Randomized Online Scheduling on Two Uniform Machines

Leah Epstein* John Noga† Steve Seiden‡ Jiří Sgall§ Gerhard Woeginger‡

Abstract

We study the problem of online scheduling on two uniform machines with speeds 1 and $s \geq 1$. For this problem, a $\phi \approx 1.61803$ competitive deterministic algorithm was already known. We present the first randomized results for this problem: We show that randomization does not help for speeds $s \geq 2$, but does help for all $s < 2$. We present a simple memoryless randomized algorithm that is $(4-s)(1+s)/4 \leq 1.5625$ competitive. We analyze other randomized algorithms that demonstrate that the randomized competitive ratio is at most 1.53 for any s. Our algorithms for $s > 1.5$ are *barely random*—they use only a constant number of random bits, and similar methods yield also barely random algorithms for scheduling on two and three identical machines. Finally, we present a $1 + s/(s^2+s+1)$ competitive deterministic algorithm for the preemptive version of this scheduling problem. For any $s \geq 1$, this algorithm is optimal even among *randomized* preemptive algorithms.

*Department of Computer Science, Tel-Aviv University. `lea@math.tau.ac.il`.

†Department of Mathematics, University of California, Riverside, CA 92521-0304. `jnoga@cs.ucr.edu`. This research has been supported by NSF grants INT9724750 and CCR9503498.

‡Institut fuer Mathematik, TU Graz, Steyrergasse 30, A-8010 Graz, Austria. `sseiden@acm.org`, `gwoegi@opt.math.tu-graz.ac.at`. This research has been supported by the START program Y43-MAT of the Austrian Ministry of Science.

§Mathematical Inst., AS CR, Žitná 25, 115 67 Praha 1, and Dept. of Applied Mathematics, Faculty of Math. and Physics, Charles University, Praha, Czech Republic. `sgall@math.cas.cz`, `http://www.math.cas.cz/~sgall/`. Partially supported by grant A1019602 of GA AV ČR, postdoctoral grant 201/97/P038 of GA ČR, and cooperative research grant INT-9600919/ME-103 from the NSF (USA) and the MŠMT (Czech republic).

1 Introduction

In this paper, we study online scheduling of independent jobs with the objective to minimize the *makespan* (i.e., the length of the schedule, or equivalently the last job completion time). An input to this problem consists of a sequence of jobs of different *sizes* (or *running times*), each of which has to be assigned to one of two machines. There are no dependencies between the jobs and no release times. The machines are *uniform* in that a job with running time t is processed in time t on the fast machine, and in time $s \cdot t$ on the slow machine. We consider the *non-preemptive* version of this problem (once a job is started, it must be processed without interruption till completion on the same machine) and the *preemptive* version (the processing of a job may be interrupted and resumed arbitrarily, and may even be switched between the two machines). In the non-preemptive version, it is never useful to introduce idle time; so the makespan is simply the maximum over machines of the sum of running times of assigned jobs.

In the online problem, jobs arrive one by one and we must assign each job before we see the next one. The performance of an online algorithm is measured by its *competitive ratio*, which is the largest ratio of the makespan achieved by the online algorithm with the minimal makespan. For randomized algorithms, we use expected makespan in the definition of the competitive ratio (this corresponds to the so-called oblivious adversary). See [4] for a general reference on online algorithms, and [13] for a survey of online scheduling.

In this paper, we are mainly interested in randomized online algorithms for non-preemptive scheduling, and in the dependency of the optimal competitive ratio on the speed s. We find this problem interesting for (at least) two reasons: First, the problem is related to the notorious problem of randomized scheduling on m identical machines. This is a very basic problem in the area of competitive analysis, and our understanding of randomized algorithms for it is very poor [3, 12, 13]. Roughly speaking, there is an analogy between assigning jobs to the slow and fast machines in our problem and to one machine (or a few machines) and the remaining machines in the m-machine problem. We believe that a better understanding of algorithms for two uniform machines will help us to attack the m-machine problem.

Second, as a general rule, our understanding of randomized algorithms is good for problems that exhibit some symmetric structure; however, once we break the symmetry, our understanding worsens rapidly. E.g., in the k-server problem we have optimal algorithms for the paging problem [9] which corresponds to the uniform metric space. Once we move away from the uniform space (and from symmetry), the problem is widely open even for 2 servers [2, 7]. In our scheduling problem we fully understand the case of $s = 1$. This is the symmetric case, where the machines can arbitrarily interchange their roles during the algorithm's execution; the solution and the analysis of this case strongly rely on this fact. The case $s \neq 1$ is the non-symmetric case. We believe that the study of algorithms for accessible non-symmetric problems is an important research area, and we hope that further progress can be made.

Let us first discuss our results for the non-preemptive version. We start by analyzing the performance of deterministic algorithms depending on the speed s. It turns out that the greedy LIST algorithm is best for any speed: The optimal competitive ratio is $(2s+1)/(s+1)$ for $s \leq \phi$ and $1 + 1/s$ for $s \geq \phi$ ($\phi \approx 1.61803$ is the golden ratio). Much of the material is implicit in [6], where the upper bound is proved for all s as well as the fact that this is tight for speed $s = \phi$. We include this analysis since the ideas used are essential for the rest of the paper.

We prove that randomization does not help at all for $s \geq 2$. For $1 < s < 2$ we prove lower bounds showing that the randomized competitive ratio for any s in this range is strictly larger than $4/3$, the optimal randomized competitive ratio for $s = 1$ [3].

On the other hand, for any $1 < s < 2$, even the simple *memoryless* algorithm which schedules each job independently with probability $s/2$ on the fast machine is better than any deterministic algorithm; its competitive ratio is $(4-s)(1+s)/4$. Is this memoryless algorithm the best possible randomized algorithm for speeds $1 < s < 2$? The answer to this question is NO, and it is even NO for every single speed in this range. We develop different algorithms for the case of a *small* speed $1 < s < 3/2$, and for the case of a *medium* speed $3/2 \leq s < 2$.

For small speeds, as s approaches 1, the competitive ratio of the memoryless algorithm approaches $3/2$ (which is the optimal *deterministic* competitive ratio for two identical machines) and stays far away from $4/3$ (which is the optimal *randomized* competitive ratio for two identical machines [3]). We design a randomized algorithm whose competitive ratio approaches $4/3$ as s approaches 1; for small speeds, this is our best algorithm for small speeds, always better than the memoryless algorithm.

As our main result we develop good randomized algorithms for medium speeds. For $s = 3/2$ the memoryless algorithm schedules each job on the slow machine with probability $1/4$. Its obvious drawback is that its decisions are independent of each other. Thus, if we get three big jobs, with significant probability it schedules two of them on the slow machine; this results in rather poor performance. To improve on this, we use a deterministic algorithm for scheduling on four imaginary machines. We pick one of the four machines at random and identify it with the slow machine, whereas the other three are identified with the fast machine. Hence, on average $1/4$ of the load goes on the slow machine (as in the memoryless algorithm), but if in the previous example the 4-machine algorithm schedules the three big jobs on different machines, then we

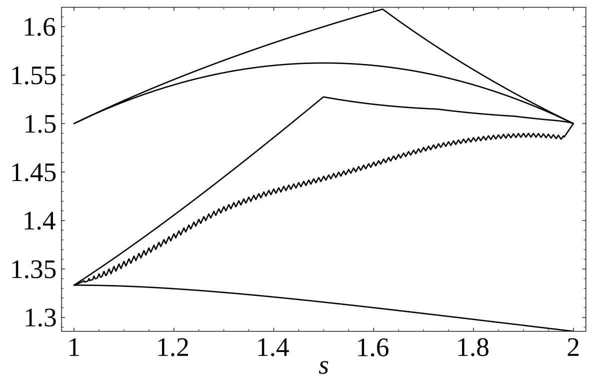

Figure 1: Deterministic, memoryless randomized, and general randomized upper bounds, the randomized lower bound, and the optimal preemptive bound (from top to bottom) for $1 \leq s \leq 2$.

manage to avoid the poor performance of memoryless algorithms. Our algorithms are *barely random* [10], i.e., they use only a constant number of random bits (2 bits for speed 3/2).

The same technique also yields barely random algorithms for two and three identical machines. For three machines, a distribution over 15 deterministic strategies yields a competitive ratio of $23/14 < 1.64286$. Although these algorithms do not perform as well as previously known ones [3, 12], they are much simpler and do outperform the best possible deterministic algorithm.

For the preemptive version, we present a complete solution: For any $s \geq 1$, there exists a deterministic online algorithm that is $1 + s/(s^2 + s + 1)$ competitive, and even randomized algorithms cannot improve on this. Note that the optimal competitive ratio strictly decreases in the whole range of $s \geq 1$. This is in contrast to the non-preemptive version, where for the deterministic and for the randomized case the optimal competitive ratio first increases with the speed and then decreases to 1. See Figure 1 for a comparison of the bounds.

Most of our lower bounds are achieved on the *critical sequence*, which is defined so that each job (except for the first one) takes as long to run on the fast machine as all the previous jobs on the slow machine. Thus we define the running times of jobs of the critical sequence as

$$\begin{aligned} r_0 &= 1, \\ r_i &= s(s+1)^{i-1} \quad \text{for } i \geq 1. \end{aligned} \tag{1}$$

For the critical sequence ending at r_i, the optimal makespan is equal to $r_i = s(s+1)^{i-1}$, and in any schedule the makespan is determined by the the machine on which the last job was scheduled.

2 Deterministic scheduling

The algorithm List was originally introduced for scheduling jobs on m identical machines. List is a greedy algorithm and schedules every job on the machine where it would finish first. This algorithm was first studied by Graham [8], who showed that List is $2 - 1/m$ competitive for all m. List is the best possible deterministic online algorithm for $m = 2, 3$, whereas for all $m \geq 4$ better algorithms are known [1, 5].

In [6] it is proved that List is $\phi \approx 1.61803$ competitive for scheduling on two uniform machines, and that no better online algorithm is possible. The following theorem gives the exact dependence of the competitive ratio on the speed and shows that List is optimal for any speed s.

Theorem 2.1 *For any $s \geq 1$, the competitive ratio of* List *is*

$$c_{\text{List}} = \min\left\{\frac{2s+1}{s+1}, 1 + \frac{1}{s}\right\}.$$

For any $s \geq 1$, the competitive ratio of any non-preemptive deterministic online algorithm is at least c_{List}.

The proof of the lower bound appears in Appendix A. The upper bound is from [6]. For $s \leq \phi$ the bound of $(2s + 1)/(s + 1)$ follows Graham's analysis of List on identical machines. For $s \geq \phi$, we consider the algorithm that schedules all jobs on the faster machine. This is clearly not worse than List. If the makespan of this algorithm is x, the optimal algorithm has to schedule jobs with total load x as well; it can do it only a factor of $1 + 1/s$ faster, since the slow machine can only process $1/s$ of the load of the fast machine. The bound of $1 + 1/s$ follows.

3 Randomized lower bounds

For $s \geq 2$, we prove that randomization does not help and still the strategy of putting all jobs on the fast machine is optimal. We make use of the Von Neumann-Yao principle [14]. This principle allows us to prove a lower bound for any randomized algorithm by exhibiting a distribution over inputs and showing that any *deterministic* algorithm incurs a cost at least c times the optimal offline cost for this distribution.

Theorem 3.1 *For $s \geq 2$, the competitive ratio of any non-preemptive randomized online algorithm is at least $c_{\mathrm{LIST}} = 1 + 1/s$.*

Proof For a fixed n, consider the following distribution on input instances. Choose uniformly at random $i = 1, \ldots, n$, and take the instance consisting of the first i jobs of the critical sequence (i.e., ending with a job with running time r_{i-1}). We prove that for any deterministic algorithm, the average value of the competitive ratio on this distribution of inputs is at least $1 + 1/s - 1/(ns)$. The theorem follows by taking sufficiently large n.

The action of a deterministic algorithm is determined by a set $S \subseteq \{0, \ldots, n-1\}$ of the indices of jobs that are scheduled on the slow machine. We show that the best average competitive ratio is achieved if $S = \emptyset$; the computation essentially shows that scheduling any single job on the fast machine only improves the performance.

If a job r_i is scheduled on the slow machine, the competitive ratio after i jobs is at least s (since the optimum is r_i but running the job on the slow machine takes sr_i). If it is scheduled on the fast machine, the competitive ratio is

$$\sum_{k \leq i, k \notin S} \frac{r_k}{r_i} = \sum_{k \leq i} \frac{r_k}{r_i} - \sum_{k < i, k \in S} \frac{r_k}{r_i}.$$

Denote $u_i = \sum_{k \leq i} r_k / r_i$. For $i > 0$ we get $u_i = 1 + 1/s$ and $u_0 = 1$. The average competitive ratio is at least

$$\frac{1}{n} \left(\sum_{j \in S} s + \sum_{i \notin S} \left(u_i - \sum_{k < i, k \in S} \frac{r_k}{r_i} \right) \right)$$

$$= \frac{1}{n} \left(\sum_{i=0}^{n-1} u_i + \sum_{j \in S} (s - u_j) - \sum_{i \notin S} \sum_{j < i, j \in S} \frac{r_j}{r_i} \right)$$

$$= 1 + \frac{1}{s} - \frac{1}{ns} + \frac{1}{n} \sum_{j \in S} \left(s - u_j - \sum_{i > j, i \notin S} \frac{r_j}{r_i} \right).$$

Now we prove that the expression in the sum is non-negative for all j. For $j > 0$ we have

$$\sum_{i > j, i \notin S} \frac{r_j}{r_i} \leq \sum_{i=1}^{\infty} (s+1)^{-i} = \frac{1}{s}.$$

Thus the whole expression is at least $s - u_j - 1/s = s - 1 - 2/s \geq 0$, since $s \geq 2$. For $j = 0$, a similar calculation yields that the expression is at least $s - 1 - (s+1)/s^2$ which is non-negative too. Therefore the average competitive ratio is at least $1 + 1/s - 1/(ns)$. ■

Next, we show a lower bound for any randomized algorithm for speeds $s < 2$. Again, we apply the von Neumann-Yao principle. With probability p_i the input sequence contains the first i jobs of the critical sequence, for $i = 1, \ldots, n$. Solving a linear program on a computer, we find the best lower bounds achievable by such distributions, for fixed values of s and n. There are 2^n possible deterministic algorithms for scheduling n jobs. The makespan for algorithm j after i jobs, denoted $M_{i,j}$, can be computed using the following recurrence.

$$\begin{aligned} L_{0,j} &= \langle 0, 0 \rangle \\ L_{i,j} &= \begin{cases} L_{i-1,\lfloor j/2 \rfloor} + \langle r_{i-1}, 0 \rangle & \text{for odd } j \\ L_{i-1,\lfloor j/2 \rfloor} + \langle 0, s \cdot r_{i-1} \rangle & \text{otherwise} \end{cases} \\ M_{i,j} &= \max L_{i,j} \end{aligned}$$

The expected competitive ratio incurred by algorithm j is $C_{n,j}$ can be expressed as a fixed linear combination of p_i's for each n and s:

$$C_{n,j} = \sum_{i=1}^{n} p_i \frac{M_{i,j}}{r_i} \tag{2}$$

Define the following linear program over variables c, $p_1, \ldots, p_n$: Maximize c subject to

$$\sum_{i=1}^{n} p_i = 1, \qquad c \leq C_{n,j} \quad \text{for } j = 0, \ldots, 2^n - 1.$$

Any feasible c is a lower bound on the competitive ratio. For $n = 15$ and $s = 1 + i/100$ for $i = 1, \ldots, 99$, we have calculated the optimal values of this linear program by using CPLEX. The actual values computed are given in Appendix E. We use the following lemma to extend these bounds to all $s \in [1, 2]$.

Lemma 3.2 *If the competitive ratio is at least c_0 at speed s_0, then $c_0 \min\{s_0/s, s/s_0\}$ is a lower bound for any s.*

Proof Suppose that there is a c-competitive algorithm ALG at speed s. First consider $s < s_0$. If we use ALG at speed s_0, its cost for any instance increases at most by a factor of s_0/s. The optimal offline cost can only increase with speed. Therefore, ALG is cs_0/s-competitive at speed s_0, and $c \geq c_0 s/s_0$. Now consider $s > s_0$. If we use ALG at speed s_0, its cost does not increase, and the optimal offline cost decreases at most by a factor of s_0/s. Therefore, ALG is cs/s_0-competitive at speed s_0, implying $c \geq c_0 s_0/s$. ∎

To get a lower bound strictly greater than 4/3 for small s, we need an analytic lower bound:

Lemma 3.3 *For $1 \leq s \leq \sqrt{2}$, the competitive ratio of any non-preemptive randomized online algorithm is at least $(1+s)^2(2+s)/(3s^2+5s+1)$.*

The proof is given in in Appendix A. All the lower bounds are illustrated in Figure 1.

4 The memoryless algorithm

In this section we analyze the memoryless algorithm BIAS: Whenever a job arrives, BIAS schedules it on the fast machine with probability $s/2$, and otherwise assigns it to the slow machine. All random choices are made independently.

Theorem 4.1 *The competitive ratio of* BIAS *is $c_{\text{BIAS}} = (4 - s)(1 + s)/4$.*

Proof We prove that the worst case competitive ratio of BIAS is achieved on some sequence of two jobs. (In fact, this is true even if the probability of assigning a job on the fast machine is a different constant than $s/2$.)

If a sequence of jobs contains three or more jobs then an optimal schedule has some two jobs scheduled on the same machine; denote their running times t, t', so that $t \geq t'$. We show that for any random choices of BIAS for the other jobs, replacing the two jobs by a single job with running time $t + t'$ does not decrease the makespan. Since the optimal makespan does not change, the competitive ratio cannot decrease.

There are three cases how the two jobs can be assigned: (i) t on the fast machine and t' on the slow machine, (ii) t' on the fast machine and t on the slow machine, and (iii) both on the same machine (fast or slow). The cases (i) and (ii) have the same probability $q = s(2 - s)/4$. If we replace the two jobs by a single job of size $t + t'$, then the probability distribution changes so that instead of (i) and (ii), we schedule the new job with probability q on the fast machine, with probability q on the slow machine, and leave the distribution in case (iii) intact. To verify this, always schedule the job $t+t'$ where we previously assigned t, ignoring the previous decision for t'.

To prove that the makespan cannot decrease after the replacement, we distinguish three cases. (I) The makespan both in (i) and (ii) is determined by the fast machine. Let l be the load of the fast machine without t and t'. The contribution of (i) and (ii) to the expected makespan in the original schedule is $q(l + t) + q(l + t')$. In the new schedule with a job of size $t + t'$ it is at least $q(l + t + t') + ql$. So the makespan does not decrease.

(II) The makespan both in (i) and (ii) is determined by the slow machine. This is identical to (I), except that all the contributions are multiplied by s.

(III) The makespan both in (i) and (ii) is determined by the machine with t. Now, if we always schedule t' on the machine with t, the makespan never decreases.

Thus the worst case is a sequence with at most two jobs. It is easy to calculate that the competitive ratio on such a sequence is c_{BIAS} (the worst case being the sequence 1, s). ∎

5 The algorithm for small s

We present an algorithm with good competitive ratio for $1 \leq s \leq 3/2$. The algorithm uses as a subroutine the randomized 4/3-competitive algorithm of Bartal, Fiat, Karloff and Vohra [3] for scheduling on two identical machines.

The algorithm FAIR is defined as follows: We flip an unbiased coin. If it comes up heads, we schedule all jobs using the Bartal *et al.* algorithm (ignoring machine speeds) with the first job going on the fast machine. If it comes up tails, we schedule all jobs using the Bartal *et al.* algorithm with the first job going on the slow machine.

While the following upper bound holds for any s, for $s > 3/2$ we have better algorithms.

Theorem 5.1 *For any $s \geq 2$,* FAIR *is c_{FAIR}-competitive, where*

$$c_{\text{FAIR}} = \frac{(3s+1)(1+s)}{6s}.$$

Proof First we briefly describe the algorithm of Bartal *et al.* for two identical machines and introduce some notation. Given a schedule, we call the machine in that schedule with greatest height the *tall* machine. We define similarly the *short* machine. At each point the algorithm keeps track of all schedules $\pi_1, \ldots, \pi_i, \ldots$ that it might have chosen so far (after j jobs there are at most 2^j schedules) and of their probability distribution. In schedule π_i, we let x_i and y_i be the heights of the tall and short machines, respectively. We let p_i be the probability of π_i. When a new job arrives, let $x = \sum_i p_i x_i$ and $y = \sum_i p_i y_i$ be the expectations of the height of the tall machine and the short machine, respectively. The algorithm schedules each job maintaining $x \geq 2y$. The intuition behind this invariant is that the algorithm should keep y, the load on the short machine, relatively low in order to be ready for the possibility of a large job arriving. For some probability p, the algorithm schedules the new job with probability p on the tall machine and with probability $1-p$ on the short machine (independently of the previous choices). The algorithm chooses the smallest p such that the invariant is preserved. For $p = 1$ (the new job is always scheduled on the tall machine) the invariant is preserved, hence "the smallest p" is well-defined.

We may assume, without loss of generality, the sum of the sizes of all jobs is 1 (we rescale the jobs after each new one; note that the behavior of the algorithm does not change by scaling). We refer to this as the *normalizing* assumption. Let

$$g(x) = \frac{3x-1}{2}.$$

Seiden [11] presents an alternative proof that the algorithm for $s = 1$ is 4/3 competitive, and shows that under the normalizing assumption the algorithm maintains the following invariants:

1. The optimal offline cost is at least $g(x)$.
2. If $g(x) > 1/2$ (i.e., $x > 2/3$) then a job of size at least $g(x)$ was scheduled.

We now begin the analysis of FAIR, using the facts above. The probability that the machine with load x_i in π_i is the slow one is 1/2. Therefore, the expected makespan in π_i is $(s \cdot x_i + \max\{x_i, s \cdot y_i\})/2$. We partition the schedules into two sets A and B according to the order of the terms in the maximum, and define the expectations x_A, y_A, x_B, y_B of x_i and y_i over these sets:

$$\begin{aligned}
A &= \{i \mid x_i < s \cdot y_i\}, \\
B &= \{i \mid x_i \geq s \cdot y_i\}, \\
P &= \sum_{i \in A} p_i, \\
x_A &= \frac{1}{P} \sum_{i \in A} p_i \cdot x_i, \\
y_A &= \frac{1}{P} \sum_{i \in A} p_i \cdot y_i, \\
x_B &= \frac{1}{1-P} \sum_{i \in B} p_i \cdot x_i, \\
y_B &= \frac{1}{1-P} \sum_{i \in B} p_i \cdot y_i.
\end{aligned}$$

Since we assume that the sum of all job sizes is 1, we have $y_A = 1 - x_A$ and $y_B = 1 - x_B$. The optimal offline makespan x_{opt} is bounded by $x_{\text{opt}} \geq g(x)$ and $x_{\text{opt}} \geq s/(s+1)$. Let X be the makespan of FAIR. We have

$$\begin{aligned}
2X &= \sum_i p_i(s \cdot x_i + \max\{x_i, s \cdot y_i\}) \\
&= \sum_{i \in A} p_i(s \cdot x_i + s \cdot y_i) + \sum_{i \in B} p_i(x_i + s \cdot x_i) \\
&= sP(x_A + y_A) + (s+1)(1-P)x_B \\
&= sP + (s+1)(1-P)x_B.
\end{aligned}$$

We begin by bounding x_B:

Lemma 5.2 *For $P < \frac{2}{3}$ we have*

$$3(1-P)x_B \leq 1 + (2-3P)x_{\text{opt}}.$$

Proof Since $g(x)$ is a lower bound on the makespan in any schedule, we have $x_A \geq g(x)$ and also $x_{\text{opt}} \geq g(s)$. Thus

$$\begin{aligned}
x_A &\geq g(x) \;=\; \tfrac{3}{2}x - \tfrac{1}{2} \\
&= \tfrac{3}{2}Px_A + \tfrac{3}{2}(1-P)x_B - \tfrac{1}{2}.
\end{aligned}$$

This implies that, for $P < \frac{2}{3}$,

$$x_A \geq \frac{3(1-P)x_B - 1}{2-3P}.$$

Together with $x_{\text{opt}} \geq g(x)$ we obtain

$$\begin{aligned}
x_{\text{opt}} &\geq g(x) \;=\; \tfrac{3}{2}Px_A + \tfrac{3}{2}(1-P)x_B - \tfrac{1}{2} \\
&\geq \tfrac{3}{2}P\frac{3(1-P)x_B - 1}{2-3P} + \tfrac{3}{2}(1-P)x_B - \tfrac{1}{2} \\
&= \frac{1}{2}\left(\frac{3P}{2-3P} + 1\right)(3(1-P)x_B - 1) \\
&= \frac{1}{2-3P}(3(1-P)x_B - 1),
\end{aligned}$$

which implies the desired bound on x_B. ∎

When $P < \frac{2}{3}$, we use Lemma 5.2 and $s \leq (s+1)x_{\text{opt}}$ to obtain

$$\begin{aligned}
6sX &= 3s^2P + 3s(s+1)(1-P)x_B \\
&\leq 3s^2P + s(s+1)(1 + (2-3P)x_{\text{opt}}) \\
&\leq (3s(s+1)P + (s+1)^2 + \\
&\qquad 2s(s+1)2 - 3s(s+1)P)x_{\text{opt}} \\
&= (s+1)(3s+1)x_{\text{opt}}.
\end{aligned}$$

For $P \geq \frac{2}{3}$ we use $x_B \leq 1$ and $s \leq (s+1)x_{\text{opt}}$:

$$\begin{aligned}
6sX &\leq 3s^2P + 3s(s+1)(1-P) \\
&= 3s(s+1) - 3sP \;\leq\; 3s(s+1) - 2s \\
&= (3s+1)s \;\leq\; (3s+1)(s+1)x_{\text{opt}}.
\end{aligned}$$

In both cases it follows that $X \leq c_{\text{FAIR}} \cdot x_{\text{opt}}$. ∎

6 Algorithms for medium s

We define a new class of randomized online scheduling algorithms for two uniform machines. Each instance of the class is defined by fixing an integer $\ell \geq 2$ and providing a deterministic online algorithm $\mathcal{A}$ for scheduling on ℓ identical machines. Call the identical machines $I_1, \ldots, I_\ell$. Given ℓ and $\mathcal{A}$ the algorithm PICK ONE behaves as follows:

1. Pick $k \in \{1, \ldots, \ell\}$ randomly and uniformly.
2. For each job J:
   (a) Give J to $\mathcal{A}$.
   (b) If $\mathcal{A}$ puts J on M_k then put J on the slow machine.
   (c) Otherwise put J on the fast machine.

Let L_i denote the load on M_i. Define

$$\mathcal{L} = \sum_{i=1}^{\ell} L_i. \tag{3}$$

Without loss of generality, we assume that

$$L_i \leq L_{i+1} \qquad \text{for } i = 1, \ldots, \ell - 1. \tag{4}$$

So L_ℓ is the makespan of $\mathcal{A}$ on the identical machines. Let T denote the size of the largest job. W.l.o.g., we assume that the optimal offline cost for the two machine problem is 1. We have

$$1 \geq \frac{s}{s+1}\mathcal{L}, \tag{5}$$

$$1 \geq T, \tag{6}$$

For a general analysis of PICK ONE we assume that the makespan of the algorithm $\mathcal{A}$ is upper bounded in terms of T and $\mathcal{L}$:

$$L_\ell \leq f(T, \mathcal{L}) \tag{7}$$

The makespan of Pick One is

$$\begin{aligned}
&\frac{1}{\ell}\sum_{i=1}^{\ell}\max\{\mathcal{L}-L_i, sL_i\} \\
=& \left(\frac{1}{\ell}\sum_{i=1}^{\ell}\max\{\mathcal{L},(s+1)L_i\}\right)-\mathcal{L} \\
=& \frac{i_0-1}{\ell}\mathcal{L}+\frac{s+1}{\ell}\sum_{i=i_0+1}^{\ell}L_i, \qquad (8)
\end{aligned}$$

where $i_0 = \max\{i \mid \mathcal{L} \geq (s+1)L_i\}$. Since $L_{\ell-1}$ is the third largest load, we have $\mathcal{L} \geq 3L_{\ell-1} \geq (s+1)L_{\ell-1}$, and thus $i_0 \in \{\ell-2, \ell-1, \ell\}$.

In order to show that Pick One is c-competitive, we show that the mathematical program: Maximize (8) subject to (3), (4), (5), (6) and (7) has solution at most c for each $i_0 \in \{\ell-2, \ell-1, \ell\}$. We call this mathematical program $\mathcal{P}$.

We are able to bound the value of $\mathcal{P}$ in the cases $i = \ell$ and $i_0 = \ell-2$, without any knowledge of the scheduling algorithm used. I.e. we do not use (6) in bounding the value of $\mathcal{P}$ in these cases.

Lemma 6.1 *For $i_0 = \ell-2$ or $i_0 = \ell$, program $\mathcal{P}$ has solution at most*

$$\frac{(s+1)(\ell+s-2)}{\ell s}. \qquad (9)$$

Proof If $i_0 = \ell - 2$ the makespan is

$$\begin{aligned}
&\frac{\ell-3}{\ell}\mathcal{L}+\frac{s+1}{\ell}(L_{\ell-1}+L_\ell) \\
\leq\ & \frac{\ell-3+s+1}{\ell} \leq \frac{(s+1)(\ell+s-2)}{\ell s}.
\end{aligned}$$

using (5). If $i_0 = \ell$ the makespan is $(\ell-1)\mathcal{L}/\ell$, and the lemma follows again from (5). ∎

To show that Pick One is c competitive, we need therefore only show that the value of $\mathcal{P}$ is at most c for $i_0 = \ell - 1$, and that c at least as large as (9).

It is possible to use List with Pick One, however, this does not yield any improvement over Bias. See Appendix C for the analysis.

We describe a new algorithm for $\ell \geq 3$ machines, which we call Prepared. Prepared is not in general a good algorithm for the ℓ machine problem, in that its competitiveness is worse than that of List. However, it does handle large jobs well. It does this by always keeping several machines lightly loaded, so that it is 'prepared' when a large job arrives. For some $k \leq \ell - 2$, the algorithm has parameters $\alpha_1, \ldots, \alpha_k$ which satisfy

$$\begin{aligned}
\alpha_i \leq \alpha_{i+1} \quad & \text{for } i = 1, \ldots, k-1, \\
0 \leq \alpha_i \leq 1 \quad & \text{for } i = 1, \ldots, k.
\end{aligned} \qquad (10)$$

Prepared is defined as follows:

1. For each job J:
   (a) Let t be the size of job J.
   (b) For $i \leftarrow 1, \ldots, k$:
   If $L_{j+1} \leq \alpha_j(\mathcal{L}+t)$ for all $j = i, \ldots, k$, then put J on I_i.
   (c) Put J on I_{k+1}.

The next lemma gives some intuition about the role of the parameters α_i. For convenience we define $\alpha_i = 1/(\ell - i + 1)$ for $i = k+1, \ldots, \ell$.

Lemma 6.2 *For all $i = 1, \ldots, \ell$,* Prepared *maintains the invariants*

$$L_i \leq \alpha_i \mathcal{L}.$$

Proof The bounds are trivial for $i = k+1, \ldots, \ell$. The proof for $i = 1, \ldots, k$ is by induction. For the empty sequence it is trivial. We let the machine loads be $L_1, \ldots, L_\ell$ before J, and $L'_1, \ldots, L'_\ell$ after J. Let $\mathcal{L} = \sum_i L_i$ and $\mathcal{L}' = \sum_i L'_i$. Assuming $L_i \leq \alpha_i \mathcal{L}$, we show that $L'_i \leq \alpha_i \mathcal{L}'$. Suppose J is scheduled on I_i. By induction, we have $L'_p = L_p \leq \alpha_p \mathcal{L} \leq \alpha_p \mathcal{L}'$ for $p = 1, \ldots, i-1$. Let $j \geq i$ be the new rank of machine i. We have $L'_j = L_i + t$ and $L'_p = L_{p+1}$ for $p = i, \ldots, j-1$. Since J was scheduled on machine i, by definition of Prepared we have that $L_{p+1} \leq \alpha_p(\mathcal{L}+t)$ for $p = i, \ldots, k$. We have two cases. First consider $j \leq k$. We have $L'_p = L_{p+1} \leq \alpha_p(\mathcal{L}+t) = \alpha_p \mathcal{L}'$ for $p = i, \ldots, j-1 < k$. Also, we have $L'_j = L_i + t \leq L_{j+1} \leq \alpha_j(\mathcal{L}+t) = \alpha_j \mathcal{L}'$. Finally, Note that $L'_p = L_p \leq \alpha_p \mathcal{L} \leq \alpha_p \mathcal{L}'$ for $p = j+1, \ldots, k$. Now consider $j > k$. We have $L'_p = L_{p+1} \leq \alpha_p(\mathcal{L}+t) = \alpha_p \mathcal{L}'$ for $p = i, \ldots, k$. ∎

The rest of the analysis of PREPARED is given in Appendix B. A special case of our main theorem concerning PREPARED is:

Lemma 6.3 *For $k = 1$, the makespan of* PREPARED *is at most*

$$f(T, \mathcal{L}) = \max\left\{\frac{s+\alpha_1}{1+s}\mathcal{L}, (1-\alpha_1)T + \alpha_1\mathcal{L}\right\},$$

where $\alpha_1 = 1/(s(\ell-2)+\ell-1)$.

By varying the number of identical machines ℓ used by PREPARED, we are able to get a family of algorithms whose competitive ratio beats FAIR for all $3/2 \leq s < 2$. For instance, for $k = 1$ we are able to prove that PICK ONE with PREPARED is $c_{\text{PICK-PREPARED-1}}$-competitive where

$$c_{\text{PICK-PREPARED-1}} = \min_{\ell \geq 3} \frac{(s+1)(\ell-2+s+\alpha_1)}{\ell s}.$$

The proof, together with better results using larger k, is given in Appendix C.

7 Identical machines

We define a class of randomized algorithms for online scheduling on m identical machines $\mathcal{M} = \{M_1, \ldots, M_m\}$. This class is very similar in spirit to PICK ONE. Each instance of the class is defined by fixing an integer r and providing a deterministic online algorithm $\mathcal{A}$ for scheduling on $\ell = rm$ machines $\mathcal{I} = \{I_1, \ldots, I_\ell\}$. We call these machines the *imaginary* machines because they do not really exist. They are simply aid in visualizing what the algorithm is doing. Given m, ℓ and $\mathcal{A}$ the algorithm, which we call IMAGINARY, behaves as follows:

1. Pick an assignment $\pi : \mathcal{I} \mapsto \mathcal{M}$ of imaginary machines to real machines, such that r imaginary machines are assigned to each real one, randomly and uniformly.

2. For each job J:

   (a) Give J to $\mathcal{A}$.

   (b) If $\mathcal{A}$ puts J on I_j then put J on $\pi(I_j)$.

Again, we use PREPARED as our subroutine algorithm $\mathcal{A}$. For $m = 2$ and $r = 2, 3, 4$, we have algorithms which are distributions over 3, 10 and 35 deterministic strategies, and which are $22/15 < 1.46667$, 1.44489 and 1.44286-competitive, respectively. Further, we show that for $r \geq 5$ the competitive ratio is strictly worse. For $m = 3$ and $r = 2$, we have a distribution over 15 deterministic strategies which is $23/14 < 1.64286$-competitive. See Appendix D.

8 Preemptive scheduling

We give tight bounds for preemptive scheduling for any speed. In this case randomization does not help: our algorithm is deterministic while the lower bound holds also for randomized algorithms. In the preemptive case it is not a priori clear that introducing idle time is not useful: for example we can not preempt the first job without introducing idle time. However, our online algorithm does not need idle time, while the lower bound holds even for algorithms with idle time (some care is needed in the formulation of the lower bound proof to achieve that). The following theorem is proved in Appendix A.

Theorem 8.1 *There exists a deterministic online algorithm for preemptive scheduling on two machines with speeds 1 and s that is c_{pmtn}-competitive, where*

$$c_{\text{pmtn}} = 1 + \frac{s}{s^2+s+1} = \frac{(s+1)^2}{s^2+s+1}.$$

For any s, no randomized online algorithms can achieve a better competitive ratio than c_{pmtn}.

9 Conclusions

In this paper, we have completely settled the preemptive version (deterministic and randomized) for all speeds $s \geq 1$, and the non-preemptive version for all speeds $s \geq 1$ (deterministic) respectively for all speeds $s \geq 2$ (randomized). For the non-preemptive, randomized case with $1 < s < 2$, we have developed a family of good algorithms and a class of almost matching lower bounds.

Settling the non-preemptive randomized case for speeds $1 < s < 2$ seems to be very hard: We definitely do conjecture that the optimal competitive ratio should be at most $3/2$ for any speed in this range; but we do not even have a reasonable guess for what the optimal competitive ratio should be in dependence on s. On the critical sequence with $1 < s < 2$, even algorithm Pick One with List would be 3/2-competitive. However, there exist other sequences on which Pick One with any deterministic scheduling algorithm is worse than 3/2 competitive at speed $s = 3/2$: consider a sequence with three jobs of sizes 4, 5, and 6, in this order, possibly followed by jobs of size $3/2 \cdot 15$ and $(3/2)^2 \cdot 15$. There are still possibilities to overcome these difficulties within the framework of Pick One: We could allow that the subroutine schedules some jobs on none or on more than one of the machines, thus changing the probability for assignments to the slow machine. Alternatively, we could use a randomized algorithm instead of a deterministic one in the subroutine. However, both modifications seem to be very hard to analyze.

Another open question concerns (preemptive and non-preemptive, deterministic and randomized) online scheduling on $m \geq 3$ machines. We conjecture that for any number $m \geq 3$ of machines and for any combination of machine speeds, the optimal competitive ratio for the deterministic preemptive version is strictly better than the optimal competitive ratio for the randomized non-preemptive version.

References

[1] S. Albers. Better bounds for online scheduling. In *Proc. of the 29th STOC*, pages 130–139. ACM, 1997.

[2] Y. Bartal, M. Chrobak, and L. L. Larmore. A randomized algorithm for two servers on the line. Manuscript, 1998.

[3] Y. Bartal, A. Fiat, H. Karloff, and R. Vohra. New algorithms for an ancient scheduling problem. *J. Comput. Syst. Sci.*, 51(3):359–366, 1995.

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

[5] B. Chen, A. van Vliet, and G. J. Woeginger. New lower and upper bounds for on-line scheduling. *Oper. Res. Lett.*, 16:221–230, 1994.

[6] Y. Cho and S. Sahni. Bounds for list schedules on uniform processors. *SIAM J. Comput.*, 9(1):91–103, 1980.

[7] M. Chrobak, L. L. Larmore, C. Lund, and N. Reingold. A better lower bound on the competitive ratio of the randomized 2-server problem. *Inf. Process. Lett.*, 1997.

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

[9] L. A. McGeoch and D. D. Sleator. A strongly competitive randomized paging algorithm. *Algorithmica*, 6:816–825, 1991.

[10] J. Noga and S. Seiden. On the power of barely random online algorithms. Manuscript, 1998.

[11] S. S. Seiden. *Randomization in Online Computation.* PhD thesis, University of California, Irvine, CA, U.S.A., 1997.

[12] S. S. Seiden. A randomized algorithm for that ancient scheduling problem. In *Proc. of the 5th WADS, LNCS 1272*, pages 210–223. Springer-Verlag, 1997.

[13] J. Sgall. Online scheduling. To appear in *Online Algorithms: The State of Art*, eds. A. Fiat and G. Woeginger, LNCS Springer-Verlag.

[14] A. C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. of the 18th FOCS*, pages 222–227. IEEE, 1977.

A Omitted proofs

Proof of Theorem 2.1 (lower bound) For $s \geq \phi$, consider the sequence of two jobs 1 and s. The first job has to be scheduled on the fast machine, since otherwise the competitive ratio would be $s \geq 1 + 1/s$ (using $s \geq \phi$). After both jobs, the optimal makespan is s. Thus the second job has to be scheduled on the fast machine as well, since otherwise the makespan is s^2 and the competitive ratio is again s. But with both jobs on the fast machine the competitive ratio is $(s+1)/s$, and the bound follows.

For $s < \phi$, let $\alpha = (s+1-s^2)/s^2 = 1/s + 1/s^2 - 1$. Note that $0 < \alpha \leq 1$. Our goal is to force the situation when the load on the fast machine is $x > 0$ and the load on the slow machine is αx. If this happens, we continue with a job of size $(1+1/s)x$. Then the makespan of the algorithm is $(2/s + 1/s^2)x$ (no matter on which machine the job is scheduled), while the optimum is $(1+1/s)x$ (scheduling only the last job on the fast machine), and the competitive ratio is $(2s+1)/(s+1)$.

Now start the sequence with a job of size 1. If it is scheduled on the slow machine, continue with a job of size α. If this job is scheduled on the fast machine, we have the desired situation with the ratio of loads equal to $1 : \alpha$. Otherwise both jobs are scheduled on the slow machine, and a calculation yields that the competitive ratio is $1 + 1/s$. If the first job is scheduled on the fast machine, we continue with jobs of sizes α, $\alpha(1+\alpha)$,..., $\alpha(1+\alpha)^i$, ... None of them can be scheduled on the slow machine, as we would get the ratio of loads $1 : \alpha$. Since the sizes of jobs are a geometric sequence with ratio in $(1, 2]$, for a sufficient number of jobs it is possible to partition them into two groups so that the ratio of total sizes of jobs in the groups is arbitrarily close to $s : 1$. This means that the optimal schedule can almost balance the machines and the competitive ratio is arbitrarily close to $1 + 1/s$. ∎

Proof of Lemma 3.3 We use the von Neumann-Yao principle. We give the algorithm the first i jobs of the critical sequence with probability p_i where

$$p_1 = 0, \qquad p_2 = \frac{s^2+s+1}{3s^2+5s+1}, \qquad p_3 = p_4 = \frac{(1+s)^2(2+s)}{3s^2+5s+1}.$$

and $p_i = 0$ for $i \geq 5$. Let $\Delta_j = C_{4,j} - c$, where $c = (1+s)^2(2+s)/(3s^2+5s+1)$ is the desired lower bound and $C_{4,j}$ for $j = 0, \ldots, 15$ is the competitive ratio of algorithm j, as defined by (2). We calculate Δ_i symbolically in terms of s using the above values of p_i. We find:

$$\begin{aligned}
\Delta_0 &= \frac{(s+1)(2s^2+2s-1)}{3s^2+5s+1}, & \Delta_1 &= \frac{(s+1)(s^2+s-1)}{3s^2+5s+1}, \\
\Delta_2 &= \frac{(s-1)(s+1)^2}{3s^2+5s+1}, & \Delta_3 &= 0, \\
\Delta_4 &= \Delta_1, & \Delta_5 &= 0, \\
\Delta_6 &= 0, & \Delta_7 &= \frac{(s+1)(-s^2+2)}{3s^2+5s+1}, \\
\Delta_8 &= \frac{(s+1)(2s^2+s-2)}{3s^2+5s+1}, & \Delta_9 &= \frac{(s-1)(s+1)(s^2+s-1)}{s(3s^2+5s+1)}, \\
\Delta_{10} &= \frac{(s-1)^2(s+1)^2}{s(3s^2+5s+1)}, & \Delta_{11} &= \frac{\Delta_7}{s}, \\
\Delta_{12} &= \frac{(s-1)(s+1)^3}{s(3s^2+5s+1)}, & \Delta_{13} &= 0, \\
\Delta_{14} &= \frac{s+1}{3s^2+5s+1}, & \Delta_{15} &= \frac{(1+s)(-s^3+3s+1)}{s(3s^2+5s+1)}.
\end{aligned}$$

Of the above values, Δ_{15} is positive for $1 \le s < 2\cos(\pi/9) \approx 1.87939$, Δ_7 and Δ_{11} are positive for $1 \le s < \sqrt{2} \approx 1.41421$, and all others are non-negative for all $s \ge 1$. Therefore, no algorithm is better than c-competitive for $1 \le s \le \sqrt{2}$. ■

Proof of Theorem 8.1 First we proof the upper bound. Let x be the total size of all jobs scheduled on the fast machine so far, y the total size of all jobs scheduled on the slow machine, and $t_{\max}$ the size of the largest job so far. The algorithm will maintain the following two invariants:

$$y \le \frac{x}{s^2+s}; \tag{11}$$

$$\text{if } y < \frac{x}{s^2+s}, \text{ then } x \le c_{\text{pmtn}} \cdot t_{\max} \tag{12}$$

Note that the first invariant implies that $x \ge sy$ and thus the makespan of our schedule is always x. If $y = x/(s^2+s)$, then the optimal schedule has makespan at least

$$\frac{1}{1+\frac{1}{s}} \cdot (x+y) = \frac{s}{1+s} \cdot \left(1 + \frac{1}{s^2+s}\right) \cdot x = \frac{1}{c_{\text{pmtn}}} \cdot x,$$

and thus the competitive ratio is at most c_{pmtn}. If $y < x/(s^2+s)$ then $x \le c_{\text{pmtn}} \cdot t_{\max}$ and the algorithm is also c_{pmtn} competitive since $t_{\max}$ is a lower bound on the optimal makespan.

It remains to prove that it is possible to maintain the invariant. It is true at the beginning. When a new job arrives we distinguish three cases. Let t be the size of the new job, and let x' and y' denote the loads after scheduling it.

If it is possible to schedule the new job so that $y' = x'/(s^2+s)$ and $y' \le x/s$, we do so. The first condition guarantees that the invariant is maintained. The second condition guarantees that the portion of the job scheduled on the slow machine runs in parallel with previous jobs on the fast machine, and we do not need to introduce any idle time. This is the case when we have no big job.

If the previous case does not apply and the size of the new job t satisfies $t \ge x/s - y$, we schedule it so that $y' = x/s$. This means that the job is big and we schedule as much of it as possible on the slow machine in parallel with previous jobs on the fast machine. The invariant (11) holds since otherwise we would schedule the job in the previous case (putting less of the job on the slow machine). This implies that $x' \ge y'(s^2+s) = x(s+1)$. Since the invariant (11) was satisfied before this step, we have $y \le x/(s^2+s)$ and we can bound t using the previous inequality:

$$t = (x'-x) + (y'-y) \ge x' - x + \left(\frac{x}{s} - \frac{x}{s^2} + s\right) = x' - \frac{x}{s+1} \ge \left(1 - \frac{1}{(s+1)^2}\right) \cdot x' = \frac{1}{c_{\text{pmtn}}} \cdot x'$$

and the invariant (12) is satisfied.

The remaining possibility is that $t \le x/s - y$ and even after scheduling the new job on the slow machine we have $y' < x'/(s^2+s)$. This happens if a small job arrives after a big job. We schedule the job on the small machine, in parallel with the previous jobs on the fast machine. The condition above guarantees that the invariant (11) is maintained. The load of the fast machine does not change, and thus also the invariant (12) is true (the size of the largest job cannot decrease).

To prove the lower bound, suppose we have a c-competitive randomized algorithm. Consider the first three jobs from the critical sequence, with running times 1, s, and s^2+s. Let T be the expected makespan after scheduling the first two jobs. Since the optimal schedule for the first two jobs has length s, it must be the case that $T \le cs$. After scheduling the last job, the expected time when both machines are busy is bounded by T, and even if the faster machine is used all the rest of the schedule, the expected makespan is at least $s^2+2s+1-T/s \ge s^2+2s+1-c$. Since the optimal makespan is s^2+s we obtain $s^2+2s+1-c \le c(s^2+s)$ which yields $c \ge (s^2+2s+1)/(s^2+s+1) = c_{\text{pmtn}}$. ■

B Analysis of Prepared

Lemma B.1 *If T is the size of the largest job scheduled by* Prepared *then $L_p \leq f_2(T, \mathcal{L})$ where*

$$\begin{aligned} T_i &= 1 - \alpha_{i-1}/\alpha_i, \\ T_1 &= 1, \\ g_i(T, \mathcal{L}) &= T + (\mathcal{L} - T)\alpha_i, \\ f_2(T, \mathcal{L}) &= \max_{i=1,\ldots,k+1} g_i\left(\min\{T, T_i \cdot \mathcal{L}\}, \mathcal{L}\right). \end{aligned}$$

Proof The proof is by induction. Consider the scheduling of a job J. Let T be the size of the largest job before J, and t be the size of J, and $T' = \max\{T, t\}$. We let the machine loads be $L_1, \ldots, L_\ell$ before J, and $L'_1, \ldots, L'_\ell$ after J. Let $\mathcal{L} = \sum_i L_i$ and $\mathcal{L}' = \sum_i L'_i$.

For the basis, if J is the first job, then after it arrives $t = T' = L'_\ell = \mathcal{L}'$ and

$$L'_\ell = g_{k+1}(L'_\ell, L'_\ell) = g_{k+1}(T', \mathcal{L}') = g_{k+1}(\min\{T', T_{k+1}\mathcal{L}'\}, \mathcal{L}') \leq f_2(T', \mathcal{L}').$$

If when J is scheduled the makespan does not increase, we have

$$L'_\ell = L_\ell \leq f_2(T, \mathcal{L}) \leq f_2(\max\{T, t\}, \mathcal{L} + t) = f_2(T', \mathcal{L}'),$$

where the first inequality is by the inductive hypothesis and the second is by the fact that $f_2(T, \mathcal{L})$ is non-decreasing in both T and $\mathcal{L}$. Therefore we may assume from this point on that the makespan increases.

If J is scheduled on I_i and the makespan increases then we have

$$L'_\ell = L_i + t \leq \alpha_i \mathcal{L} + t = t + \alpha_i(\mathcal{L}' - t) = g_i(t, L').$$

First consider $i > 1$. By definition of the algorithm, if J was scheduled on I_i then $L_{j+1} \leq \alpha_j(\mathcal{L}+t)$ for $j = i, \ldots, k$. Also, if J was not scheduled on I_{i-1} then $L_{j+1} \leq \alpha_j(\mathcal{L} + t)$ does not hold for all $j = i-1, \ldots, k$. ¿From these two facts we have $L_i > \alpha_{i-1}(\mathcal{L} + t)$. This implies that

$$t \leq \frac{L_i}{\alpha_{i-1}} - \mathcal{L} \leq \left(\frac{\alpha_i}{\alpha_{i-1}} - 1\right)\mathcal{L}.$$

This holds if and only if

$$\frac{\alpha_i}{\alpha_{i-1}} t \leq \left(\frac{\alpha_i}{\alpha_{i-1}} - 1\right)(\mathcal{L} + t),$$

which implies

$$t \leq \left(1 - \frac{\alpha_{i-1}}{\alpha_i}\right)(\mathcal{L} + t) = T_i \mathcal{L}'.$$

Therefore

$$L'_\ell = g_i(t, L') = g_i(\min\{t, T_i\mathcal{L}'\}, L') \leq g_i(\min\{T', T_i\mathcal{L}'\}, L') \leq f_2(T', L').$$

Now consider $i = 1$. In this case,

$$g_i(t, L') \leq g_{k+1}(T', L') = g_{k+1}(\min\{T', \mathcal{L}'\}, L') = g_{k+1}(\min\{T', T_{k+1}\mathcal{L}'\}, L') \leq f_2(T', L').$$

■

We pick $\alpha_1, \ldots, \alpha_k$ to satisfy

$$
\begin{aligned}
x_1 &= x_2 = \cdots = x_{k+1}, && (13)\\
x_1 &\geq \alpha_1, && (14)\\
x_i &= 1 - \frac{\alpha_{i-1}(1-\alpha_i)}{\alpha_i}, && (15)\\
x_{k+1} &= 1 - (\ell - k - 1)\alpha_k. && (16)
\end{aligned}
$$

Further, we restrict $x_1(\alpha_1) : [0,1] \mapsto [0,1]$ to be a monotone increasing rational function of α_1 with no singularities and with $x_1(0) \neq 1$. Note (16) is simply a consequence of (15) and $\alpha_{k+1} = 1/(\ell - k)$. To find such α_i's we solve the equation

$$
(\ell - k - 1)\alpha_k = 1 - x_1(\alpha_1), \tag{17}
$$

for α_1 where α_k is calculated via the recurrence:

$$
\begin{aligned}
\alpha_2 &= \frac{\alpha_1}{\alpha_1 + 1 - x_1(\alpha_1)},\\
\alpha_i &= \frac{\alpha_{i-1}^2}{\alpha_{i-1}^2 + \alpha_{i-2}(1-\alpha_{i-1})}.
\end{aligned}
$$

This recurrence is easily derived from (13), (14) and (15).

Lemma B.2 *If $\alpha_1, \ldots, \alpha_k$ satisfy (13), (14), (15) and (16) and $1 \geq \alpha_1 \geq 0$ then $1 \geq \alpha_i$ and $\alpha_{i+1} \geq \alpha_i$ for $i = i, \ldots, k$.*

Proof The proof is by induction. For the basis we note that

$$
\alpha_2 - \alpha_1 = \frac{(x_1 - \alpha_1)\alpha_1}{1 + \alpha_1 - x_1} \geq 0.
$$

since $1 \geq x_1 \geq \alpha_1 \geq 0$. Further, α_2 is increasing in α_1 and achieves a maximum of 1 at $\alpha_1 = 1$. For the inductive step we assume that $1 \geq \alpha_{i-1} \geq \alpha_{i-2}$. First we show that $\alpha_i \geq \alpha_{i-1}$. We have

$$
\alpha_i - \alpha_{i-1} = \frac{\alpha_{i-1}^2}{\alpha_{i-1}^2 + \alpha_{i-2}(1-\alpha_{i-1})} - \alpha_{i-1} = \frac{\alpha_{i-1}(1-\alpha_{i-1})(\alpha_{i-1}-\alpha_{i-2})}{\alpha_{i-1}(\alpha_{i-1}-\alpha_{i-2}) + \alpha_{i-2}} \geq 0.
$$

Now consider

$$
\alpha_i = \frac{\alpha_{i-1}^2}{\alpha_{i-1}^2 + \alpha_{i-2}(1-\alpha_{i-1})}.
$$

This is increasing in α_{i-1} and decreasing in α_{i-2}. A maximum of 1 is achieved when $\alpha_{i-1} = 1$ and $\alpha_{i-2} = 0$. ∎

Lemma B.3 *For all $1 \leq k \leq \ell - 2$ there exists an α_1 with $0 \leq \alpha_1 \leq 1$ which satisfies (17).*

Proof Note that the right hand side of (17) is a continuous map from $[0,1]$ to $[0,1]$ as a function of α_1. We have $(\ell - k - 1) \geq 1$. $\alpha_k(\alpha_1)$ is continuous for $\alpha_1 \in (0,1]$, and $\alpha_k(1) = 1$. Our problem is that $\alpha_1 = 0$ is a singularity of $\alpha_k(\alpha_1)$. If we could remove this singularity, then $\alpha_k(\alpha_1)$ would be continuous on $[0,1]$. If further $\lim_{\alpha_1 \to 0} \alpha_k(\alpha_1) = 0$, then by the mean value theorem the curves $(\ell - k - 1)\alpha_k(\alpha_1)$ and $1 - x_1(\alpha_1)$ must intersect, and we are done.

We show that the singularity of $\alpha_k(\alpha_1)$ at $\alpha_1 = 0$ is removable. To accomplish this, we show that $\alpha_k(\alpha_1) = \alpha_1 p_i/q_i$ for α_1 in some neighborhood of 0 and all $k \geq 1$, where p_i and q_i are polynomials in α_1 and m which have no root at $\alpha_1 = 0$. The proof is by induction. For $k = 1, 2$ we have $\alpha_k = \alpha_1$ and $\alpha_k = \alpha_1/(\alpha_1 + 1 - x_1(\alpha_1))$, respectively, and so the required conditions hold. For $k \geq 3$ we have

$$\begin{aligned}
\alpha_k &= \frac{\alpha_{i-1}^2}{\alpha_{i-1}^2 + \alpha_{i-2}(1-\alpha_{i-1})} \\
&= \frac{\alpha_1^2 p_{i-1}^2/q_{i-1}^2}{\alpha_1^2 p_{i-1}^2/q_{i-1}^2 + \alpha_1 p_{i-2}/q_{i-2}(1-\alpha_1 p_{i-1}/q_{i-1})} \\
&= \alpha_1 \frac{p_{i-1}^2 q_{i-2}}{p_{i-2}q_{i-1}^2 + \alpha_1 p_{i-1}(p_{i-1}q_{i-2} - p_{i-2}q_{i-1})},
\end{aligned}$$

for $\alpha_1 \neq 0$, $p_{i-1} \neq 0$, $p_{i-2} \neq 0$, $q_{i-1} \neq 0$ and $q_{i-2} \neq 0$. Note that $p_{i-1} \neq 0$, $p_{i-2} \neq 0$, $q_{i-1} \neq 0$ and $q_{i-2} \neq 0$ at $\alpha_1 = 0$, by the inductive hypothesis. Therefore, the above holds in some neighborhood around $\alpha_1 = 0$. This is of the required form, since $p_{i-1}^2 q_{i-2}$ and $p_{i-2}q_{i-1}^2 + \alpha_1 p_{i-1}(p_{i-1}q_{i-2} - p_{i-2}q_{i-1})$ are both polynomials in α_1 and s and they are not 0 at $\alpha_1 = 0$. This also implies that $\lim_{\alpha_1 \to 0} \alpha_k(\alpha_1) = 0$. ■

Lemma B.4 *Under conditions (13), (14), (15) and (16) we have*

$$f_2(T, \mathcal{L}) \leq \max\{x_1\mathcal{L}, (1-\alpha_1)T + \alpha_1\mathcal{L}\}.$$

Proof The first term in the above maximum is $g_1(T, \mathcal{L})$. We note that

$$g_i(\min\{T, T_i\mathcal{L}\}, \mathcal{L}) \leq g_i(T_i\mathcal{L}, \mathcal{L}) = T_i\mathcal{L} + (\mathcal{L} - T_i\mathcal{L})\alpha_i = (1-\alpha_i)T_i\mathcal{L} + \alpha_i\mathcal{L}.$$

For $i = 2, \ldots, k+1$ this is

$$(1-\alpha_i)\left(1 - \frac{\alpha_{i-1}}{\alpha_i}\right)\mathcal{L} + \alpha_i\mathcal{L} = \mathcal{L} - \frac{\alpha_{i-1}(1-\alpha_i)}{\alpha_i}\mathcal{L} = x_i\mathcal{L} = x_1\mathcal{L}.$$

■

We now see the intuition behind our choice of picking $\alpha_1, \ldots, \alpha_k$ to satisfy $x_1 = x_2 = \cdots = x_{k+1}$. For $i = 2 \ldots k+1$, $g_i(T_i\mathcal{L}, \mathcal{L}) = x_1\mathcal{L}$. As we shall see, the value of x_1 is chosen so that $g_1(T, \mathcal{L}) = x_1\mathcal{L}$ for the worst case value of T.

C Algorithms for medium s

We first analyze the combination of Pick One and List, using the general analysis developed in Section 6. From [8], one can easily derive that the size of the largest job scheduled by List is at least

$$L_\ell \leq f_1(T, \mathcal{L}) = \frac{\ell - 1}{\ell}T + \frac{1}{\ell}\mathcal{L}.$$

Lemma C.1 *For $f = f_1$ and $i_0 = \ell - 1$, program $\mathcal{P}$ has solution at most*

$$\frac{(s+1)(\ell^2 + \ell s - 2\ell + 1)}{\ell^2 s}. \tag{18}$$

Proof In this case the makespan is

$$\begin{aligned}
\frac{\ell-2}{\ell}\mathcal{L}+\frac{s+1}{\ell}L_\ell &\le \frac{\ell-2}{\ell}\mathcal{L}+\frac{s+1}{\ell}f_1(T,\mathcal{L})\\
&= \frac{\ell-2}{\ell}\mathcal{L}+\frac{s+1}{\ell}\left(\frac{\ell-1}{\ell}T+\frac{1}{\ell}\mathcal{L}\right)\\
&= \frac{\ell^2-2\ell+s+1}{\ell^2}\mathcal{L}+\frac{(\ell-1)(s+1)}{\ell^2}T\\
&\le \frac{(s+1)(\ell^2-2\ell+s+1)}{\ell^2 s}+\frac{(\ell-1)(s+1)}{\ell^2}\\
&= \frac{(s+1)(\ell^2+\ell s-2\ell+1)}{\ell^2 s}.
\end{aligned}$$

The first inequality follows from (5) while the second follows from (6) and (7). ■

Theorem C.2 Pick One *is* $c_{\text{Pick-List}}$*-competitive for* $1 \le s \le 2$ *when* List *is used, where*

$$\begin{aligned}
c(\ell) &= \frac{(s+1)(\ell^2+\ell s-2\ell+1)}{\ell^2 s},\\
c_{\text{Pick-List}} &= \min_{\ell\ge 2} c(\ell).
\end{aligned}$$

Proof This follows directly from Lemmas 6.1 and C.1, and the fact that $c(\ell)$ is greater than (9) for all $\ell \ge 2$ and $1 \le s \le 2$. ■

Note that the theorem implies that Pick One with List is c_{Bias}-competitive for $s = 2 - 2/\ell$ for all integers $\ell \ge 2$. For these values of s it is not better, as shown by an example of a number of small jobs of total size 1 followed by a job of size s. We display $c_{\text{Pick-List}}$ in Figure 2.

We now consider Pick One with Prepared. Again, we apply the our general analysis from Section 6. We choose

$$x_1 = \frac{s+\alpha_1}{1+s}$$

We bound the value of $\mathcal{P}$ for $i_0 = \ell - 1$.

Lemma C.3 *For* $f(T,\mathcal{L}) = f_2(T,\mathcal{L})$*, program* $\mathcal{P}$ *has solution at most*

$$\frac{(s+1)(\ell-2+s+\alpha_1)}{\ell s}. \tag{19}$$

Proof If $i_0 = \ell - 1$ the makespan is at most

$$\begin{aligned}
\frac{\ell-2}{\ell}\mathcal{L}+\frac{s+1}{\ell}f_2(T,\mathcal{L}) &\le \frac{\ell-2}{\ell}\mathcal{L}+\frac{s+1}{\ell}\max\left\{\frac{s+\alpha_1}{1+s}\mathcal{L},(1-\alpha_1)T+\alpha_1\mathcal{L}\right\}\\
&\le \frac{(s+1)(\ell-2)}{\ell s}+\frac{s+1}{\ell}\left(1+\frac{\alpha_1}{s}\right)\\
&= \frac{(s+1)(\ell-2+s+\alpha_1)}{\ell s}.
\end{aligned}$$

The second inequality follows from (5) and (6). Note that this is greater than (9) for $\alpha_1 > 0$. Therefore, by Lemma 6.1 we are done. ■

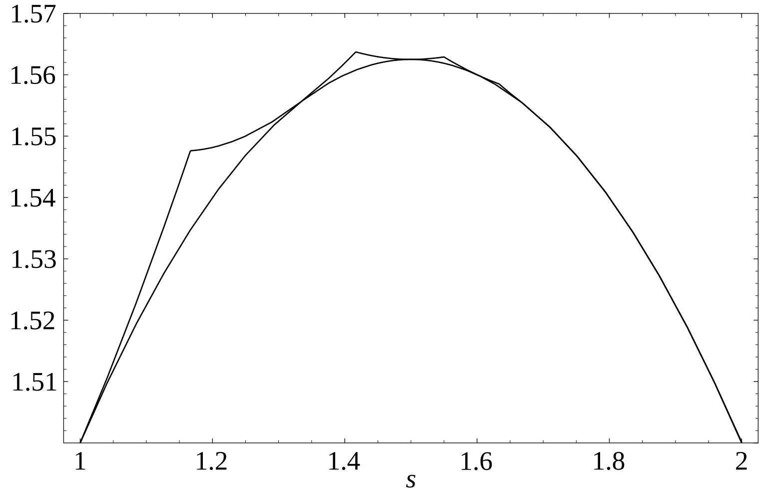

Figure 2: The functions $c_{\textsc{Bias}}$ and $c_{\textsc{Pick-List}}$ for $1 \leq s \leq 2$.

Let

$$c_{\textsc{Pick-Prepared-}k} = \min_{\ell \geq k+2} \frac{(s+1)(\ell - 2 + s + \alpha_1)}{\ell s}.$$

Note that $c_k(\ell)$ is greater than (9) since $\alpha_1 > 0$. For $k = 1$ we define

$$\alpha_1 = \frac{1}{s(\ell-2) + \ell - 1}.$$

For $k = 2$ we define

$$\begin{aligned} A &= (3s+1)(s+2) - (s+1)^2 \ell, \\ \alpha_1 &= \frac{A + \sqrt{4s + A^2}}{2s}. \end{aligned}$$

For $k \geq 3$, we are unable to give a compact closed form expression for α_1.

Theorem C.4 *For all $k \geq 1$,* Pick One *is $c_{\textsc{Pick-Prepared-}k}$-competitive for $1 \leq s \leq 2$ when* Prepared *is used.*

Proof The theorem follows directly from Lemma C.3, and the fact that the algorithm can pick whichever ℓ is best for a particular s. ■

We display $c_{\textsc{Pick-Prepared-}k}$ for $k = 1, \ldots, 4$ in Figure 3.

D Analysis of Imaginary

We formulate a general analysis for Imaginary. Let L_i be the load on I_i. Define

$$\mathcal{L} = \sum_{i=1}^{\ell} L_i. \tag{20}$$

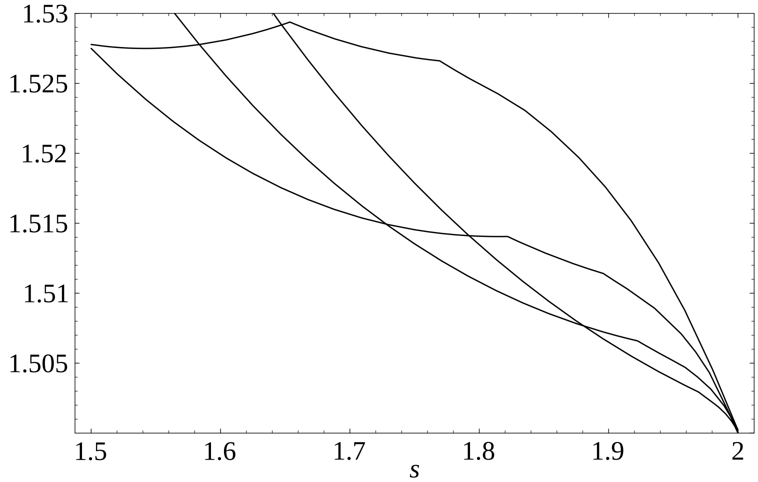

Figure 3: The curves $c_{\textsc{Pick-Prepared-}k}$ with $k = 1, \ldots, 4$ for $\frac{3}{2} \leq s \leq 2$. (The curves are in decreasing order in the neighborhood of 2.)

Without loss of generality, we can assume that

$$L_i \leq L_{i+1} \qquad \text{for } i = 1, \ldots, \ell - 1. \tag{21}$$

So L_ℓ is the makespan of $\mathcal{A}$ on the ℓ machine problem. Without loss of generality, define 1 to be the optimal offline cost for the m machine problem. For algorithm $\mathcal{A}$, let f be a function which upper bounds the makespan as a function of the size of the largest job T and the total size of all jobs $\mathcal{L}$. For any sequence of jobs scheduled by $\mathcal{A}$ we have

$$1 \geq \frac{1}{m}\mathcal{L}, \tag{22}$$

$$1 \geq T, \tag{23}$$

$$L_\ell \leq f(T, \mathcal{L}). \tag{24}$$

For $m > 2$, we make the assumption that $\mathcal{A}$ schedules each job on one of the $\ell - m - 1$ machines which are least loaded. Under this assumption, we have $m+1$ jobs with sizes at least $L_\ell - L_{\ell-m-1}, L_{\ell-1} - L_{\ell-m-1}, \ldots, L_{\ell-m} - L_{\ell-m-1}$, respectively. Two of these must go on the same machine, and certainly the adversary can do no better than having the two smallest on the same machine. We therefore get the bound:

$$1 \geq L_{\ell-m+1} + L_{\ell-m} - 2L_{\ell-m-1}. \tag{25}$$

Let Π be the set of all possible assignments $\pi : \mathcal{I} \mapsto \mathcal{M}$, such that r imaginary machines are assigned to each real one. Define $\pi^{-1}(M_i)$ to be the set of imaginary machines which are assigned to M_i in π. The makespan of Imaginary is

$$\frac{1}{|\Pi|} \sum_{\pi \in \Pi} \max_{1 \leq i \leq m} \left\{ \sum_{I_j \in \pi^{-1}(M_i)} L_j \right\} \tag{26}$$

In order to show that Imaginary is c-competitive, we show that the mathematical program: Maximize (26) subject to (20), (21), (22), (23), (24) and (for $m > 2$) (25) has solution at most c.

We are able to show the following lower bound for Imaginary with $m = 2$, which holds regardless of the scheduling algorithm used:

Lemma D.1 *For two machines, the competitive ratio of* IMAGINARY *is at least*

$$\frac{3r-2}{2r-1}.$$

Proof Consider the performance of IMAGINARY on ℓ jobs of size $1/\ell$ followed by a single job of size 1. In this case, the makespan is always achieved by the machine that is assigned I_ℓ. The expected makespan is

$$L_\ell + \sum_{i=2}^{\ell} \frac{r-1}{2r-1} L_i = \frac{r}{2r-1} L_\ell + \frac{r-1}{2r-1}\mathcal{L}.$$

Since $L_\ell \geq 1$, no matter how the jobs are scheduled, this is at least

$$\frac{r}{2r-1} + \frac{r-1}{2r-1} 2 = \frac{3r-2}{2r-1}.$$

■

So the competitive ratio of IMAGINARY approaches $\frac{3}{2}$ as r grows. The deterministic LIST algorithm achieves this ratio. Therefore, if we wish to beat the deterministic bounds, we must choose r small.

We use PREPARED as our subroutine algorithm $\mathcal{A}$ and fix

$$x_1 = \frac{1+(m-1)\alpha_1}{m}$$

D.1 Two identical machines

We start with the simplest non-trivial example, two real machines and four imaginary ones. We show that IMAGINARY combined with PREPARED is $\frac{22}{15} < 1.46667$-competitive with $k = 1$ and $\alpha_1 = \frac{1}{5}$. In this case, IMAGINARY is a uniform distribution over only three deterministic strategies. Note that no uniform distribution over two deterministic strategies is better than $\frac{3}{2}$-competitive, and that this bound is achieved by the deterministic LIST algorithm.

When $m = 2$ and $l = 4$ the makespan is

$$\begin{aligned}
&\frac{1}{|\Pi|} \sum_{\pi \in \Pi} \max \left\{ \sum_{I_j \in \pi^{-1}(M_1)} L_j, \sum_{I_j \in \pi^{-1}(M_2)} L_j \right\} \\
= \;& \tfrac{1}{3} \max\{L_4 + L_3, L_2 + L_1\} + \tfrac{1}{3} \max\{L_4 + L_2, L_3 + L_1\} + \tfrac{1}{3} \max\{L_4 + L_1, L_3 + L_2\} \\
= \;& \tfrac{1}{3}(L_4 + L_3) + \tfrac{1}{3}(L_4 + L_2) + \tfrac{1}{3} \max\{L_4 + L_1, L_3 + L_2\} \\
= \;& \tfrac{2}{3} L_4 + \tfrac{1}{3} L_3 + \tfrac{1}{3} L_2 + \tfrac{1}{3} \max\{L_4 + L_1, L_3 + L_2\}.
\end{aligned}$$

If $L_4 + L_1 \geq L_3 + L_2$ this is

$$\begin{aligned}
L_4 + \tfrac{1}{3} L_3 + \tfrac{1}{3} L_2 + \tfrac{1}{3} L_1 &= \tfrac{2}{3} L_4 + \tfrac{1}{3}\mathcal{L} \\
&\leq \tfrac{2}{3} f(T, \mathcal{L}) + \tfrac{1}{3}\mathcal{L} \\
&= \tfrac{2}{3} \max\left\{\tfrac{3}{5}\mathcal{L}, \tfrac{4}{5} T + \tfrac{1}{5}\mathcal{L}\right\} + \tfrac{1}{3}\mathcal{L}.
\end{aligned}$$

If $\frac{3}{5}\mathcal{L} \geq \frac{4}{5} T + \frac{1}{5}\mathcal{L}$ then this is $\frac{11}{15}\mathcal{L} \leq \frac{22}{15}$ using (22). Otherwise, it is $\frac{8}{15} T + \frac{7}{15}\mathcal{L} \leq \frac{8}{15} + \frac{7}{15}\mathcal{L} \leq \frac{22}{15}$, using (23) and then (22). If, on the other hand, $L_4 + L_1 < L_3 + L_2$ the makespan is

$$\tfrac{2}{3}(L_4 + L_3 + L_2) \leq \tfrac{2}{3}\mathcal{L} \leq \tfrac{4}{3},$$

again using (22).

We now consider $m = 2$ and $l = 6$. In this case IMAGINARY is a distribution over 10 deterministic strategies. Fixing $k = 3$, we find that $\alpha_1 \approx 0.0748032$ is the unique root in the interval $(0, 1]$ of $\alpha_1^3 - 5\alpha_1^2 - 13\alpha_1 + 1$. We approximate this value, setting $\alpha_1 = 19/254$. We show that IMAGINARY is $602942/417295 < 1.44489$-competitive.

The makespan is

$$\begin{aligned}
&\frac{1}{|\Pi|}\sum_{\pi\in\Pi}\max\left\{\sum_{I_j\in\pi^{-1}(M_1)} L_j, \sum_{I_j\in\pi^{-1}(M_2)} L_j\right\} \\
&= \tfrac{1}{10}\max\{L_6+L_5+L_4, L_3+L_2+L_1\} + \tfrac{1}{10}\max\{L_6+L_5+L_3, L_4+L_2+L_1\} + \\
&\quad \tfrac{1}{10}\max\{L_6+L_5+L_2, L_4+L_3+L_1\} + \tfrac{1}{10}\max\{L_6+L_5+L_1, L_4+L_3+L_2\} + \\
&\quad \tfrac{1}{10}\max\{L_6+L_4+L_3, L_5+L_2+L_1\} + \tfrac{1}{10}\max\{L_6+L_4+L_2, L_5+L_3+L_1\} + \\
&\quad \tfrac{1}{10}\max\{L_6+L_4+L_1, L_5+L_3+L_2\} + \tfrac{1}{10}\max\{L_6+L_3+L_2, L_5+L_4+L_1\} + \\
&\quad \tfrac{1}{10}\max\{L_6+L_3+L_1, L_5+L_4+L_2\} + \tfrac{1}{10}\max\{L_6+L_2+L_1, L_5+L_4+L_3\} \\
&= \tfrac{1}{2}L_6 + \tfrac{3}{10}L_5 + \tfrac{3}{10}L_4 + \tfrac{1}{5}L_3 + \tfrac{1}{5}L_2 + \tfrac{1}{10}\sum_{i=1}^{5} z_i,
\end{aligned}$$

where $z_i = \max\{z_{i,1}, z_{i,2}\}$ and

$$\begin{aligned}
z_{1,1} &= L_6+L_5+L_1, & z_{1,2} &= L_4+L_3+L_2,\\
z_{2,1} &= L_6+L_4+L_1, & z_{2,2} &= L_5+L_3+L_2,\\
z_{3,1} &= L_6+L_3+L_2, & z_{3,2} &= L_5+L_4+L_1,\\
z_{4,1} &= L_6+L_3+L_1, & z_{4,2} &= L_5+L_4+L_2,\\
z_{5,1} &= L_6+L_2+L_1, & z_{5,2} &= L_5+L_4+L_3.
\end{aligned}$$

We further define

$$z_{0,1} = \frac{1+\alpha_1}{2}\mathcal{L}, \qquad z_{0,2} = (1-\alpha_1)T + \alpha_1\mathcal{L}.$$

Now we rewrite (24) as

$$L_\ell \le z_0. \tag{27}$$

For $i = 0, \dots, 5$, we have either $z_i = z_{i,1} \ge z_{i,2}$ or $z_i = z_{i,2} \ge z_{i,1}$. For each of the $2^6 = 64$ possibilities, maximize (26) subject to (20), (21), (22), (23) and (27) is a linear program. Using `Mathematica` we have shown that each of these has solution at most $602942/417295$.

Finally, we consider $m = 2$ and $\ell = 8$. In this case, IMAGINARY is a distribution over 35 deterministic strategies, and it is $101/70 < 1.44286$-competitive. α_1 should be the real root of $\alpha_1^5 - 7\alpha_1^4 + 22\alpha_1^3 - 42\alpha_1^2 - 39\alpha_1 + 1$ in $(0, 1]$. We approximate this, choosing $\alpha_1 = 1/40$. Lemma D.1 implies the competitive ratio is at least $\frac{13}{9} > 1.44444$ for $r \ge 5$. Therefore, $\ell = 8$ is the best possible choice for two machines.

The makespan is

$$\begin{aligned}
&\frac{1}{|\Pi|}\sum_{\pi\in\Pi}\max\left\{\sum_{I_j\in\pi^{-1}(M_1)} L_j, \sum_{I_j\in\pi^{-1}(M_2)} L_j\right\} \\
&= \tfrac{14}{35}L_8 + \tfrac{9}{35}L_7 + \tfrac{9}{35}L_6 + \tfrac{7}{35}L_5 + \tfrac{7}{35}L_4 + \tfrac{5}{35}L_3 + \tfrac{5}{35}L_2 + \tfrac{1}{35}\sum_{i=1}^{5} z_i,
\end{aligned}$$

where $z_i = \max\{z_{i,1}, z_{i,2}\}$ and

$$\begin{aligned}
z_{1,1} &= L_8+L_7+L_6+L_1, & z_{1,2} &= L_5+L_4+L_3+L_2,\\
z_{2,1} &= L_8+L_7+L_5+L_1, & z_{2,2} &= L_6+L_4+L_3+L_2,\\
z_{3,1} &= L_8+L_7+L_4+L_1, & z_{3,2} &= L_6+L_5+L_3+L_2,\\
z_{4,1} &= L_8+L_7+L_3+L_2, & z_{4,2} &= L_6+L_5+L_4+L_1,\\
z_{5,1} &= L_8+L_7+L_3+L_1, & z_{5,2} &= L_6+L_5+L_4+L_2,\\
z_{6,1} &= L_8+L_7+L_2+L_1, & z_{6,2} &= L_6+L_5+L_4+L_3,\\
z_{7,1} &= L_8+L_6+L_5+L_1, & z_{7,2} &= L_7+L_4+L_3+L_2,\\
z_{8,1} &= L_8+L_6+L_4+L_1, & z_{8,2} &= L_7+L_5+L_3+L_2,\\
z_{9,1} &= L_8+L_6+L_3+L_2, & z_{9,2} &= L_7+L_5+L_4+L_1,\\
z_{10,1} &= L_8+L_6+L_3+L_1, & z_{10,2} &= L_7+L_5+L_4+L_2,\\
z_{11,1} &= L_8+L_6+L_2+L_1, & z_{11,2} &= L_7+L_5+L_4+L_3,\\
z_{12,1} &= L_8+L_5+L_4+L_3, & z_{12,2} &= L_7+L_6+L_2+L_1,\\
z_{13,1} &= L_8+L_5+L_4+L_2, & z_{13,2} &= L_7+L_6+L_3+L_1,\\
z_{14,1} &= L_8+L_5+L_4+L_1, & z_{14,2} &= L_7+L_6+L_3+L_2,\\
z_{15,1} &= L_8+L_5+L_3+L_2, & z_{15,2} &= L_7+L_6+L_4+L_1,\\
z_{16,1} &= L_8+L_5+L_3+L_1, & z_{16,2} &= L_7+L_6+L_4+L_2,\\
z_{17,1} &= L_8+L_5+L_2+L_1, & z_{17,2} &= L_7+L_6+L_4+L_3,\\
z_{18,1} &= L_8+L_4+L_3+L_2, & z_{18,2} &= L_7+L_6+L_5+L_1,\\
z_{19,1} &= L_8+L_4+L_3+L_1, & z_{19,2} &= L_7+L_6+L_5+L_2,\\
z_{20,1} &= L_8+L_4+L_2+L_1, & z_{20,2} &= L_7+L_6+L_5+L_3,\\
z_{21,1} &= L_8+L_3+L_2+L_1, & z_{21,2} &= L_7+L_6+L_5+L_4.
\end{aligned}$$

and z_0 is defined as in the case of $\ell = 6$. If we do things as in the previous analysis, there would be $2^{22} = 4194304$ cases. Even using a computer, this is a prohibitively large number of cases. We therefore need need to restrict the number of cases further. We accomplish this as follows: Note that

$$z_{1,1} \geq z_{2,1} \geq z_{3,1} \geq z_{5,1} \geq z_{6,1}.$$

Further, we have

$$z_{1,2} \leq z_{2,2} \leq z_{3,2} \leq z_{5,2} \leq z_{6,2}.$$

We therefore have six possibilities:

- $z_i = z_{i,2}$ for each $i \in \{1,2,3,5,6\}$.
- $z_1 = z_{i,1}$ and $z_i = z_{i,2}$ for each $i \in \{2,3,5,6\}$.
- $z_i = z_{i,1}$ for each $i \in \{1,2\}$ and $z_i = z_{i,2}$ for each $i \in \{3,5,6\}$.
- $z_i = z_{i,1}$ for each $i \in \{1,2,3\}$ and $z_i = z_{i,2}$ for each $i \in \{5,6\}$.
- $z_i = z_{i,1}$ for each $i \in \{1,2,3,5\}$ and $z_6 = z_{6,2}$.
- $z_i = z_{i,1}$ for each $i \in \{1,2,3,5,6\}$.

Using the same type of argument, there are five ways to assign the variables in each of the sets $\{z_7, z_8, z_{10}, z_{11}\}$ and $\{z_{18}, z_{19}, z_{20}, z_{21}\}$, and six ways to assign the variables in $\{z_{12}, z_{13}, z_{14}, z_{16}, z_{17}\}$. We further observe that $z_{4,1} \geq z_{5,1}$ and $z_{4,2} \leq z_{5,2}$. This implies that if $z_5 = z_{5,1}$ then $z_4 = z_{4,1}$. Similarly, if $z_{10} = z_{10,1}$ then $z_9 = z_{9,1}$ and if $z_{16} = z_{16,1}$ then $z_{15} = z_{15,1}$. These observations reduce the number of possible cases to $10 \times 8 \times 10 \times 5 \times 2 = 8000$. We have verified that each of the linear programs has solution at most $101/70$ using `Mathematica`.

D.2 Three identical machines

We now consider three machines and $l = 6$. Imaginary combined with Prepared is a distribution over 15 deterministic strategies. We set $k = 1$ and $\alpha_1 = \frac{1}{7}$ and show that Imaginary is $\frac{23}{14} < 1.64286$-competitive.

The analysis is much the same as that in the previous section. The makespan is

$$\frac{1}{|\Pi|} \sum_{\pi \in \Pi} \max_{1 \leq i \leq 3} \left\{ \sum_{I_j \in \pi^{-1}(M_i)} L_j \right\} = \tfrac{8}{15} L_6 + \tfrac{1}{5} L_5 + \tfrac{1}{5} L_4 + \tfrac{2}{15} L_3 + \tfrac{1}{15} \sum_{i=1}^{7} z_i,$$

where $z_i = \max_j \{z_{i,j}\}$ and

$$\begin{array}{rclcrcl}
z_{1,1} & = & L_6 + L_3, & & z_{1,2} & = & L_5 + L_4, \\
z_{2,1} & = & L_6 + L_2, & & z_{2,2} & = & L_5 + L_4, \\
z_{3,1} & = & L_6 + L_2, & & z_{3,2} & = & L_5 + L_3, \\
z_{4,1} & = & L_6 + L_2, & & z_{4,2} & = & L_4 + L_3, \\
z_{5,1} & = & L_6 + L_1, & & z_{5,2} & = & L_5 + L_4, \\
z_{6,1} & = & L_6 + L_1, & & z_{6,2} & = & L_5 + L_3,
\end{array}$$

$$z_{7,1} = L_6 + L_1, \qquad z_{7,2} = L_5 + L_2, \qquad z_{7,3} = L_4 + L_3,$$

$$z_{0,1} = \frac{1 + 2\alpha_1}{3} \mathcal{L}, \qquad z_{0,2} = (1 - \alpha_1) T + \alpha_1 \mathcal{L}.$$

In this case there are $3 \times 2^8 = 768$ linear programs with objective function (26) and constraints (20), (21), (22), (23), (25) and (27). Each has value at most $\frac{23}{14}$.

E Lower bounds for $s < 2$

In the following table we display the actual lower bounds calculated from the linear program described in Section 3. The value $c_{\text{low}}(i)$ is a lower bound for any randomized online algorithm at speed $s = 1 + i/100$. The bounds are valid for all decimal places shown, i.e. the last digit is always rounded down.

i	$c_{\text{low}}(i)$	$c_{\text{low}}(i+1)$	$c_{\text{low}}(i+2)$	$c_{\text{low}}(i+3)$
1	1.33333	1.33558	1.33787	1.34022
5	1.34261	1.34505	1.34754	1.35006
9	1.35263	1.35523	1.3579	1.36059
13	1.36331	1.36608	1.36888	1.37171
17	1.37457	1.37747	1.38039	1.38335
21	1.38632	1.38932	1.39233	1.39535
25	1.39837	1.40122	1.40407	1.40679
29	1.40948	1.4121	1.41433	1.41648
33	1.41853	1.42038	1.42226	1.42411
37	1.42594	1.42772	1.42948	1.43107
41	1.43255	1.43381	1.43524	1.43654
45	1.43783	1.43912	1.44042	1.44172
49	1.44305	1.44439	1.44569	1.44705
53	1.44844	1.44984	1.45111	1.45268
57	1.45416	1.45563	1.45711	1.45865
61	1.46018	1.46172	1.46342	1.46501
65	1.46657	1.46812	1.46964	1.47113
69	1.47258	1.47401	1.47539	1.4767
73	1.47799	1.47916	1.4802	1.48125
77	1.48221	1.48311	1.48398	1.48474
81	1.48549	1.48618	1.48677	1.48737
85	1.48793	1.48839	1.48885	1.48921
89	1.48947	1.4897	1.48986	1.48993
93	1.48989	1.4897	1.48943	1.48904
97	1.48853	1.48798	1.48731	1.48652