove a lower bound for the case $\sqrt{2} \leq s \leq 2$ and $K = 1$. This lower bound does not hold for larger values of K. The first two jobs have sizes 1 and $s + 1$. At this time, since $K = 1$, one job must be assigned. There are four cases.

If the first job is assigned to the first machine, a third job of size $s^2 + s - 1$ arrives. An optimal solution assigns the second job to the first machine, and the other jobs to the second machine, and has a makespan of $s + 1$. If at least one additional job is assigned by the algorithm to the first machine, we get a makespan of at least $\min\{s + 2, s^2 + s\}$, which gives a competitive ratio of at least $\min\{\frac{s+2}{s+1}, s\} = \frac{s+2}{s+1}$. Otherwise, the makespan is at least $\frac{1}{s}(s^2 + 2s) = s + 2$ as well, which proves the lower bound in this case.

If the second job is assigned to the second machine, a third job of size $(s + 1)^2$ arrives. At this time, the cost of an optimal solution is $\frac{(s+1)^2}{s}$, by assigning the last job to the second machine, and the other jobs to the first machine, and since $s + 2 \leq \frac{(s+1)^2}{s}$. The competitive ratio is at least $1 + \frac{s+1}{(s+1)^2} = 1 + \frac{1}{s+1} = \frac{s+2}{s+1}$, if the last job is assigned to the second machine, and at least s otherwise.

If the second job is assigned to the first machine, no further jobs arrive. The cost of an optimal solution is at most $\frac{s+1}{s}$, by using the solution which assigns the first job to the first machine, and the second job to the second machine. The cost of the algorithm is $s + 1$. The competitive ratio is s.

If the first job is assigned to the second machine, no further jobs arrive. If the second job is assigned to the first machine, we get the previous case again. Otherwise, we get a makespan of $\frac{s+2}{s}$, and a competitive ratio of at least $\frac{s+2}{s+1}$. ■

Note that the last lower bound construction can be used for the interval $s \in (1, \sqrt{2})$, and yields a lower bound of s.

3 Tight bounds for $K \geq 2$

We have shown that using a buffer of size 1 allows to design an algorithm of best possible competitive ratio for any $K \geq 1$, for $s \geq 2$. The same holds for $s = 1$ by the results of [15, 19]

Therefore, we consider the case $1 < s < 2$, in this section, and design algorithms which use a buffer of size $K = 2$, which have the best possible competitive ratio for $K \geq 2$.

Let $C_2(s) = \frac{(s+1)^2}{s^2+s+1}$, if $s \leq \frac{\sqrt{5}+1}{2}$, and if $\frac{\sqrt{5}+1}{2} \leq s < 2$, $C_2(s) = \frac{s^2}{s^2-s+1}$. Note that $C_2(s) < \frac{4}{3}$ for $1 < s < 2$.

The algorithm always keeps the two biggest jobs seen so far in the buffer. In addition to $C_2(s)$, it uses a parameter $c_2(s)$ which is defined to be $\frac{s+1}{s^2}$ if $s \leq \frac{\sqrt{5}+1}{2}$, and otherwise, $\frac{1}{s^2-s}$. Note that since $s > 1$, $c_2(s)$ is well defined and positive.

Algorithm *Three-Jobs* **(***TJ***)**
Store the first job in the buffer. If the sequence stops, assign this job to the fast machine. Otherwise, store the second job in the buffer as well. Let $X_2 \geq Y_2$ be the sizes of these two jobs and $P_2 = X_2 + Y_2$. In addition, let $L_3^1 = L_3^2 = 0$.

For any arriving job of index $t \geq 3$ act as follows.

1. Let $P_t = P_{t-1} + p_t$.
2. Let $Z_t \leq Y_t \leq X_t$ be the sorted list of sizes p_t, Y_{t-1}, X_{t-1}. We have $L_t^1 + sL_t^2 + Z_t + Y_t + X_t = P_t$. The jobs of sizes X_t and Y_t become the contents of the buffer.
3. The job of size Z_t is assigned as follows.
   (a) If $Y_t > (C_2(s) - 1)P_t$, then assign the job of size Z_t to the second machine, let $L_1^{t+1} = L_1^t$ and $L_2^{t+1} = L_2^t + \frac{Z_t}{s}$.
   (b) If $L_t^1 + Y_t \geq c_2(s)\left(sL_t^2 + Z_t\right)$, then assign the job of size Z_t to the second machine, let $L_1^{t+1} = L_1^t$ and $L_2^{t+1} = L_2^t + \frac{Z_t}{s}$.
   (c) Otherwise assign the job of size Z_t to the first machine, let $L_1^{t+1} = L_1^t + Z_t$ and $L_2^{t+1} = L_2^t$.

After all jobs have arrived, let n be the number of jobs. We consider all four assignment of the remaining two jobs and choose the one with minimum makespan. Specifically, the four assignments are as follows. In the first assignment, the job of size X_n is assigned to the fast machine and the job of size Y_n to the slow machine. In the second assignment, the job of size Y_n is assigned to the fast machine and the job of size X_n to the slow machine. In the third assignment, both jobs are assigned to the fast machine and in the fourth assignment, both jobs are assigned to the slow machine.

We prove the following theorem.

Theorem 3 *TJ has a competitive ratio of $C_2(s)$ for any $1 < s < 2$, which is best possible for any $K \geq 2$.*

Proof. If the sequence consists of a single job, then the assignment is optimal. If it consists of two jobs, then all possible assignments are considered, which results in an optimal solution as well. We therefore assume that at least one job was assigned by Step 3.

Suppose that the statement is not true and consider an instance what violates it. We scale this instance so that the makespan of an optimal solution is 1. Then in the optimal solution the total sizes of jobs assigned to the the first machine and second machine (respectively) are at most 1 and at most s. The total sum of the jobs is therefore at most $s + 1$. By our assumption on the instance, the algorithm terminates with a makespan of more than $C_2(s)$. If the competitive ratio is violated, this means that either the first machine receives a total size of jobs of more than $C_2(s)$ or that the second machine receives a total size of jobs of at most $sC_2(s)$

We first claim that a makespan of more than $C_2(s)$ cannot be created as long as jobs are being assigned by Step 3. We consider three cases. If a job of size Z_t is assigned in Step 3(a), then the total size of jobs in the buffer is at least $X_t + Y_t \geq 2Y_t \geq 2(C_2(s) - 1)P_t$. Assume that $L_{t+1}^2 > C_2(s)$. We have $sL_{t+1}^2 = sL_t^2 + Z_t \leq P_t - X_t - Y_t \leq (3-2C_2(s))P_t \leq (3-2C_2(s))P_n \leq (3-2C_2(s))(s+1)$, which gives $(3 - 2C_2(s))(s + 1) > sC_2(s)$, that is equivalent to $C_2(s) < \frac{3s+3}{3s+2}$. This is impossible for $1 < s \leq \frac{\sqrt{5}+1}{2}$, since $C_2(s) - 1 = \frac{s}{s^2+s+1} > \frac{1}{3s+2}$, and for $\frac{\sqrt{5}+1}{2} < s < 2$, since $C_2(s) - 1 = \frac{s-1}{s^2-s+1} > \frac{1}{3s+2}$ for $s > \sqrt{\frac{3}{2}} \approx 1.225$.

If a job of size Z_t is assigned in Step 3(b), then by the assignment condition, $c_2(s)(sL_t^2 + Z_t) \leq P_t - sL_t^2 - Z_t - X_t$. Therefore $(c_2(s) + 1)(sL_t^2 + Z_t) \leq P_t \leq P_n \leq s + 1$. Thus $sL_{t+1}^2 \leq \frac{s+1}{c_2(s)+1}$. For $1 < s \leq \frac{\sqrt{5}+1}{2}$, we get $\frac{s+1}{c_2(s)+1} = \frac{s^2(s+1)}{s^2+s+1} \leq \frac{s(s+1)^2}{s^2+s+1} = sC_2(s)$, and for $\frac{\sqrt{5}+1}{2} < s < 2$, we have $\frac{s+1}{c_2(s)+1} = \frac{(s+1)(s^2-s)}{s^2-s+1} < \frac{s^3}{s^2-s+1} = sC_2(s)$.

If a job of size Z_t is assigned in Step 3(c), then by the assignment condition, $L_t^1 + Y_t < c_2(s)(P_t - L_t^1 - Y_t - X_t)$. Therefore, using $Z_t \leq Y_t \leq X_t$, $(c_2(s) + 1)(L_t^1 + Z_t) \leq (c_2(s) + 1)(L_t^1 + Y_t) \leq c_2(s)P_t \leq c_2(s)P_n \leq c_2(s)(s+1)$. Thus $L_{t+1}^1 \leq \frac{c_2(s)(s+1)}{c_2(s)+1}$. For $1 < s \leq \frac{\sqrt{5}+1}{2}$, we get $\frac{c_2(s)(s+1)}{c_2(s)+1} = \frac{(s+1)^2}{s^2+s+1} = C_2(s)$, and for $\frac{\sqrt{5}+1}{2} < s < 2$, we have $\frac{c_2(s)(s+1)}{c_2(s)+1} = \frac{s+1}{s^2-s+1} < \frac{s^2}{s^2-s+1} = C_2(s)$, by $s^2 > s + 1$.

We consider the assignment of the last two jobs. Denote the sizes of jobs, which remain in the buffer after the job of size Z_n has been assigned, by $Y^* = Y_n$ and $X^* = X_n$. All other jobs are called regular. As shown above, at the time just before the assignment of the jobs of size Y^* and X^*, the makespan is no larger than $C_2(s)$. If by assigning one or two of the last two jobs to the first machine, the load of the first machine exceeds $C_2(s)$, this means that the total size of jobs assigned to the second machine does not exceed $s + 1 - C_2(s)$. On the other hand, if by assigning one or two jobs of the last two jobs to the second machine, the load of the second machine exceeds $C_2(s)$, this means that the load of the first machine does not exceed $s + 1 - sC_2(s)$. Note that $C_2(s) < \frac{s+1}{s}$ in both cases, thus $s + 1 - sC_2(s) > 0$.

We claim that we can assume that prior to the assignment of the last two jobs, the loads of the first machine and the second machine respectively do not exceed $s + 1 - sC_2(s)$ and $\frac{s+1-C_2(s)}{s}$, respectively. We already showed that these loads do not exceed $C_2(s)$. If the first load is at least $s + 1 - sC_2(s)$, then assigning both last jobs to the second machine would result in a load of less than $C_2(s)$. If the second load is at least $\frac{s+1-C_2(s)}{s}$, then assigning both last jobs to the first machine would result in a load of less than $C_2(s)$.

For the last two jobs, we define a notion of being **big** or **small** as follows. A job is called **big**, if assigning it (temporarily) to the first machine, the load of the first machine would exceed $C_2(s)$, otherwise it is called **small**. It follows that each big job has a size of more than $(s + 1)(C_2(s) - 1)$. Recall that the two jobs remaining in the buffer at the end of the process are the two largest jobs among all jobs. We consider three cases, according to the number of small jobs and the number of big jobs.

Case 1. If both remaining jobs are small, we consider two options. If by assigning the job of size X^* to the first machine, its load becomes larger than $s + 1 - sC_2(s)$, then assigning the job of size Y^* to the second machine would result in a total size of jobs of at most $P_n - (s + 1 - sC_2(s)) = sC_2(s)$. Otherwise, we have $Y^* \leq X^* \leq s + 1 - sC_2(s)$. Thus, assigning both these jobs to the first machine would result in a load of at most $2(s + 1 - sC_2(s))$. Assume by contradiction $2(s + 1 - sC_2(s)) > C_2(s)$, or equivalently, $C_2(s)(2s + 1) < 2(s + 1)$. For $1 < s \leq \frac{\sqrt{5}+1}{2}$, we have $C_2(s) - 1 = \frac{s}{s^2+s+1} > \frac{1}{2s+1}$. For $\frac{\sqrt{5}+1}{2} < s < 2$, we have $C_2(s) - 1 = \frac{s-1}{s^2-s+1} > \frac{1}{2s+1}$, which leads to a contradiction.

Case 2. Next, suppose that there is exactly one big job, then this must be the job of size X^* and the job of size Y^* is a small job. We consider the assignment of the big job to the second machine and the small job to the first machine. In this assignment, the load of the first machine does not exceed $C_2(s)$ and therefore the load of the second machine must be more than $C_2(s)$. By our assumption on the instance, $X^* \leq s$. Thus $sL_{n+1}^2 > sC_2(s) - X^* \geq sC_2(s) - s > 0$, since $C_2(s) > 1$ for $1 < s < 2$. Therefore at least one regular job was assigned to the second machine. Consider the moment during the execution of the algorithm when the last such job was assigned to the second machine, and assume that this was the job of size Z_t. There are two cases considered according to whether it was assigned to the second machine in Step 3(a) or 3(b).

Subcase 2a. The job of size Z_t is assigned to the second machine in Step 3(a). We show that the job of size Y_t will be later assigned to the first machine. If it becomes a regular job, this holds since the job of size Z_t is the last job regular job which assigned to the second machine. Otherwise, it is the job of size Y^* and we assume that it is assigned to the first machine in the last step.

If the job of size X_t does not become the job of size X^*, then the job of size X_t also will be either assigned later to the first machine as a regular job, or if it becomes the job of size Y^* is it assigned to the same machine as well. In this case the final load of the first machine is therefore at least $X_t + Y_t > 2Y_t \geq 2(C_2(s) - 1)P_t$. It follows that $sL_t^2 + Z_t \leq P_t - X_t - Y_t \leq (3 - 2C_2(s))P_t$. On the other hand, we have $sL_{n+1}^2 + X^* = sL_t^2 + Z_t + X^* \geq sC_2(s)$, or $sL_t^2 + Z_t \geq sC_2(s) - s$. Thus the final load of the first machine including all jobs would be more than $2(C_2(s)-1)P_t \geq \frac{2s(C_2(s)-1)^2}{3-2C_2(s)}$. By our assumption, this load is smaller than $s+1-sC_2(s)$. We get $C_2(s) < \frac{s+3}{s+2}$. If $1 < s \leq \frac{\sqrt{5}+1}{2}$, then $C_2(s) - 1 = \frac{s}{s^2+s+1} > \frac{1}{s+2}$. If $\frac{\sqrt{5}+1}{2} < s < 2$, then $C_2(s) - 1 = \frac{s-1}{s^2-s+1} > \frac{1}{s+2}$ (which holds for $s > \frac{3}{2}$). Therefore, we get a contradiction.

We next consider the case where the job of size X_t becomes the job of size X^*. In this case, the total load of the second machine would remain at most $P_t - Y_t \leq P_t(1 - (C_2(s) - 1)) \leq (2 - C_2(s))P_n \leq (2 - C_2(s))(s + 1)$. If $1 < s \leq \frac{\sqrt{5}+1}{2}$, then $(2 - C_2(s))(s + 1) = \frac{(s^2+1)(s+1)}{s^2+s+1} = C_2(s)\frac{s^2+1}{s+1} < sC_2(s)$. If $\frac{\sqrt{5}+1}{2} < s < 2$, then $(2 - C_2(s))(s + 1) = \frac{(s^2-2s+2)(s+1)}{s^2-s+1} = C_2(s)\frac{s^3-s^2+2}{s^2} < sC_2(s)$, since $s > \sqrt{2}$. Therefore, we get a contradiction.

Subcase 2b. The job of size Z_t is assigned to the second machine in Step 3(b). Therefore, by the assignment condition, at this moment $L_t^1 + Y_t \geq c_2(s)\left(sL_t^2 + Z_t\right)$ holds. Since the final load of the second machine, excluding the job of size X^*, satisfies $sL_t^2 + Z_t = sL_{n+1}^2 > sC_2(s) - s$, we get that the final load of the first machine is at least $L_t^1 + Y^* \geq L_t^1 + Y_t \geq sc_2(s)(C_2(s) - 1)$, and thus, the final load of the second machine is at most $\frac{s+1-sc_2(s)(C_2(s)-1)}{s}$.

If $1 < s \leq \frac{\sqrt{5}+1}{2}$, then $sc_2(s)(C_2(s) - 1) = \frac{s+1}{s^2+s+1}$, and $s + 1 - \frac{s+1}{s^2+s+1} = \frac{s(s+1)^2}{s^2+s+1} = sC_2(s)$. If $\frac{\sqrt{5}+1}{2} < s < 2$, then $sc_2(s)(C_2(s) - 1) = \frac{1}{s^2-s+1}$, and $s + 1 - \frac{1}{s^2-s+1} = \frac{s^3}{s^2+s+1} = sC_2(s)$. Therefore, we get a contradiction.

Case 3. There are two big jobs at the end of the algorithm, whose sizes are Y^* and X^*. We would like to show that the ratio between the total size of jobs assigned to the first machine and the the total size of jobs assigned to the second machine is at most $c_2(s)$ just after all regular jobs have been assigned. If no regular jobs are ever assigned to the first machine we are done.

Otherwise, consider the moment when the last regular job of size Z_t is assigned to the first machine. Then by the assignment rule, $L_t^1 + Y_t < c_2(s)\left(sL_t^2 + Z_t\right)$ holds, from which it follows that $L_t^1 + Z_t < c_2(s)\left(sL_t^2 + Y_t\right)$ since $Z_t \leq Y_t$ holds for all $t \leq n$. The job of size Y_t cannot be the big job of size Y^* since $P_t \leq P_n \leq s + 1$, and $Y_t \leq (C_2(s) - 1)P_t \leq (C_2(s) - 1)(s + 1)$, but a big job has a size of more than $(C_2(s) - 1)(s + 1)$. Thus, the job of size Y_t is assigned the second machine as a regular job at some time $t' > t$. Furthermore, all other further jobs that are assigned as regular jobs are assigned to the second machine as well, so just before assigning the jobs of sizes Y^* and X^*, the load of the first machine is no larger than $c_2(s)$ times the total size of jobs on the second machine.

The sum of sizes of all regular jobs (i.e., all jobs excluding the jobs of sizes Y^* and X^*) is at most $s + 1 - 2Y^*$.

The load of the first machine, after all regular jobs are assigned, is therefore at most $\frac{c_2(s)}{c_2(s)+1}(s+1-2Y^*)$. Since both last jobs are big, assigning the job of size Y^* to the first machine would increase its load to more than $C_2(s)$, we have $\frac{c_2(s)}{c_2(s)+1}(s + 1 - 2Y^*) + Y^* > C_2(s)$.

If $1 < s \leq \frac{\sqrt{5}+1}{2}$, then $\frac{c_2(s)}{c_2(s)+1}(s+1-2Y^*)+Y^*-C_2(s) = \frac{s+1}{s^2+s+1}(s+1-2Y^*)+\frac{s^2+s+1}{s^2+s+1}Y^*-\frac{(s+1)^2}{s^2+s+1} = Y^*\frac{s^2-s-1}{s^2+s+1} > 0$ leads to a contradiction since $Y^* > 0$ and $s^2 \leq s + 1$.

If $\frac{\sqrt{5}+1}{2} < s < 2$, then $\frac{c_2(s)}{c_2(s)+1}(s+1-2Y^*)+Y^*-C_2(s) = \frac{1}{s^2-s+1}(s+1-2Y^*)+\frac{s^2-s+1}{s^2-s+1}Y^*-\frac{s^2}{s^2-s+1} =$

$(Y^* - 1)\frac{s^2-s-1}{s^2+s+1} > 0$. We consider the third lower bound on OPT. If the two large jobs are assigned to the second machine in an optimal solution, then $2Y^* \leq Y^* + X^* \leq s$, and therefore $Y^* \leq \frac{s}{2} \leq 1$. Otherwise, the job assigned to the first machine has size of at most 1, so again $Y^* \leq 1$. Therefore, we get a contradiction since $Y^* \leq 1$ and $s^2 \geq s+1$.

We next prove matching lower bounds. Consider the interval $1 < s \leq \frac{\sqrt{5}+1}{2}$, and let $K \geq 1$ be an arbitrary integer size of buffer. We give a sequence for which any algorithm has a competitive ratio of at least $\frac{(s+1)^2}{s^2+s+1} = 1 + \frac{s}{s^2+s+1}$.

The sequence starts with many very small jobs of size $\varepsilon > 0$, of total size 1. Denote the total size of jobs assigned to the first and second machines, respectively, after the arrival of these jobs by α and β, where $\alpha + \beta \geq 1 - K\varepsilon$. The cost of an optimal solution at this moment is $\frac{1}{s+1}$. Therefore, if $\alpha \geq \frac{s+1}{s^2+s+1}$ then the lower bound follows. Moreover, if $\beta \geq \frac{s^2+s}{s^2+s+1}$, the lower bound is implied as well. Thus suppose that $\alpha \leq \frac{s+1}{s^2+s+1}$, and $\beta \leq \frac{s^2+s}{s^2+s+1}$, i.e. $\frac{1}{s^2+s+1} - K\varepsilon \leq \alpha \leq \frac{s+1}{s^2+s+1}$ and $\frac{s^2}{s^2+s+1} - K\varepsilon \leq \beta \leq \frac{s^2+s}{s^2+s+1}$. One further job of size s arrives. The cost of an optimal solution becomes 1.

The machine which receives the last job would have a larger completion time than the other machine.

If the second machine receives this job, then the makespan is $\frac{\beta+s}{s} = 1 + \frac{\beta}{s} \geq 1 + \frac{s}{s^2+s+1} - \frac{K\varepsilon}{s}$. If the first machine receives this job, then the makespan is $\alpha + s \geq \frac{1}{s^2+s+1} - K\varepsilon + s = \frac{s^3+s^2+s+1}{s^2+s+1} \geq \frac{s^2+2s+1}{s^2+s+1} - K\varepsilon = \frac{(s+1)^2}{s^2+s+1} - K\varepsilon$.

In both cases the statement follows from letting ε tend to zero.

Consider the interval $\frac{\sqrt{5}+1}{2} < s < 2$, and let $K \geq 1$ be an arbitrary integer size of buffer. We give a sequence for which any algorithm has a competitive ratio of at least $\frac{s^2}{s^2-s+1}$ (note that $\frac{s^2}{s^2-s+1} < \frac{s+2}{s+1} \leq s$ in this range). The first phase is as in the previous case, and the values α and β are defined similarly.

Assume first that $\beta \geq \frac{s^2-s}{s^2-s+1} - K\varepsilon$, then one last job of size s arrives. The makespan in this case is at least

$$\min\left\{\alpha + s, \frac{\beta+s}{s}\right\} \geq \min\left\{s, 1 + \frac{\beta}{s}\right\} \geq \min\left\{1 + \frac{1}{s}, 1 + \frac{\beta}{s}\right\} = 1 + \frac{\beta}{s} \geq \frac{s^2}{s^2-s+1} - \frac{K\varepsilon}{s} .$$

Consider next the case where $\alpha > \frac{1}{s^2-s+1}$. Then let two further jobs arrive, where the sizes of the jobs are $Y = \frac{s^2-s+1}{s-1}\alpha$ and $X = sY - 1$. Note that $X - Y = (s-1)Y - 1 = (s-1)\frac{s^2-s+1}{s-1}\alpha - 1 = \left(s^2 - s + 1\right)\alpha - 1 > 0$, from which we get $Y < X$.

An optimal solution would be to assign Y to the first machine, and the other jobs to the second machine, which gives a makespan of Y. If at least one of the two last jobs is assigned to the first machine by the algorithm, then the makespan is at least $\alpha + Y = (1 + \frac{s^2-s+1}{s-1})\alpha = \frac{s^2}{s-1}\alpha$, which gives the required competitive ratio since OPT $= \frac{s^2-s+1}{s-1}\alpha$. Otherwise both of them are assigned to the second machine, and the total size of jobs assigned to the second machines will be $\beta + Y + X = 1 - \alpha - K\varepsilon + (s+1)Y - 1 = (s+1)\frac{s^2-s+1}{s-1}\alpha - \alpha - K\varepsilon = \frac{s^3+2-s}{s-1}\alpha - K\varepsilon$, thus the makespan tends to at least $\frac{s^2+2/s-1}{s-1}\alpha \geq \frac{s^2}{s-1}\alpha$ again (for $\varepsilon \to 0$). ■

References

[1] S. Albers. Better bounds for online scheduling. *SIAM Journal on Computing*, 29, 1999.

[2] E. Angelelli, A. Nagy, M. G. Speranza, and Zs. Tuza. The on-line multiprocessor scheduling problem with known sum of the tasks. *Journal of Scheduling*, 7(6):421–428, 2004.

[3] E. Angelelli, M. G. Speranza, and Zs. Tuza. Semi-online scheduling on two uniform processors. *Theoretical Computer Science*, 393(1-3):211–219, 2008.

[4] Y. Azar and O. Regev. Online bin stretching. *Theorectial Computer Science*, 268:17–41, 2001.

[5] M. Englert, D. Özmen, and M. Westermann. The power of reordering for online minimum makespan scheduling. In *Proc. 48th Symp. Foundations of Computer Science (FOCS)*, 2008. To appear.

[6] L. Epstein. Bin stretching revisted. *Acta Informatica*, 39(2):97–117, 2003.

[7] L. Epstein and L. M. Favrholdt. Optimal non-preemptive semi-online scheduling on two related machines. *J. Algorithms*, 57(1):49–73, 2005.

[8] L. Epstein, J. Noga, S. S. Seiden, J. Sgall, and G. J. Woeginger. Randomized Online Scheduling on Two Uniform Machines. *Journal of Scheduling*, 4(2):71–92, 2001.

[9] L. Epstein and D. Ye. Semi-online scheduling with "end of sequence" information. *Journal of Combinatorial Optimization*, 14(1):45–61, 2007.

[10] U. Faigle, W. Kern, and G. Turan. On the performance of online algorithms for partition problems. *Acta Cybernetica*, 9:107–119, 1989.

[11] T. Gormley, N. Reingold, E. Torng, and J. Westbrook. Generating adversaries for request-answer games. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 564–565, 2000.

[12] R. L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical J.*, 45:1563–1581, 1966.

[13] R.L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math*, 17:416–429, 1969.

[14] Y. He and G. Zhang. Semi on-line scheduling on two identical machines. *Computing*, 62(3):179–187, 1999.

[15] H. Kellerer, V. Kotov, M. G. Speranza, and Zs. Tuza. Semi online algorithms for the partition problem. *Operations Research Letters*, 21:235–242, 1997.

[16] S. Li, Y. Zhou, G. Sun, and G. Chen. Study on parallel machine scheduling problem with buffer. In *Proc. of the 2nd International Multisymposium on Computer and Computational Sciences (IMSCCS2007)*, pages 278–281, 2007.

[17] S. Seiden, J. Sgall, and G. Woeginger. Semi-online scheduling with decreasing job sizes. *Operations Research Letters*, 27(5):215–221, 2000.

[18] J. Sgall. On-line scheduling. In A. Fiat and G. Woeginger, editors, *Online Algorithms - The State of the Art*, chapter 9, pages 196–231. Springer, 1998.

[19] G. Zhang. A simple semi on-line algorithm for $P2//C_{\max}$ with a buffer. *Information Processing Letters*, 61:145–148, 1997.

[20] G. Zhang and D. Ye. A note on on-line scheduling with partial information. *Computers & Mathematics with Applications*, 44(3-4):539–543, 2002.

A The case $K = 1$ and $1 < s < \sqrt{2}$

We consider the range for which the previous bounds are not tight, which is $s \in (1, \sqrt{2})$. We conjecture that neither the lower bound of $\max\{s, \frac{(s+1)^2}{s^2+s+1}\}$ nor the upper bound of $\frac{2(s+1)}{s+2}$ are tight. We motivate this conjecture by considering several special cases. A case for which a better upper bound can be achieved ($s = \frac{4}{3}$), a case where $s < \frac{(s+1)^2}{s^2+s+1}$ ($s = \frac{6}{5}$), and an upper bound higher than $\frac{(s+1)^2}{s^2+s+1}$ can be achieved, and a case where $s > \frac{(s+1)^2}{s^2+s+1}$, and an upper bound higher than s can be achieved.

A.1 Algorithm

We define a new algorithm. The algorithm uses a parameter $C(s) > s$. We later analyze it for $s = \frac{4}{3}$.

Algorithm *Two-Conditions* **(***TC***)**

1. Store the first job in the buffer. Let $L_2^1 = L_2^2 = 0$, $P_1 = p_1$.
2. For each arriving job of index t act as follows.

   2.1 Let $P_t = P_{t-1} + p_t$. Let $P_t = P_{t-1} + p_t$, $M_t = \max\{M_{t-1}, p_t\}$. Consider the new job of index t and the job in the buffer. Let $X_t \geq Y_t$ be their sizes. The job of size X_t is stored in the buffer.

   2.2 Consider the following conditions.

   - $sL_t^2 + Y_t \leq s(C(s) - 1)P_t$
   - $sL_t^2 + Y_t + X_t \leq s \cdot C(s) \cdot LB_t$

   If both conditions hold, then assign the job of size Y_t to the second machine, and let $L_{t+1}^1 = L_t^1$, $L_{t+1}^2 = L_t^2 + \frac{Y_t}{s}$.

   2.3 Otherwise, assign it to the first machine and let $L_{t+1}^1 = L_t^1 + Y_t$, $L_{t+1}^2 = L_t^2$.

3. The last job which remains in the buffer is assigned to the second machine.

We show that the algorithm performs slightly better than *LL* for $s = \frac{4}{3}$, which was shown to have a competitive ratio of exactly 1.4 for this value of s.

Theorem 4 *For $s = \frac{4}{3}$, the competitive ratio of the algorithm using $C(s) = \gamma \approx 1.3907364$ is at most γ, where γ is the solution of $7x^3 - 4x^2 + 10x - 25 = 0$.*

Proof. Note that $s = \frac{4}{3} < \gamma < 1.4 = \frac{2(s+1)}{s+2} < \frac{s+1}{s} = \frac{7}{4}$, therefore $s + 1 - s\gamma > 0$. We also use $\gamma > \frac{(s+1)^2}{s^2+s+1} = \frac{49}{37}$, $\gamma \geq \frac{2s+1}{2s} = \frac{11}{8}$ and $\gamma \geq \frac{3s+2}{s+3} = \frac{18}{13}$. Suppose that the statement is not true and consider an instance what violates it. Assume that $\text{OPT} = 1$, then we have $L_1^n + sL_2^n + X_n + Y_n = P_n \leq s + 1 = \frac{7}{3}$.

Due to the definition of the algorithm, the competitive ratio can be violated either by the last job assigned to the second machine, or by some job which is assigned to the first machine. Assume first that the violation is caused by the last assigned job. Consider the situation at the time when the last job assigned to the second machine. If this machine contains no jobs at all, clearly the load of this machine does not exceed 1, since $X_n \leq s$. Otherwise, consider the previous job assigned to the second machine. Let the time of assignment be t'. We claim that the job of size $X_{t'}$ and the last job which remains in the buffer are not the same job, otherwise by the definition of the algorithm, the total size of jobs assigned to the second machine is $sL_n^2 + Y_{t'} + X_{t'} \leq s \cdot \gamma \cdot LB_{t'} \leq s\gamma\text{OPT} = s\gamma$.

We get that the job of size $X_{t'}$ is assigned to the first machine at some later time. Therefore, $L_n^1 \geq L_{t'}^1 + X_{t'}$. Since the job of size $Y_{t'}$ is assigned to the second machine, then by the first condition, $sL_{t'}^2 + Y_{t'} \leq s(\gamma - 1)P_{t'}$, and due to the violation of competitive ratio, $sL_n^2 + X_n = sL_{t'}^2 + Y_{t'} + X_n > s\gamma$. Using $X_n \leq s$ we get $s(\gamma - 1)P_{t'} > s\gamma - s$ and $P_{t'} > 1$.

We have $sL_{t'}^2 + Y_{t'} \leq s(\gamma - 1)P_{t'}$, i.e., $P_{t'} = sL_{t'}^2 + Y_{t'} + L_{t'}^1 + X_{t'} \leq s(\gamma - 1)P_{t'} + L_{t'}^1 + X_{t'}$, or $L_{t'}^1 + X_{t'} \geq P_{t'}(s + 1 - s\gamma) \geq s + 1 - s\gamma$. We get that the total size of all jobs is at least $P_{t'} + X_n = (L_{t'}^1 + X_{t'}) + (sL_{t'}^2 + Y_{t'} + X_n) > s + 1 - s\gamma + s\gamma = s + 1$, which is a contradiction.

Assume next that at some time t, a job of size Y_t is assigned to the first machine and violates the competitive ratio. We can in fact assume that $t = n$, since removing some jobs and scaling the input if necessary may only increase the competitive ratio. Therefore, $L_n^1 + Y_n > \gamma$ and $sL_n^2 + X_n = P_n - L_n^1 - Y_n < P_n - \gamma \leq s + 1 - \gamma$. Thus $Y_n \leq X_n < P_n - \gamma \leq s + 1 - \gamma$ implies $L_n^1 > \gamma - Y_n \geq 2\gamma - s - 1 > 0$, since $\gamma > s > 1$. Therefore, the first machine contains at least one job in addition to the job of size Y_n.

We analyze the conditions which led to the assignment of the job of size Y_n to the first machine. The first condition must hold since $sL_n^2 + X_n < P_n - \gamma \leq s(\gamma-1)P_n$. To prove the last inequality, assume by contradiction that $P_n - \gamma > s(\gamma-1)P_n$ or equivalently $P_n(s+1) > \gamma(sP_n+1)$. Using $\gamma > \frac{(s+1)^2}{s^2+s+1}$ we get $P_n(s^2+s+1) > (s+1)(sP_n+1)$ or $P_n > s+1$ which is a contradiction. Thus, it must be the case that the second condition does not hold, i.e., $P_n - L_n^1 = sL_n^2 + Y_n + X_n > s\gamma \cdot LB_n \geq s\gamma\frac{P_n}{s+1}$, or $L_n^1 < P_n(1-\frac{s\gamma}{s+1}) \leq s+1-s\gamma$.

Since $L_n^1 + Y_n > \gamma$ we have $X_n \geq Y_n > (s+1)(\gamma-1)$. Therefore, $sL_2^n = P_n - L_1^n - Y_n - X_n \leq s+1-\gamma-(s+1)(\gamma-1) = 2s+2-(s+2)\gamma$. Note that in an optimal schedule, each one of the two jobs of sizes Y_n and X_n must be assigned to different machines, since $X_n+Y_n > 2(s+1)(\gamma-1) \geq 2(s+1)\cdot\frac{s}{s^2+s+1} > s$, since $s^2 < s+1$. Note that the first machine must actually contain at least three jobs. Otherwise, if it contains two jobs, then the first job assigned to it has a size of more than $2\gamma - s - 1$ and it must be assigned to one of the machine in an optimal solution together with a job of size more than $(s+1)(\gamma-1)$. We get a total size of more than $(s+3)\gamma - 2(s+1) \geq s$, by the value of γ, which leads to a contradiction.

Denote the sizes of the two jobs assigned to the first machine before the last job by Y_t and $Y_{t'}$, where $t' < t < n$. Consider first the job of size X_t. If this job is not one of the two jobs of sizes X_n and Y_n, then it is eventually assigned to the second machine in step 2.2, and $sL_n^2 \geq sL_t^2 + X_t$. Due to the choice of the time t, we have $L_n^1 = L_t^1 + Y_t$. We test the two conditions at the time of assignment of the job of size Y_t.

Assume that the first condition does not hold. Then we have $(s+1-s\gamma)(sL_t^2+Y_t) > s(\gamma-1)(L_t^1+X_t)$. Using $sL_t^2 + Y_t \leq sL_t^2 + X_t \leq L_n^2 \leq s+1-\gamma-X_n$ and $L_t^1 + X_t \geq L_t^1 + Y_t = L_n^1 > \gamma - Y_n \geq \gamma - X_n$ we get $(s+1-s\gamma)(s+1-\gamma-X_n) > s(\gamma-1)(\gamma-X_n)$. Simplifying, we get $(s+1)^2 - \gamma(s^2+s+1) + X_n(2s\gamma-2s-1) > 0$. We have $\gamma \geq \frac{2s+1}{2s}$, so using $X_n \leq s+1-\gamma$ we get $(s+1)^2 - \gamma(s^2+s+1) + (s+1-\gamma)(2s\gamma-2s-1) > 0$. Simplifying the last expression gives $s+1 < (s+3-2\gamma)\gamma$, or $6\gamma^2 - 13\gamma + 7 < 0$, which does not hold for $\gamma > \frac{4}{3}$ and leads to a contradiction. Assume next that the second property does not hold. We get $sL_t^2 + Y_t + X_t > s\cdot\gamma\cdot LB_t \geq \frac{s\gamma}{s+1}P_t = \frac{s\gamma}{s+1}(sL_t^2 + L_t^1 + Y_t + X_t)$, i.e., $(s+1-s\gamma)(sL_t^2+X_t) + (s+1)Y_t > s\gamma(L_t^1+Y_t)$. Since $sL_n^2 \geq sL_t^2 + X_t \geq Y_t$, we have $(s+1-s\gamma)(sL_t^2+X_t) + (s+1)Y_t \leq (2s+2-s\gamma)sL_n^2 \leq (2s+2-s\gamma)(s+1-\gamma-X_n)$. Using $L_t^1 + Y_t = L_n^1 > \gamma - X_n$, we get $(2s+2-s\gamma)(s+1-\gamma-X_n) > s\gamma(\gamma-X_n)$. Simplifying we get $2(s+1)^2 - \gamma(s+2)(s+1) > 2X_n(s+1-s\gamma) > 2(s+1)(\gamma-1)(s+1-s\gamma)$. This results in $32\gamma^2 + 94\gamma - 154 < 0$, which does not hold for $\gamma > \frac{4}{3}$, thus we reach a contradiction.

Therefore we are left with the case where the job of size X_t is one of the jobs of sizes X_n and Y_n, thus $X_t \geq (s+1)(\gamma-1)$, and there is another job of at least this size to arrive. Thus $P_t \leq P_n - Y_n$. We have $P_t = L_n^1 + sL_n^2 + Y_t + X_t$. We again check which condition led to the assignment of the job of size Y_t to the first machine.

Assume that the first condition does not hold. We get $P_t - X_t \geq sL_n^2 + Y_t > s(\gamma-1)P_t$ and therefore $X_t < P_t(s+1-s\gamma) \leq (s+1-s\gamma)(P_n - Y_t)$. We get $(s+1)(\gamma-1)(s+2-s\gamma) \leq Y_t(s+2-s\gamma) \leq (s+1-s\gamma)P_n \leq (s+1-s\gamma)(s+1)$. Simplifying, we get $s\gamma^2 - \gamma(3s+2) + 2s + 3 \geq 0$, or $4\gamma^2 - 18\gamma + 17 \geq 0$, which does not hold for $\gamma > 1.35$.

Therefore it must be the case that the second condition does not hold, we get $s+1-\gamma-X_n+Y_t \geq sL_n^2 + Y_t \geq sL_t^2 + Y_t > s\cdot\gamma\cdot LB_t - X_t \geq (\gamma-1)Y_n$, using $LB_t \geq \frac{X_t}{s}$. Thus $Y_t > \gamma(\gamma-1)(s+1) + \gamma - s - 1 = (s+1)\gamma^2 - s\gamma - (s+1)$.

Note that $Y_t + Y_n > (s+1)\gamma^2 - s\gamma - (s+1) + (s+1)(\gamma-1) = (s+1)\gamma^2 + \gamma - 2(s+1) > 1$, and $2Y_t + Y_n > 2(s+1)\gamma^2 - 2s\gamma - 2(s+1) + (s+1)(\gamma-1) = 2(s+1)\gamma^2 + (1-s)\gamma - 3(s+1) > s$, so the three jobs of sizes Y_t, Y_n and X_n are the three largest jobs, out of which two must be assigned to the fast machine in an optimal solution. These two cannot be the jobs of sizes Y_n and X_n. Thus $Y_t + Y_n \leq s$, and therefore $L_t^1 = L_n^1 - Y_t > \gamma - Y_n - Y_t > \gamma - s$.

Recall that a job of size $Y_{t'}$ is the last job which was assigned to the first machine before the job of size Y_t. We consider the job of size $X_{t'}$. If this job is not one of the three largest jobs, then it is assigned to the fast machine at step 2.2 and we have $sL_n^2 \geq sL_{t'}^2 + X_{t'}$ and $L_n^1 = L_{t'}^1 + Y_{t'} + Y_t$. We test the two conditions at the time of assignment of the job of size $Y_{t'}$.

Assume that the first condition does not hold. Then we have $(s+1-s\gamma)(sL_{t'}^2+Y_{t'}) > s(\gamma-1)(L_{t'}^1+X_{t'})$. Using $sL_{t'}^2+Y_{t'} \leq sL_{t'}^2+X_{t'} \leq sL_n^2 \leq s+1-\gamma-X_n \leq s+1-\gamma-(s+1)(\gamma-1)$, $L_{t'}^1+Y_{t'}+Y_t+Y_n = L_n^1 > \gamma$ and $Y_t + Y_n \leq s$ gives $(s+1-s\gamma)(2s+2-(s+2)\gamma) > s(\gamma-1)(\gamma-s)$. Simplifying, we get $(s^2+s)\gamma^2 - 2\gamma(s+1)^2 + (s^2+4s+2) > 0$, which does not hold.

Next, assume that the second condition does not hold. We use $LB_{t'} \geq \frac{P_{t'}}{s+1}$ and get $(s+1-s\gamma)(sL_{t'}^2 + X_{t'}) + (s+1)Y_{t'} > s\gamma(L_{t'}^1 + Y_{t'}) = s\gamma L_t^1 > s\gamma(\gamma - s)$. Since $Y_{t'} \leq sL_{t'}^2 + X_{t'} \leq sL_n^2 - X_n \leq s+1-\gamma-(s+1)(\gamma-1)$, we get $(2s+2-s\gamma)(s+1-\gamma-(s+1)(\gamma-1)) > s\gamma(\gamma-s)$. The left hand side is equal to approximately 0.08684 whereas the right hand side is equal to approximately 0.10644, which leads to a contradiction.

Thus the job of size $X_{t'}$ is one of the three largest jobs, and we have $Y_{t'} \leq s - Y_n - Y_t \leq s - (s+1)\gamma^2 - \gamma + 2(s+1)$ if the job of size $Y_{t'}$ is assigned to the fast machine in an optimal solution, and otherwise $Y_{t'} \leq 1 - Y_n \leq 1-(s+1)(\gamma-1)$. The first bound is larger, therefore we use $Y_{t'} \leq s - Y_n - Y_t \leq s-(s+1)\gamma^2-\gamma+2(s+1) \approx 0.096525$.

At this time, the job $Y_{t'}$ is assigned to the first machine, therefore at least one of the two conditions does not hold. Assume by contradiction that the first condition does not hold. We have $sL_{t'}^2 \leq sL_n^2 - X_n \leq s+1-\gamma-(s+1)(\gamma-1)$, and $X_{t'} \geq Y_t \geq (s+1)\gamma^2 - s\gamma - (s+1)$. Thus $sL_{t'}^2 + Y_{t'} \leq 0.127131$ and $X_{t'} \geq 0.32536276$. We get $(\frac{7}{3} - \frac{4}{3}\gamma)(sL_{t'}^2 + Y_{t'}) > \frac{4}{3}(\gamma-1)(X_{t'} + L_{t'}^1)$, which does not hold.

Assume by contradiction that the second condition does not hold. Using $s\gamma LB_{t'} \geq \gamma X_{t'}$, we get $sL_{t'}^2 + Y_{t'} > (\gamma-1)X_{t'}$. However, the bounds on the two values are equal (by the definition of γ), which is a contradiction as well.

Since none of the options is possible, we conclude that the counter example does not exist. ■

A.2 Improved lower bounds for some small values of s

We consider two values of s, namely $s = \frac{4}{3}$ and $s = \frac{6}{5}$ and show slightly higher lower bounds than those shown in the body of the paper. The lower bound for $s = \frac{6}{5}$ shown in the body of the paper is approximately 1.3296, and the lower bound which was shown for $s = \frac{4}{3}$ is $\frac{4}{3}$.

Theorem 5 *The competitive ratio of any algorithm which uses a buffer of size $K = 1$ is at least $\frac{4}{3}$ for $s = \frac{6}{5}$. The competitive ratio of any algorithm which uses a buffer of size $K = 1$ is at least 1.37 for $s = \frac{4}{3}$.*

Proof. We first consider the case $s = \frac{6}{5}$. Let $0 < \varepsilon < \frac{1}{16500}$ be a number such that $\frac{1}{\varepsilon}$ is an integer.

The sequence starts with many very small jobs of size $\varepsilon > 0$, of total size 1. Denote the total size of jobs assigned to the first and second machines, respectively, after the arrival of these jobs by α and β, where $1-\varepsilon \leq \alpha+\beta \leq 1$. At this time, $\text{OPT} = \frac{1}{1+s} = \frac{5}{11}$ and therefore, in order not to achieve a competitive ratio of at least $\frac{4}{3}$, it follows that $\alpha < \frac{4}{3} \cdot \frac{5}{11} = \frac{20}{33}$, and $\beta < \frac{6}{5} \cdot \frac{20}{33} = \frac{24}{33}$. Thus $\frac{9}{33} \leq \alpha < \frac{20}{33}$.

If the next and last job has a size of $\frac{6}{5}$, then after its arrival $\text{OPT} = 1$. Assigning this job to to the slow machine would create a makespan of at least $\frac{9}{33} + \frac{6}{5} > \frac{4}{3}$, thus the last job must be assigned to the fast machine. However, in order not to achieve a competitive ratio of $\frac{4}{3}$, the load of the second machine must be lower than $\frac{4}{3} \cdot \frac{6}{5}$, thus $\beta < \frac{1}{3} \cdot \frac{6}{5} = \frac{2}{5}$ must hold, which implies $\frac{3}{5} = \frac{99}{165} \leq \alpha < \frac{20}{33} = \frac{100}{165}$.

Assume that the job of size $\frac{6}{5}$ does not arrive after all, and instead two jobs of sizes $X = 3$ and $Y = \frac{9}{5}$ arrive. At this time $\text{OPT} = \frac{1+1.8+3}{2.2} = \frac{5.8}{2.2} = \frac{29}{11} \approx 2.6364$, by assigning the job of size X to the fast machine, the job of size Y to the slow machine, and spreading the small jobs to allow the two machines equal completion times.

At this time, at least one of the two larger jobs must be assigned to one of the machines. We consider several cases. If the job of size X is assigned to the slow machine, then the makespan will be at least

$\alpha + 3 \geq 3.6 > \frac{4}{3}\textsc{opt}$ since $\frac{4}{3} \cdot \frac{29}{11} \approx 3.5152$. Similarly, if at this time, the job of size Y is assigned to the fast machine, then either the job of size X will be assigned to the slow machine, and the previous proof can be used, or the load of the fast machine would become $\beta + X + Y > 4.8$, whereas $\frac{4}{3} \cdot \frac{6}{5} \cdot \frac{29}{11} < 4.8$.

Consider the case that the job of size Y is assigned to the slow machine. Then an additional job of size $X' = 3.11$ arrives, which is the last job. At this moment $\textsc{opt} = 4.05$, by assigning a job of size X, a job of size Y, and a total of 0.06 of the small jobs to the fast machine, and all other jobs to the slow machine. Next, if one of the two remaining jobs is assigned to the slow machine, then we get a load of at least $\alpha + 1.8 + 3 \geq 5.4 = \frac{4}{3}\textsc{opt}$, and otherwise both these jobs are assigned to the fast machine and its load will be $\beta + 6.11 \geq \frac{13}{33} + 6.11 > \frac{6}{5} \cdot \frac{4}{3}\textsc{opt}$, and the lower bound holds.

Thus only one case remains, where one job of size Y and one job of size X had arrived, and the job of size $X = 3$ is assigned to the fast machine. In this case an additional job of size $Y = 1.8$ arrives, and two such jobs are present. One such job must be assigned to one of the machines.

Suppose that a job of size Y is assigned to the slow machine. Then a final job of size $Z = \frac{6}{5}(1+3+1.8+1.8) = 9.12$ arrives. We have $\textsc{opt} = 7.6$. The makespan will be at least $\min\left\{\alpha + 1.8 + Z, \frac{\beta+3+Z}{1.2}\right\} \geq \min\left\{0.6 + 1.8 + 9.12, \frac{13/33+3+9.12}{1.2}\right\} \approx 10.42 > \frac{4}{3} \cdot 7.6$.

Finally, consider the case where the job of size Y is assigned to the fast machine. A third job of size $Y = 1.8$ arrives, which results in two pending jobs of this size. At this moment $\textsc{opt} = \frac{47}{11}$, where in an optimal solution two jobs of size Y are assigned to the slow machine, the other two jobs are are assigned to the fast machine, and the small jobs are spread to let the two machines have equal loads.

If an additional job of size Y is assigned to the fast machine, then the total size of jobs assigned to it is $\beta + 3 + 1.8 + 1.8 > 6.99 > \frac{6}{5}\frac{4}{3}\frac{47}{11}$. Otherwise, a job of size Y is assigned to the slow machines, and the last job of size $U = \frac{6}{5}(1 + 3 + 1.8 + 1.8 + 1.8) = 11.28$ arrives and $\textsc{opt} = 9.4$. The makespan will be at least $\min\left\{\alpha + 1.8 + U, \frac{\beta+3+1.8+U}{1.2}\right\} \geq \min\left\{0.6 + 1.8 + 11.28, \frac{13/33+3+1.8+11.28}{1.2}\right\} = 13.68 > \frac{4}{3} \cdot 9.4$. We have thus showed that in all possible cases the lower bound holds as required.

To prove a lower bound for $s = \frac{4}{3}$, Let $\frac{4}{3} < c \leq 1.4$ the value of the lower bound which is proved. The sequence starts as in the previous case, with the possibility that an additional job of size $\frac{4}{3}$ may arrive. In this case, the resulting bounds on α are $1 - \frac{c}{3} \leq \alpha < \frac{3c}{7}$. We next have a job of size X and a job of size Y, where $Y < X$, and the values of X and Y are chosen so that $s(Y + 1) \leq X$ and $Y \leq (s - 1)X + s$.

At this time there are three possible inputs. In the first input the sequence stops, in the second input an additional job of size X arrives, and in the third input, a job of size $Z = s(sX + s + 1)$ arrives. We give bounds on $\textsc{opt}$ in the three cases. In the first case, it is possible to assign the job of size X to the fast machine, and the other jobs to the slow machine. Since $Y + 1 \leq \frac{X}{s}$, we get $\textsc{opt} \leq \frac{X}{s}$. In the second case, it is possible to assign jobs of sizes X and Y to the fast machine, and all other jobs to the slow machine. Since $\frac{Y+X}{s} \leq X + 1$, we have $\textsc{opt} \leq X + 1$. In the third case the large job is assigned to the fast machine and $\textsc{opt} = sX + s + 1$.

If the job of size X is assigned to the slow machine or the job of size Y is assigned to the fast machine, the first input is used. The makespan is at least $\min\{\alpha + X, \frac{\beta+X+Y}{s}\}$.

If the job of size Y is assigned to the slow machine, the second input is used. The makespan is at least $\min\{\alpha + Y + X, \frac{\beta+2X}{s}\}$.

If the job of size X is assigned to the fast machine, the third input is used. The makespan is at least $\min\{\alpha + Z, \frac{\beta+X+Z}{s}\}$.

It is not difficult to verify that the choice $X = 8.6$ and $Y = 4.2$ yields the required lower bound. ■