ELSEVIER

Information Processing Letters 81 (2002) 297–304

Information Processing Letters

www.elsevier.com/locate/ipl

On an on-line scheduling problem for parallel jobs

Edwin Naroska, Uwe Schwiegelshohn *

Computer Engineering Institute, University Dortmund, 44221 Dortmund, Germany

Received 26 July 2000; received in revised form 25 May 2001
Communicated by S. Albers

Abstract

This paper addresses the non-preemptive on-line scheduling of parallel jobs. In particular we assume that the release dates and the processing times of the jobs are unknown. It is already known that for this problem Garey and Graham's list scheduling algorithm achieves the competitive factor $2 - \frac{1}{m}$ for the makespan if m identical machines are available and if each job requires only a single machine for processing. Here, we show that the same factor also holds in the case of parallel jobs.

Keywords: On-line algorithms; Analysis of algorithms; Scheduling

1. Introduction

We address a well known on-line problem: A job system τ consisting of independent parallel jobs must be scheduled on m identical machines without preemption. Each job $j \in \tau$ is characterized by its fixed (integer) degree of parallelism $1 \leqslant m_j \leqslant m$, its processing time $p_j > 0$ and its release date $r_j \geqslant 0$. Exactly m_j machines must be allocated to a job j at its start and will be released altogether once the processing is finished. Each machine can execute at most one job at a time and the execution of job j requires the same time on any subset of m_j machines. Further, a job is not known before it is released and its processing time is unknown until the job has finished. We denote the completion time of job j in a schedule S by $C_j(S)$. Therefore, job j starts at time $C_j(S) - p_j$ in schedule S. In this paper it is the goal of a scheduling algorithm to minimize the makespan $C_{\max}(S) = \max_{j\in\tau} C_j(S)$ of S. Further, $C^*_{\max}(\tau)$ is the minimal makespan of all valid schedules on m machines for job system τ.

On-line problems are often evaluated with the help of the so called competitive ratio ρ. Here, we say that an on-line algorithm has a competitive ratio ρ or is *ρ-competitive* if $C_{\max}(S) \leqslant \rho C^*_{\max}(\tau)$ holds for any job system τ and for any schedule S generated by this algorithm. Note that it may be a hard problem to find an optimal schedule S' with $C_{\max}(S') = C^*_{\max}(\tau)$ even if all information about τ is already available at time 0.

Variants of this problem have already been addressed in the past. Shmoys, Wein and Williamson [1] proved a lower bound of $2 - \frac{1}{m}$ for the competitive ratio using an example of Graham. This bound also holds for the restricted cases where all jobs are sequential ($m_j = 1$ for all $j \in \tau$) or are immediately available ($r_j = 0$ for all

* Corresponding author.
E-mail addresses: edwin@ds.e-technik.uni-dortmund.de (E. Naroska), uwe@ds.e-technik.uni-dortmund.de (U. Schwiegelshohn).

0020-0190/02/$ – see front matter
PII: S0020-0190(01)00241-1

$j \in \tau$). Feldmann, Sgall and Teng [2] assumed $r_j = 0$ for all jobs and proved that the on-line version of Garey and Graham's list scheduling algorithm [3] is $2 - \frac{1}{m}$ competitive, see Lemma 3. For the sake of completeness we repeat this algorithm that has been presented first by Graham in 1966 [4]:

Algorithm *List Scheduling*

repeat
  for any unscheduled and released job j **do**
    if at least m_j machines are idle
      start j immediately on m_j idle machines
until all jobs in τ are scheduled

For the rest of the paper we say that a schedule is a *list schedule* if it is produced by Algorithm *List Scheduling*. Note that the algorithm does not require a specific processing order of the jobs and that another processing order may result in a different list schedule and makespan. Specifically, the job order need not be based on the release dates.

Further, Shmoys, Wein and Williamson [1] showed that introducing unknown release dates into the scheduling model of Feldmann, Sgall and Teng will increase the competitive factor by at most a factor of 2. They also state that a natural extension of the on-line version of Graham's list scheduling [5] has a competitive factor of at most $2 - \frac{1}{m}$ in the presence of unknown release dates and unknown processing times. However, Shmoys, Wein and Williamson only discussed the sequential case ($m_j = 1$ for all $j \in \tau$).

As Garey and Graham's list scheduling algorithm yields an identical off-line performance guarantee for parallel and sequential jobs it may seem reasonable to assume the same behavior in the on-line case as well. However, to our knowledge no better competitive factor than 3 has been proven so far in the parallel job case, see Feldmann et al. [6].

2. Basic results

In this section we give a definition and some already known results that are later used to prove our main theorem. We start by introducing a simple lower bound for $C^*_{\max}(\tau)$.

$$C^*_{\max}(\tau) \geqslant \overline{C}^*_{\max}(\tau) = \max_{0 \leqslant t \leqslant \max_{j\in\tau}\{p_j + r_j\}} \left\{ t + \frac{1}{m} \sum_{j \in \tau \mid p_j + r_j > t} m_j \min\{p_j, p_j + r_j - t\} \right\}.$$

To see the validity of this lower bound consider a job j and a time t with $p_j + r_j > t > r_j$. In any valid schedule at least the workload $m_j(p_j + r_j - t)$ of job j must be executed after time t. Also note that $\overline{C}^*_{\max}(\tau) \geqslant \max_{j\in\tau}\{p_j + r_j\}$.

Next, we define non overlapping time intervals of the schedule. These intervals are called *phases*.

Definition 1. A phase P of schedule S is a time interval $[t_P, t_P + l_P)$ such that in S at least one job from τ starts or completes at times t_P and $t_P + l_P$ while no job from τ starts or completes at any other time of the interval.

If at most $\lfloor \frac{1}{2} m \rfloor$ machines are used during a phase P we say that P is *thin*.

Note that any machine is either busy or idle during a whole phase. Further, the time interval $[0, C_{\max}(S))$ can be uniquely partitioned into the phases of S.

With the help of these phases we introduce a necessary and sufficient condition for list schedules:

Lemma 2. *Assume a phase $P = [t_P, t_P + l_P)$ of a schedule S and a job $j \in \tau$ such that $C_j(S) \geqslant t_P + l_P$ and at least m_j machines are idle during P. Iff S is a list schedule then for any such pair (P, j)*

- *either job j is executed during P,*
- *or it is not released before the end of P.*

Proof. ($\Rightarrow$) In a list schedule S a machine is only left idle at any time instant if there is no unscheduled job that can be started at this time instant.

($\Leftarrow$) Sort the jobs in τ in increasing order by their starting date in S. Assume that Algorithm *List Scheduling* is using this order for the processing of jobs in its for-loop. Due to the ordering of the jobs and the conditions for P and j Algorithm *List Scheduling* cannot start job j earlier than j is started in schedule S. On the other hand, machines, that execute job j in schedule S, will not be left idle by Algorithm *List Scheduling* (not considering a possible permutation of idle and busy machines within one or several phases). Therefore, Algorithm *List Scheduling* generates schedule S. □

As already mentioned the next lemma is well known, see Feldmann, Sgall and Teng [2]. Here, we repeat the lemma in a slightly different form and also give a version of its proof.

Lemma 3. *Assume a job system τ with $r_j = 0$ for all jobs $j \in \tau$. Then for any list schedule S there is* $C_{\max}(S) \leqslant (2 - \frac{1}{m})\overline{C}^*_{\max}(\tau)$.

Proof. If list schedule S does not contain any thin phase, that is, if at least $\lceil \frac{1}{2}(m+1) \rceil$ machines are used at any time instance in S, we have

$$\overline{C}^*_{\max}(\tau) \geqslant \frac{1}{m} \sum_{j \in \tau} m_j p_j \geqslant \frac{m+1}{2m} C_{\max}(S) \geqslant \frac{m}{2m-1} C_{\max}(S) = \frac{1}{2 - \frac{1}{m}} C_{\max}(S).$$

Otherwise assume that P is the last thin phase of schedule S with $m_P > 0$ machines being used during P. Further, let job j execute during P. Because of Lemma 2 we know that

- all phases in the interval $[0, C_j(S) - p_j)$ use at least $m - m_j + 1 \geqslant m - m_P + 1$ machines,
- all phases in the interval $[C_j(S) - p_j, C_j(S))$ use at least $m_P \geqslant m_j$ machines as all jobs executing during P must also execute during all previous thin phases and
- for $C_j(S) < C_{\max}(S)$ all phases in the interval $[C_j(S), C_{\max}(S))$ use at least $m - m_P + 1$ machines as P is the last thin phase and all jobs j' starting not before the end of P have $m_{j'} \geqslant m - m_P + 1$.

Therefore, a total machine-time product of at least $m_P p_j + (m - m_P + 1)(C_{\max}(S) - p_j) \leqslant m\overline{C}^*_{\max}(\tau)$ is required for τ. If $p_j \geqslant \frac{C_{\max}(S)}{2 - \frac{1}{m-m_P+1}}$ this results in

$$\frac{C_{\max}(S)}{2 - \frac{1}{m}} \leqslant \frac{C_{\max}(S)}{2 - \frac{1}{m-m_P+1}} \leqslant p_j \leqslant \overline{C}^*_{\max}(\tau).$$

Otherwise as $2m_P < m + 1$ we have

$$\begin{aligned}
\overline{C}^*_{\max}(\tau) &\geqslant \frac{1}{m}\big((m - m_P + 1)C_{\max}(S) - (m - 2m_P + 1)p_j\big) \\
&\geqslant \left(m - m_P + 1 - \frac{m - 2m_P + 1}{2 - \frac{1}{m-m_P+1}}\right)\frac{C_{\max}(S)}{m} \\
&= \frac{m - m_P + 1}{2m - 2m_P + 1} C_{\max}(S) \geqslant \frac{1}{2 - \frac{1}{m}} C_{\max}(S). \qquad \square
\end{aligned}$$

3. Main result

We now claim that the property of Lemma 3 holds as well in the presence of unknown release dates.

Theorem 4. *Assume a job system τ. Then for any list schedule S there is* $C_{\max}(S) \leqslant (2 - \frac{1}{m})\overline{C}^*_{\max}(\tau)$.

Proof. If the first release date in a job system τ is $r > 0$ we simply subtract r from the release dates and the completion times of all jobs in any schedule S. For the resulting job system τ' and schedule S' we obtain the ratio

$$\frac{C_{\max}(S')}{\overline{C}^*_{\max}(\tau')} = \frac{C_{\max}(S) - r}{\overline{C}^*_{\max}(\tau) - r} \geqslant \frac{C_{\max}(S)}{\overline{C}^*_{\max}(\tau)} \geqslant 1.$$

Therefore, it is sufficient to consider only those job systems where the first release date is always 0.

The theorem is proven in an inductive fashion. To this end, we assume that the theorem is valid for all job systems with at most k different release dates. Note that the initial case $k = 1$ is addressed in Lemma 3. Next, we consider a list schedule S for an arbitrary job system τ with $k + 1$ release dates where 0 and $r > 0$ are the first two different release dates.

If there is a job $j \in \tau$ such that $0 < C_{j'}(S) \leqslant r_j$ for all jobs $j' \in \tau$ with $r_{j'} < r_j$ we obtain schedule S' and job system τ' from S and τ by simply removing all those jobs j' with $r_{j'} < r_j$ from τ and S. This results in $C_{\max}(S') = C_{\max}(S)$ and $\overline{C}^*_{\max}(\tau') = \overline{C}^*_{\max}(\tau)$ with τ' having at most k different release dates and a positive first release date. Therefore, we assume in the remaining parts of the proof that at least one machine is busy at any time instant in $[0, C_{\max}(S))$.

In the main part of the proof we are looking for a way to transform job system $\tau = \tau_0$ and list schedule $S = S_0$ into a new job system τ_n and a list schedule S_n such that the following two conditions are satisfied:

$$\tau_n \text{ has at most } k \text{ different release dates.} \tag{1}$$

$$\overline{C}^*_{\max}(\tau) - \overline{C}^*_{\max}(\tau_n) \geqslant r \geqslant \frac{C_{\max}(S) - C_{\max}(S_n)}{2 - \frac{1}{m}}. \tag{2}$$

Due to our induction assumption we have $C_{\max}(S_n) \leqslant (2 - \frac{1}{m})\overline{C}^*_{\max}(\tau_n)$. This yields

$$C_{\max}(S) \leqslant C_{\max}(S_n) + \left(2 - \frac{1}{m}\right)r \leqslant \left(2 - \frac{1}{m}\right)\overline{C}^*_{\max}(\tau_n) + \left(2 - \frac{1}{m}\right)r \leqslant \left(2 - \frac{1}{m}\right)\overline{C}^*_{\max}(\tau).$$

It remains to be shown that such a transformation from τ and S into τ_n and S_n exists. In order to obtain schedule S_n we repeatedly remove time intervals from schedule S. To this end we first define a basic *removal* operation: the removal of a time interval $[t_a, t_b) \subseteq P$ from schedule S, with P being a phase in S, generates a job system τ' by transforming every job $j \in \tau$ into a job j' and produces a new schedule S'. This transformation has the following properties:

$$m_{j'} = m_j, \qquad p_{j'} = \begin{cases} p_j - (t_b - t_a) & \text{if job } j \text{ executes during } P \text{ in } S, \\ p_j & \text{otherwise,} \end{cases}$$

$$r_{j'} = \begin{cases} \max\{r_j - (t_b - t_a), t_a\} & \text{if } r_j > t_a \text{ in } S, \\ r_j & \text{otherwise,} \end{cases} \qquad C_{j'}(S') = \begin{cases} C_j(S) - (t_b - t_a) & \text{if } C_j(S) \geqslant t_b, \\ C_j(S) & \text{otherwise.} \end{cases}$$

If S is a list schedule then S' is a list schedule as well due to Lemma 2. For the sake of an easier notation we allow jobs j' with $p_{j'} = 0$ in job system τ' if $r_{j'} = 0$ holds. Note that those jobs do not affect the makespan of schedule S'.

We will use one or more removal operations within a Step i to generate schedule S_i and job system τ_i from schedule S_{i-1} and job system τ_{i-1}. To express that a job $j \in \tau_{i-1}$ is transformed into job $j' \in \tau_i$ we use the notation $j' = f_i(j)$. As long as we apply only removal operations to transform job system τ into job system τ_i, every job $j' \in \tau_i$ is originally derived from a job $j \in \tau$. We denote this relation by $j = g(j')$. Further, we define the so called set of first jobs $\tilde{\tau}_i = \{j \in \tau_i | r_{g(j)} = 0\}$, that is, the set of those jobs that are originally derived from jobs with release date 0.

First, we want to satisfy condition (1). To this end we remove the time interval $[0, r)$ from S in Step 1 of the transformation and obtain list schedule S_1 and job system τ_1. This step transforms every job $j \in \tau$ with $r_j \geqslant r$ into

a job $j' = f_1(j) \in \tau_1$ with $r_{j'} = r_j - r$ and $p_{j'} = p_j$. As 0 and r are the first two different release dates of job system τ, job system τ_1 has only k different release dates.

In order to prove the validity of condition (2) we need to remove more intervals from S_1. While it is easy to see whether the inequality $\frac{C_{\max}(S)-C_{\max}(S_n)}{2-\frac{1}{m}} \leqslant r$ holds it is rather difficult to address the inequality $\overline{C}^*_{\max}(\tau) - r \geqslant \overline{C}^*_{\max}(\tau_n)$. Therefore in the latter case, we look for an equivalent formulation that is easier to handle. To this end let us assume that there is a time $\bar{t}$ in interval $[r, C_{\max}(S)]$ such that the following properties hold for job system τ_n and schedule S_n:

$$\text{For all jobs } j \in \tilde{\tau}_n \text{ there is } p_j \leqslant \max\{p_{g(j)} - r, \bar{t} - r\}. \tag{3}$$

$$\sum_{j' \in \tau} m_{j'} p_{j'} - mr \geqslant \sum_{j \in \tau_n} m_j p_j \quad \text{if } \bar{t} > r. \tag{4}$$

$$\sum_{j \in \tau_n | r_j < t} m_j \min\{p_j, t - r_j\} \geqslant mt \quad \text{for all } t \text{ with } t \leqslant \bar{t} - r \text{ provided } \bar{t} > r. \tag{5}$$

Intuitively, property (3) guarantees that after the removal operations the processing time of each job with release date 0 is either reduced by r or is not more than $\bar{t} - r$. As a removal operation cannot increase the processing time or the release date of any job, the first transformation step (for all jobs $j \in \tau_n \setminus \tilde{\tau}_n$) and property (3) (for all jobs $j \in \tilde{\tau}_n$) together yield

$$\max\{0, r_{g(j)} + p_{g(j)} - t\} \geqslant r_j + p_j - (t - r)$$

for all $t \geqslant \bar{t}$ and for each job $j \in \tau_n$ with $r_j + p_j \geqslant t - r$. This results in

$$\sum_{j' \in \tau | r_{j'} + p_{j'} \geqslant t} m_j \min\{p_{j'}, r_{j'} + p_{j'} - t\} \geqslant \sum_{j \in \tau_n | r_j + p_j \geqslant t - r} m_j \min\{p_j, r_j + p_j - (t - r)\} \tag{6}$$

for all $t \geqslant \bar{t}$. If $\bar{t} = r$ then inequality (6) already guarantees $\overline{C}^*_{\max}(\tau) - r \geqslant \overline{C}^*_{\max}(\tau_n)$.

If $\bar{t} > r$ then property (3) only addresses $t \geqslant \bar{t} - r$ in the definition of the lower bound. Therefore, we further need properties (4) and (5) to handle $t < \bar{t} - r$. First note that $\sum_{j \in \tau_n} m_j p_j = \sum_{j \in \tau_n | r_j < t} m_j \min\{p_j, t - r_j\} + \sum_{j \in \tau_n | r_j + p_j \geqslant t} m_j \min\{p_j, r_j + p_j - t\}$ holds for all $t \geqslant 0$. Then properties (4) and (5) yield

$$\begin{aligned}
\overline{C}^*_{\max}(\tau) - r &\geqslant \frac{1}{m} \sum_{j' \in \tau} m_{j'} p_{j'} - r \geqslant \frac{1}{m} \sum_{j \in \tau_n} m_j p_j \\
&= \frac{1}{m} \left(\sum_{j \in \tau_n | r_j < t} m_j \min\{p_j, t - r_j\} + \sum_{j \in \tau_n | r_j + p_j \geqslant t} m_j \min\{p_j, p_j + r_j - t\} \right) \\
&\geqslant \max_{0 \leqslant t' \leqslant \bar{t} - r} \left\{ t' + \frac{1}{m} \sum_{j \in \tau_n | r_j + p_j \geqslant t'} m_j \min\{p_j, r_j + p_j - t'\} \right\}.
\end{aligned} \tag{7}$$

Therefore, inequalities (6) and (7) combined yield $\overline{C}^*_{\max}(\tau) - r \geqslant \overline{C}^*_{\max}(\tau_n)$. Hence, condition (2) becomes valid if such a $\bar{t}$ exists and if $(2 - \frac{1}{m})r \geqslant C_{\max}(S) - C_{\max}(S_n)$ holds.

As already mentioned such a $\bar{t}$ may not exist for schedule S_1. As $C_{\max}(S) - C_{\max}(S_1) = r$ we may remove more time intervals with a combined length of up to $(1 - \frac{1}{m})r$ from S_1 such that a suitable $\bar{t}$ can be found. Before describing those removal steps in detail we introduce another definition with respect to job system τ_i and the corresponding list schedule S_i:

Definition 5. A job $j \in \tilde{\tau}_i$ is called *dominant* or *strongly dominant* in schedule S_i if $2p_j \geqslant C_j(S_i)$ or $2p_j > C_j(S_i)$ holds, respectively. $t_i^t = \max_{j \in \tilde{\tau}_i | 2p_j \geqslant C_j(S_i)} \{C_j(S_i) - p_j\}$ denotes the highest starting time of any dominant job in schedule S_i.

Note that S_i always contains a dominant job as we allow jobs with processing time 0 in this proof. Further, if there is a job $j \in \tilde{\tau}_i$ that starts before t_i^t in schedule S_i we use $t_i^b = \max_{j \in \tilde{\tau}_i | C_j(S_i) - p_j < t_i^t} \{C_j(S_i) - p_j\}$ to describe the highest starting time of any such job.

We briefly discuss the meaning of strongly dominant jobs in the context of our proof. If a job $j \in \tilde{\tau}_n$ is not strongly dominant in schedule S_n then the selection $\bar{t} = p_j + r$ guarantees the validity of property (5) as j does not start before $\bar{t} - r$ in schedule S_n and at least $m + 1 - m_j$ machines must be busy at any time instance up to the starting time of j in S_n, see Lemma 2. Therefore, we can simply pick the job with the largest processing time among those jobs for the selection of $\bar{t}$. Of course property (3) is also correct for all those jobs. However, we still need to make sure that property (3) holds for all strongly dominant jobs $j \in \tilde{\tau}_n$.

As we further require $\bar{t} \geqslant t_i^t + r$, we execute the following series of steps until subsequent condition is valid for schedule S_i:

There is no strongly dominant job $j \in \tilde{\tau}_i$ in schedule S_i with $p_j > \max\{t_i^t, p_{g(j)} - r\}$. (8)

If condition (8) does not hold for schedule S_i we pick the job $j_t \in \tilde{\tau}_i$ starting at time t_i^t in S_i with the largest processing time. We need to consider two cases:

1. If j_t is strongly dominant we remove the time interval $[t_i^t, t_i^t + x)$ in Step $i + 1$ to obtain τ_{i+1} and S_{i+1}. x is the smallest value such that anyone of the following conditions holds:
   - $x = p_{j_t} - t_i^t$, that is, $f_{i+1}(j_t)$ is dominant but not strongly dominant in schedule S_{i+1}.
   - $t_{i+1}^t > t_i^t$.
   - Condition (8) is valid for schedule S_{i+1}.

   This step results in $p_{j_t} - p_{f_{i+1}(j_t)} = x$. Note that $t_{i+1}^t \geqslant t_i^t$ holds as $f_{i+1}(j_t)$ is always a dominant job. If $t_{i+1}^t > t_i^t$ then no strongly dominant job can start in the time interval $(t_i^t, t_{i+1}^t]$ in schedule S_{i+1} as otherwise the condition $t_{i+1}^t > t_i^t$ can be satisfied with a smaller value of x.
2. If j_t is not strongly dominant, that is, if there is no strongly dominant job starting at time t_i^t in schedule S_i, then there is a time t_i^b in schedule S_i as $\tilde{\tau}_i$ must contain a strongly dominant job in S_i. We pick any job $j_b \in \tilde{\tau}_i$ that starts at time t_i^b and remove the time intervals $[t_i^b, t_i^b + \frac{1}{2}x)$ and $[t_i^t, t_i^t + \frac{1}{2}x)$ to obtain τ_{i+1} and S_{i+1}. Again, x is the smallest value such that anyone of the following conditions holds:
   - $x = 2p_{j_b}$, that is, $p_{f_{i+1}(j_b)} = 0$.
   - $x = 2(t_i^t - t_i^b)$, that is, both removed intervals merge.
   - $t_{i+1}^t > t_i^t - \frac{1}{2}x$.
   - Condition (8) is valid for schedule S_{i+1}.

   Again, $f_{i+1}(j_t)$ is always a dominant job. Therefore, $t_{i+1}^t \geqslant t_i^t - \frac{1}{2}x$. If $t_{i+1}^t > t_i^t - \frac{1}{2}x$ then no strongly dominant job can start in the time interval $(t_i^b \leqslant t_i^t - \frac{1}{2}x, t_{i+1}^t]$ in schedule S_{i+1} as otherwise the condition $t_{i+1}^t > t_i^t - \frac{1}{2}x$ can be satisfied with a smaller value of x.

Intuitively we can say that our target group consist those strongly dominant jobs that prevent the validity of condition (8). Each job in the target group completes after time t_i^t in schedule S_i. Let us discuss what happens to this target group when a removal step is executed. First note that no strongly dominant job can start in schedule S_i after the beginning of any interval removed in Step $i + 1$. Further, any time instance $t > t_{i+1}^t$ in schedule S_{i+1} corresponds with a time instance in schedule S_i that comes after the end of any interval removed in Step i. Hence, if there is a strongly dominant job $j' = f_{i+1}(j) \in \tilde{\tau}_{i+1}$ in schedule S_{i+1} with $C_{j'}(S_{i+1}) > t_{i+1}^t$ then its ancestor $j \in \tilde{\tau}_i$ is strongly dominant in S_i with $C_j(S_i) > t_i^t$ and $p_j = p_{j'} + x$. Therefore, a removal step cannot increase the number of jobs in our target group and the processing time of each job in the target group after this step is reduced by the combined length of the removed intervals in this step in comparison to its ancestor job.

Assume that condition (8) holds for schedule $S_{i'}$. Then we have $l' = C_{\max}(S) - C_{\max}(S_{i'}) = C_{\max}(S_1) + r - C_{\max}(S_{i'}) \leqslant 2r$ as the combined length of the time intervals removed in Steps 2 to i' is at most r due to our observations above.

Finally, we again consider two cases:

1. $l' > (2-\frac{1}{m})r$. In this case the first inequality of condition (2) does not hold as we have removed too much from schedule S. Therefore, we introduce an additional job $\hat{j}$ with $m_{\hat{j}} = m$, $r_{\hat{j}} = 0$ and $p_{\hat{j}} = l' - (2-\frac{1}{m})r$ to $\tau_{i'}$ such that $\hat{j}$ is always considered last in the job order of Algorithm *List Scheduling*. This generates $\tau_{i'+1} = \tau_n$ and $S_{i'+1} = S_n$. Remember that at most $m-1$ machines can be idle at any moment in $[0, C_{\max}(S))$. This property also holds for any schedule S_i with $1 \leqslant i \leqslant i'$. Therefore, $\hat{j}$ will be started at time $C_{\max}(S_{i'})$ in schedule S_n resulting in $C_{\max}(S) - C_{\max}(S_n) = (2-\frac{1}{m})r$. Note that job $\hat{j}$ will not affect the completion time of any other job in S_n. For the sake of easier handling we define $p_{g(\hat{j})} = 0$, $m_{g(\hat{j})} = m$ and $r_{g(\hat{j})} = 0$. Further, we assume that $g(\hat{j})$ is a job in τ although it has a processing time of 0. Therefore, we have $\hat{j} \in \tilde{\tau}_n$. For the validity of condition (2) it remains to be shown that properties (3) to (5) hold.
We set $\bar{t} = r + \max\{t^t_{i'}, \max_{j\in\tilde{\tau}_n | C_j(S_n)-p_j > t^t_{i'}}\{p_j\}\}$. Property (3) holds as there is $p_j \leqslant \max\{t^t_{i'}, p_{g(j)} - r\}$ for all jobs in $\tilde{\tau}_{i'}$ that do not start after time $t^t_{i'}$ in $S_{i'}$ due to the validity of condition (8) and $p_j \leqslant \bar{t} - r$ for job $\hat{j}$ and for all jobs in $\tilde{\tau}_{i'}$ that start later than $t^t_{i'}$.
Let job $j \in \tilde{\tau}_{i'}$ be the last job that satisfies condition (8) in Step i'. Then we have $p_{g(j)} - p_j = r$ as it is otherwise possible to satisfy condition (8) in Step i' by removing a smaller time interval. Remember that job $g(j)$ executes in S during all time intervals that are later removed in Steps 2 to i' and that the combined length of these time intervals is $l' - r$. Therefore, job $g(j)$ does not start before time $l' - (p_{g(j)} - p_j) = l' - r$ in schedule S. Hence, at least $m - m_j + 1$ machines are busy at any time instance in time interval $[0, l' - r)$ of schedule S. Therefore, jobs $j' \in \tau$ with $r_{j'} = 0$ use at least a machine-time product of

$$rm_j + (l'-r)(m+1-m_j) = (l'-r)m + r + (2r-l')(m_j - 1) \geqslant (l'-r)m + r,$$

in the removed time intervals of schedule S resulting in

$$\sum_{j\in\tau | r_j = 0} m_j p_j - \sum_{j'\in\tilde{\tau}_n} m_{j'} p_{j'} \geqslant (l'-r)m + r - \left(l' - \left(2-\frac{1}{m}\right)r\right)m = rm.$$

This leads to $\sum_{j\in\tau} m_j p_j - mr \geqslant \sum_{j'\in\tau_n} m_{j'} p_{j'}$ (property (4)).
Property (5) holds if $p_{\hat{j}} = \bar{t} - r$. Otherwise there is at least one job $j' \in \tilde{\tau}_n$ with $p_{j'} \geqslant \bar{t} - r$ that does not start before $\bar{t} - r$. Remember that at least one dominant job starts at time $t^t_{i'}$. Then at least $m + 1 - m_{j'}$ machines are used at any moment in schedule S_n during the time interval $[0, \bar{t} - r)$. Therefore, property (5) is valid as well.
2. $l' \leqslant (2-\frac{1}{m})r$. In this case we may not have removed enough intervals so that property (4) holds. On the other hand, we may also not be able to set $\bar{t} = r$ in order to ignore properties (4) and (5). Therefore, more intervals must be removed. To this end we replace condition (8) with the following condition (9) and use the same removing method as in Steps 2 to i':

$$C_{\max}(S) - C_{\max}(S_i) = \left(2-\frac{1}{m}\right)r \quad \text{or}$$
$$p_{g(j)} - r < p_j \text{ holds for every strongly dominant job } j \in \tilde{\tau}_i \text{ in schedule } S_i. \tag{9}$$

Assume that condition (9) holds for schedule $S_{i''}$. If $l'' = C_{\max}(S) - C_{\max}(S_{i''}) = (2-\frac{1}{m})r$ we set $n = i''$ and prove the validity of properties (3) and (5) in the same way as in the first case.
As condition (9) holds for schedule $S_{i''}$ there is a (dominant) job $j \in \tilde{\tau}_{i''}$ with $p_j \geqslant p_{g(j)} - r$. Otherwise it is possible to satisfy condition (9) by removing at least one smaller interval. Next consider all intervals of schedule S that are removed in Steps 2 to i''. If $g(j)$ starts after time r in schedule S then each of those intervals also starts after time r in S, that is, each of those intervals contains at least one job with release date 0 that starts after time r in S. If job $g(j)$ does not start later than time r in S then an interval of schedule S_i contains a job j' with $g(j') = g(j)$ if it is removed in Step $i+1$ and starts at time 0 in schedule S_i. Remember that an interval of schedule S_i contains a job $j' \in \tilde{\tau}_i$ such that $g(j')$ starts after time r in schedule S if the interval is removed

in Step i and begins after time 0 in schedule S_i. Hence, we have 4 possible types of intervals that are removed in Steps 1 to i'':

- *Type* 1: An interval that is in $[0, r)$ and does not contain job $g(j)$.
- *Type* 2: An interval that is in $[0, r)$ and contains job $g(j)$.
- *Type* 3: Intervals that are not in $[0, r)$ and do not contain any jobs with release date 0 that start after time r in schedule S.
- *Type* 4: Intervals that are not in $[0, r)$ and contain at least one job with release date 0 that starts after time r in schedule S.

Assume that at least m_i machines are busy at any time instance of a Type i interval and that l_i is the combined length of all Type i intervals. Then, the following inequalities hold: $m_1 + m_3 \geqslant m + 1$, $m_1 + m_4 \geqslant m + 1$, $m_2 + m_4 \geqslant m + 1$, $l_1 + l_2 = r$, $l_2 + l_3 \leqslant p_{g(j)} - p_j \leqslant r$ and $l_3 + l_4 = (1 - \frac{1}{m})r$. Therefore, jobs in the removed time intervals of schedule S use at least a machine-time product of

$$\begin{aligned}\sum_{i=1}^{4} l_i m_i &\geqslant l_3(m_1 + m_3) + (r - l_3)\min\{m_1, m_2\} + \left(\left(1 - \frac{1}{m}\right)r - l_3\right)m_4 \\ &\geqslant l_3(m+1) + (r - l_3)(m+1) - \frac{1}{m}rm_4 \\ &= rm + \left(1 - \frac{m_4}{m}\right)r \geqslant rm.\end{aligned}$$

Hence, property (4) holds as well.

Otherwise if $l'' < (2 - \frac{1}{m})r$ then property (3) holds for all strongly dominant jobs. But for the validity of property (4) we need to remove even more intervals without generating new strongly dominant jobs. To this end we pick a job $j \in \tilde{\tau}_{i''}$ with the highest starting time $t' > 0$ in $S_{i''}$ among all jobs from $\tilde{\tau}_{i''}$ and remove the smallest time interval $[t', t' + x)$ such that either $x = p_j$ (job j is completely removed) or $C_{\max}(S) - C_{\max}(S_{i''+1}) = (2 - \frac{1}{m})r$. Step $i'' + 1$ cannot generate a new dominant job. If $C_{\max}(S) - C_{\max}(S_{i''+1}) < (2 - \frac{1}{m})r$ we repeat this step with schedule $S_{i''+1}$ until we eventually obtain a job system τ_n and a schedule S_n such that either $C_{\max}(S) - C_{\max}(S_n) = (2 - \frac{1}{m})r$ or there is no job $j \in \tilde{\tau}_n$ that starts later than time 0 in schedule S_n.

If no job $j \in \tilde{\tau}_n$ starts later than time 0 in S_n then we set $\bar{t} = r$ and are done as properties (4) and (5) are not relevant.

If $C_{\max}(S) - C_{\max}(S_n) = (2 - \frac{1}{m})r$ we set $\bar{t} = r + \min\{t_n^t, \max_{j \in \tilde{\tau}_n | C_j(S_n) - p_j > t_n^t}\{p_j\}\}$. If $\bar{t} > r$ then property (5) holds for the same reasons already explained in the first case. Finally, the validity of property (4) is shown in exactly the same way as described above as all intervals of schedule S, that are removed in Steps $i'' + 1$ to n, contain at least one job with release date 0 that starts after time r in S. □

References

[1] D. Shmoys, J. Wein, D. Williamson, Scheduling parallel machines on-line, SIAM J. Comput. 24 (6) (1995) 1313–1331.
[2] A. Feldmann, J. Sgall, S.-H. Teng, Dynamic scheduling on parallel machines, Theoret. Comput. Sci. 130 (1994) 49–72.
[3] M. Garey, R.L. Graham, Bounds for multiprocessor scheduling with resource constraints, SIAM J. Comput. 4 (2) (1975) 187–200.
[4] R.L. Graham, Bounds for certain multiprocessor anomalies, Bell System Technical J. 45 (1966) 1563–1581.
[5] R.L. Graham, Bounds on multiprocessor timing anomalies, SIAM J. Appl. Math. 17 (1969) 416–429.
[6] A. Feldmann, B. Maggs, J. Sgall, D.D. Sleator, A. Tomkins, Competitive analysis of call admission algorithms that allow delay, Technical Report CMU-CS-95-102, Carnegie-Mellon University, Pittsburgh, PA, 1995.