Online, Non-preemptive Scheduling of Equal-Length Jobs on Two Identical Machines*

Michael H. Goldwasser and Mark Pedigo

Saint Louis University, Dept. of Mathematics and Computer Science
221 North Grand Blvd.; St. Louis, MO 63103-2007
{goldwamh, pedigom}@slu.edu

Abstract. We consider the non-preemptive scheduling of two identical machines for jobs with equal processing times yet arbitrary release dates and deadlines. Our objective is to maximize the number of jobs completed by their deadlines. Using standard nomenclature, this problem is denoted as $P2 \mid p_j = p, r_j \mid \sum \overline{U}_j$. The problem is known to be polynomially solvable in an offline setting.

In an online variant of the problem, a job's existence and parameters are revealed to the scheduler only upon that job's release date. We present an online, deterministic algorithm for the problem and prove that it is $\frac{3}{2}$-competitive. A simple lower bound shows that this is the optimal deterministic competitiveness.

Keywords: Algorithms, Online, Scheduling.

1 Introduction

We present an online, non-preemptive, deterministic algorithm for scheduling two machines in the following setting. Each job j is specified by three non-negative integer parameters, with r_j denoting its release time, p_j its processing time, and d_j its deadline. For this paper, we assume all processing times are equal, thus $p_j = p$ for a fixed constant p. In order to successfully complete a job j, the scheduler must devote a machine to it for p consecutive units of time during the interval $[r_j, d_j)$.

We examine an *online* model for the problem, in which the scheduler is oblivious to a job's existence and characteristics until that job's release time. We use competitive analysis to measure the performance of an algorithm $\mathcal{A}$ by comparing the quality of the schedule it produces to that of an optimal *offline* scheduler that has a priori knowledge of the entire instance [3]. Our main result is the presentation of a $\frac{3}{2}$-competitive, deterministic online algorithm for the two-machine problem. A simple lower bound shows that this is the best possible result.

Preliminaries and Notations. We let $x_j = d_j - p$ denote a job's *expiration time*, namely the last possible time it can be started, and we fix a canonical linear

* This material is based upon work supported by the National Science Foundation under Grant No. CCR-0417368.

L. Arge and R. Freivalds (Eds.): SWAT 2006, LNCS 4059, pp. 113–123, 2006.

ordering of jobs, $\prec$, such that $i \prec j$ implies $x_i \leq x_j$. As all job parameters are presumed to be integral, we consider time as a series of discrete steps. Our algorithm provides what has been termed *immediate notification* [8]. At the moment a job is released, the scheduler must either accept or reject that job. An accepted job need not be scheduled precisely at that moment, but the scheduler must guarantee that it will be successfully completed by its deadline. In the case where several jobs are released at a time t, we assume that they are considered in arbitrary order with the scheduler providing notification to each in turn.

We introduce notation FEASIBLE$(\mathcal{J}, t_1, t_2)$ to represent a combinatorial boolean property, namely whether a set $\mathcal{J}$ of released jobs can be achieved on two machines, given that one of the machines cannot be used prior to time t_1 nor the other prior to time t_2. We do not make any assumption about the relationship between t_1 and t_2, though conventionally we will order them with $t_1 \leq t_2$ when known. A further discussion of this feasibility property is provided in Section 3.

Related Work. In recent independent work, a different algorithm and analysis has been presented claiming the same $\frac{3}{2}$-competitive upper bound [5]. The single machine version of this problem is well studied. A deterministic, non-preemptive greedy algorithm is known to be 2-competitive with equal-length jobs, and this is the best possible deterministic result [2, 6]. If each job satisfies a *patience* requirement, namely that $d_j - r_j \geq (1+\kappa) \cdot p_j$ for some constant $\kappa > 0$, then the deterministic greedy algorithm is $(1 + \frac{1}{\lfloor \kappa \rfloor + 1})$-competitive [7]. A similar algorithm achieves the same competitiveness while also providing immediate notification [8]. When considering randomized online algorithms, there exists a $\frac{5}{3}$-competitive algorithm [4] versus a $\frac{4}{3}$-competitive lower bound [6].

In the offline setting, checking the feasibility of a set of equal-length jobs with release dates and deadlines is polynomially solvable, even for an arbitrary number of identical machines [10, 11]. The optimization problem is polynomially solvable for any *fixed* number of machines, even with weighted utility $(Pm \mid p_j = p, r_j \mid \sum w_j U_j)$ [1], yet open with *arbitrary* number of machines, even when unweighted $(P \mid p_j = p, j \mid \sum U_j)$.

2 Algorithm Definition

The algorithm, $\mathcal{A}$, maintains a queue of jobs which have been accepted but not yet started. As the composition of the queue changes over time, we introduce the following notation. We let $Q_t^{\mathcal{A}}$ denote the queue as it exists at the onset of time step t. For each job j released at time t, we let $Q_{r_j}^{\mathcal{A}}$ denote the queue as it exists when j's release was considered (thus $Q_{r_j}^{\mathcal{A}} \supseteq Q_t^{\mathcal{A}}$ may contain newly accepted jobs which were considered prior to j). Job j is accepted into the system precisely if it is feasible to do so. Specifically, we check FEASIBLE$(Q_{r_j}^{\mathcal{A}} \cup \{j\}, c, \dot{c})$, where c (resp. $\dot{c}$) represents the time until which the first (resp. second) machine is committed to a currently running job. In the case where a machine is not running a job, we considered it trivially committed until time t.

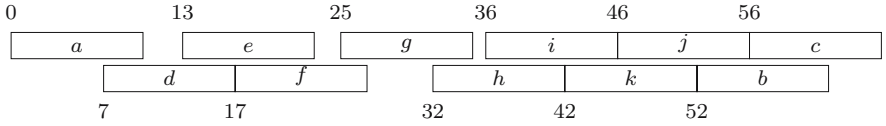

Fig. 1. The schedule produced by $\mathcal{A}$ on an instance with $p = 10$ and jobs: $a = \langle 0, 60\rangle$, $b = \langle 0, 71\rangle$, $c = \langle 0, 71\rangle$, $d = \langle 3, 30\rangle$, $e = \langle 3, 31\rangle$, $f = \langle 3, 33\rangle$, $g = \langle 3, 37\rangle$, $h = \langle 3, 45\rangle$, $i = \langle 3, 52\rangle$, $j = \langle 3, 56\rangle$, $k = \langle 38, 55\rangle$

After considering all newly released jobs at time t, the scheduling policy is as follows. If neither machine is currently committed to a job and the queue is non-empty, the $\prec$-minimal job is started on an arbitrary machine. If one machine is committed to a job, yet the other is uncommitted (including the case when a job was just started by the preceding rule), a decision is made as to whether or not to start a job on the other machine. For the sake of analysis, we will refer to this as a *secondary decision*. Specifically, let $\dot{Q}_t^{\mathcal{A}}$ denote the queue at the point this decision is made and let $c > t$ denote the time until which the running machine is committed. We begin the $\prec$-minimal job of $\dot{Q}_t^{\mathcal{A}}$ on the available machine if the test $\textsc{Feasible}(\dot{Q}_t^{\mathcal{A}}, c, t + p + 1)$ fails. Intuitively, if there is enough flexibility, the algorithm prefers to idle for the moment, leaving open the possibility of starting a more urgent job should one soon arrive. Figure 1 shows the schedule produced by this algorithm on an example.

3 A Supplemental Feasibility Test

In this section, we discuss the feasibility test, $\textsc{Feasible}(\mathcal{J}, t_1, t_2)$. Such a feasibility condition is satisfied if and only if it can be achieved by scheduling jobs according to an earliest deadline first (EDF) rule.

A classic result of Jackson proves this for a single machine and a set of jobs with arbitrary processing times and deadlines [9]. With arbitrary job lengths and two or more machines, that argument no longer applies. However with equal-length jobs, the EDF schedule suffices. Since all jobs of $\mathcal{J}$ are presumed to have been released, any idleness in a feasible schedule beyond t_i on a machine M_i can be removed. Furthermore, if any job $j \in \mathcal{J}$ is started before some other job with earlier deadline, those two jobs of equal length can be transposed while still guaranteeing a feasible schedule. Based on this structure, we provide the following lemmas, specifically in the context of $\textsc{Feasible}(\mathcal{J}, t_1, t_2)$.

Lemma 1. *For arbitrary set $\mathcal{J}$, $t_1 \le t_1'$ and $t_2 \le t_2'$,* $\textsc{Feasible}(\mathcal{J}, t_1', t_2')$ *implies* $\textsc{Feasible}(\mathcal{J}, t_1, t_2)$.

Lemma 2. *For arbitrary set $\mathcal{J}$ and $t_1 \le t_2$, let job f be $\prec$-minimal of $\mathcal{J}$.* $\textsc{Feasible}(\mathcal{J}, t_1, t_2)$ *if and only if $x_f \ge t$ and* $\textsc{Feasible}(\mathcal{J} \setminus \{f\}, t_1 + p, t_2)$.

Lemma 3. *Assume* $\textsc{Feasible}(\mathcal{J}, t_1, t_2)$ *for $t_1 \le t_2$. Let $h \in \mathcal{J}$ be an element for which there are i other elements of $\mathcal{J}$ which precede it as per $\prec$-order. Then $x_h \ge t_1 + \lfloor \frac{i}{2} \rfloor \cdot p$.*

Lemma 4. *For arbitrary $\mathcal{J}$ and t, if* FEASIBLE$(\mathcal{J}, t, t+p+1)$ *yet not* FEASIBLE$(\mathcal{J}, t+1, t+p)$*, then there must exist a job $f \in \mathcal{J}$ with $x_f = t$.*

4 Competitive Analysis

We fix an arbitrary instance $\mathcal{I}$ and an optimal schedule, OPT, for that instance. For a job j achieved by OPT, we let s_j^{OPT} denote the time at which it is started; we similarly define $s_j^{\mathcal{A}}$ for jobs achieved by $\mathcal{A}$. Though OPT is not constructed in online fashion, we introduce a formal notation of Q_t^{OPT} to be symmetric to that of $Q_t^{\mathcal{A}}$. Namely Q_t^{OPT} consists of all jobs which are released strictly before time t yet started by OPT on or after time t. Our analysis is based upon a form of a potential argument. Specifically, we will soon define functions $\Phi_t^{\mathcal{A}}$ and Φ_t^{OPT} which measure the quality of the schedules being developed as of time t, by $\mathcal{A}$ and OPT respectively. We will view these two potential functions as payment to the respective schedules, with a handicap given to the online algorithm. Specifically, $\mathcal{A}$ will receive 3 units for each job it starts whereas OPT receives 2 units for each job. To prove a $\frac{3}{2}$-competitive ratio in the end, we must show that $\mathcal{A}$ collects at least as much payment as OPT. To properly compare the merits of the schedules at interim times, our potential functions contain full payment for jobs which have been started and a partial payment to account for the inherit potential of each queue.

Before defining the exact potential functions, we introduce some additional notations. We let $F_t^{\mathcal{A}}$ (resp. F_t^{OPT}) designate the set of jobs started strictly before time t by $\mathcal{A}$ (resp. OPT). We define $W_t^{\mathcal{A}} = Q_t^{\mathcal{A}} \setminus Q_t^{\text{OPT}}$ and symmetrically, $W_t^{\text{OPT}} = Q_t^{\text{OPT}} \setminus Q_t^{\mathcal{A}}$, to denote jobs which are currently waiting in one queue but not the other (presumably because they were never accepted or were already started). Intuitively, our potential functions ignore jobs which are common to the two queues but account for the difference between those queues. However there is one more anomaly which may arise.

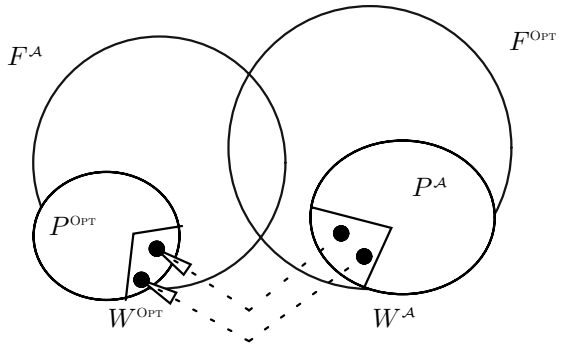

Fig. 2. Relationship among the sets used in analysis at a given time. The dashed lines represents matches between pairs of items from $W^{\mathcal{A}} \setminus P^{\mathcal{A}}$ to $W^{\text{OPT}} \setminus P^{\text{OPT}}$ respectively.

If we identify a job waiting in $W^{\mathcal{A}}$ which has an expiration time at least as large as some other job waiting in W^{OPT}, we choose not to let either job effect the potential functions. Intuitively, such a pairing can only be to the advantage of the algorithm. Therefore, for each time t we maintain a partial matching $M_t : W_t^{\mathcal{A}} \to W_t^{\text{OPT}}$ (the precise rule for establishing a match is omitted from this version due to space limitations). We introduce notation $P_t^{\mathcal{A}}$ and P_t^{OPT} to identify respective subsets of the waiting jobs which do not participate in the matching.

Namely, we let $P_t^{\mathcal{A}} = \{j \in W_t^{\mathcal{A}} : M_t(j) \text{ is undefined}\}$. Similarly, we let $P_t^{\text{OPT}} = \{j \in W_t^{\text{OPT}} : M_t^{-1}(j) \text{ is undefined}\}$. A typical Venn diagram of the various sets we have defined is shown in Figure 2. We now define two potential functions as follows:

$$\Phi_t^{\mathcal{A}} = \begin{cases} 3 \cdot |F_t^{\mathcal{A}}| & \text{if } P_t^{\mathcal{A}} = \emptyset \\ 3 \cdot |F_t^{\mathcal{A}}| + 1 \cdot |P_t^{\mathcal{A}}| + 1 & \text{if } P_t^{\mathcal{A}} \neq \emptyset \end{cases}$$
$$\Phi_t^{\text{OPT}} = \quad 2 \cdot (|F_t^{\text{OPT}} \cup P_t^{\text{OPT}}|)$$

In effect, we take an advance credit of two for the first job in $P_t^{\mathcal{A}}$, and an advanced credit of one for each additional such job. This means that when these jobs are later scheduled, most provide a balance of two additional credits. The last job of P^{OPT} only provides a balance of one credit, yet we will see that an adversary cannot as readily exploit the existence of that last job. Looking at Φ_t^{OPT}, we make full payment for jobs which OPT has started as well as for unmatched jobs which OPT holds in P_t^{OPT}.

Our end goal is to show that $\Phi_\infty^{\text{OPT}} \leq \Phi_\infty^{\mathcal{A}}$. This inequality suffices to prove the $\frac{3}{2}$-competitiveness, since $P_\infty^{\mathcal{A}} = P_\infty^{\text{OPT}} = \emptyset$. Ideally, we would like to show that $\Phi_t^{\text{OPT}} \leq \Phi_t^{\text{OPT}}$ for all times t; unfortunately this cannot be assured.

Instead we break the overall time period into distinct regions, $[s, t)$, such that $\Phi_t^{\text{OPT}} \leq \Phi_t^{\mathcal{A}}$ at the end of each such region. We consider two types of regions: in an idle region both machines of $\mathcal{A}$ are idle, in a busy region at least one machine of $\mathcal{A}$ is at work throughout. If both machines of $\mathcal{A}$ are idle at a time s, we consider idle region $[s, t)$ where t is the next time at which either machine starts a job, if any.

Lemma 5. *For an idle region $[s, t)$, $\Phi_t^{\mathcal{A}} = \Phi_s^{\mathcal{A}}$ and $\Phi_t^{\text{OPT}} = \Phi_s^{\text{OPT}}$.*

Proof. As both machines idle at time s, $Q_s^{\mathcal{A}} = \emptyset$, and as they remain idle, no further jobs are released prior to time t. Thus $Q_t^{\mathcal{A}} = \emptyset$ as well. With the empty queue, sets $W_s^{\mathcal{A}}$, $P_s^{\mathcal{A}}$, $W_t^{\mathcal{A}}$ and $P_t^{\mathcal{A}}$ must be empty and the matchings $M_s(\cdot)$ and $M_t(\cdot)$ trivially empty. As no jobs are completed, $F_t^{\mathcal{A}} = F_s^{\mathcal{A}}$ and so we conclude that $\Phi_t^{\mathcal{A}} = \Phi_s^{\mathcal{A}}$. Because no new jobs are released throughout the region and no matches exist, the only jobs which OPT can schedule are those in $Q_s^{\text{OPT}} = P_s^{\text{OPT}}$. This implies that $F_t^{\text{OPT}} \cup P_t^{\text{OPT}} = F_s^{\text{OPT}} \cup P_s^{\text{OPT}}$. □

For a time s at which $\mathcal{A}$ starts executing a job, we define a busy region $[s, t)$ as follows. We define $t > s$ as the first subsequent time when either both machines are idle, or when one machine starts a job yet the other remains idle (that is, for at least one unit). A trivial consequence of this definition is that there is never

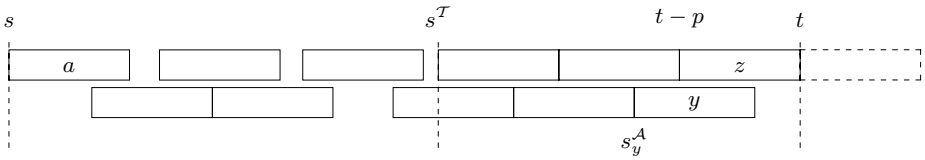

Fig. 3. Typical configuration of algorithm's two machines during a region $[s,t)$. In this example, $|\mathcal{R}| = 11$ and $|\mathcal{T}| = 5$.

a time when both machines are idle within such a region. For ease of exposition throughout the analysis of region $[s,t)$, we let $\mathcal{R}$ denote the set of jobs started by $\mathcal{A}$ during the region. We let a denote the first job of $\mathcal{R}$ to be started, namely at time s. We let z denote the last job of $\mathcal{R}$ to be started, namely at time $t-p$.

In the case where $|\mathcal{R}| = 1$, the region is composed trivially of job $a = z$. For $|\mathcal{R}| \geq 2$, we further let y denote the second-to-last job of $\mathcal{R}$ to be started. We note that y must run on the opposite machine as z and be executing at the time that z is started, as otherwise z would be starting at a time when the other machine is idle, contradicting our definition of t. Thus, $s_y^{\mathcal{A}} \leq s_z^{\mathcal{A}} = t-p < s_y^{\mathcal{A}} + p$.

For $|\mathcal{R}| \geq 2$, we define the *tail of the region*, denoted as $\mathcal{T}$, as follows. Let $s^{\mathcal{T}}$ be the latest time at which $\mathcal{A}$ starts a job of $\mathcal{R}$ following a time step $[s^{\mathcal{T}}-1, s^{\mathcal{T}})$ at which a machine was idle. We note that $s^{\mathcal{T}}$ is well defined as at least one job of $\mathcal{R}$ must follow such an idle time step. In particular, if the second job of the region is started strictly after time s, then it suffices. Alternatively, the region begins with two jobs starting precisely at time s. Such a region could only follow an idle region, as a previous busy region could not have ended at such a time s. Having defined $s^{\mathcal{T}}$, tail $\mathcal{T} \subseteq \mathcal{R}$ is the set of jobs started by $\mathcal{A}$ during the interval $[s^{\mathcal{T}}, t)$. Figure 3 demonstrates a typical region.

Structural Properties of Busy Regions Produced by $\mathcal{A}$

Lemma 6. *If there exists a time t at which at least one machine is idle in $\mathcal{A}$ and a job j such that $r_j \leq t \leq x_j$, then j cannot be rejected by $\mathcal{A}$.*

Proof. By the algorithm definition, a machine is left idle at time t only if it is feasible to achieve its current queue even when that machine is left idle until time $t+p+1$ or later. Since j could be feasibly achieved over the interval $[t, t+p]$ without disrupting the completion of any other jobs in the system at that time, it must be accepted. □

Lemma 7. *For busy region $[s,t)$,* FEASIBLE$(Q_t^{\mathcal{A}}, t, t+p+1)$.

Proof. By the definition of a region, neither machine was committed to a job at the onset of time t, and at least one remains idle during the time $[t, t+1)$. If both machines remain idle at time t, then $Q_t^{\mathcal{A}} = \emptyset$ and thus the lemma trivially true. Alternatively some job, f, starts on one machine at t, while the secondary decision is to remain idle. Based on the algorithm definition, it must be that the test FEASIBLE$(\dot{Q}_t^{\mathcal{A}}, t+p, t+p+1)$ succeeded, and by Lemma 2,

FEASIBLE($\dot{Q}_t^{\mathcal{A}} \cup \{f\}, t, t+p+1$). Since $Q_t^{\mathcal{A}} \subseteq \{f\} \cup \dot{Q}_t^{\mathcal{A}}$, this implies FEASIBLE($Q_t^{\mathcal{A}}, t, t+p+1$). □

Lemma 8. *For busy region $[s,t)$ and arbitrary time t', assume $j \neq a$ is the latest job of $\mathcal{R}$ to be started by $\mathcal{A}$ such that $x_j \geq t'$. In this case, j must start due to a secondary decision at some time s_j, and further it must be that $\dot{Q}_{s_j}^{\mathcal{A}} \setminus \{j\} \subseteq Q_t^{\mathcal{A}}$.*

Lemma 9. *For any $j \neq a$ started by $\mathcal{A}$ during busy region $[s,t)$, $x_j \leq t$.*

Proof. For contradiction, let j with $x_j \geq t+1$ be the latest such job to be started by $\mathcal{A}$. By Lemma 8, j is started through a secondary decision at a time s_j, and $\dot{Q}_{s_j}^{\mathcal{A}} \setminus \{j\} \subseteq Q_t^{\mathcal{A}}$. By Lemma 7, FEASIBLE($Q_t^{\mathcal{A}}, t, t+p+1$) and thus FEASIBLE($\dot{Q}_{s_j}^{\mathcal{A}} \setminus \{j\}, t, t+p+1$). Since $x_j \geq t+1$, we can schedule j over the interval $[t+1, t+p+1]$ while still achieving $\dot{Q}_{s_j}^{\mathcal{A}} \setminus \{j\}$ starting the machines respectively at t and $t+p+1$. Therefore FEASIBLE($\dot{Q}_{s_j}^{\mathcal{A}}, t, t+1$). Since $t \geq s_j + p$ and $t \geq c$ where c denotes the time until which the opposite machine was committed as j started, this demonstrates the feasibility of FEASIBLE($\dot{Q}_{s_j}^{\mathcal{A}}, c, s_j+p+1$). This contradicts the fact that $\mathcal{A}$ starts j with a secondary decision at s_j. □

Lemma 10. *For busy region $[s,t)$, if $\exists j \in \mathcal{R}$ with $j \neq a$ and $x_j \geq s_y^{\mathcal{A}} + p$, then $x_z \geq s_y^{\mathcal{A}} + p$.*

Lemma 11. *For busy region $[s,t)$ with $|\mathcal{R}| \geq 2$, we consider the tail $\mathcal{T}$. If all jobs of $\mathcal{T}$ are released on or before time $s^{\mathcal{T}} - 1$, then $x_z \geq s_y^{\mathcal{A}} + p$.*

Lemma 12. *For busy region $[s,t)$ with $|\mathcal{R}| \geq 2$, if $x_z \geq s_y^{\mathcal{A}} + p$, then each of the following are true.*

(A) Any job k with $r_k \leq t - p \leq x_k$ must have been accepted by $\mathcal{A}$.
(B) There exist job $f \in Q_t^{\mathcal{A}}$ such that $s_f^{\mathcal{A}} = x_f = t$.

Comparing Progress of $\mathcal{A}$ Versus OPT over a Busy Region

For ease of exposition in the analysis of busy region $[s,t)$, we let set $\mathcal{G} = (F_t^{\text{OPT}} \cup P_t^{\text{OPT}}) \setminus (F_s^{\text{OPT}} \cup P_s^{\text{OPT}})$ denote those jobs which contribute to Φ^{OPT} during the region $[s,t)$. We let $\mathcal{G}_+$ (resp. $\mathcal{G}_-$) denote those jobs of $\mathcal{G}$ which were accepted (resp. rejected) by $\mathcal{A}$. We further let $\tilde{j}_t$ represent the element with which j is matched at time t, if such element exists, and j otherwise. An element's match often serves as a substitute in our analysis.

Lemma 13. *Each $j \in \mathcal{G}$ satisfies precisely one of the following conditions*

(A) $s \leq s_j^{\text{OPT}} < t$ and $j \notin P_s^{\text{OPT}}$;
(B) $s_j^{\text{OPT}} = t$, $\tilde{j}_s \in F_t^{\mathcal{A}} \setminus F_s^{\mathcal{A}}$ and $\tilde{j}_s \neq a$;
(C) $s_j^{\text{OPT}} \geq t$ and $\tilde{j}_s = a$.

Based upon the three conditions of Lemma 13, we can define another partition of the set $\mathcal{G}$ into sets $\mathcal{G}_A$, $\mathcal{G}_B$ and $\mathcal{G}_C$. We can superimpose this partition together with our partition of $\mathcal{G}_+$ and $\mathcal{G}_-$ to denote sets such as $\mathcal{G}_{A^+} \equiv \mathcal{G}_A \cap \mathcal{G}_+$ and $\mathcal{G}_{A^-} \equiv \mathcal{G}_A \cap \mathcal{G}_-$. Though we might use similar notations for $\mathcal{G}_B$ and $\mathcal{G}_C$, the next lemma shows that this is unnecessary.

Lemma 14. *All jobs of $\mathcal{G}_B$ and $\mathcal{G}_C$ are accepted by $\mathcal{A}$.*

Proof. For $j \in \mathcal{G}$, $r_j < t$, yet for $j \in \mathcal{G}_B \cup \mathcal{G}_C$, $x_j \geq s_j^{\text{OPT}} \geq t$. By the definition of the region, at least one machine of $\mathcal{A}$ is idle at time t. Therefore, we may apply Lemma 6 to assure the acceptance of j by $\mathcal{A}$. □

Our next lemma specifically examines sets $\mathcal{G}_{A^+}$ and $\mathcal{G}_{A^-}$. We can further partition each of these sets based upon which machine OPT uses to achieve such a job. For machine m, we let $\mathcal{G}_{A^+}^m$ denote the number of jobs of $\mathcal{G}_{A^+}$ which are run by OPT, and $\mathcal{G}_{A^-}^m$ the corresponding number from $\mathcal{G}_{A^-}$.

Lemma 15. *For busy region $[s,t)$ and arbitrary machine m,*

(A) $|\mathcal{G}_B^m| + |\mathcal{G}_{A^+}^m| + 2 \cdot |\mathcal{G}_{A^-}^m| \leq |\mathcal{R}|$.

(B) Furthermore, if inequality (A) is tight and $|\mathcal{R}| \geq 2$, both of the following are true.

(1) Either m starts a job of $\mathcal{G}_{A^-}^m$ at $t-p$ or is processing a job of $\mathcal{G}_A^m \cup \mathcal{G}_B^m$ during $[t, t+1)$.

(2) No job of $\mathcal{G}_{A^+}^m$ can be started during the interval $[s^{\mathcal{T}}, s_y^{\mathcal{A}} + p)$.

Proof (part A only). We use a basic counting argument to establish the lemma. We initially consider all elements of $\mathcal{R}$ to be unmarked. For each job $j \in \mathcal{G}_A$ started on m by OPT, we mark those jobs of $\mathcal{R}$ which are currently being processed at the time j starts. Because jobs have equal length, it is impossible for OPT to start two jobs on a single machine, both of which are started while a single job of $\mathcal{A}$ executes. Therefore, each job of $\mathcal{A}$ is marked at most once. By Lemma 6, if j were rejected by $\mathcal{A}$, two distinct jobs of $\mathcal{A}$ must be running at time s_j^{OPT}. Therefore two jobs of $\mathcal{R}$ are marked in association with j and $2 \cdot |\mathcal{G}_{A^-}^m|$ jobs are marked overall in association with rejected jobs. Any job of $\mathcal{G}_{A^+}^m$ must mark at least one further job of $\mathcal{R}$, since there is never a time during $[s,t)$ when both machines of $\mathcal{A}$ are idle. Therefore $|\mathcal{G}_{A^+}^m| \leq |\mathcal{R}| - 2 \cdot |\mathcal{G}_{A^-}^m|$ and thus $|\mathcal{G}_{A^+}^m| + 2 \cdot |\mathcal{G}_{A^-}^m| \leq |\mathcal{R}|$.

This establishes condition (A) in the case where $\mathcal{G}_B^m = \emptyset$. If $\mathcal{G}_B^m \neq \emptyset$, then by definition $\tilde{j}_s \in F_t^{\mathcal{A}} \setminus F_s^{\mathcal{A}}$ and $\tilde{j}_s \neq a$ for any $j \in \mathcal{G}_B^m$. Since $j \neq a$, this already requires that $|\mathcal{R}| \geq 2$. By Lemma 9, $x_{\tilde{j}_s} \leq t$ and thus $x_j \leq t$, as $j \prec \tilde{j}_s$ in the case when $j \in W_t^{\text{OPT}} \setminus P_t^{\text{OPT}}$. Such a job must then be started precisely at time t on m and so set $\mathcal{G}_B^m$ must consist of a single such element, if non-empty. Notice that if z was not marked by an element of $\mathcal{G}_A^m$, then $|\mathcal{G}_{A^+}^m| + 2 \cdot |\mathcal{G}_{A^-}^m| \leq |\mathcal{R}| - 1$ and so $|\mathcal{G}_B^m| + |\mathcal{G}_{A^+}^m| + 2 \cdot |\mathcal{G}_{A^-}^m| \leq |\mathcal{R}|$. If z is marked, the only way this can be accomplished is by a job which starts precisely at time $t-p$, given that $\tilde{j}_s$ must be started at time t. As $x_{\tilde{j}_s} \geq t \geq s_y^{\mathcal{A}} + p$, by combining Lemmas 10 and 12, we have that such a job starting at $t-p$ must have been a job accepted by $\mathcal{A}$. As this

accepted job marks both y and z, again we find that $|\mathcal{G}^m_{A^+}| + 2 \cdot |\mathcal{G}^m_{A^-}| \leq |\mathcal{R}| - 1$ and so $|\mathcal{G}^m_B| + |\mathcal{G}^m_{A^+}| + 2 \cdot |\mathcal{G}^m_{A^-}| \leq |\mathcal{R}|$. This establishes part (A) in general. □

Lemma 16. *For $u \in \mathcal{G}_+$, $\tilde{u}_s \in (F^{\mathcal{A}}_t \cup P^{\mathcal{A}}_t) \setminus (F^{\mathcal{A}}_s \cup P^{\mathcal{A}}_s)$.*

Lemma 17. *For distinct $u, v \in \mathcal{G}$, $\tilde{u}_s$ and $\tilde{v}_s$ are distinct.*

Lemma 18. *If $\tilde{a}_s \in Q^{\text{OPT}}_t$ for region $[s,t)$, then $\tilde{a}_s \in W^{\text{OPT}}_t$ and $a \notin P^{\mathcal{A}}_s$.*

Lemma 19. *For busy region $[s,t)$, assume that $\tilde{a}_s \in Q^{\text{OPT}}_t$ and there exists some $b \in P^{\mathcal{A}}_s \cup W^{\mathcal{A}}_t$. We may legally establish a match, $M_t(b) = \tilde{a}_s$.*

Lemma 20. *If $\Phi^{\text{OPT}}_s \leq \Phi^{\mathcal{A}}_s$ for busy region $[s,t)$, then $\Phi^{\text{OPT}}_t \leq \Phi^{\mathcal{A}}_t$.*

Proof (sketch). For ease of exposition, we introduce notation $\Phi^{\mathcal{A}}_{[s,t)}$ to denote $(\Phi^{\mathcal{A}}_t - \Phi^{\mathcal{A}}_s)$, and likewise $\Phi^{\text{OPT}}_{[s,t)} = (\Phi^{\text{OPT}}_t - \Phi^{\text{OPT}}_s)$. Our goal is to show that $\Phi^{\text{OPT}}_{[s,t)} \leq \Phi^{\mathcal{A}}_{[s,t)}$. In accordance with Lemmas 13 and 14, we rewrite $\Phi^{\text{OPT}}_{[s,t)}$ as,

$$\begin{aligned}
\Phi^{\text{OPT}}_{[s,t)} &= 2 \cdot |\mathcal{G}| = 2 \cdot |\mathcal{G}^{m_1}_{A^+} \cup \mathcal{G}^{m_2}_{A^+} \cup \mathcal{G}^{m_1}_{A^-} \cup \mathcal{G}^{m_2}_{A^-} \cup \mathcal{G}^{m_1}_B \cup \mathcal{G}^{m_2}_B \cup \mathcal{G}_C| \\
&= (|\mathcal{G}^{m_1}_B| + |\mathcal{G}^{m_1}_{A^+}| + 2 \cdot |\mathcal{G}^{m_1}_{A^-}|) + (|\mathcal{G}^{m_2}_B| + |\mathcal{G}^{m_2}_{A^+}| + 2 \cdot |\mathcal{G}^{m_2}_{A^-}|) + |\mathcal{G}_C| \\
&\quad + |\mathcal{G}^{m_1}_{A^+} \cup \mathcal{G}^{m_2}_{A^+} \cup \mathcal{G}^{m_1}_B \cup \mathcal{G}^{m_2}_B \cup \mathcal{G}_C| \\
&= (|\mathcal{G}^{m_1}_B| + |\mathcal{G}^{m_1}_{A^+}| + 2 \cdot |\mathcal{G}^{m_1}_{A^-}|) + (|\mathcal{G}^{m_2}_B| + |\mathcal{G}^{m_2}_{A^+}| + 2 \cdot |\mathcal{G}^{m_2}_{A^-}|) + |\mathcal{G}_C| + |\mathcal{G}_+|
\end{aligned}$$

By applying Lemma 15(A) to each of the two machines of OPT, we conclude that $\Phi^{\text{OPT}}_{[s,t)} \leq 2 \cdot |\mathcal{R}| + |\mathcal{G}_C| + |\mathcal{G}_+|$. In contrast, we claim that $\Phi^{\mathcal{A}}_{[s,t)} \geq 2 \cdot |\mathcal{R}| + \delta + |\mathcal{G}_+|$, where δ is defined as

$$\delta = \begin{cases} 1 & \text{if } P^{\mathcal{A}}_s = \emptyset \text{ and } P^{\mathcal{A}}_t \neq \emptyset \\ -1 & \text{if } P^{\mathcal{A}}_s \neq \emptyset \text{ and } P^{\mathcal{A}}_t = \emptyset \\ 0 & \text{otherwise} \end{cases}$$

The δ term adjusts for the discontinuity inherent in our definition of $\Phi^{\mathcal{A}}$, based upon whether or not set $P^{\mathcal{A}}$ is empty. With that aside, each job of $\mathcal{R}$ results in a relative contribution of at least 2, as such job is added to $F^{\mathcal{A}}$ though perhaps removed from $P^{\mathcal{A}}$. The only other way in which a job can be removed from $P^{\mathcal{A}}$ is if it becomes matched. We consider the creation of a new match in one particular case. If $\tilde{a}_s \in Q^{\text{OPT}}_t$ and there exists some $b \in P^{\mathcal{A}}_s \cup W^{\mathcal{A}}_t$, we create a match, $M_t(b) = \tilde{a}_s$, with the validity of such a match established as per Lemma 19. In this special case there is indeed a relative loss of one unit due to job b. However as $\tilde{a}_s \in W^{\text{OPT}}_t$, we show that $a \notin P^{\mathcal{A}}_s$. If it were, then a would be unmatched at time s, thus $\tilde{a}_s = a$, yet since $a \in Q^{\text{OPT}}_t$ it is not possible that $a \in P^{\mathcal{A}}_s \subseteq W^{\mathcal{A}}_s = Q^{\mathcal{A}}_s \setminus Q^{\text{OPT}}_t$. Given that $a \notin P^{\mathcal{A}}_s$, its presence in $\mathcal{R}$ results in a profit of 3 rather than the minimally assumed profit of 2 and so this offsets the loss of 1 due to the matching of b.

Next, we consider the impact of set $\mathcal{G}_+$ on $\Phi^{\mathcal{A}}_{[s,t)}$. By Lemma 16, for any $u \in \mathcal{G}_+$ there exists $\tilde{u}_s \in (F^{\mathcal{A}}_t \cup P^{\mathcal{A}}_t) \setminus (F^{\mathcal{A}}_s \cup P^{\mathcal{A}}_s)$ and by Lemma 17, those associated jobs are distinct. Each such job provides an additional point towards $\Phi^{\mathcal{A}}_{[s,t)}$ beyond

the presumed $2 \cdot |\mathcal{R}|$, either providing 1 if in $P_t^{\mathcal{A}}$, or providing 3 rather than 2 if in $F_t^{\mathcal{A}}$. It is important to note that these $|\mathcal{G}_+|$ additional points are also distinct from the possible point associated with a in the preceding paragraph, as in that case neither a nor $\tilde{a}_s$ can lie in $\mathcal{G}_+$. We have thus far established that $\Phi^{\mathcal{A}}_{[s,t)} \geq 2 \cdot |\mathcal{R}| + \delta + |\mathcal{G}_+|$.

For the remainder of the proof, we focus on the expression $(\delta - |\mathcal{G}_C|)$, recalling that $|\mathcal{G}_C|$ is either 0 or 1. In the case that $(\delta - |\mathcal{G}_C|) \geq 0$ the theorem follows, as $\Phi^{\text{OPT}}_{[s,t)} \leq 2 \cdot |\mathcal{R}| + |\mathcal{G}_C| + |\mathcal{G}_+| \leq 2 \cdot |\mathcal{R}| + \delta + |\mathcal{G}_+| \leq \Phi^{\mathcal{A}}_{[s,t)}$.

If $(\delta - |\mathcal{G}_C|) < 0$ we undertake a detailed case analysis to counterbalance this deficit either by showing a gap in our upper bound on $\Phi^{\text{OPT}}_{[s,t)}$ due to a strict inequality when applying Lemma 15(A) to one or both of the two machines of OPT, or by showing a gap in our lower bound on $\Phi^{\mathcal{A}}_{[s,t)}$. Details are omitted here. □

Theorem 1. *Algorithm $\mathcal{A}$ is $\frac{3}{2}$-competitive.*

Proof. We show that $\Phi^{\text{OPT}}_t \leq \Phi^{\mathcal{A}}_t$ by induction. Initially, $\Phi^{\text{OPT}}_0 = \Phi^{\mathcal{A}}_0 = 0$. Neither potential function changes during regions for which $\mathcal{A}$ remains completely idle, as per Lemma 5. The times during which $\mathcal{A}$ uses one or more machines can be partitioned into regions $[s,t)$, for which Lemma 20 applies, thereby extending the induction from time s to time t for each such region. We conclude that $\Phi^{\text{OPT}}_\infty \leq \Phi^{\mathcal{A}}_\infty$. Since the queues of both $\mathcal{A}$ and OPT are presumed to be empty at $t = \infty$, $P^{\mathcal{A}}_\infty = P^{\text{OPT}}_\infty = \emptyset$. Therefore, $\Phi^{\text{OPT}}_\infty \leq \Phi^{\mathcal{A}}_\infty$ is equivalent to $2 \cdot |F^{\text{OPT}}_\infty| \leq 3 \cdot |F^{\mathcal{A}}_\infty|$, thus $\frac{|\text{OPT}|}{|\mathcal{A}|} \leq \frac{3}{2}$. □

Theorem 2. *For $m = 2$, no non-preemptive, deterministic algorithm can be better than $\frac{3}{2}$-competitive.*

Proof. Consider the release of a single job $j = \langle 0, 3p - 1 \rangle$. A deterministic algorithm must start j at some time $0 \leq t \leq 2p - 1$, or else have unbounded competitiveness. Yet if two identical jobs with parameters $\langle t + 1, t + 1 + p \rangle$ are subsequently released, one must be rejected. It is easy to verify that OPT can achieve all three jobs. □

Acknowledgments

We gratefully acknowledge the correspondence of Jiří Sgall.

References

1. P. Baptiste, P. Brucker, S. Knust, and V. G. Timkovsky. Ten notes on equal-processing-time scheduling. *4OR: Quarterly J. Belgian, French and Italian Operations Research Societies*, 2(2):111–127, July 2004.
2. S. K. Baruah, J. R. Haritsa, and N. Sharma. On-line scheduling to maximize task completions. *J. Combin. Math. and Combin. Computing*, 39:65–78, 2001.

3. A. Borodin and R. El-Yaniv. *Online Computation and Competitive Analysis.* Cambridge University Press, New York, 1998.
4. M. Chrobak, W. Jawor, J. Sgall, and T. Tichý. Online scheduling of equal-length jobs: Randomization and restarts help. In J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, editors, *Proc. 31st Int. Colloquium on Automata, Languages and Programming (ICALP)*, volume 3142 of *Lecture Notes in Computer Science*, pages 358–370, Turku, Finland, July 2004. Springer-Verlag.
5. J. Ding and G. Zhang. Online scheduling with hard deadlines on parallel machines. In *Proc. Second Int. Conference on Algorithmic Aspects in Information and Management*, Lecture Notes in Computer Science, Hong Kong, China, June 2006. Springer-Verlag. To appear.
6. S. Goldman, J. Parwatikar, and S. Suri. On-line scheduling with hard deadlines. *J. Algorithms*, 34(2):370–389, Feb. 2000.
7. M. H. Goldwasser. Patience is a virtue: The effect of slack on competitiveness for admission control. *J. Scheduling*, 6(2):183–211, Mar./Apr. 2003.
8. M. H. Goldwasser and B. Kerbikov. Admission control with immediate notification. *J. Scheduling*, 6(3):269–285, May/June 2003.
9. J. R. Jackson. Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles, Jan. 1955.
10. B. B. Simons. Multiprocessor scheduling of unit length jobs with arbitrary release times and deadlines. *SIAM J. Comput.*, 12:294–299, 1983.
11. B. B. Simons and M. K. Warmuth. A fast algorithm for multiprocessor scheduling of unit-length jobs. *SIAM J. Comput.*, 18(4):690–710, 1989.