Online Non-preemptive Scheduling of Equal-length Jobs on Two Identical Machines

MICHAEL H. GOLDWASSER and MARK PEDIGO
Saint Louis University

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.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*sequencing and scheduling*; F.2.1 [**Computation by Abstract Devices**]: Modes of Computation—*online computation*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Admission control, competitive analysis, 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 nonnegative 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 in which a 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 [Borodin and El-Yaniv 1998]. In the context of this maximization problem, we say that an online (deterministic) algorithm is c-competitive so long

This material is based upon work supported by the National Science Foundation under Grant No. CCR-0417368.
Authors' address: M. Goldwasser and M Pedigo, Department of Mathematics and Computer Science, Saint Louis University, 221 North Grand Blvd., St. Louis, MO 63103-2007.
A preliminary version of this work appeared in *Proceedings of the 10th Scandinavian Workshop on Algorithm Theory (SWAT)*, Riga, Latvia, July 2006, pp. 113–123.

as the number of jobs achieved by the optimal schedule is at most c times that of the online algorithm for any given problem instance. Our main result is the presentation of a $\frac{3}{2}$-competitive online algorithm for the two-machine problem. A simple lower bound shows that this is the best possible deterministic result.

1.1 Related Work

For the single-machine version of the problem, a deterministic non-preemptive greedy algorithm is known to be 2-competitive, and this is the best possible deterministic result [Baruah et al. 2001; Goldman et al. 2000]. 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 optimally $(1 + \frac{1}{\lfloor \kappa \rfloor + 1})$-competitive [Goldwasser 2003]. A similar algorithm by Goldwasser and Kerbikov [2003] achieves the same competitiveness while providing *immediate notification* (defined below). For randomized online algorithms, there exists a $\frac{5}{3}$-competitive algorithm [Chrobak et al. 2004] versus a $\frac{4}{3}$-competitive lower bound [Goldman et al. 2000].

For the two-machine version of the problem, Ding and Zhang [2006] claim a $\frac{3}{2}$-competitive bound in recent independent work. However, their algorithm description and analysis are significantly more complex than ours. For three or more machines, they demonstrate that the algorithm and analysis of Goldman et al. generalizes to a 2-competitive algorithm for an arbitrary number of machines. They also provide a deterministic lower-bound that approaches $\frac{6}{5}$ for large m.

In the offline version of the problem, 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 [Simons 1983; Simons and Warmuth 1989]. The optimization version of the problem is polynomially solvable for any *fixed* number of machines, even with weighted utility (i.e., $Pm \mid p_j = p, r_j \mid \sum w_j U_j$) [Baptiste et al. 2004]. The status of the problem is open for an arbitrary number of machines, even in the unweighed case (i.e., $P \mid p_j = p, r_j \mid \sum U_j$).

1.2 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 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* [Goldwasser and Kerbikov 2003]. 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 that 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 $\textsc{Feasible}(\mathcal{J}, t_1, t_2)$ 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, although 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.

2. ALGORITHM DEFINITION

Our algorithm $\mathcal{A}$ maintains a queue of jobs that 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 $Q_{t|j}^{\mathcal{A}}$ denote the queue as it exists when j's release is considered (thus $Q_{t|j}^{\mathcal{A}} \supseteq Q_t^{\mathcal{A}}$ may contain newly accepted jobs that were considered prior to j). Job j is accepted into the system precisely if it is feasible to do so. Specifically, we check $\textsc{Feasible}(Q_{t|j}^{\mathcal{A}} \cup \{j\}, c, \dot{c})$, 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.

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 nonempty, 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 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.

To reinforce the algorithm definition, we consider a trace of a small example. We let $p = 10$ and consider an instance with jobs having respective release times and deadlines as follows:

$$a = \langle 0, 60\rangle \quad b = \langle 0, 71\rangle \quad c = \langle 0, 71\rangle \quad d = \langle 3, 30\rangle \quad e = \langle 3, 31\rangle \quad f = \langle 3, 33\rangle$$
$$g = \langle 3, 37\rangle \quad h = \langle 3, 45\rangle \quad i = \langle 3, 52\rangle \quad j = \langle 3, 56\rangle \quad k = \langle 38, 55\rangle$$

At time 0 jobs a, b, and c are accepted. a is started on the first machine, but the second machine remains idle as $\textsc{Feasible}(\{b, c\}, 10, 11)$ is easily satisfied. At time 3 the set of newly released jobs is considered, and each one is accepted as feasible. At this point, the queue of waiting jobs is $\{d, e, f, g, h, i, b, c\}$ as per $\prec$-order. Since one machine is committed to a until time 10, we face a secondary decision as to whether to use the other machine. The condition $\textsc{Feasible}(\dot{Q}_3^{\mathcal{A}}, 10, 14)$ is true as evidenced by the schedule in Figure 1, so the other machine is not used at this time. By similar reasoning, the secondary machine is left idle at time-steps 4, 5 and 6. Yet at time 7, $\textsc{Feasible}(\dot{Q}_7^{\mathcal{A}}, 10, 18)$ fails, as there are four jobs $\{d, e, f, g\}$ all of which must be completed by time 37. Therefore, the secondary machine is committed at time 7 to the $\prec$-minimal job, namely d.

The next decision occurs at time 10 when job a completes. Since job d is still running, the decision of whether to immediately start another job is viewed as a secondary decision. In this case, the $\textsc{Feasible}(\dot{Q}_{10}^{\mathcal{A}}, 17, 21)$ condition is true as evidenced by the schedule in Figure 2. It is not until time 13 when the rules again dictate the start of a secondary job. While we do not detail the rest of the process for this example, the resulting schedule is shown in Figure 3.

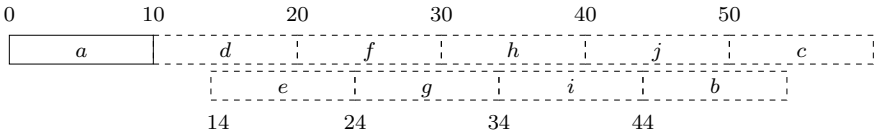

Fig. 1. A potential schedule for $\dot{Q}_3^{\mathcal{A}} = \{d, e, f, g, h, i, j, b, c\}$ that demonstrates that FEASIBLE($\dot{Q}_3^{\mathcal{A}}, 10, 14$) is true. A similar schedule validates FEASIBLE($\dot{Q}_6^{\mathcal{A}}, 10, 17$), yet fails for FEASIBLE($\dot{Q}_7^{\mathcal{A}}, 10, 18$) as g would miss its deadline of 37.

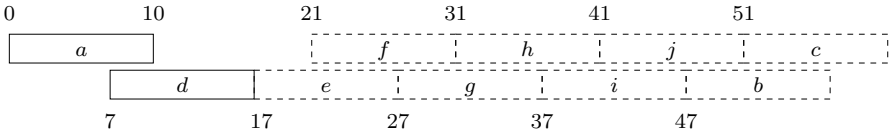

Fig. 2. Hypothetical schedule for $\dot{Q}_{10}^{\mathcal{A}} = \{e, f, g, h, i, j, b, c\}$, confirming FEASIBLE($\dot{Q}_{10}^{\mathcal{A}}, 17, 21$) as part of the secondary decision at time 10. A similar schedule validates FEASIBLE($\dot{Q}_{12}^{\mathcal{A}}, 10, 23$), yet fails for FEASIBLE($\dot{Q}_{13}^{\mathcal{A}}, 10, 24$) as f would miss its deadline of 33.

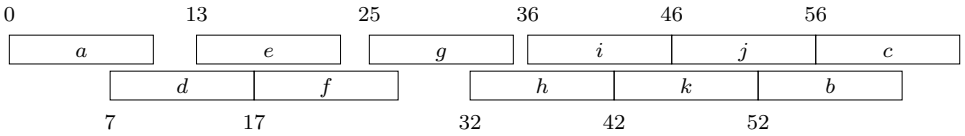

Fig. 3. The complete schedule produced by $\mathcal{A}$ for our sample instance.

3. A SUPPLEMENTAL FEASIBILITY TEST

In this section, we discuss the feasibility test FEASIBLE($\mathcal{J}, t_1, t_2$). Such a condition is satisfied if and only if the jobs can be achieved when scheduled according to an earliest deadline first (EDF) rule. A classic result of Jackson [1955] proves this for a set of jobs with arbitrary processing times and deadlines when scheduled on a single machine. 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 time t_i on 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 in the context of FEASIBLE($\mathcal{J}, t_1, t_2$).

LEMMA 3.1. *For sets $\mathcal{J} \subseteq \mathcal{J}'$ and times $t_1 \leq t_1'$ and $t_2 \leq t_2'$,* FEASIBLE($\mathcal{J}', t_1', t_2'$) *implies* FEASIBLE($\mathcal{J}, t_1, t_2$).

PROOF. Such monotonicity is obvious, as $\mathcal{J}$ can be scheduled using the relevant portion of the presumed feasible schedule for superset $\mathcal{J}'$. □

LEMMA 3.2. *For arbitrary set $\mathcal{J}$ and times $t_1 \leq t_2$, let j be the $\prec$-minimal job of $\mathcal{J}$.* FEASIBLE($\mathcal{J}, t_1, t_2$) *if and only if $x_j \geq t_1$ and* FEASIBLE($\mathcal{J} \setminus \{j\}, t_1 + p, t_2$).

PROOF. Immediate, as an EDF schedule witnessing FEASIBLE($\mathcal{J}, t_1, t_2$) starts j at time t_1 with $\mathcal{J} \setminus \{j\}$ to follow. □

LEMMA 3.3. *If* FEASIBLE$(\mathcal{J}, t, t+p+1)$ *yet not* FEASIBLE$(\mathcal{J}, t+1, t+p)$*, then there must exist a job* $j \in \mathcal{J}$ *with* $x_j = t$*.*

PROOF. Let j be the $\prec$-minimal job of $\mathcal{J}$. We assume FEASIBLE$(\mathcal{J}, t, t+p+1)$. Applying Lemma 3.2 to t and $t+p+1$ implies FEASIBLE$(\mathcal{J} \setminus \{j\}, t+p, t+p+1)$ and $x_j \geq t$. For the sake of contradiction, we consider the possibility that $x_j \geq t+1$. In that case, the fact that FEASIBLE$(\mathcal{J} \setminus \{j\}, t+p+1, t+p)$, together with an application of Lemma 3.2 to $t+1$ and $t+p$, implies FEASIBLE$(\mathcal{J}, t+1, t+p)$. This contradicts the conditions of our claim and therefore we conclude that $x_j = t$. □

4. STRUCTURAL PROPERTIES OF THE ALGORITHM'S SCHEDULE

LEMMA 4.1. *If there exists a time* t *at which at least one machine of* $\mathcal{A}$ *is idle and a job* j *such that* $r_j \leq t \leq x_j$*, then* j *cannot be rejected by* $\mathcal{A}$*.*

PROOF. For j to be rejected, condition FEASIBLE$(Q^{\mathcal{A}}_{r_j|j} \cup \{j\}, c, \dot{c})$ must fail at time r_j. For all subsequent times, jobs of $Q^{\mathcal{A}}_{r_j|j}$ must either be completed, running, or remaining in $Q^{\mathcal{A}}$. Since we assume that at least one machine idles at time t, let c_t denote the time until which the other machine is committed at that time. The idleness in the schedule implies FEASIBLE$(\dot{Q}^{\mathcal{A}}_t, c_t, t+p+1)$. Since j can be scheduled over the interval $[t, t+p)$, this implies FEASIBLE$(\dot{Q}^{\mathcal{A}}_t \cup \{j\}, c_t, t)$. Since $c_t \geq \max(c, \dot{c})$ and $t \geq \min(c, \dot{c})$, the monotonicity of Lemma 3.1 contradicts the assumed failure of FEASIBLE$(Q^{\mathcal{A}}_{r_j|j} \cup \{j\}, c, \dot{c})$. □

We analyze a schedule produced by algorithm $\mathcal{A}$ on a fixed instance of the problem by partitioning time into consecutive regions of the form $[u, v)$ defined as follows. We consider two types of regions: idle and busy. If both machines are idle at time u, we designate idle region $[u, v)$ where v is the next time at which either machine starts a job, if any. For a time u at which $\mathcal{A}$ starts executing a job, we designate a busy region $[u, v)$ as follows. We define $v > u$ as the first subsequent time when both machines are idle or when one machine starts a job yet the other remains idle (that is, for at least one unit).

For ease of exposition throughout the analysis of a busy region $[u, v)$, 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 u. We let z denote the last job of $\mathcal{R}$ to be started, namely at time $v - p$. 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 v. Thus $s^{\mathcal{A}}_y \leq s^{\mathcal{A}}_z = v - p < s^{\mathcal{A}}_y + p$, where notation $s^{\mathcal{A}}_j$ denotes the time at which job j is started by algorithm $\mathcal{A}$.

For $|\mathcal{R}| \geq 2$, we define the *tail of the region*, denoted as $\mathcal{T}$, as follows. Let $u^{\mathcal{T}}$ be the latest time at which $\mathcal{A}$ starts a job of $\mathcal{R}$ following a time-step $[u^{\mathcal{T}} - 1, u^{\mathcal{T}})$ at which a machine was idle. We note that $u^{\mathcal{T}}$ 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 u, then it suffices. Alternatively, the region begins with two jobs starting precisely at time u. Such a region could only follow an idle region,

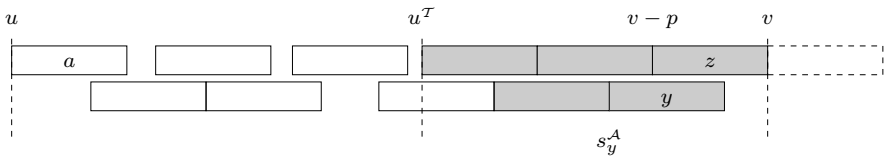

Fig. 4. Typical configuration of algorithm's two machines during a region $[u, v)$. In this example, $|\mathcal{R}| = 11$ and the tail is shaded, with $|\mathcal{T}| = 5$.

as a previous busy region could not have ended at such a time u. Having defined $u^{\mathcal{T}}$, tail $\mathcal{T} \subseteq \mathcal{R}$ is the set of jobs started by $\mathcal{A}$ during the interval $[u^{\mathcal{T}}, v)$. Figure 4 demonstrates a typical region.

LEMMA 4.2. *For a busy region* $[u, v)$, FEASIBLE$(Q_v^{\mathcal{A}}, v, v + p + 1)$.

PROOF. By the definition of a region, neither machine was committed to a job at the onset of time v and at least one remains idle during the time $[v, v+1)$. If $Q_v^{\mathcal{A}} = \emptyset$, the lemma is trivially true. Alternatively, some job j starts on one machine at v, while the secondary decision is to remain idle. Based on the algorithm definition, it must be that the test FEASIBLE$(\dot{Q}_v^{\mathcal{A}}, v + p, v + p + 1)$ succeeded and that j is $\prec$-minimal for $\dot{Q}_v^{\mathcal{A}} \cup \{j\}$. By Lemma 3.2, FEASIBLE$(\dot{Q}_v^{\mathcal{A}} \cup \{j\}, v, v + p + 1)$. Since $Q_v^{\mathcal{A}} \subseteq \dot{Q}_v^{\mathcal{A}} \cup \{j\}$, Lemma 3.1 implies FEASIBLE$(Q_v^{\mathcal{A}}, v, v + p + 1)$. □

LEMMA 4.3. *For a busy region* $[u, v)$ *and an arbitrary time* t*, consider the subset of* $\mathcal{R} \setminus \{a\}$ *consisting of jobs that expire on or after time* t*. If this subset is nonempty, let* j *denote the last of those elements to be started by* $\mathcal{A}$*. This* j *must start due to a secondary decision and* $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\} \subseteq Q_v^{\mathcal{A}}$.

PROOF. First we show that j must be started as part of a secondary decision. Had $j \neq a$ been started through a primary decision, the other machine could not remain idle by the definition of our region, and thus some other job would have been started. Yet since j is $\prec$-minimal at the time it is started and that other job was in the queue, that other job expires at least as late as j, contradicting our assumption that j was the latest such job to be started. We conclude that j is started through a secondary decision.

By the definition of j, all subsequently started jobs in the region expire strictly before time t and thus precede j as per $\prec$. Since j was $\prec$-minimal when started, jobs of $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\}$ cannot be subsequently started in the region, thus $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\} \subseteq Q_v^{\mathcal{A}}$. □

LEMMA 4.4. *For busy region* $[u, v)$*, any job* $j \in \mathcal{R} \setminus \{a\}$ *satisfies* $x_j \leq v$.

PROOF. For contradiction, let j with $x_j \geq v + 1$ be the latest such job to be started by $\mathcal{A}$. Applying Lemma 4.3 with $t = v+1$ implies that j is started through a secondary decision and $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\} \subseteq Q_v^{\mathcal{A}}$. By Lemma 4.2, FEASIBLE$(Q_v^{\mathcal{A}}, v, v + p + 1)$ and thus FEASIBLE$(\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\}, v, v + p + 1)$. Since $x_j \geq v + 1$, we can schedule j over the interval $[v+1, v+p+1)$ while still achieving $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\}$ starting the machines respectively at v and $v + p + 1$. Therefore FEASIBLE$(\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}}, v, v + 1)$. Since $v \geq c$, where c denotes the time until which the opposite machine is committed as j starts,

and $v \geq s_j^{\mathcal{A}} + p$, Lemma 3.1 implies FEASIBLE($\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}}, c, s_j^{\mathcal{A}} + p + 1$). This contradicts the fact that $\mathcal{A}$ starts j with a secondary decision. □

LEMMA 4.5. *For a busy region $[u, v)$, if $\exists j \in \mathcal{R} \setminus \{a\}$ with $x_j \geq s_y^{\mathcal{A}} + p$, then $x_z \geq x_j$.*

PROOF. Let $t = \max_{j \in \mathcal{R} \setminus \{a\}} x_j$. We henceforth let $j \in \mathcal{R} \setminus \{a\}$ with $x_j = t$ denote the latest such job to be started by $\mathcal{A}$. If $j = z$, the result is trivial. Otherwise by applying Lemma 4.3 to t, we have that j is started via a secondary decision and $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \setminus \{j\} \subseteq Q_v^{\mathcal{A}}$. Lemma 4.2 assures that FEASIBLE($Q_v^{\mathcal{A}}, v, v + p + 1$). Since $\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}} \subseteq Q_v^{\mathcal{A}} \cup \{j\}$, we claim that FEASIBLE($\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}}, v - p, v + p + 1$). This is witnessed by taking the relevant portion of the witness for $Q_v^{\mathcal{A}}$ together with j over the region $[v-p, v)$; note that $r_j \leq s_j^{\mathcal{A}} \leq v - p < s_y^{\mathcal{A}} + p \leq x_j$. Let c denote the time until which the opposite machine was committed when the secondary decision for $s_j^{\mathcal{A}}$ was made. Since $j \neq z$, $c \leq v - p$. Therefore Lemma 3.1 implies FEASIBLE($\dot{Q}^{\mathcal{A}}_{s_j^{\mathcal{A}}}, c, s_j^{\mathcal{A}} + p + 1$) given that $s_j^{\mathcal{A}} \leq v$. This contradicts the secondary decision to start j at $s_j^{\mathcal{A}}$. □

LEMMA 4.6. *For a busy region $[u, v)$ with $|\mathcal{R}| \geq 2$, we consider the tail $\mathcal{T}$. If all jobs of $\mathcal{T}$ are released on or before time $u^{\mathcal{T}} - 1$, then $x_z \geq s_y^{\mathcal{A}} + p$.*

PROOF. Since jobs of $\mathcal{T}$ were released by time $u^{\mathcal{T}} - 1$ yet the tail begins at time $u^{\mathcal{T}}$, one machine must be busy at time $u^{\mathcal{T}} - 1$ while the other machine was left idle through a secondary decision. If we let c denote the time until which the first machine was committed at that time, it must be that FEASIBLE($\dot{Q}^{\mathcal{A}}_{u^{\mathcal{T}}-1}, c, u^{\mathcal{T}} + p$). Notice that both the tail and the witnessing schedule for this feasibility execute jobs of $\mathcal{T}$ in $\prec$-minimal order. Furthermore, these two scenarios rely on precisely the same starting times with the exception of the tail's use of $u^{\mathcal{T}}$ replaced by a start time of $s_y^{\mathcal{A}} + p$; for illustration, the five jobs of the tail from Figure 4 could have hypothetically been scheduled starting at $c, u^{\mathcal{T}} + p, \ldots, s_y^{\mathcal{A}} + p$. Since z is the last job of $\mathcal{T}$, the secondary decision implies $x_z \geq s_y^{\mathcal{A}} + p$. □

LEMMA 4.7. *For a busy region $[u, v)$ with $|\mathcal{R}| \geq 2$ and $x_z \geq s_y^{\mathcal{A}} + p$, each of the following are true.*

(i) Any job k with $r_k \leq v - p \leq x_k$ must have been accepted by $\mathcal{A}$.

(ii) There exists job $j \in Q_v^{\mathcal{A}}$ such that $s_j^{\mathcal{A}} = x_j = v$.

PROOF. By Lemma 4.2, FEASIBLE($Q_v^{\mathcal{A}}, v, v + p + 1$). We also note that $s_y^{\mathcal{A}} + 2p \leq v + p$, given that $s_y^{\mathcal{A}} \leq s_z^{\mathcal{A}} = v - p$.

Assume for contradiction to (i) that $\mathcal{A}$ rejects a job k with $r_k \leq v - p \leq x_k$. The rejection of k upon its release implies the infeasibility of achieving set $Q^{\mathcal{A}}_{r_k|k} \cup \{k\}$, pending the completion of any jobs running at that time. Moving to time $v - p$, when z was started by $\mathcal{A}$, we note that all jobs of $Q^{\mathcal{A}}_{r_k|k}$ have either been started or remain in the current queue. Therefore, the earlier infeasibility involving the addition of job k must imply the infeasibility of achieving k together with the current queue. Yet we can feasibly schedule k in place of z starting at $v - p$, z following y over the interval $[s_y^{\mathcal{A}} + p, s_y^{\mathcal{A}} + 2p)$, and the remaining $Q_v^{\mathcal{A}}$ scheduled on two machines starting at times v and $v + p + 1 \geq s_y^{\mathcal{A}} + 2p$. This contradicts the existence of any such k, thereby proving result (i).

By applying Lemma 4.3 with $t = x_z$, we see that z must be started through a secondary decision at $v - p$. By definition, y was the job executing on the opposite machine at that time. The start of z implies that FEASIBLE($\dot{Q}^{\mathcal{A}}_{v-p}, s^{\mathcal{A}}_y + p, v + 1$) fails. By Lemma 3.2, FEASIBLE($\dot{Q}^{\mathcal{A}}_{v-p}, s^{\mathcal{A}}_y + p, v + 1$) is equivalent to the compound condition $x_z \geq s^{\mathcal{A}}_y + p$ and FEASIBLE($\dot{Q}^{\mathcal{A}}_{v-p} \setminus \{z\}, v + 1, s^{\mathcal{A}}_y + 2p$). Since we have assumed that $x_z \geq s^{\mathcal{A}}_y + p$ it must be that FEASIBLE($\dot{Q}^{\mathcal{A}}_{v-p} \setminus \{z\}, v + 1, s^{\mathcal{A}}_y + 2p$) fails. This further implies the failure of FEASIBLE($Q^{\mathcal{A}}_v, v + 1, v + p$), in accordance with Lemma 3.1 given that that $\dot{Q}^{\mathcal{A}}_{v-p} \setminus \{z\} \subseteq Q^{\mathcal{A}}_v$ and $s^{\mathcal{A}}_y + 2p \leq v + p$. As FEASIBLE($Q^{\mathcal{A}}_v, v, v + p + 1$), Lemma 3.3 implies the existence of $j \in Q^{\mathcal{A}}_v$ with $x_j = v$, and thus $s^{\mathcal{A}}_j = v$ as $\mathcal{A}$ completes this job. □

5. COMPETITIVE ANALYSIS

We compare the performance of algorithm $\mathcal{A}$ on a given instance to that of an optimal schedule denoted as OPT. To compare the progress of the two schedules over time, we introduce some additional notations. We let s^{OPT}_j denote the time at which OPT starts some job j, just as $s^{\mathcal{A}}_j$ represents a job's starting time in $\mathcal{A}$. We let $F^{\mathcal{A}}_t$ (resp. F^{OPT}_t) designate the set of jobs started *strictly before* time t by $\mathcal{A}$ (resp. OPT). Notice that by this definition, $\mathcal{R} = F^{\mathcal{A}}_v \setminus F^{\mathcal{A}}_u$ for region $[u, v)$.

Although OPT is not constructed in online fashion, we introduce a formal notation of a presumed queue Q^{OPT}_t consisting of all jobs that are released strictly before time t yet started by OPT on or after time t. A job that is simultaneously in both queues will be ignored by our analysis until a time when it is started in one or both of the schedules. Yet we wish to account for jobs that are waiting in one queue but not in the other (presumably because they were already started or never accepted). In this regard, we define $W^{\mathcal{A}}_t = Q^{\mathcal{A}}_t \setminus Q^{\text{OPT}}_t$, and symmetrically $W^{\text{OPT}}_t = Q^{\text{OPT}}_t \setminus Q^{\mathcal{A}}_t$.

We must introduce one additional technical definition to account for a certain anomaly that may arise. In some cases, when identifying a job waiting in $W^{\mathcal{A}}$ that 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 immediate analysis. Intuitively, such a pairing can only be to the advantage of the algorithm. To track such scenarios over time, we constructively define a function $M_t(j)$ representing a partial matching so that $M_t(j) = j$ for most jobs but with $M_t(j) = k$ and $M_t(k) = j$ for jobs that we want offset in the analysis. The precise rules for this function will be established later in this section. We introduce notation $P^{\mathcal{A}}_t$ and P^{OPT}_t to identify respective subsets of the waiting jobs that do not participate in the matching. Namely, we let $P^{\mathcal{A}}_t = \{j \in W^{\mathcal{A}}_t : M_t(j) = j\}$. Similarly, we let $P^{\text{OPT}}_t = \{j \in W^{\text{OPT}}_t : M_t(j) = j\}$.

Our competitive analysis is based upon the following two functions that measure the quality of the schedules being developed as of time t by $\mathcal{A}$ and OPT respectively.

$$\Phi^{\mathcal{A}}_t = \begin{cases} 3 \cdot |F^{\mathcal{A}}_t| & \text{if } P^{\mathcal{A}}_t = \emptyset \\ 3 \cdot |F^{\mathcal{A}}_t| + 1 \cdot |P^{\mathcal{A}}_t| + 1 & \text{if } P^{\mathcal{A}}_t \neq \emptyset \end{cases}$$

$$\Phi^{\text{OPT}}_t = 2 \cdot |F^{\text{OPT}}_t \cup P^{\text{OPT}}_t|$$

We view these functions as payment to the respective schedules, with a handicap given to the online algorithm. In the end, since both queues are empty we have that $\Phi^{\text{OPT}}_\infty = 2 \cdot |F^{\text{OPT}}_t|$ and $\Phi^{\mathcal{A}}_\infty = 3 \cdot |F^{\mathcal{A}}_t|$. By showing that $\Phi^{\text{OPT}}_\infty \leq \Phi^{\mathcal{A}}_\infty$ we will have

proven the $\frac{3}{2}$-competitive ratio. 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 perform analysis based upon the regions $[u, v)$, as defined in Section 4, showing that $\Phi_v^{\text{OPT}} \leq \Phi_v^{\mathcal{A}}$ at the end of each such region.

Intuition. Our functions represent a combination of full payment for jobs that have already been started (i.e., $F^{\mathcal{A}}$ and F^{OPT}) and supplemental payments for the potential of particular jobs in the respective queues (i.e., $P^{\mathcal{A}}$ and P^{OPT}). In the case of Φ_t^{OPT}, we *fully* credit the potential jobs in advance. These typically result from a scenario in which $\mathcal{A}$ begins a region with a job that has a far away deadline, while OPT ignores that job momentarily to wait for new arrivals. We essentially concede that OPT will be able to complete such jobs at a later time and so we credit them proactively.

The potential function $\Phi_t^{\mathcal{A}}$ is defined more subtly. In effect, we take one advance credit for a job in $P_t^{\mathcal{A}}$, receiving the remaining two credits once that job is started and thus moved from $P^{\mathcal{A}}$ to $F^{\mathcal{A}}$. Because $P^{\mathcal{A}}$ represent jobs that $\mathcal{A}$ still needs to complete yet which OPT may have already started, the two credits can be used to offset the possibility that OPT gets to accept a future job that $\mathcal{A}$ rejects due to the existing commitment.

The additional $+1$ term in the case when $P_t^{\mathcal{A}} \neq \emptyset$ is necessary to balance two boundary cases. We motivate it by considering an instance with $p = 10$, beginning with three jobs $a = \langle 0, 1000\rangle$, $b = \langle 1, 30\rangle$, and $c = \langle 1, 30\rangle$. Facing this instance at time 0, $\mathcal{A}$ would being executing job a. It would accept both b and c into the queue at time 1 yet since there is ample flexibility without any further arrivals, b would be started at time 10 when a completes and the secondary machine would remain idle. By our regional analysis, the first region is defined simply as $[0, 10)$.

Depending upon the composition of the full instance, an optimal schedule might wait to start both b and c on the two machines at time 1, with a achieved at some later time. By our regional analysis $\Phi_{10}^{\text{OPT}} = 6$, as OPT receives full credit for b, c, and $a \in P_{10}^{\text{OPT}}$. In contrast, $\mathcal{A}$ would receive 3 for starting a yet only 1 each for b and c which are added to $P_{10}^{\mathcal{A}}$. In this case, since $P^{\mathcal{A}}$ has changed from an empty set to a nonempty set, the algorithm receives the additional $+1$ term bringing $P_{10}^{\mathcal{A}} = 6 = P_{10}^{\text{OPT}}$.

In the converse, presume hypothetically that the algorithm has built up a surplus of 10 jobs in $P^{\mathcal{A}}$. When it later schedules those jobs it only receives a reserve of 19 credits rather than 20 because eventually $P^{\mathcal{A}}$ returns to an empty set. We earlier stated that the 2 credits per job were to offset jobs that $\mathcal{A}$ might have to reject due to being backlogged. If the algorithm were critically backlogged with no remaining flexibility, then all such credits would be needed. However, the algorithm is defined to start a secondary job slightly before losing all flexibility. For this reason, an adversary cannot fully exploit the last remaining job in $P^{\mathcal{A}}$.

5.1 Matching Function

Because our analysis will proceed on a region-by-region basis, we are only concerned with the value of function $M_t(\cdot)$ at regional boundaries. Initially, $M_0(j) = j$ for all jobs, even those not yet released. For a region $[u, v)$, we constructively define $M_v(\cdot) = M_u(\cdot)$ by default. However, there are two notable variations.

Breaking a match. For any $j \neq k$ for which $M_u(j) = k$, if either j or k is started by $\mathcal{A}$ or OPT during $[u, v)$, we break the match by assigning $M_v(j) \leftarrow j$ and $M_v(k) \leftarrow k$.

Creating a match. Recall that a denotes the first job of the region started by $\mathcal{A}$. If $M_u(a) \in Q_v^{\text{OPT}}$ and $P_u^{\mathcal{A}} \cap W_v^{\mathcal{A}} \neq \emptyset$, we select an arbitrary $b \in P_u^{\mathcal{A}} \cap W_v^{\mathcal{A}}$ and establish a match between $M_u(a)$ and b by assigning $M_v(M_u(a)) \leftarrow b$ and $M_v(b) \leftarrow M_u(a)$. We note that when $M_u(a) \neq a$, the match-breaking rule dictates that $M_v(M_u(a)) \leftarrow M_u(a)$, given that a is started by $\mathcal{A}$. In this case, we let the match-creation assignment $M_v(M_u(a)) \leftarrow b$ prevail. Note that b could not have been simultaneously involved in a broken match as $b \in P_u^{\mathcal{A}}$.

LEMMA 5.1. *For an arbitrary region boundary t, if $M_t(j) = k \neq j$, then*

(i) $M_t(k) = j$

(ii) if we assume w.l.o.g that $j \prec k$, then $j \in W_t^{\text{OPT}}$ and $k \in W_t^{\mathcal{A}}$

PROOF. We prove the claim by induction over time. At time 0, the claim is trivially true as $M_0(j) = j$ for all jobs. Otherwise, consider the earliest region $[u, v)$ for which one or both of the conditions fails for $M_v(j) = k \neq j$.

In regard to condition (i) we note that if $M_u(j) \neq M_v(j)$ it must be that $M_v(j)$ was reassigned by the match-creation rule; in this case $M_v(k) = j$ is assigned by symmetry. If $M_u(j) = M_v(j) = k$, we can assume the inductive hypotheses at time u. Since $M_u(k) = j$, the concern would be if $M_v(k)$ were reassigned. That reassignment could not have been due to the match-breaking rule, as $M_v(j)$ would have been reassigned back to j in that case. The remaining concern is that k served the role of $M_u(a)$ or b in establishing a different match. If $k = M_u(a)$ then by induction $a = M_u(k) = j$. Yet in this case, a is scheduled by $\mathcal{A}$ and the match-breaking rule would result in $M_v(j) = j$, contradicting our assumption. Alternatively, if $k = b$ the match-creation rule would require $k \in P_u^{\mathcal{A}}$ in which case k must be unmatched at time u. This contradicts our assumption that $M_u(k) = j$. We therefore conclude that k did not take part in a new match during this region and so $M_v(k) = j$, re-establishing condition (i) at time v.

For condition (ii), we make the additional assumption that $j \prec k$ without loss of generality. If $M_u(j) = M_v(j) = k$ then we can apply induction for time u, implying $j \in W_u^{\text{OPT}}$ and $k \in W_u^{\mathcal{A}}$. The fact that $M_v(j) = k$ ensures that the match-breaking rule was not applied, and thus that neither j nor k were started by $\mathcal{A}$ or OPT during this region. Therefore both remain waiting at time v and thus $j \in W_v^{\text{OPT}}$ and $k \in W_v^{\mathcal{A}}$. In the case that $M_u(j) \neq M_v(j) = k$, the match must have been established during this region with j and k taking the roles (irrespectively) of $M_u(a)$ and b. If j had served as $b \in P_u^{\mathcal{A}} \subseteq Q_u^{\mathcal{A}}$ and k as $M_u(a)$, we note that k could not also serve as a, as it would not have been selected as $\prec$-minimal given that $j \in Q_u^{\mathcal{A}}$ and $j \prec k$. So it must be that $M_u(a) = k \neq a$. Applying induction to that pair it must be that $a \in W_u^{\mathcal{A}}$ and thus $k \prec a$. Yet by transitivity we have that $j \prec k \prec a$, again contradicting the selection of a as $\prec$-minimal in $Q_u^{\mathcal{A}}$. Therefore it must be that $j = M_u(a) \in Q_v^{\text{OPT}}$ and $k = b \in P_u^{\mathcal{A}} \cup W_v^{\mathcal{A}}$. This immediately establishes that $k \in W_v^{\mathcal{A}}$. If we can show that $j \notin Q_v^{\mathcal{A}}$ then we would have that $j \in Q_v^{\text{OPT}} \setminus Q_v^{\mathcal{A}} = W_v^{\text{OPT}}$ as desired. The condition $j \notin Q_v^{\mathcal{A}}$ is obvious if $j = a$ and thus scheduled by $\mathcal{A}$ to start the region. Otherwise, $M_u(a) = j \neq a$ and by

induction it must be that $a \in W_u^{\mathcal{A}}$ and $j \in W_u^{\text{OPT}}$. Since this requires that j had arrived strictly before time u and yet $j \notin Q_u^{\mathcal{A}}$, certainly $j \notin Q_v^{\mathcal{A}}$. This completes the re-establishment of condition (ii) for our induction. □

COROLLARY 5.2. *For an arbitrary region boundary t, the following are true:*

(i) If $j \in Q_t^{\text{OPT}}$ and $M_t(j) \neq j$, then
$j \in W_t^{\text{OPT}} \setminus P_t^{\text{OPT}}$, $M_t(j) \in W_t^{\mathcal{A}} \setminus P_t^{\mathcal{A}}$, and $j \prec M_t(j)$.
(ii) If $j \in Q_t^{\mathcal{A}}$ and $M_t(j) \neq j$, then
$j \in W_t^{\mathcal{A}} \setminus P_t^{\mathcal{A}}$, $M_t(j) \in W_t^{\text{OPT}} \setminus P_t^{\text{OPT}}$, and $M_t(j) \prec j$.
(iii) If $j \in Q_t^{\text{OPT}} \cap Q_t^{\mathcal{A}}$ or $j \notin Q_t^{\text{OPT}} \cup Q_t^{\mathcal{A}}$ then $M_t(j) = j$.

PROOF. By Lemma 5.1, if $M_t(j) \neq j$ then either $j \in W_t^{\text{OPT}} = Q_t^{\text{OPT}} \setminus Q_t^{\mathcal{A}}$ or $j \in W_t^{\mathcal{A}} = Q_t^{\mathcal{A}} \setminus Q_t^{\text{OPT}}$. Therefore, membership $j \in Q_t^{\text{OPT}}$ assures that $j \in W_t^{\text{OPT}}$ and, since matched, that $j \in W_t^{\text{OPT}} \setminus P_t^{\text{OPT}}$ as in (i). A similar argument shows (ii) in the case that $j \in Q_t^{\mathcal{A}}$. For (iii) we note that if j is in both queues or neither queue, then it cannot be a member of either $W_t^{\mathcal{A}}$ or W_t^{OPT} and therefore must be unmatched. □

5.2 Regional Analysis

For ease of exposition, we let $\mathcal{G} = (F_v^{\text{OPT}} \cup P_v^{\text{OPT}}) \setminus (F_u^{\text{OPT}} \cup P_u^{\text{OPT}})$ denote the set of jobs that newly contribute to Φ^{OPT} during a busy region $[u, v)$. We begin by showing that the potential functions are unchanged during an idle region.

LEMMA 5.3. *For an idle region $[u, v)$, $\Phi_v^{\mathcal{A}} = \Phi_u^{\mathcal{A}}$ and $\Phi_v^{\text{OPT}} = \Phi_u^{\text{OPT}}$.*

PROOF. As both machines idle at time u, $Q_u^{\mathcal{A}} = \emptyset$. Given that both machines continue to idle, no further jobs were released prior to time v and so $Q_v^{\mathcal{A}} = \emptyset$. As no jobs are completed, $F_v^{\mathcal{A}} = F_u^{\mathcal{A}}$, and since $Q_u^{\mathcal{A}} = Q_v^{\mathcal{A}} = \emptyset$, then trivially $W_u^{\mathcal{A}} = P_u^{\mathcal{A}} = W_v^{\mathcal{A}} = P_v^{\mathcal{A}} = \emptyset$. We conclude that $\Phi_v^{\mathcal{A}} = \Phi_u^{\mathcal{A}}$.

Since $Q_u^{\mathcal{A}} = Q_v^{\mathcal{A}} = \emptyset$, we have that $W_u^{\text{OPT}} = Q_u^{\text{OPT}}$ and $W_v^{\text{OPT}} = Q_v^{\text{OPT}}$, and since no new jobs arrive during the region, $Q_u^{\text{OPT}} \supseteq Q_v^{\text{OPT}}$. Furthermore, since $W_u^{\mathcal{A}} = W_v^{\mathcal{A}} = \emptyset$, the contrapositive of Lemma 5.1(ii) assures that $M_u(j) = M_v(j) = j$ for all jobs. Therefore, $P_u^{\text{OPT}} = W_u^{\text{OPT}} = Q_u^{\text{OPT}}$ and $P_v^{\text{OPT}} = W_v^{\text{OPT}} = Q_v^{\text{OPT}}$. Under such conditions, we can reformulate $\mathcal{G}$ more simply as $F_v^{\text{OPT}} \setminus (F_u^{\text{OPT}} \cup Q_u^{\text{OPT}})$, which is necessarily empty as OPT has no jobs to start other than those already existing in its queue. Therefore $\Phi_v^{\text{OPT}} = \Phi_u^{\text{OPT}}$. □

LEMMA 5.4. *For a busy region $[u, v)$, each job $j \in \mathcal{G}$ satisfies precisely one of the following conditions*

(α) $u \leq s_j^{\text{OPT}} < v$ and $j \notin P_u^{\text{OPT}}$;
(β) $s_j^{\text{OPT}} = x_j = v$, $M_u(j) \in \mathcal{R} \setminus \{a\}$;
(γ) $s_j^{\text{OPT}} \geq v$ and $M_u(j) = a$.

PROOF. Due to the constraints on s_j^{OPT} the first condition is clearly distinct from either of the remaining conditions. Due to the constraint on $M_u(j)$ the second and third are distinct from each other. We henceforth show that any such j satisfies one of the conditions. The set $\mathcal{G}$ can be decomposed into sets $F_v^{\text{OPT}} \setminus (F_u^{\text{OPT}} \cup P_u^{\text{OPT}})$ and $P_v^{\text{OPT}} \setminus (F_u^{\text{OPT}} \cup P_u^{\text{OPT}})$, as sets F_v^{OPT} and P_v^{OPT} are disjoint by definition. Items in $F_v^{\text{OPT}} \setminus (F_u^{\text{OPT}} \cup P_u^{\text{OPT}})$ are precisely those matching condition (α) of the lemma.

We show that job $j \in P_v^{\text{OPT}} \setminus (F_u^{\text{OPT}} \cup P_u^{\text{OPT}})$ satisfies either condition (β) or (γ). As $j \in P_v^{\text{OPT}}$, we have that $r_j < v \leq s_j^{\text{OPT}} \leq x_j$. If $M_u(j) = a$, then condition (γ) is satisfied. Alternatively, we must show that condition (β) is satisfied. Since at least one machine of $\mathcal{A}$ idles at time v based upon our region definition, Lemma 4.1 assures that j is accepted. Since $j \in P_v^{\text{OPT}} \subseteq W_v^{\text{OPT}} = Q_v^{\text{OPT}} \setminus Q_v^{\mathcal{A}}$, we have that $j \notin Q_v^{\mathcal{A}}$, and thus $s_j^{\mathcal{A}} < v$. We consider two subsequent cases, depending on whether $s_j^{\mathcal{A}} \geq u$. If $u \leq s_j^{\mathcal{A}} < v$ then both $\mathcal{A}$ and OPT start it on or after time u, and thus j is in both $Q_u^{\mathcal{A}}$ and Q_u^{OPT} or in neither (if not previously released). By Corollary 5.2(iii), it must be unmatched, and so $M_u(j) = j \in \mathcal{R} \setminus \{a\}$. Lemma 4.4 implies that $x_j \leq v$. Yet as $x_j \geq s_j^{\text{OPT}} \geq v$ we have that $s_j^{\text{OPT}} = x_j = v$, satisfying condition ($\beta$).

Alternatively, if $s_j^{\mathcal{A}} < u$, it must be that $j \in W_u^{\text{OPT}} \setminus P_u^{\text{OPT}}$ and thus matched at time u. Since $j \in W_u^{\text{OPT}}$, Corollary 5.2(i) implies that $M_u(j) \in W_u^{\mathcal{A}}$ and $j \prec M_u(j)$. Because $j \in P_v^{\text{OPT}}$ is unmatched at time v, it must be that either j or $M_u(j)$ is scheduled by $\mathcal{A}$ or OPT during this region. Based on our current assumptions $s_j^{\mathcal{A}} < u$ and $s_j^{\text{OPT}} \geq v$, so $M_u(j) \in W_u^{\mathcal{A}}$ must be started during the region by $\mathcal{A}$. Because $M_u(j) \neq a$, Lemma 4.4 implies that $x_{M_u(j)} \leq v$. Therefore $x_j \leq x_{M_u(j)} \leq v \leq s_j^{\text{OPT}}$, ensuring that $s_j^{\text{OPT}} = x_j = v$, thereby satisfying condition (β). □

Based upon the three conditions of Lemma 5.4, we define a partition of the set $\mathcal{G}$ into sets $\mathcal{G}_\alpha$, $\mathcal{G}_\beta$, and $\mathcal{G}_\gamma$. Independently we let $\mathcal{G}_+$ (resp. $\mathcal{G}_-$) denote the jobs of $\mathcal{G}$ that were accepted (resp. rejected) by $\mathcal{A}$. Superimposing these partitions, we define sets such as $\mathcal{G}_{\alpha^+} \equiv \mathcal{G}_\alpha \cap \mathcal{G}_+$ and $\mathcal{G}_{\alpha^-} \equiv \mathcal{G}_\alpha \cap \mathcal{G}_-$. Although we could use similar notations for $\mathcal{G}_\beta$ and $\mathcal{G}_\gamma$, the next lemma shows that this is unnecessary.

LEMMA 5.5. *Any job of $\mathcal{G}_\beta$ or $\mathcal{G}_\gamma$ is accepted by $\mathcal{A}$.*

PROOF. By the definition of the region, at least one machine of $\mathcal{A}$ is idle at time v. For $j \in \mathcal{G}$, $r_j < v$, and for $j \in \mathcal{G}_\beta \cup \mathcal{G}_\gamma$, $x_j \geq s_j^{\text{OPT}} \geq v$. Therefore, we may apply Lemma 4.1 to assure the acceptance of j by $\mathcal{A}$. □

LEMMA 5.6. *If $M_u(a) \in Q_v^{\text{OPT}}$ for a busy region $[u, v)$, then $M_u(a) \in W_v^{\text{OPT}}$ and $a \notin P_u^{\mathcal{A}}$.*

PROOF. There are two cases, depending on whether $M_u(a) = a$. If so, then membership of $M_u(a) = a \in Q_v^{\text{OPT}}$ assures that $s_a^{\text{OPT}} \geq v \geq u$ and thus $a \notin W_u^{\mathcal{A}} \supseteq P_u^{\mathcal{A}}$. Furthermore, as a is started by $\mathcal{A}$ during the region, $a \in Q_v^{\text{OPT}} \setminus Q_v^{\mathcal{A}} = W_v^{\text{OPT}}$.

Alternatively, if $M_u(a) \neq a$, the existence of the match directly precludes membership of a in $P_u^{\mathcal{A}}$. Furthermore, since both of these jobs must have arrived prior to u to be matched, condition $M_u(a) \in Q_v^{\text{OPT}}$ implies $M_u(a) \in Q_u^{\text{OPT}}$. Combining this with Corollary 5.2(i) we have that $M_u(a) \in W_v^{\text{OPT}} \setminus P_v^{\text{OPT}}$. □

LEMMA 5.7. *If $P_u^{\mathcal{A}} \neq \emptyset$ and $\mathcal{G}_\gamma \neq \emptyset$, then*

(i) $\exists b \in P_u^{\mathcal{A}} \cap \mathcal{R} \setminus \{a\}$

(ii) $s_{M_a(u)}^{\text{OPT}} = x_{M_a(u)} = x_b = v$

(iii) $x_z = v$

PROOF. By definition, the only potential element of $\mathcal{G}_\gamma$ is $M_a(u)$ with $s_{M_a(u)}^{\text{OPT}} \geq v$. For this element to contribute to $\mathcal{G}$ it must be in P_v^{OPT}, thus unmatched at time v. Yet given that it is in Q_v^{OPT}, the match-creation rule would apply so long as there

exists some $b \in P_u^{\mathcal{A}} \cap W_v^{\mathcal{A}}$. By our assumption $P_u^{\mathcal{A}} \neq \emptyset$, and by Lemma 5.6 $a \notin P_u^{\mathcal{A}}$, so there must exists some $b \neq a$ in $P_u^{\mathcal{A}}$. To avoid the match, it must be that $b \notin W_v^{\mathcal{A}}$ and thus b started by $\mathcal{A}$ during this region, proving (i).

For (ii) we note that $x_b \leq v$ by Lemma 4.4. Furthermore, $x_a \leq x_b$, given that a was started by $\mathcal{A}$ at time u even though $b \in P_u^{\mathcal{A}} \subseteq Q_u^{\mathcal{A}}$. We also claim that $x_{M_a(u)} \leq x_a$. This is trivially so if $M_a(u) = a$. Otherwise, since $a \in Q_u^{\mathcal{A}}$ Corollary 5.2(ii) assures that $M_a(u) \prec a$. By the definition of $\mathcal{G}_\gamma$, $s^{\text{OPT}}_{M_a(u)} \geq v$. Combining these facts, $v \leq s^{\text{OPT}}_{M_a(u)} \leq x_{M_a(u)} \leq x_a \leq x_b \leq v$, thus all equal to v.

For (iii) we note that by Lemma 4.4, $x_z \leq v$. Since $b \in \mathcal{R} \backslash \{a\}$ has $x_b = v \geq s_y^{\mathcal{A}} + p$, Lemma 4.5 implies $x_z \geq v$. □

LEMMA 5.8. *For a busy region $[u, v)$, if $x_z < s_y^{\mathcal{A}} + p$, then $\mathcal{G}_\beta = \emptyset$.*

PROOF. Conversely, we show that if $\exists j \in \mathcal{G}_\beta$, then $x_z \geq s_y^{\mathcal{A}} + p$. By definition, $s_j^{\text{OPT}} = x_j = v$ and $M_u(j) \in \mathcal{R} \setminus \{a\}$. Next we claim that $x_j \leq x_{M_u(j)}$. This is trivially so if $M_u(j) = j$. Otherwise, $j \in Q_u^{\text{OPT}}$ and Corollary 5.2(i) assures that $j \prec M_u(j)$. Combining facts we have that $s_y^{\mathcal{A}} + p \leq v = s_j^{\text{OPT}} = x_j \leq x_{M_u(j)}$. This allows us to apply Lemma 4.5 to $M_u(j)$, implying that $x_z \geq s_y^{\mathcal{A}} + p$. □

LEMMA 5.9. *For $j \in \mathcal{G}_+$, $M_u(j) \in (F_v^{\mathcal{A}} \cup P_v^{\mathcal{A}}) \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$.*

PROOF. By definition, $j \in \mathcal{G}_+$ is released prior to v, accepted by both schedules, yet with $j \notin F_u^{\text{OPT}} \cup P_u^{\text{OPT}}$. We consider two cases, depending on whether $j \in F_u^{\mathcal{A}}$.

We first consider when $j \notin F_u^{\mathcal{A}}$. In this case, at time u it must be that j is either in both $Q_u^{\mathcal{A}}$ and Q_u^{OPT} (assuming it arrives strictly before u) or else in neither queue. Therefore $j \notin Q_u^{\mathcal{A}} \setminus Q_u^{\text{OPT}} = W_u^{\mathcal{A}} \supseteq P_u^{\mathcal{A}}$ and by Corollary 5.2, $M_u(j) = j$. We must still show that $M_u(j) = j \in F_v^{\mathcal{A}} \cup P_v^{\mathcal{A}}$. Since j is started by $\mathcal{A}$ on or after u, either $j \in F_v^{\mathcal{A}}$ satisfies the condition, or else $j \in Q_v^{\mathcal{A}}$. In the latter case, $j \in Q_v^{\mathcal{A}} \setminus Q_v^{\text{OPT}} = W_v^{\mathcal{A}}$, as $j \in Q_v^{\mathcal{A}} \cap Q_v^{\text{OPT}}$ would contradict $j \in \mathcal{G}_+$. So long as j remains unmatched at time v, then $j \in P_v^{\mathcal{A}}$ satisfies the condition. By the match-creation rule, a new match must be between an element of Q_v^{OPT} and $P_u^{\mathcal{A}} \cap W_v^{\mathcal{A}}$. Yet we have already established that $j \notin Q_v^{\text{OPT}}$ and $j \notin W_u^{\mathcal{A}} \supseteq P_u^{\mathcal{A}}$. This completes the proof for the case that $j \notin F_u^{\mathcal{A}}$.

When $j \in F_u^{\mathcal{A}}$ it must be that $j \in Q_u^{\text{OPT}} \setminus Q_u^{\mathcal{A}} = W_u^{\text{OPT}}$, given that $j \notin F_u^{\text{OPT}}$ by assumption. As we assume such $j \notin P_u^{\text{OPT}}$, it must be that $j \in W_u^{\text{OPT}} \setminus P_u^{\text{OPT}}$ and matched at time u. By Corollary 5.2(i), $M_u(j) \in W_u^{\mathcal{A}} \setminus P_u^{\mathcal{A}}$ and thus, $M_u(j) \notin F_u^{\mathcal{A}}$. If $M_u(j) \in F_v^{\mathcal{A}}$, the claim is complete. Otherwise, we have that $M_u(j) \in W_v^{\mathcal{A}}$ and we must show that $M_v(M_u(j)) = M_u(j)$ so that $M_u(j) \in P_v^{\mathcal{A}}$. Since $\mathcal{A}$ starts j under the current assumptions, the match from time u is indeed broken. Furthermore, the only case in which one match is broken and another re-established is if $M_u(j) = a \in F_v^{\mathcal{A}}$, a condition that we have already precluded. □

LEMMA 5.10. *If $|\mathcal{R}| = 1$ and $P_v^{\mathcal{A}} = \emptyset$ then $|\mathcal{G}| \leq 1$.*

PROOF. Since one machine of $\mathcal{A}$ is idle throughout the region, Lemma 4.1 assures that $\mathcal{G}_{\alpha^-} = \emptyset$ and thus $\mathcal{G} = \mathcal{G}_+$. If there were two distinct jobs in $\mathcal{G}_+$, Lemma 5.9 guarantees the existence of two (distinct) jobs of $(F_v^{\mathcal{A}} \cup P_v^{\mathcal{A}}) \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$. Since we assume that $P_v^{\mathcal{A}}$ is empty, both of these jobs must lie in $F_v^{\mathcal{A}} \setminus F_u^{\mathcal{A}}$, contradicting the assumption that $|\mathcal{R}| = 1$. We therefore conclude that $|\mathcal{G}| \leq 1$. □

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

LEMMA 5.11. *For a busy region $[u, v)$ and arbitrary machine m,*

(i) $|\mathcal{G}^m_\beta| + |\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}|$.

(ii) If inequality (i) is tight and $|\mathcal{R}| \geq 2$, then OPT *either starts a job of $\mathcal{G}^m_{\alpha^-}$ at $v - p$ or is processing a job of $\mathcal{G}^m_\alpha \cup \mathcal{G}^m_\beta$ during $[v, v + 1)$.*

(iii) If inequality (i) is tight, $\mathcal{G}_\beta = \emptyset$, and $|\mathcal{R}| \geq 2$, then no job of $\mathcal{G}^m_+$ can be started by OPT *during the interval $[u^\tau, s^{\mathcal{A}}_y + p)$.*

PROOF. We use a counting argument to establish the lemma. We initially consider all elements of $\mathcal{R}$ to be unmarked. For each job $j \in \mathcal{G}^m_\alpha$, we mark those jobs of $\mathcal{R}$ that are currently being processed at s^{OPT}_j. 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 4.1, if j were rejected by $\mathcal{A}$, two distinct jobs of $\mathcal{A}$ must be running at time s^{OPT}_j. Therefore two jobs of $\mathcal{R}$ are marked in association with a rejected j, and $2 \cdot |\mathcal{G}^m_{\alpha^-}|$ jobs are marked overall in association with rejected jobs. Any job of $\mathcal{G}^m_{\alpha^+}$ must mark at least one further job of $\mathcal{R}$, since there is never a time during $[u, v)$ when both machines of $\mathcal{A}$ are idle. Therefore $|\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}|$. Furthermore, we note that if any job of $\mathcal{R}$ goes unmarked or if any job of $\mathcal{G}^m_{\alpha^+}$ is responsible for marking two jobs of $\mathcal{R}$, then $|\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}| - 1$.

The remainder of the proof is divided into two cases, depending upon whether $\mathcal{G}^m_\beta = \emptyset$. If so, we have established condition (i). To establish (ii), we note that z must have been marked. This either requires that OPT start a job during interval $(v - p, v)$ and hence continue running strictly beyond time v, or else that z be marked by a job of OPT starting precisely at $v - p$. However, since $|\mathcal{R}| \geq 2$, such a job would mark y as well and therefore must be in $\mathcal{G}^m_{\alpha^-}$, as no job of $\mathcal{G}^m_{\alpha^+}$ can mark two jobs if the inequality is tight. To establish (iii), we simply note that both machines of $\mathcal{A}$ are busy throughout the period $[u^\tau, s^{\mathcal{A}}_y + p)$ based on our definition of the tail, and so any job started by OPT during that region must be a rejected job for the above inequality to be tight. This concludes the proof under the assumption that $\mathcal{G}^m_\beta = \emptyset$.

If there exists $j \in \mathcal{G}^m_\beta$, then by definition $s^{\text{OPT}}_j = v$ and $M_u(j) \in \mathcal{R} \setminus \{a\}$. Since only one job can be started by OPT on machine m at time v, $|\mathcal{G}^m_\beta| = 1$. Notice that if z were not marked by an element of $\mathcal{G}^m_\alpha$, then $|\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}| - 1$ and so $|\mathcal{G}^m_\beta| + |\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}|$. For z to be marked, given that $s^{\text{OPT}}_j = v$, OPT must start a job k precisely at time $v - p$. Lemma 5.8 implies that $x_z \geq s^{\mathcal{A}}_y + p$ and so Lemma 4.7(i) guarantees that k is accepted by $\mathcal{A}$. As this job marks both y and z, again we find that $|\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}| - 1$ and so $|\mathcal{G}^m_\beta| + |\mathcal{G}^m_{\alpha^+}| + 2 \cdot |\mathcal{G}^m_{\alpha^-}| \leq |\mathcal{R}|$. This establishes part (i). Part (ii) is trivial in this case as $j \in \mathcal{G}^m_\beta$ is started at time v by OPT, and part (iii) does not apply. □

LEMMA 5.12. *If $|\mathcal{R}| \geq 2$, $P^{\mathcal{A}}_v = \emptyset$, and $x_z \geq s^{\mathcal{A}}_y + p$, then there exists a job $j \notin \mathcal{G}$ that must be started by* OPT *at time v on some machine, and the inequality of Lemma 5.11(i) is not tight for that machine.*

PROOF. By Lemma 4.7(ii), there exists $j' \in Q_v^{\mathcal{A}}$ with $s_{j'}^{\mathcal{A}} = x_{j'} = v$. Yet since we currently presume $P_v^{\mathcal{A}} = \emptyset$, it must be that either $j' \in Q_v^{\text{OPT}}$ and thus $M_v(j') = j'$ by Corollary 5.2(iii), or that $M_v(j') \neq j'$ and thus $M_v(j') \in W_v^{\text{OPT}} \setminus P_v^{\text{OPT}}$ with $M_v(j') \prec j'$ by Corollary 5.2(ii). In either case, we see that $M_v(j') \in Q_v^{\text{OPT}}$ and that $M_v(j') \notin \mathcal{G}$. Since $x_{M_v(j')} \leq x_{j'} = v$, OPT must start $M_v(j')$ at time v on a machine m. Therefore $M_v(j')$ satisfies the conditions of j in the first part of the lemma result.

If the inequality of Lemma 5.11(i) were tight for m, part (ii) of that lemma implies that a job of $\mathcal{G}_{\alpha^-}^m$ must be started precisely at $v-p$, given that $M_v(j') \notin \mathcal{G}$ occupies the interval $[v, v+p)$. However the rejection of a job that OPT starts at time $v-p$ contradicts Lemma 4.7(i). Therefore we conclude that the inequality of Lemma 5.11(i) is not tight for m. □

LEMMA 5.13. *If $\Phi_u^{\text{OPT}} \leq \Phi_u^{\mathcal{A}}$ for a busy region $[u,t)$, then $\Phi_v^{\text{OPT}} \leq \Phi_v^{\mathcal{A}}$.*

PROOF. For ease of exposition, we introduce notation $\Phi_{[u,v)}^{\mathcal{A}}$ to denote $(\Phi_v^{\mathcal{A}} - \Phi_u^{\mathcal{A}})$, and likewise $\Phi_{[u,v)}^{\text{OPT}} = (\Phi_v^{\text{OPT}} - \Phi_u^{\text{OPT}})$. Our goal is to show that $\Phi_{[u,v)}^{\text{OPT}} \leq \Phi_{[u,v)}^{\mathcal{A}}$. In accordance with Lemmas 5.4 and 5.5, we rewrite $\Phi_{[u,v)}^{\text{OPT}}$ as,

$$\begin{aligned}
\Phi_{[u,v)}^{\text{OPT}} &= 2 \cdot |\mathcal{G}| = 2 \cdot |\mathcal{G}_{\alpha^+}^{m_1} \cup \mathcal{G}_{\alpha^+}^{m_2} \cup \mathcal{G}_{\alpha^-}^{m_1} \cup \mathcal{G}_{\alpha^-}^{m_2} \cup \mathcal{G}_{\beta}^{m_1} \cup \mathcal{G}_{\beta}^{m_2} \cup \mathcal{G}_{\gamma}| \\
&= (|\mathcal{G}_{\beta}^{m_1}| + |\mathcal{G}_{\alpha^+}^{m_1}| + 2 \cdot |\mathcal{G}_{\alpha^-}^{m_1}|) + (|\mathcal{G}_{\beta}^{m_2}| + |\mathcal{G}_{\alpha^+}^{m_2}| + 2 \cdot |\mathcal{G}_{\alpha^-}^{m_2}|) \\
&\quad + |\mathcal{G}_{\alpha^+}^{m_1} \cup \mathcal{G}_{\alpha^+}^{m_2} \cup \mathcal{G}_{\beta}^{m_1} \cup \mathcal{G}_{\beta}^{m_2} \cup \mathcal{G}_{\gamma}| + |\mathcal{G}_{\gamma}| \\
&= (|\mathcal{G}_{\beta}^{m_1}| + |\mathcal{G}_{\alpha^+}^{m_1}| + 2 \cdot |\mathcal{G}_{\alpha^-}^{m_1}|) + (|\mathcal{G}_{\beta}^{m_2}| + |\mathcal{G}_{\alpha^+}^{m_2}| + 2 \cdot |\mathcal{G}_{\alpha^-}^{m_2}|) + |\mathcal{G}_{+}| + |\mathcal{G}_{\gamma}|
\end{aligned}$$

By applying Lemma 5.11(i) to each of the two machines of OPT, we conclude that $\Phi_{[u,v)}^{\text{OPT}} \leq 2 \cdot |\mathcal{R}| + |\mathcal{G}_+| + |\mathcal{G}_\gamma|$. In contrast, we argue that $\Phi_{[u,v)}^{\mathcal{A}} \geq 2 \cdot |\mathcal{R}| + |\mathcal{G}_+| + \delta$, where δ is defined as

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

Informally, each job of $\mathcal{R}$ results in a relative contribution of at least 2, as such job is added to $F^{\mathcal{A}}$ although potentially taken from $P^{\mathcal{A}}$. We will demonstrate an additional bonus of $|\mathcal{G}_+|$ credits, either from jobs of $F_v^{\mathcal{A}} \setminus P_u^{\mathcal{A}}$ that produce 3 credits rather than 2, or from jobs of $P_v^{\mathcal{A}} \setminus P_u^{\mathcal{A}}$ that receive 1 for being newly added to $P^{\mathcal{A}}$. The δ term adjusts for the discontinuity inherent in our definition of $\Phi^{\mathcal{A}}$, based upon whether set $P^{\mathcal{A}}$ is empty.

More formally, $\Phi_{[u,v)}^{\mathcal{A}} = 3(|F_v^{\mathcal{A}}| - |F_u^{\mathcal{A}}|) + |P_v^{\mathcal{A}}| - |P_u^{\mathcal{A}}| + \delta$. Since $F_v^{\mathcal{A}} \supseteq F_u^{\mathcal{A}}$, we have that $|F_v^{\mathcal{A}}| - |F_u^{\mathcal{A}}| = |F_v^{\mathcal{A}} \setminus F_u^{\mathcal{A}}| = |\mathcal{R}|$. Since $P_v^{\mathcal{A}}$ and $P_u^{\mathcal{A}}$ are not necessarily subsets of each other, $|P_v^{\mathcal{A}}| - |P_u^{\mathcal{A}}| = |P_v^{\mathcal{A}} \setminus P_u^{\mathcal{A}}| - |P_u^{\mathcal{A}} \setminus P_v^{\mathcal{A}}|$. If we ignore, for the moment, the one scenario in which a new match is established during a region, then $P_u^{\mathcal{A}} \setminus P_v^{\mathcal{A}} \subseteq \mathcal{R}$. Lemma 5.9 states that for each $j \in \mathcal{G}_+$ there exists $M_u(j) \in (F_v^{\mathcal{A}} \cup P_v^{\mathcal{A}}) \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$. These $|\mathcal{G}_+|$ jobs are distinct, as the definition of $M_u(\cdot)$ assures that $M_u(j) \neq M_u(k)$ for $j \neq k$. For each such $M_u(j) \in F_v^{\mathcal{A}} \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$, we have a relative decrease by one in the quantity $|P_u^{\mathcal{A}} \setminus P_v^{\mathcal{A}}|$ as such a job is newly started by $\mathcal{A}$ yet not drawn from $P_u^{\mathcal{A}}$. For each such $M_u(j) \in P_v^{\mathcal{A}} \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$, we have a relative increase by one in the quantity $|P_v^{\mathcal{A}} \setminus P_u^{\mathcal{A}}|$. Therefore, modulo the creation of a new match, we have

established that $|P_v^{\mathcal{A}}|-|P_u^{\mathcal{A}}| \geq |\mathcal{G}_+|-|\mathcal{R}|$ and thus $\Phi_{[u,v)}^{\mathcal{A}} \geq 2\cdot|\mathcal{R}|+|\mathcal{G}_+|+\delta$. The only scenario in which a match is established is when $M_u(a) \in Q_v^{\text{OPT}}$ becomes matched to some $b \in P_u^{\mathcal{A}}$. In this case, b is removed from $P^{\mathcal{A}}$ resulting in a one-credit deficit in the analysis. However, by Lemma 5.6, $a \notin P_u^{\mathcal{A}}$ and thus $a \in F_v^{\mathcal{A}} \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$. This results in an additional credit, distinct from those associated with Lemma 5.9. The only concern would be if $M_u(a) \in \mathcal{G}_+$ and thus $a = M_u(M_u(a))$ double-counted. Yet by the matching-rule preconditions, $M_u(a) \in Q_v^{\text{OPT}}$ and re-matched at time v, thus it does not contribute to $\mathcal{G}_+$.

We have thus far established that $\Phi_{[u,v)}^{\mathcal{A}} \geq 2\cdot|\mathcal{R}|+|\mathcal{G}_+|+\delta$. For the remainder of the proof, we focus on the expression $\delta-|\mathcal{G}_\gamma|$. In cases when $\delta-|\mathcal{G}_\gamma| \geq 0$ the theorem follows immediately, as $\Phi_{[u,v)}^{\text{OPT}} \leq 2\cdot|\mathcal{R}|+|\mathcal{G}_+|+|\mathcal{G}_\gamma| \leq 2\cdot|\mathcal{R}|+|\mathcal{G}_+|+\delta \leq \Phi_{[u,v)}^{\mathcal{A}}$. Otherwise we undertake a detailed case analysis to counterbalance this deficit by showing a gap in our upper bound on $\Phi_{[u,v)}^{\text{OPT}}$ due to a strict inequality when applying Lemma 5.11(i) to one or both of the two machines of OPT, or by showing a gap in our lower bound on $\Phi_{[u,v)}^{\mathcal{A}}$.

We begin by considering the case when $|\mathcal{R}| = 1$. If $P_v^{\mathcal{A}} \neq \emptyset$, the only way in which $\delta - |\mathcal{G}_\gamma| < 0$ is when $P_u^{\mathcal{A}} \neq \emptyset$ and $|\mathcal{G}_\gamma| = 1$. Yet then Lemma 5.7(i) contradicts $|\mathcal{R}| = 1$. With $P_v^{\mathcal{A}} = \emptyset$, Lemma 5.10 implies that $|\mathcal{G}| \leq 1$. In this case, the inequality of Lemma 5.11(i) must be loose for at least one of the machines. Such a gap counterbalances the deficit when $\delta - |\mathcal{G}_\gamma| \geq -1$. The only way in which $\delta - |\mathcal{G}_\gamma| = 2$ is when $|\mathcal{G}_\gamma| = 1$ and $P_v^{\mathcal{A}} = 1$, yet in this case $\mathcal{G}_\alpha = \mathcal{G}_\beta = \emptyset$ and so the inequality is loose for *both* machines, counterbalancing the deficit of -2.

We therefore assume $|\mathcal{R}| \geq 2$ for the remainder of this proof. We consider two sub-cases, starting with one for which $x_z \geq s_y^{\mathcal{A}} + p$. In this case, we begin by examining the possibility that $\delta - |\mathcal{G}_\gamma| = -2$. This occurs when $\mathcal{G}_\gamma \neq \emptyset$, $P_u^{\mathcal{A}} \neq \emptyset$, and $P_v^{\mathcal{A}} = \emptyset$. With $P_v^{\mathcal{A}} = \emptyset$, Lemma 5.12 guarantees the existence of $j \notin \mathcal{G}$ that must be started by OPT at time v on a machine m for which the inequality of Lemma 5.11(i) is not tight. We show a similar gap for the inequality of the other machine m'. With $\mathcal{G}_\gamma \neq \emptyset$ and $P_u^{\mathcal{A}} \neq \emptyset$, Lemma 5.7(ii) states that $s_{M_a(u)}^{\text{OPT}} = v$. Since $M_a(u) \in \mathcal{G}$, $M_a(u) \neq j$ which is started by OPT on m at time u. So $M_a(u)$ must be started on m'. If the inequality for m' were tight, Lemma 5.11(ii) requires either that machine m' in OPT starts a job of $\mathcal{G}_{\alpha^-}^{m'}$ at $v-p$ or is processing a job of $\mathcal{G}_\alpha^{m'} \cup \mathcal{G}_\beta^{m'}$ during $[v, v+1)$. The former is precluded by Lemma 4.7(i) and the latter is precluded by the start of $M_a(u) \in \mathcal{G}_\gamma$ at v. Therefore, the inequality is loose for both m and m', providing the necessary gap to counterbalance the deficit of $\delta - |\mathcal{G}_\gamma| = -2$. In this case that $\delta - |\mathcal{G}_\gamma| = -1$, it must either be that $P_v^{\mathcal{A}} = \emptyset$ or that both $\mathcal{G}_\gamma \neq \emptyset$ and $P_u^{\mathcal{A}} = \emptyset$. In either scenario, we can apply a part of the above argument to demonstrate a loose inequality on one of the machines, so we conclude the proof under the assumption $x_z \geq s_y^{\mathcal{A}} + p$.

When $x_z < s_y^{\mathcal{A}} + p$, Lemma 4.6 implies that some job of tail $\mathcal{T}$ is released on or after time $u^{\mathcal{T}}$. In particular, let $j \in \mathcal{T}$ with $r_j \geq u^{\mathcal{T}}$ be the latest such job to be started by $\mathcal{A}$. We note that $x_j \leq x_z$ by this definition, as either $j = z$ or else z arrived before $u^{\mathcal{T}}$ and was in the queue when j was started implying that $j \prec z$. Therefore, we have $u \leq u^{\mathcal{T}} \leq r_j \leq x_j < s_y^{\mathcal{A}} + p$ for $j \in \mathcal{R}$. This implies that $j \in F_v^{\mathcal{A}} \setminus (F_u^{\mathcal{A}} \cup P_u^{\mathcal{A}})$, providing an extra credit as a job of $F_v^{\mathcal{A}}$ that produces

3 credits toward $\Phi^{\mathcal{A}}_{[u,v)}$. We argue that this credit is distinct from those in our earlier accounting. Since $x_z < s^{\mathcal{A}}_y + p$ Lemma 5.8 implies that $\mathcal{G}_\beta = \emptyset$, and thus by Lemma 5.11(iii) no job of $\mathcal{G}_+$ can be started during interval $[u^\tau, s^{\mathcal{A}}_y + p)$. Therefore $j \notin \mathcal{G}_+$ and since it was released during the region, $M_u(j) = j$. This makes it distinct from the $|\mathcal{R}|$ jobs whose existence was justified through Lemma 5.9. We also note that this additional credit is distinct from the one associated when a becomes matched, as j arrives strictly after u and thus $j \neq a$. This credit allows us to complete the analysis for the case when $\delta - |\mathcal{G}_\gamma| = -1$. The only remaining concern is that $\delta - |\mathcal{G}_\gamma| = -2$, but this is impossible as it requires that $P^{\mathcal{A}}_u \neq \emptyset$ and $\mathcal{G}_\gamma \neq \emptyset$, which implies $x_z = v \geq s^{\mathcal{A}}_y + p$ by Lemma 5.7(iii). □

THEOREM 5.14. *Algorithm $\mathcal{A}$ is $\frac{3}{2}$-competitive.*

PROOF. We show by induction that $\Phi^{\text{OPT}}_t \leq \Phi^{\mathcal{A}}_t$. Initially, $\Phi^{\text{OPT}}_0 = \Phi^{\mathcal{A}}_0 = 0$. We then consider successive regions of the form $[u, v)$, with $\Phi^{\text{OPT}}_v \leq \Phi^{\mathcal{A}}_v$ guaranteed by Lemma 5.3 for an idle region or Lemma 5.13 for a busy region. We conclude that $\Phi^{\text{OPT}}_\infty \leq \Phi^{\mathcal{A}}_\infty$. Since the queues of both $\mathcal{A}$ and OPT are eventually empty, $P^{\mathcal{A}}_\infty = P^{\text{OPT}}_\infty = \emptyset$. The inequality $\Phi^{\text{OPT}}_\infty \leq \Phi^{\mathcal{A}}_\infty$ is equivalently stated as $2 \cdot |F^{\text{OPT}}_\infty| \leq 3 \cdot |F^{\mathcal{A}}_\infty|$, thus $\frac{|\text{OPT}|}{|\mathcal{A}|} \leq \frac{3}{2}$. □

THEOREM 5.15. *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 with $r_j = 0$ and $d_j = 3p - 1$. 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 are subsequently released at time $t + 1$ with deadlines $t + 1 + p$, one must be rejected. It is easy to verify that OPT can achieve all three jobs for any value of t. □

ACKNOWLEDGMENTS

We appreciate the correspondence of Jiří Sgall at early stages of this work. We also gratefully acknowledge the anonymous referee's attention to detail. The organization and presentation of this paper have been greatly improved as a result of the valuable feedback.

REFERENCES

BAPTISTE, P., BRUCKER, P., KNUST, S., AND TIMKOVSKY, V. G. 2004. Ten notes on equal-processing-time scheduling. *4OR: Quarterly J. Belgian, French and Italian Operations Research Societies 2,* 2 (July), 111–127.

BARUAH, S. K., HARITSA, J. R., AND SHARMA, N. 2001. On-line scheduling to maximize task completions. *J. Combin. Math. and Combin. Computing 39*, 65–78.

BORODIN, A. AND EL-YANIV, R. 1998. *Online Computation and Competitive Analysis*. Cambridge University Press, New York.

CHROBAK, M., JAWOR, W., SGALL, J., AND TICHÝ, T. 2004. Online scheduling of equal-length jobs: Randomization and restarts help. In *Proc. 31st Int. Colloquium on Automata, Languages and Programming (ICALP)*, J. Díaz, J. Karhumäki, A. Lepistö, and D. Sannella, Eds. Lecture Notes in Computer Science, vol. 3142. Springer-Verlag, Turku, Finland, 358–370.

DING, J. AND ZHANG, G. 2006. 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, vol. 4041. Springer-Verlag, Hong Kong, China, 32–42.

Goldman, S., Parwatikar, J., and Suri, S. 2000. On-line scheduling with hard deadlines. *J. Algorithms 34,* 2 (Feb.), 370–389.

Goldwasser, M. H. 2003. Patience is a virtue: The effect of slack on competitiveness for admission control. *J. Scheduling 6,* 2 (Mar./Apr.), 183–211.

Goldwasser, M. H. and Kerbikov, B. 2003. Admission control with immediate notification. *J. Scheduling 6,* 3 (May/June), 269–285.

Jackson, J. R. 1955. Scheduling a production line to minimize maximum tardiness. Research Report 43, Management Science Research Project, University of California, Los Angeles. Jan.

Simons, B. B. 1983. Multiprocessor scheduling of unit length jobs with arbitrary release times and deadlines. *SIAM J. Comput. 12*, 294–299.

Simons, B. B. and Warmuth, M. K. 1989. A fast algorithm for multiprocessor scheduling of unit-length jobs. *SIAM J. Comput. 18,* 4, 690–710.