Online and offline preemptive two-machine job shop scheduling

Tracy Kimbrel
IBM T.J. Watson Research Center
PO Box 704
Yorktown Heights, NY 10598
USA

914-784-7513 (voice) 914-784-6040 (fax)
kimbrel@watson.ibm.com

Jared Saia
Department of Computer Science and Engineering
University of Washington
Seattle, Washington

Research Division
Almaden · Austin · China · Haifa · T.J. Watson · Tokyo · Zurich

Online and offline preemptive two-machine job shop scheduling

Tracy Kimbrel Jared Saia

Abstract

We consider online and offline algorithms for special cases of preemptive job shop scheduling to minimize makespan. These special cases are of interest because they commonly arise in the scheduling of computer systems. We give a randomized online algorithm for the two-machine preemptive job shop that is 1.5-competitive against oblivious adversaries. The algorithm assigns priority for one machine according to an arbitrary permutation of the jobs, and in the opposite order for the other machine. A single random bit is used to select which machine is assigned which of the two orders. Thus, our algorithm is easily derandomized in the offline case to yield a simple linear-time 1.5-approximation algorithm. Simple arguments yield lower bounds showing that the randomized online bound of 1.5 and the trivial deterministic online upper bound of 2 are asymptotically tight.

keywords: online algorithms, job shop scheduling, preemption, randomized algorithms

1 Introduction and related work

Job shop scheduling is one of the most notoriously difficult combinatorial optimization problems [1]. In this paper, we consider some of the more tractable special cases of the problem. Our interest in these variants stems from the fact that they arise naturally in the scheduling of computer systems. We shall be concerned with the case of two machines in which preemption is allowed. The two machines model the CPU processing power and the I/O processing power of a computer system, respectively.

We will consider both the cases of an arbitrary number of jobs and of a constant number of jobs. The latter case, arises in coscheduling [2] of multiprocessors for tightly synchronized parallel applications. Under coscheduling, all processors of (a partition of) a multiprocessor are scheduled as a unit, i.e., allocated to the same job at the same time. Performance improves if the I/O subsystem is explicitly scheduled as well [3]. Many I/O-intensive parallel applications alternate between CPU-intensive and I/O-intensive phases [4]. Moreover, the large memory requirements of many parallel applications limit the number of jobs that can run concurrently [5].

Most previous work on job shop scheduling concerns the nonpreemptive case [1]. A recent series of results [6, 7, 8] applies to the preemptive case (equivalently, the case in which all tasks are unit length, and a job is allowed to request the same machine twice without intervening requests to another machine). These results apply to the more general case of more than two machines, and give approximation bounds polylogarithmic in the number of machines and the maximum number of tasks in a job. Shmoys, Stein, and Wein [7] give a $2 + \epsilon$-approximation algorithm for the case in which the number of machines and the maximum number of tasks in a job are constants. Very recently, Jansen, Solis-Oba, and Sviridenko [9] gave a PTAS for the special case of a constant numbers of machines and a constant number of tasks per job. Sevastianov and Woeginger [10] give an offline linear-time 1.5-approximation for the two machine case (without restriction on the number of tasks in a job). The derandomization of our online algorithm achieves the same performance and is much simpler.

It is interesting that our randomized online algorithm requires only a single random bit and is asymptotically optimal as the number of jobs increases, whereas any deterministic online algorithm is able to achieve no better asymptotic competitive ratio than the most naive algorithm. Ours is not the first randomized online algorithm to achieve good performance while requiring only a constant number of random bits. The COMB algorithm of Albers *et al.* [11], currently the best known randomized algorithm for the list update problem, requires a number of random bits that depends only on the size of the list and not on the length of the request sequence. The algorithm of Kalyanasundaram and Pruhs [12], for maximizing the number of jobs on a single machine that complete before their deadlines, uses only a single random bit; their algorithm achieves a constant competitive ratio. Like our algorithm, a recent algorithm for online interval scheduling described by Ben-David *et al.* [13] requires only a single random bit to be strongly competitive.

1.1 Results and organization of the paper

The remainder of this paper is organized as follows.

In Section 2, we define our problems formally.

In Section 3, we give our randomized algorithm RPRI against oblivious adversaries and an asymptotically matching lower bound. The algorithm assigns priority for one machine according to an arbitrary permutation of the jobs (e.g., the input order), and in the opposite order for the other machine. One random bit is used to select which machine is assigned which of the two orders. We prove the following: *RPRI is $\frac{3}{2}$-competitive against oblivious adversaries. No randomized online algorithm for the 2-machine, n-job preemptive job shop problem $J2|pmtn|C_{max}$ achieves a competitive ratio less than $\frac{3}{2} - \frac{1}{2n}$ against oblivious adversaries.* For the case of two jobs, we give a somewhat tighter bound: *No randomized online algorithm for the 2-machine, 2-job preemptive job shop problem $J2|pmtn; n = 2|C_{max}$ achieves a competitive ratio of r for $r < \frac{4}{3}$ against oblivious adversaries.*

In Section 4, we observe offline bounds implied by the randomized online algorithm. The algorithm finds a schedule of length at most $\frac{3}{2}\max(P_{max}, \Pi_{max})$, where P_{max} and Π_{max} denote the greatest job length and the maximum machine load, respectively, and are trivial lower bounds on the optimal schedule length. Thus we have: *For every instance of $J2|pmtn|C_{max}$, there is a schedule of length $\frac{3}{2}\max(P_{max}, \Pi_{max})$. Moreover, such a schedule can be found in linear time. Thus, there is a deterministic linear-time $\frac{3}{2}$-approximation algorithm for the NP-hard problem $J2|pmtn|C_{max}$.*

In Section 5, we consider deterministic online algorithms, and show that the trivial upper bound of 2 on the competitive ratio is asymptotically tight: *No deterministic online algorithm for the* 2*-machine,* n*-job preemptive job shop problem* $J2|pmtn|C_{max}$ *achieves a competitive ratio less than* $2 - \frac{1}{n}$. For the case of two jobs, we give a somewhat tighter bound: *No deterministic online algorithm for the* 2*-machine,* 2*-job preemptive job shop problem* $J2|pmtn; n = 2|C_{max}$ *achieves a competitive ratio of* c *for* $c < 1.64$.

Section 6 concludes the paper with directions for further work.

2 Preliminaries

We consider offline and online algorithms for the job shop scheduling problem with two machines and two or more jobs. We will denote the two machines r and b, for red and blue. In the case of two jobs, job 1 will be denoted by a list of tasks $x_1, x_2, \ldots$ and job 2 by a list $y_1, y_2, \ldots$, where odd-subscripted entries represent demands for the red machine and even-subscripted entries represent demands for the blue machine. (Dummy entries equal to zero can be added as needed in case a job does not begin with a request for r.) Each task must be serviced only after all preceding tasks in the same job have been served. The goal is to minimize the makespan, the time at which the last job finishes. We consider nonclairvoyant online algorithms; that is, an algorithm learns of the existence of a task only on completion of the previous task in the same job, and learns the service time of the task only when it finishes serving the task.

We begin with some standard definitions. In the standard notation for scheduling problems (see, for example, the survey by Lawler *et al.* [1]), our problems are denoted $J2|pmtn|C_{max}$, and so on; we will use this notation for each problem we consider. The J in the first field indicates the job shop problem; a number, if present, denotes the number of machines. Modifiers in the second field include $pmtn$ for preemption and $n = c$ to indicate that the number of jobs is restricted to c or fewer. We use n throughout to denote the number of jobs. C_{max} in the last field indicates that our optimality criterion is the completion time of the last job to finish. As usual, P_{max} denotes the maximum, over all jobs, of the sum of task lengths in the job, and Π_{max} denotes the maximum load on any machine. A schedule in which there is never a time at which all machines are idle is said to be *busy*.

See Borodin and El-Yaniv [14] for background on competitive analysis of online algorithms. An online algorithm A is *c-competitive* if there is some constant d such that for any input x,

$$cost_A(x) \leq c \cdot cost_{OPT}(x) + d,$$

where $cost_A(x)$ is the cost incurred by algorithm A on input x and $cost_{OPT}(x)$ is the optimal (offline) cost on x. The *competitive ratio* of A is the infimum over c such that A is c-competitive. If A is c-competitive and no online algorithm is c'-competitive for $c' < c$, algorithm A is said to be *strongly competitive.*

In the literature on randomized online algorithms, three types of adversarially generated inputs have been considered. We will be concerned with only two here. An oblivious adversary generates an input without knowledge of the randomized algorithm's online choices (i.e., the outcomes of its random events), although it does know the randomized algorithm and the probability distributions it uses. An offline adaptive adversary is allowed to generate its i^{th} input after observing the online algorithm's responses to the first $i-1$ inputs. The offline adaptive adversary serves the input offline, with full knowledge of the entire input sequence. Clearly, the latter adversary is at least as powerful as the former; also, lower bounds using an offline adaptive adversary apply to deterministic algorithms as well. Randomization does not add power to either type of adversary (Theorem 7.1 of [14]). For randomized online algorithms, we replace the costs in the definition of competitiveness by their expectations.

3 Randomized online algorithms against oblivious adversaries

3.1 Upper bound

We will now describe a randomized algorithm that achieves a competitive ratio of $\frac{3}{2}$ against oblivious adversaries for the 2-machine preemptive job shop problem $J2|pmtn|C_{max}$.

Definition: Let $1 \ldots n$ be an arbitrary ordering of the n jobs. Let R_i denote the total service time of all requests in job i for the red machine, and let B_i denote the total service time of all requests in job i for the blue machine. Let PRI1 denote the algorithm that assigns priority for the red machine in the order $1 \ldots n$ (i.e., job i has priority over jobs $i+1 \ldots n$) and for the blue machine in the order $n \ldots 1$, and let PRI2 denote the algorithm that reverses these orders. Let RPRI denote the randomized algorithm that runs PRI1 and

PRI2 each with probability 1/2.

Theorem 1 *RPRI is $\frac{3}{2}$-competitive against oblivious adversaries.*

Proof: Let i_1 be the index of the last job to complete under PRI1, and let i_2 be the index of the last job to complete under PRI2.

Claim: PRI1's cost is at most $\sum_{j=1}^{i_1} R_j + \sum_{j=i_1}^{n} B_j$.

To see this, note that at any time job j, $j > i_1$ is running on the red machine, job i_1 (which has not yet completed) must be either running on the blue machine or waiting for the blue machine; otherwise, i_1 would preempt the red machine from j. Thus some job j', $i_1 \le j' \le n$ must be running on the blue machine, since jobs $1 \ldots i_1 - 1$ have priority lower than i_1, so that $\sum_{j=i_1+1}^{n} R_j$ can be charged against $\sum_{j=i_1}^{n} B_j$. Similarly $\sum_{j=1}^{i_1-1} B_j$ can be charged to $\sum_{j=1}^{i_1} R_j$.

A similar argument shows that PRI2's cost is at most $\sum_{j=1}^{i_2} B_j + \sum_{j=i_2}^{n} R_j$.

Thus we need to show

$$1/2 * (\sum_{j=1}^{i_1} R_j + \sum_{j=i_2}^{n} R_j + \sum_{j=1}^{i_2} B_j + \sum_{j=i_1}^{n} B_j)$$

is at most 3/2 the cost of an optimal schedule OPT, or

$$\sum_{j=1}^{i_1} R_j + \sum_{j=i_2}^{n} R_j + \sum_{j=1}^{i_2} B_j + \sum_{j=i_1}^{n} B_j <= 3 * cost(OPT).$$

case 1 - $i_1 > i_2$: Each of the first two terms on the left side of the inequality is at most $\sum_{j=1}^{n} R_i$, and the sum of the last two terms is at most $\sum_{j=1}^{n} B_j$. Each of these is a lower bound on the optimal cost.

case 2 – $i_1 < i_2$: Similar to case 1.

case 3 – $i_1 = i_2$: The expression on the left side of the inequality is equal to

$$\sum_{j=1}^{n} R_i + \sum_{j=1}^{n} B_i + R_{i_1} + B_{i_1}.$$

The first two terms are each lower bounds on the optimal cost, and so is the sum of the last two terms. □

3.2 Lower bounds

Theorem 2 *No randomized online algorithm for the* 2*-machine, n-job preemptive job shop problem $J2|pmtn|C_{max}$ achieves a competitive ratio less than $\frac{3}{2} - \frac{1}{2n}$ against oblivious adversaries.*

Proof: We describe an adversary ADV which, for any $T > 0$, and for any randomized online algorithm ALG, will construct an input for which the optimal schedule has length at least T and for which ALG's expected schedule length is at least $\frac{3}{2} - \frac{1}{2n}$ times the optimum. For convenience, assume T/n is an integer; otherwise, round up to the next larger integer.

The adversary chooses one job i uniformly at random. All jobs request the same machine (say the red one) at time 0. All jobs request T/n units of service on the red machine. All jobs other than i end after their first request. Job i requests $T(n-1)/n$ on the blue machine after its red request is satisfied.

By the adversary's choice, job i is equally likely to be the j^{th} job to finish receiving its service on the red machine under ALG, for each $1 \leq j \leq n$. Thus the expected length of ALG's schedule is at least

$$\sum_{j=1}^{n} \frac{1}{n}[\frac{jT}{n} + T\frac{n-1}{n}] = \frac{1}{n}\frac{n(n+1)}{2}\frac{T}{n} + \frac{n-1}{n}T = T(\frac{3}{2} - \frac{1}{2n}).$$

The optimal schedule length is easily seen to be T, giving the desired ratio. □

For the two-job case we are able to give a tighter bound.

Theorem 3 *No randomized online algorithm for the 2-machine, 2-job preemptive job shop problem $J2|pmtn; n = 2|C_{max}$ achieves a competitive ratio of r for $r < \frac{4}{3}$ against oblivious adversaries.*

Proof: For convenience, assume that time is divisible into arbitrarily small units. We can remove this assumption by approximating it to within an arbitrarily small $\epsilon > 0$, and then scaling all values by $1/\epsilon$ to obtain integer inputs.

We describe a randomized adversary ADV such that, for any randomized algorithm ALG, $E(cost(ALG)) = \frac{4}{3}E(cost(OPT))$, where OPT denotes an optimal algorithm. ADV first uses one random bit to select job 1 or job 2 as the "good" job, i.e. the one that the optimal algorithm will execute first. Let G denote this choice. ADV then selects z uniformly at random from the interval $(0, 1]$. If $G = 1$, ADV sets $x_1 = z$, $x_2 = y_1 = 1$, and $y_2 = 0$; otherwise, ADV sets $y_1 = z$, $y_2 = x_1 = 1$, and $x_2 = 0$. ADV sets $x_i = y_i = 0$ for $i > 2$.

It is easy to see that $E(cost(OPT)) = 3/2$. Now, we show that $E(cost(ALG)) - E(cost(OPT)) = 1/2$. ALG pays OPT's cost, plus whatever amount of service it gives to the bad job before reaching the end of the bad job, or the end of the good job's requirement on r. Consider the curve contained in the unit square which denotes ALG's action in servicing jobs 1 and 2 when presented with an input for which the two jobs each have a single task for the red machine for 1 unit of time. The curve starts at the origin. From this point, movement in the x direction corresponds to executing job 1 on the red machine, and movement in the y direction corresponds to executing job 2 on the red machine. The curve is continuous, nondecreasing in x and y, and ends when $x = 1$ or $y = 1$.

We note that this curve can be used to determine ALG's behavior on any type of job sequence that ADV might generate. For example, fix z and let $G = 1$. Then on this sequence, ALG will follow the behavior specified by the curve until $x = z$. At this point, ALG will service the remainder of both jobs in parallel. The key observation is that the excess time ALG spends on this input sequence (beyond the optimal time) is the y value of the curve when $x = z$. Hence the average excess time of ALG over OPT when $G = 1$ is the area under the curve. Similarly, the average excess time when $G = 2$ is the area to the left of the curve.

Finally, we note that sum of the two areas is 1, and the events $G = 1$ and $G = 2$ each occur with probability 1/2; thus, the expected excess time is 1/2. $\square$

4 An offline 1.5-approximation

Theorem 4 *For every instance of $J2|pmtn|C_{max}$, there is a schedule of length $\frac{3}{2}\max(P_{max}, \Pi_{max})$. Moreover, such a schedule can be found in linear time. Thus, there is a deterministic linear-time $\frac{3}{2}$-approximation algorithm for the NP-hard problem $J2|pmtn|C_{max}$.*

Proof: The first claim is evident from an examination of the proof of Theorem 1. It remains to illustrate that the algorithm can be implemented to run in linear time.

The offline approximation algorithm runs an offline version of PRI1 and then PRI2 to produce two schedules, and returns the schedule with the shortest makespan. Below we give a linear time algorithm for PRI1.

In the following, let $above(j)$ be the next job in the current priority ordering for the red machine that has priority greater than job j, and let $below(j)$ be the next job with priority less than j. Initially, $above(j) = j - 1$ and $below(j) = j + 1$; as jobs complete, they will create gaps in this list. The jobs are stored in a doubly linked list ordered by priority, so that jobs which have completed can be removed and the priority ordering reestablished in constant time. For convenience, Offline PRI1 assumes there are two dummy jobs, job 0, which has the least priority for the blue machine and which always requests blue, and job $n + 1$ which has least priority on the red machine and which always requests red. We also append dummy tasks of length 0 for the red machine to the beginning of each real job.

The algorithm starts with the blue machine assigned to job 0 and the red machine assigned to $below(0) = 1$; this is an instance of state A below. The algorithm will always be in one of the two following states:

state A: Some job j executes on the blue machine and $below(j)$ executes on the red machine.

state B: Some job j executes on the red machine and $below(j)$ executes on the blue machine.

In either state, jobs less than j all wait for the blue machine, and jobs greater than $below(j)$ all wait for the red machine.

In both states, the jobs j and $below(j)$ run in parallel. In state A, if one job's current task completes before the other, the algorithm remains in state A, but j increases or decreases by 1, i.e., the pointer moves up or down by one job on the priority list of jobs. If both jobs complete their tasks at the same time but one or both of the jobs is finished at this point, we similarly stay in state A with the new j increasing or decreasing by 1. However if both jobs' tasks finish at the same time and neither job is finished, the algorithm moves to state B with j and $below(j)$ interchanged.

¿From state B, the algorithm always moves to state A. If one task completes before the other, the new j for state A is a job one position lower or one position higher in the priority list. If they complete at the same time, the new j (in state A) is the same as in state B unless the job j has ended in which case the new j is 1 above the old j.

Each transition to a new state requires constant time, and at least one complete task is scheduled by each transition; thus the running time is linear in the total number of tasks.

$\square$

5 Deterministic online algorithms

In this section, we consider deterministic online algorithms and randomized online algorithms against adaptive offline adversaries. It is easy to show that any deterministic algorithm that produces only busy schedules is 2-competitive.We now show that, this bound is asymptotically tight as the number of jobs increases.

Theorem 5 *No randomized online algorithm for the* 2*-machine,* n*-job preemptive job shop problem* $J2|pmtn|C_{max}$ *achieves a competitive ratio less than* $2 - \frac{1}{n}$ *against offline adaptive adversaries.*

Proof: We describe an adversary ADV which, for any $T > 0$ and for any randomized online algorithm ALG, will construct an input for which the optimal schedule has length at least T and for which ALG constructs a schedule of length at least $2 - \frac{1}{n}$ times the optimum. For convenience, assume T/n is an integer; otherwise, round up to the next larger integer.

All jobs request the same machine (say the red one) at time 0. All jobs will continue to request the red machine until the last one (say i) receives T/n units of service under ALG. At that time, all jobs other than i end. Let $T' \geq T(n-1)/n$ denote the sum of the amounts of service received by jobs other than i on the red machine. Job i now requests the blue machine for time T'.

The length of ALG's schedule is $2T' + T/n$. It is easy to see that an optimal schedule OPT is obtained by serving job i first on the red machine, and then serving the other jobs on the red machine in parallel with job i's request for the blue machine. The length of this schedule is $T' + T/n$. Thus we have

$$\frac{cost(ALG)}{cost(OPT)} = \frac{2T' + T/n}{T' + T/n} \geq \frac{2T(n-1)/n + T/n}{T(n-1)/n + T/n} = 2 - \frac{1}{n}.$$

$\Box$

Corollary 6 *No deterministic online algorithm for the* 2*-machine,* n*-job preemptive job shop problem* $J2|pmtn|C_{max}$ *achieves a competitive ratio less than* $2 - \frac{1}{n}$.

For the two-job case we are able to give a somewhat tighter bound.

Theorem 7 *No randomized online algorithm for the* 2*-machine,* 2*-job preemptive job shop problem* $J2|pmtn; n = 2|C_{max}$ *achieves a competitive ratio of* r *for* $r < 1.64$ *against offline adaptive adversaries.*

Proof: For convenience, assume that time is divisible into arbitrarily small units. We can remove this assumption by approximating it to within an arbitrarily small $\epsilon > 0$, and then scaling all values by $1/\epsilon$ to obtain integer inputs.

The adversary ADV will generate k tasks in each job for some k, followed by a final task in only one of the two jobs. ADV's offline schedule will be obtained by executing either x_i in parallel with y_{i+1} or y_i in parallel with x_{i+1}, for $1 \leq i \leq k$.

ADV causes both jobs to request the red machine at time 0. ADV determines the lengths of tasks x_1 and y_1 for the red machine in response to the online algorithm ALG's actions. Both jobs continue to request the red machine until they have each received time at least

1; thus, $\min(x_1, y_1) = 1$. In the following stages, ADV generates tasks on the opposite machine to that of the previous stage. ADV does so until ALG has given each job as much service as the opposite job in the previous stage; i.e., so that $x_{i+1} \geq y_i$ and $y_{i+1} \geq x_i$; note that either the cost of x_i or that of y_i can be hidden in the offline schedule. Finally, ADV sets either $x_{k+1} = y_k$, $y_k = 0$ or $y_{k+1} = x_k$, $x_k = 0$, and $x_i = y_i = 0$ for $i > k+1$. Notice that ALG never executes the two jobs in parallel, whereas the length of the offline schedule is equal to P_{max}. In the former case, ALG's cost is $y_k + \sum_{i=1}^{k} x_i + \sum_{i=1}^{k} y_i$ and the offline cost is $y_k + \sum_{i=1}^{k} x_i$; in the latter, ALG's cost is $x_k + \sum_{i=1}^{k} x_i + \sum_{i=1}^{k} y_i$ and the offline cost is $x_k + \sum_{i=1}^{k} y_i$.

It remains to describe how ADV determines k. If at any time ADV is able to terminate the game and cause the ratio of the online cost to the offline cost to exceed the stated bound, it does so. The following exhaustive analysis shows that by choosing $k \in \{1, 2\}$, this must be possible for ADV, and thus yields the stated ratio.

We can view this adversarial process as a two-player game in which the ALG chooses the values x_i and y_i subject to constraints imposed by ADV. ADV imposes the constraints

$$\begin{aligned} x_1 &\geq 1 \\ y_1 &\geq 1 \end{aligned}$$

$$\begin{aligned} x_{i+1} &\geq y_i \\ y_{i+1} &\geq x_i \end{aligned}$$

along with the (nonlinear) condition that at least one member of each pair of constraints is tight.

In order to achieve a competitive ratio $2-\delta$, ALG must ensure that

$$\frac{y_k + \sum_{i=1}^k x_i + \sum_{i=1}^k y_i}{y_k + \sum_{i=1}^k x_i} \le 2 - \delta$$

and

$$\frac{x_k + \sum_{i=1}^k x_i + \sum_{i=1}^k y_i}{x_k + \sum_{i=1}^k y_i} \le 2 - \delta.$$

Rearranging these, ALG must ensure that

$$\delta x_k + \sum_{i=1}^{k-1} x_i \le (1-\delta) \sum_{i=1}^{k} y_i$$

and

$$\delta y_k + \sum_{i=1}^{k-1} y_i \le (1-\delta) \sum_{i=1}^{k} x_i.$$

We will show that every solution to this set of constraints for $k \in \{1,2\}$ yields a value of δ around 0.352 or less.

Assume without loss of generality that $y_2 = x_1$; the case $x_2 = y_1$ is symmetric. We can write

$$\delta y_1 \le (1-\delta)x_1 \tag{1}$$

$$x_1 + \delta x_2 \le (1-\delta)(y_1 + y_2) \tag{2}$$

$$y_1 + \delta y_2 \le (1-\delta)(x_1 + x_2) \tag{3}$$

$$x_1 \geq 1 > 0 \tag{4}$$

$$y_2 = x_1 \tag{5}$$

¿From these it follows that

$$\begin{aligned}
\delta(x_1 + x_2) &\leq (1-\delta) \cdot y_1 && \text{(by 2 and 5)} \\
\delta x_1 + y_1 &\leq \tfrac{(1-\delta)^2}{\delta} \cdot y_1 && \text{(by 3 and 5)} \\
\delta x_1 &\leq (\tfrac{(1-\delta)^2}{\delta} - 1) \cdot y_1 && \\
\delta x_1 &\leq (\tfrac{(1-\delta)^2}{\delta} - 1) \cdot \tfrac{1-\delta}{\delta} \cdot x_1 && \text{(by 1)} \\
\delta &\leq \tfrac{(1-\delta)^2 - \delta}{\delta} \cdot \tfrac{1-\delta}{\delta} && \text{(by 4)}
\end{aligned}$$

or $2\delta^3 - 4\delta^2 + 4\delta - 1 \leq 0$.

The polynomial on the left has a single root at approximately 0.352. □

Corollary 8 *No deterministic online algorithm for the* 2*-machine,* 2*-job preemptive job shop problem* $J2|pmtn; n = 2|C_{max}$ *achieves a competitive ratio of* c *for* $c < 1.64$.

Our exhaustive analysis for small k surely does not yield the greatest lower bound that can be obtained; however, it can be shown that unless we consider more than just two offline schedules, the method cannot yield a bound greater than 1.8.

6 Discussion

In this paper, we have studied some practical and tractable special cases of the job shop scheduling problem. An obvious open question is whether there is a PTAS for the general case of $J2|pmtn|C_{max}$. We would like to obtain tighter bounds (both deterministic and randomized) for the online problem with a fixed number of jobs. Optimization criteria other than minimum makespan are other directions for further work.

Acknowledgements

We thank Anna Karlin for stimulating discussions relating to this work, Jayram Thathachar for discussions and for simplifying the proof of Theorem 7, and Gerhard Woeginger for pointing out related results.

References

[1] E.L. Lawler, J.K. Lenstra, A.H.G. Rinnooy Kan, and D.B. Shmoys. Sequencing and Scheduling: Algorithms and Complexity. In S.C. Graves, A.H.G. Rinnooy Kan, and P.H. Zipkin (eds.), *Logistics of Production and Inventory*, Handbooks in Operations Research and Management Science 4, North-Holland, Amsterdam, 1993, 445-522.

[2] J. Ousterhout. Scheduling Techniques for Concurrent Systems. In *Proceedings of the International Conference on Distributed Computing Systems*, pp. 22-30, 1982.

[3] E. Rosti, G. Serazzi, E. Smirni, and M. Squillante. The Impact of I/O on Program Behavior and Parallel Scheduling. In *Proceedings of the ACM SIGMETRICS / IFIP WG 7.3 Joint International Conference on Measurement and Modeling of Computer Systems*, pp.56-65, 1998.

[4] E. Smirni and D. Reed. Lessons from Characterizing the Input/Output Behavior of Parallel Scientific Applications. *Performance Evaluation* 33, pp. 27-44, 1998.

[5] D. Burger, R. Hyder, B. Miller, and D. Wood. Paging Tradeoffs in Distributed-Shared-Memory Multiprocessors. In *Proceedings of Supercomputing '94*, 1994.

[6] F.T. Leighton, B. Maggs, and S. Rao. Packet routing and jobshop scheduling in O(congestion + dilation) steps. *Combinatorica* 14, pp. 167–186, 1994.

[7] D. Shmoys, C. Stein, and J. Wein. Improved Approximation Algorithms for Shop Scheduling Problems. *SIAM Journal on Computing* 23:3, 617-632, 1994.

[8] L.A. Goldberg, M. Paterson, A. Srinivasan, and E. Sweedyk. Better Approximation Guarantees for Job-Shop Scheduling. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 599–608, 1997.

[9] K. Jansen, R. Solis-Oba, and M. Sviridenko. A linear time approximation scheme for the job shop scheduling problem. In *Proceedings of the The Second International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, August, 1999.

[10] S.V. Sevastianov and G.J. Woeginger. Makespan minimization in preemptive two machine job shops. *Computing* 60 (1998), 73-79.

[11] S. Albers, B. von Stengel, and R. Werchner. A combined BIT and TIMESTAMP Algorithm for the List Update Problem. *Information Processing Letters*, 56:135-139, 1995.

[12] B. Kalyanasundaram and K. Pruhs. Maximizing Job Completions Online. In *Proceedings of the European Symposium on Algorithms (ESA)*, 1998.

[13] S. Ben-David, E. Dichterman, J. Noga, and S. Seiden. On the Power of Barely Random Online Algorithms. Unpublished manuscript, 1998.

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