Special Cases of Online Parallel Job Scheduling

Johann L. Hurink and Jacob Jan Paulus

University of Twente, P.O. box 217,
7500AE Enschede, The Netherlands,
j.j.paulus@utwente.nl

Beta working paper WP-235
November 26, 2007

Abstract

In this paper we consider the online scheduling of jobs, which require processing on a number of machines simultaneously. These jobs are presented to a decision maker one by one, where the next job becomes known as soon as the current job is scheduled. The objective is to minimize the makespan. For the problem with three machines we give a 2.8-competitive algorithm, improving upon the 3-competitive greedy algorithm. For the special case with arbitrary number of machines, where the jobs appear in non-increasing order of machine requirement, we give a 2.4815-competitive algorithm, improving the 2.75-competitive greedy algorithm.

1 Introduction

In this paper we consider two special cases of online scheduling jobs which require processing on a number of machines simultaneously (parallel jobs). Jobs are characterized by their processing time p_j and the number of machines m_j simultaneously required for processing, and are presented one by one to a decision maker. As soon as a job becomes known, it has to be scheduled irrevocably (i.e. its start time has to be set) without knowledge of successive jobs. Preemption is not allowed and the objective is to minimize the makespan. We study two special cases of this online problem. First we consider the case with three machines and, next, the case where jobs appear in non-increasing order of machine requirement.

In contrast to an online algorithm, an *offline* scheduling algorithm has complete knowledge of the list of jobs to construct the optimal offline schedule. This optimal offline objective value is used to measure the quality of online algorithms. An online algorithm is ρ-competitive if for any list of jobs it produces a schedule with makespan at most ρ times the makespan of the optimal offline schedule.

Determining the competitive ratio can be seen as a game between the online scheduling algorithm and an adversary who determines the characteristics of the jobs in the list and the length of this list. The online algorithm tries to schedule the jobs such that the competitive ratio is minimized, while the adversary aims to maximize the competitive ratio of the online algorithm.

An online problem is called semi-online if there is some a priori knowledge of the list of jobs, e.g., the jobs appear in non-increasing order of machine requirement. Because of such knowledge smaller competitive ratios might be obtained.

Besides the mentioned online model (called the *online-list* model), other online models are considered in the literature. One important model is the *online-time* model. In this model jobs have a release date and become known to the online algorithm when the online schedule has been executed upto this release date. However, the online algorithm does not have to make a decision on a job directly at its release date, only scheduling decisions before the current point in time are irrevocable. In this model, the optimal offline schedule is also restricted by the release dates. For more online models and background on online scheduling we refer to [11].

Using the three-field notation originating from [5], the considered problem is denoted by $P|\text{online}-\text{list}, m_j|C_{\max}$, see also [8, 11]. In the literature the concept of parallel jobs is known by many different names, such as *parallel tasks*, *parallelizable tasks*, *multiprocessor tasks*, *multiple-job-on-one-processor*, and *1-job-on-r-processors*. In some literature the machine requirement m_j of a job is called the width or the size of a job. And in stead of m_j the term $size_j$ or simply s_j is used to denote the parallel machine requirement of job j.

There is a great deal of similarity between $P|\text{online}-\text{list}, m_j|C_{\max}$ and the online orthogonal strip packing problem. The orthogonal strip packing problem is a two-dimensional packing problem. Without rotation rectangles have to be packed on a strip with fixed width and unbounded height. The objective is to minimize the height of the strip in which the rectangles are packed. In the online setting one rectangle is presented after the other and has to be assigned without knowledge of successive rectangles. To see the similarity, let each machine correspond to one unit of the width of the strip, and time to the height of the strip. The width of a rectangle j corresponds to the machine requirement of job j and its height to the processing time. Minimizing the height of the strip used is equivalent to minimizing the makespan of the machine scheduling problem. The difference lies in the choice of machines. In $P|\text{online}-\text{list}, m_j|C_{\max}$ any m_j machines suffice for job j, where rectangles can not be split up into several rectangles together having width m_j. Therefore, algorithms for strip packing can be used for parallel job scheduling [6], but in general not the other way around.

The most simple online algorithm for the considered problem is a greedy algorithm. This algorithm schedules each job j at the earliest time t for which at any time in the interval $[t, t+p_j)$ at least m_j machines are available. Unfortunately, for the online scheduling of parallel jobs the greedy algorithm has no constant competitive ratio, as illustrated by the following instance with m machines and $2m$ jobs. The odd jobs have processing time $p_j = 1 + \frac{1}{2}\epsilon(j+1)$ and machine requirement $m_j = 1$ and the even jobs have processing time $p_j = \epsilon$ and machine requirement $m_j = m$. The optimal schedule has length $1 + 2\epsilon m$ and the 'greedy schedule' has makespan $\epsilon m + \sum_{i=1}^{m}(1+\epsilon i)$, see Figure 1. For ϵ going to 0, this results in a competitive ratio of m. On the other hand, as in the online schedule there is at any point in time at least one machine processing a job, the competitive ratio of a greedy algorithm is also at most m.

Given the above observation, a greedy strategy does not seem to be a good one. Nevertheless, for the two special case of the online parallel job scheduling problem considered in this paper, the best known algorithms up to now are greedy algorithms. Furthermore, the improved algorithms presented, also have some greedy component.

In the following we give an overview of the current state of the research on problem $P|\text{online}-\text{list}, m_j|C_{\max}$ and its various semi-online versions. The results are summarized in Table 1. The first online algorithm for online parallel job scheduling with a constant competitive ratio is presented in [8] and is 12-competitive. In [15], an improvement to a 7-competitive algorithm is given. This *dynamic waiting algorithm* schedules jobs with a small machine requirement greedily and delays the jobs with a large machine requirement.

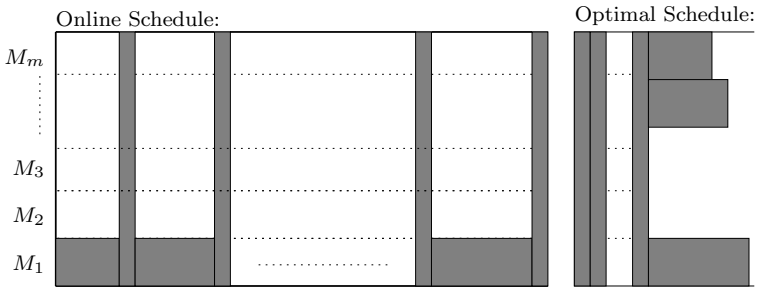

Figure 1: A greedy algorithm is no better than m-competitive.

For the strip packing problem in [1] a 6.99-competitive online algorithm is given under the assumption that jobs have a processing time of at most 1. This *shelf algorithm* groups rectangles of similar height together. The currently best known algorithm is designed by combining the ideas of separating jobs with large and small machine requirement, and using a shelf structure. This results in a 6.6623-competitive algorithm which is independently obtained in [6] and [13], and due to its structure it can be applied to online orthogonal strip packing as well.

The best known analytical lower bound on the competitive ratio for $P|\text{online}-\text{list}, m_j|C_{\max}$ is a bound of 2 resulting from the strip packing problem [2], which applies directly to the parallel job problem with $m \geq 3$. In [7] a tight lower bound of 2 is given for the two machines case. Furthermore, a computerized proof, based on an ILP-formulation, resulting in a lower bound of 2.43 for $P|\text{online}-\text{list}, m_j|C_{\max}$ is given.

Until now, the best known algorithm for the case with 3 machines is the 3-competitive greedy algorithm. In this paper we propose an improved algorithm:

Theorem 1 *For* $P3|\text{online}-\text{list}, m_j|C_{\max}$ *a* 2.8*-competitive algorithm exists.*

In the literature a number of semi-online variants of online parallel job scheduling are considered. In case the jobs appear in non-increasing order of machine requirement the best known lower bound is 1.88 from classical parallel machine scheduling, i.e. this bound uses only jobs with $m_j = 1$ [12]. Furthermore, for this case in [14] it is shown, that greedy scheduling the jobs is 2.75-competitive and no better than 2.5-competitive. In this paper we show that slightly modifying the greedy algorithm yields a better algorithm.

Theorem 2 *For* $P|\text{online}-\text{list}, m_j|C_{\max}$ *with jobs appearing in non-increasing order of machine requirement, a* 2.4815*-competitive algorithm exists.*

Furthermore, we show that for 2 and 3 machines and jobs appearing in non-increasing order of machine requirement the greedy algorithm is $(2-\frac{1}{m})$-competitive. As we know from classical parallel machine scheduling [4], this is the best possible; these bounds are tight. Finally, we show that for 4 and 5 machines greedy is 2-competitive.

In case the jobs appear in non-increasing order of processing time a greedy algorithm is 2-competitive [14]. The best know lower bound on the competitive ratio is $\frac{5}{3}$ from the strip packing problem [2]. For the two machine case with non-increasing processing times a lower bound of $\frac{9}{7}$ and a $\frac{4}{3}$-competitive online algorithm are known [3]. For the case where jobs appear in non-decreasing order of processing times and two machines, an optimal (best possible ratio) $\frac{3}{2}$-competitive algorithm is given in [3]. Optimality follows from a lower bound from classical parallel machine scheduling [4].

For the semi-online case where jobs appear in non-increasing order of processing time and also non-increasing order of machine requirement, a generalization of the the First-Fit-Decreasing heuristic for 1-dimensional bin-packing developed in [9], gives a 1.875-competitive algorithm. If $m_j \leq \frac{m}{2}$ for all j, the algorithm is in fact 1.75-competitive [10]. This result is improved in [16], showing that the algorithm is 1.75-competitive for arbitrary machine requirements. No special lower bound are known.

If the jobs appear in non-decreasing order of processing time and also in non-decreasing order of machine requirement, a lower bound of $\frac{1+\sqrt{7}}{2} > 1.82$ is given in [2]. No special upper bounds are known.

$P\|\text{online} - \text{list}, m_j\|C_{\max}$		
Model	Lower Bound	Upper Bound
-	2.43, [7]	6.6623, [6, 13]
$m = 2$	2, [7]	2, (Greedy)
$m = 3$	2, [2]	2.8, (This paper)
$3 \leq m \leq 6$	2, [2]	m, (Greedy)
Semi-online $P\|online - list, m_j\|C_{max}$		
Model	Lower Bound	Upper Bound
-non-increasing m_j	1.88, [12]	2.4815, (This paper)
$m = 2$ or 3	$2 - \frac{1}{m}$, [4]	$2 - \frac{1}{m}$ (Greedy)
$m = 4$ or 5	-	2 (Greedy)
-non-increasing p_j	$\frac{5}{3}$, [2]	2, [14]
$m = 2$	$\frac{9}{7}$, [3]	$\frac{4}{3}$, [3]
-non-decreasing p_j	-	-
$m = 2$	$\frac{3}{2}$, [4]	$\frac{3}{2}$, [3]
-non-increasing p_j and m_j	$\frac{4}{3}$ $(m_j = 1)$	$\frac{7}{4}$, [16]
-non-decreasing p_j and m_j	$\frac{1+\sqrt{7}}{2}$, [2]	-

Table 1: Results on online scheduling of $P|\text{online} - \text{list}, m_j|C_{\max}$

The results, summarized in Table 1, show that in only a few special cases the gap between the lower and upper bound on the competitive ratio is closed. In particular the gap for the general problem $P|\text{online} - \text{list}, m_j|C_{\max}$ is large.

In Section 2 we introduce some notation and basic results. Sections 3 and 4 deal with the special cases of online parallel job scheduling with three machines and jobs appearing in non-increasing order of machine requirement, respectively.

2 Bounding the offline solution

To be able to derive a bound on the competitive ratio of an online algorithm, the online solutions have to be compared with the optimal offline solutions. However, mostly the solutions are not compared to the actual optimal offline solution but to lower bounds on the values of these solutions, i.e. inequalities like the following are derived:

$$C_A \leq \rho \text{ (Lower Bound on } C^*) \leq \rho C^* \ ,$$

where C_A denotes the makespan of the schedule created by online algorithm A and C^* the makespan of the optimal offline schedule. The value ρ is the resulting competitive

ratio.

There are two straightforward lower bounds on the optimal offline makespan for the parallel job scheduling problem. The first is a simple length argument, i.e. the longest job has to be processed:

$$\max_j\{p_j\} \leq C^* \ . \tag{1}$$

The second is a load argument, i.e. all jobs need to be processed and are at best evenly divided over the m machines:

$$\frac{1}{m}\sum_j m_j p_j \leq C^* \ . \tag{2}$$

To obtain the results in this paper, it is a key issue to combine these two lower bound and improve them where possible. Improvements are found by studying the structure of the online and offline schedules.

3 Parallel job scheduling with 3 machines

In this section we present a 2.8-competitive algorithm for online parallel job scheduling with 3 machines, $P3|\text{online}-\text{list}, m_j|C_{max}$. Till now, the best known algorithm is the 3-competitive greedy algorithm and the best lower bound on the competitive ratio is 2.

Algorithm 3M
Schedules job j by the following rules:

- If $m_j = 1$ or 2, then schedule job j in a greedy fashion.
- If $m_j = 3$, consider:
  - if there is an empty interval within the current online schedule large enough to accommodate for job j, then schedule job j in the first of these intervals and as late as possible within that interval. (This will be immediately before another job with machine requirement 3.)
  - else, if the last job in the schedule has machine requirement of 3, then concatenate job j to this job at the back.
  - else, delay job j for a period d (which we define later) after the last scheduled job.

This algorithm differs from the greedy algorithm only by the way of scheduling jobs which need all three machines for processing. Each schedule for the 3 machine problem consists of intervals of four different types: *full* intervals F, *high* loaded intervals H, *low* loaded intervals L, and *empty* intervals E. The F-intervals contain the jobs with $m_j = 3$, the H-intervals are the intervals containing jobs with $m_j = 2$ or 1 and in which at least 2 machines busy, the L-intervals are the intervals which have exactly 1 machine busy (and, thus, contain only jobs with $m_j = 1$), and the E-intervals are the intervals with no machine busy.

Using this classification, each online schedule created by Algorithm 3M can be partitioned into consecutive blocks where the i^{th} block consists of four consecutive intervals H_i, L_i, E_i, F_i, where some of the intervals H_i, L_i or E_i may be empty. Since we schedule jobs with $m_j = 1$ and $m_j = 2$ in a greedy fashion, the interval H_i, L_i and E_i always occur in this order between two consecutive non-empty F-intervals F_{i-1} and F_i. We use

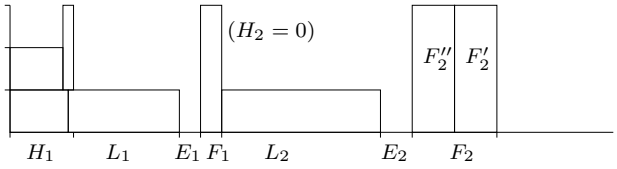

Figure 2: Structure of an online schedule created by Algorithm 3M.

the terms H_i, L_i, E_i, F_i to indicate both the interval and to indicate the length of the respective interval. In Figure 2 an example of the structure of an online schedule is given.

Each interval F_i contains one job that was delayed. This job is that job of F_i which was scheduled first by the online algorithm. Let this job together with all jobs concatenated after it form the interval F_i', and let the jobs that are concatenated before this job form the interval F_i''. Thus, $F_i = F_i'' + F_i'$ (see Figure 2).

Now consider the situation that a job with $m_j = 3$ is delayed by Algorithm 3M. At that moment the online schedule ends with an H_i or L_i interval. We define the delay d for this job as $\left(\frac{1}{2}L_i - \frac{1}{4}H_i\right)^+ := \max\{0, \left(\frac{1}{2}L_i - \frac{1}{4}H_i\right)\}$. As soon as this job is scheduled, we have created the interval E_i of length d, and F_i consists only of the last job scheduled. During the course of the algorithm E_i may decrease in length and F_i may increase in length (but not vice versa). With $\tilde{H}_i$, $\tilde{L}_i$, and $\tilde{E}_i$ we refer to the values of H_i, L_i, and E_i at the moment that interval F_i is created.

In the following, we evaluate a given online schedule created by Algorithm 3M. Let n be the number of F_i intervals in the online schedule. The makespan $C_{3\mathrm{M}}$ of the online schedule is given by

$$C_{3\mathrm{M}} = \sum_{i=1}^{n} (H_i + L_i + E_i + F_i) + H_{n+1} + L_{n+1} \ ,$$

where H_{n+1} and L_{n+1} may have length 0. To get a more helpful description of the makespan, we introduce intervals I_i by

$$I_i := H_i \cup L_i \cup E_i \cup F_i'' \cup F_{i+1}'$$

(see Figure 3).

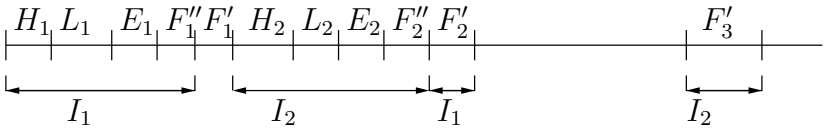

Figure 3: Definition of I_i.

Using this definition, the makespan $C_{3\mathrm{M}}$ can be expressed by

$$C_{3\mathrm{M}} = \sum_{i}^{n-1} I_i + F_1' + H_n + L_n + E_n + F_n'' + H_{n+1} + L_{n+1} \ . \tag{3}$$

Now, let $l(t)$ be the load (the number of machines in use) at time t in the schedule. The total load of the schedule in I_i can be bounded from below by the following lemma:

Lemma 1 *For $i \leq n-1$ we have:*

$$\int_{I_i} l(t)dt > \frac{5}{3}I_i - \frac{5}{3}F'_{i+1} \ .$$

Proof: The definition of interval I_i implies:

$$\begin{aligned}\int_{I_i} l(t)dt &= \int_{H_i} l(t)dt + \int_{L_i} l(t)dt + \int_{F''_i} l(t)dt + \int_{F'_{i+1}} l(t)dt \\ &\geq 2H_i + L_i + 3F''_i + 3F'_{i+1} \\ &= \frac{5}{3}I_i + \frac{1}{3}H_i - \frac{2}{3}L_i - \frac{5}{3}E_i + \frac{4}{3}\left(F''_i + F'_{i+1}\right) \ . \end{aligned} \tag{4}$$

At the time the first job in F'_{i+1} was placed, it had to be delayed since it did not fit in the empty intervals before F_i. Together with the fact that E_i is non-increasing this yields

$$F'_{i+1} > E_i \ . \tag{5}$$

There are two cases to distinguish.

Case 1: $H_i + L_i = \tilde{H}_i + \tilde{L}_i$

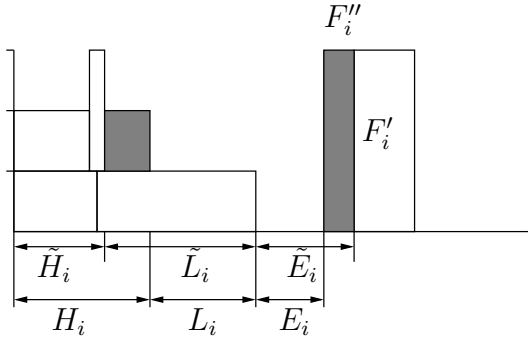

Figure 4: In the case $H_i + L_i = \tilde{H}_i + \tilde{L}_i$

This implies that $\tilde{E}_i$ has not decreased due to the insertion of jobs with $m_j = 1$ or 2, i.e. $\tilde{E}_i = E_i + F''_i <_{(5)} F'_{i+1} + F''_i$. Since furthermore $H_i \geq \tilde{H}_i$ and $L_i \leq \tilde{L}_i$ (see Figure 4), we get

$$\begin{aligned}\int_{I_i} l(t)dt &\geq \frac{5}{3}I_i + \frac{1}{3}\tilde{H}_i - \frac{2}{3}\tilde{L}_i - \frac{5}{3}E_i + \frac{4}{3}\left(F''_i + F'_{i+1}\right) \\ &> \frac{5}{3}I_i + \frac{1}{3}\tilde{H}_i - \frac{2}{3}\tilde{L}_i - \frac{5}{3}E_i + 3\tilde{E}_i - \frac{5}{3}\left(F''_i + F'_{i+1}\right) \\ &=_{(-E_i - F''_i = -\tilde{E}_i)} \frac{5}{3}I_i + \frac{1}{3}\tilde{H}_i - \frac{2}{3}\tilde{L}_i + \frac{4}{3}\tilde{E}_i - \frac{5}{3}F'_{i+1} \ . \end{aligned}$$

Since $\tilde{E}_i$ is equal to the delay of the first scheduled job of F_i, we have $\tilde{E}_i = (\frac{1}{2}\tilde{L}_i - \frac{1}{4}\tilde{H}_i)^+$, and thus,

$$\int_{I_i} l(t)dt > \frac{5}{3}I_i - \frac{5}{3}F'_{i+1} \ .$$

Case 2: $H_i + L_i > \tilde{H}_i + \tilde{L}_i$

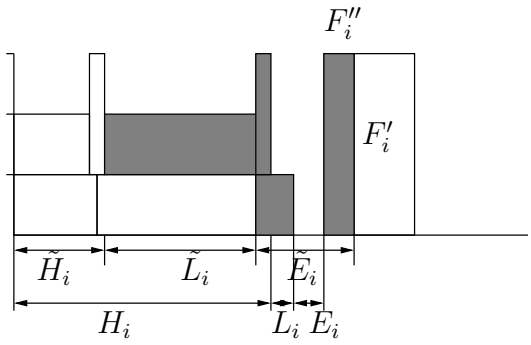

Figure 5: In the case $H_i + L_i > \tilde{H}_i + \tilde{L}_i$

In this case $\tilde{E}_i > 0$ and $\tilde{E}_i$ has been decreased (partially) due to the insertion of $m_j = 1$ or $m_j = 2$ jobs. Due to the greedy nature of Algorithm A this can happen only if the whole interval $\tilde{L}_i$ becomes part of H_i (see Figure 5).

Defining $H_i = \tilde{H}_i + \tilde{L}_i + e$ we get $L_i = \tilde{E}_i - e - E_i - F_i''$. Starting from (4), we have

$$\begin{aligned}
\int_{I_i} l(t)dt &\geq \frac{5}{3}I_i + \frac{1}{3}H_i - \frac{2}{3}L_i - \frac{5}{3}E_i + \frac{4}{3}\left(F_i'' + F_{i+1}'\right) \\
&= \frac{5}{3}I_i + \frac{1}{3}(\tilde{H}_i + \tilde{L}_i + e) - \frac{2}{3}(\tilde{E}_i - e - E_i - F_i'') - \frac{5}{3}E_i + \frac{4}{3}\left(F_i'' + F_{i+1}'\right) \\
&= \frac{5}{3}I_i + \frac{1}{3}(\tilde{H}_i + \tilde{L}_i) + e - E_i - \frac{2}{3}\tilde{E}_i + 2F_i'' + \frac{4}{3}F_{i+1}' \\
&>_{(5)} \frac{5}{3}I_i + \frac{1}{3}(\tilde{H}_i + \tilde{L}_i) + e - \frac{2}{3}\tilde{E}_i + 2F_i'' + \frac{1}{3}F_{i+1}' \ .
\end{aligned}$$

Since $\tilde{E}_i > 0$, the delay is given by $\tilde{E}_i = \frac{1}{2}\tilde{L}_i - \frac{1}{4}\tilde{H}_i$. This yields

$$\begin{aligned}
\int_{I_i} l(t)dt &> \frac{5}{3}I_i + \frac{1}{2}\tilde{H}_i + e + 2F_i'' + \frac{1}{3}F_{i+1}' \\
&\geq \frac{5}{3}I_i + \frac{1}{2}\tilde{H}_i + \frac{1}{3}F_{i+1}' \\
&\geq \frac{5}{3}I_i - \frac{5}{3}F_{i+1}' \ .
\end{aligned}$$

Thus, in both cases the lemma holds. □

Lemma 1 is a useful tool to connect the makespan of $C_{3\mathrm{M}}$ with the lower bound on C^* based on the load argument. Using this connection, the competitive ratio of Algorithm 3M can be bounded to 2.8.

Theorem 1 *For $P3|\text{online} - \text{list}, m_j|C_{\max}$ Algorithm 3M is 2.8-competitive.*

Proof: Combining (2) with Lemma 1 we get

$$\begin{aligned}
C^* &\geq_{(2)} \frac{1}{3}\sum m_j p_j = \frac{1}{3}\int_0^{C_{3\mathrm{M}}} l(t)dt \\
&\geq_{(3)} \frac{1}{3}\sum_{i=1}^{n-1}\int_{I_i} l(t)dt + \frac{2}{3}\left(H_n + H_{n+1}\right) + \frac{1}{3}\left(L_n + L_{n+1}\right) + F_1' + F_n''
\end{aligned}$$

$$\geq_{\text{(Lem. 1)}} \quad \sum_{i=1}^{n-1} \left(\frac{5}{9} I_i - \frac{5}{9} F'_{i+1} \right) + \frac{2}{3} \left(H_n + H_{n+1} \right) + \frac{1}{3} \left(L_n + L_{n+1} \right) + F'_1 + F''_n \quad (6)$$

Besides the lower bound (2) on the optimal makespan C^*, another lower bound resulting from an improved length argument can be used. Let $p_{\max}$ denote the longest processing time of the jobs with $m_j < 3$, i.e. $p_{\max} = \max_{(j|m_j<3)} p_j$. In the optimal schedule all jobs with $m_j = 3$ and this long job have to be processed one after the other. This gives

$$C^* \geq \sum_{j=1}^{n} F_j + p_{\max} \ . \quad (7)$$

The lower bounds (6) and (7) on C^* enable us to show that $C_{3\text{M}} \leq \frac{14}{5} C^*$ holds. With the load bound (6) we can 'get rid' of the I_i intervals upto $i = n-1$ in the expression (3) for $C_{3\text{M}}$ by loosing only a factor $\frac{9}{5}$ compared to C^*, i.e. rewriting (6) gives

$$\sum_{i=1}^{n-1} I_i \quad \leq \quad \frac{9}{5} C^* + \sum_{i=1}^{n-1} F'_{i+1} - \frac{6}{5} \left(H_n + H_{n+1} \right) - \frac{3}{5} \left(L_n + L_{n+1} \right) - \frac{9}{5} (F'_1 + F''_n) \ . \quad (8)$$

Thus,

$$\begin{aligned} C_{3\text{M}} \quad &=_{(3)} \quad \sum_{i=1}^{n-1} I_i + F'_1 + H_n + L_n + E_n + F''_n + H_{n+1} + L_{n+1} \\ &\leq_{(8)} \quad \frac{9}{5} C^* + \sum_{i=2}^{n} F'_i - \frac{1}{5} \left(H_n + H_{n+1} \right) + \frac{2}{5} \left(L_n + L_{n+1} \right) + E_n - \frac{4}{5} \left(F'_1 + F''_n \right) \\ &\leq \quad \frac{9}{5} C^* + \sum_{i=1}^{n} F_i - \frac{1}{5} \left(H_n + H_{n+1} \right) + \frac{2}{5} \left(L_n + L_{n+1} \right) + E_n - \frac{9}{5} F''_n \ . \end{aligned}$$

Let

$$\Delta = -\frac{1}{5} \left(H_n + H_{n+1} \right) + \frac{2}{5} \left(L_n + L_{n+1} \right) + E_n - \frac{9}{5} F''_n \ .$$

By a number of case distinctions we will show that $\Delta \leq p_{\max}$, which implies

$$\begin{aligned} C_{3\text{M}} \quad &\leq \quad \frac{9}{5} C^* + \sum_{i=1}^{n} F_i + \Delta \\ &\leq \quad \frac{9}{5} C^* + \sum_{i=1}^{n} F_i + p_{\max} \quad \leq_{(7)} \quad \frac{14}{5} C^* \ . \end{aligned}$$

Due to the greedy nature of Algorithm 3M we know that no job starts in the interior of an interval L_i. Therefore, $p_{\max}$ is larger than L_{n+1} and $\tilde{L_n}$. Furthermore,

$$E_n \leq \tilde{E_n} = \left(\frac{1}{2} \tilde{L_n} - \frac{1}{4} \tilde{H_n} \right)^+ \leq \frac{1}{2} \tilde{L_n} \leq \frac{1}{2} p_{\max} \ .$$

To show that $\Delta \leq p_{\max}$, we consider 4 cases.

Case 1: $\tilde{E_n} = 0$

Since $\tilde{E}_i = (\frac{1}{2}\tilde{L}_i - \frac{1}{4}\tilde{H}_i)^+ = 0$, we know that $\frac{1}{4}\tilde{H}_n \geq \frac{1}{2}\tilde{L_n}$. Therefore, $\frac{1}{4}H_n \geq \frac{1}{2}L_n$. Since, furthermore $E_n \leq \tilde{E_n} = 0$ and $F''_n \leq \tilde{E_n} = 0$, we get

$$\begin{aligned}\Delta &\leq -\frac{1}{5}(H_n + H_{n+1}) + \frac{2}{5}\left(\frac{1}{2}H_n + L_{n+1}\right) \\ &\leq -\frac{1}{5}H_{n+1} + \frac{2}{5}L_{n+1} \quad \leq \quad \frac{2}{5}L_{n+1} \quad \leq \quad \frac{2}{5}p_{\max} \ .\end{aligned}$$

Case 2: $H_{n+1} > 0$ (and $\tilde{E_n} > 0$)

Due to the greedy nature of Algorithm 3M we have $H_{n+1} > L_n + E_n$. So,

$$\begin{aligned}\Delta &\leq -\frac{1}{5}H_n - \frac{1}{5}(L_n + E_n) + \frac{2}{5}(L_n + L_{n+1}) + E_n \\ &\leq \frac{1}{5}L_n + \frac{2}{5}L_{n+1} + \frac{4}{5}E_n \\ &\leq \left(\frac{1}{5} + \frac{2}{5} + \frac{4}{5}\frac{1}{2}\right)p_{\max} = p_{\max} \ .\end{aligned}$$

Case 3: $H_n + L_n > \tilde{H}_n + \tilde{L_n}$, (and $H_{n+1} = 0$, $\tilde{E_n} > 0$)

This case is depicted in Figure 5. We have $L_n + E_n < \tilde{E_n} \leq \frac{1}{2}\tilde{L_n}$ and $H_n > \tilde{L_n}$. Thus,

$$\begin{aligned}\Delta &\leq -\frac{1}{5}H_n + \frac{2}{5}(L_n + L_{n+1}) + E_n \\ &\leq -\frac{1}{5}\tilde{L_n} - \frac{3}{5}L_n + \frac{2}{5}L_{n+1} + \tilde{E_n} \\ &\leq -\frac{1}{5}\tilde{L_n} + \frac{2}{5}L_{n+1} + \tilde{E_n} \\ &\leq -\frac{1}{5}\tilde{L_n} + \frac{2}{5}L_{n+1} + \frac{1}{2}\tilde{L_n} \\ &\leq \frac{3}{10}\tilde{L_n} + \frac{2}{5}L_{n+1} \quad \leq \quad \frac{7}{10}p_{\max} \ .\end{aligned}$$

Case 4: $H_n + L_n = \tilde{H}_n + \tilde{L_n}$, (and $H_{n+1} = 0$, $\tilde{E_n} > 0$)

This case is depicted in Figure 4. Let $\gamma \geq 0$ be such that $L_n = \tilde{L_n} - \gamma\tilde{E_n}$. Then $H_n = \tilde{H_n} + \gamma\tilde{E_n}$. Due to the greedy nature of Algorithm 3M we know that L_{n+1} consists only of one job and, thus, is larger than $L_n + E_n$. This gives,

$$\begin{aligned}L_{n+1} &> L_n + E_n \\ &= \tilde{L_n} - \gamma\tilde{E_n} + E_n \\ &= \tilde{L_n} - \gamma\tilde{E_n} + \tilde{E_n} - F''_n \\ &\geq (3-\gamma)\,\tilde{E_n} - F''_n \ .\end{aligned}$$

As long as $\gamma \leq 3$ we have:

$$\tilde{E_n} \quad \leq \quad \frac{L_{n+1} + F''_n}{3-\gamma} \ . \tag{9}$$

So,

$$\begin{aligned}\Delta &\leq -\frac{1}{5}H_n + \frac{2}{5}(L_n + L_{n+1}) + E_n - \frac{9}{5}F''_n \\ &\leq -\frac{1}{5}\left(\tilde{H_n} + \gamma\tilde{E_n}\right) + \frac{2}{5}\left(\tilde{L_n} - \gamma\tilde{E_n} + L_{n+1}\right) + (\tilde{E_n} - F''_n) - \frac{9}{5}F''_n \\ &\leq -\frac{1}{5}\tilde{H_n} + \frac{2}{5}\left(\tilde{L_n} + L_{n+1}\right) + \left(1 - \frac{3\gamma}{5}\right)\tilde{E_n} - \frac{14}{9}F''_n \ .\end{aligned} \tag{10}$$

j	1	2	3	4	5	6	7	8
p_j	$\frac{2}{7}$	$\frac{2}{7}-\epsilon$	ϵ	1	ϵ	$\frac{3}{7}$	ϵ	1
m_j	1	1	2	1	3	1	2	1

Table 2: Instance for lower bound on Algorithm 3M

Since $\tilde{E_n} > 0$, we have by definition $\tilde{E_n} = \frac{1}{2}\tilde{L_n} - \frac{1}{4}\tilde{H_n}$. This implies

$$\frac{2}{5}\tilde{L_n} \quad = \quad \frac{2}{5}\left(2\tilde{E_n} + \frac{\tilde{H_n}}{2}\right) \quad = \quad \frac{4}{5}\tilde{E_n} + \frac{1}{5}\tilde{H_n} \ .$$

Combining this with (10) gives

$$\Delta \ \leq \ \frac{2}{5}L_{n+1} + \left(\frac{9}{5} - \frac{3\gamma}{5}\right)\tilde{E_n} - \frac{14}{5}F_n'' \ .$$

For $\gamma \in [0,3]$ we can use (9), leading to

$$\begin{aligned}\Delta \ &\leq \ \frac{2}{5}L_{n+1} + \frac{\frac{9}{5} - \frac{3\gamma}{5}}{3-\gamma}(L_{n+1} + F_n'') - \frac{14}{5}F_n'' \\ &= \ \frac{2}{5}L_{n+1} + \frac{3}{5}(L_{n+1} + F_n'') - \frac{14}{5}F_n'' \\ &\leq \ \frac{2}{5}L_{n+1} + \frac{3}{5}L_{n+1} \quad \leq \quad p_{\max} \ .\end{aligned}$$

For $\gamma > 3$ we can use $\tilde{E_n} \leq 2\tilde{L_n}$, leading to

$$\begin{aligned}\Delta \ &\leq \ \frac{2}{5}L_{n+1} + \left(\frac{9}{5} - \frac{3\gamma}{5}\right)2\tilde{L_n} - \frac{14}{5}F_n'' \\ &\leq \ \frac{2}{5}L_{n+1} \leq \frac{2}{5}p_{\max} \ .\end{aligned}$$

So, for each case we have $\Delta \leq p_{\max}$, proving that, Algorithm 3M is 2.8-competitive. □

To show a lower bound on the performance of Algorithm 3M we give an instance leading to a competitive ratio of at least $2\frac{10}{14} \approx 2.714$.

Theorem 2 *Algorithm* 3M *had competitive ratio of at least* $2\frac{10}{14}$.

To achieve a lower bound on the performance of Algorithm 3M, consider the following instance with 8 jobs. The processing times of the jobs and machine requirements are given in Table 2. The online schedule for Algorithm 3M and the optimal offline schedule are displayed in Figure 6. Job 5 is delayed for a duration of $\frac{4}{7}$. The online makespan is $\frac{38}{14} + \epsilon$ and the optimal makespan is $1 + 3\epsilon$. As ϵ goes to 0, the ratio between online and optimal makespan goes to $2\frac{18}{14}$. □

In the Algorithm 3M the definition of the delay d is crucial to do the analysis in Lemma 1 and Theorem 1. Defining the delay as $(xL_i - yH_i)^+$, an optimization on the values of x and y shows that the delay defined as $(\frac{1}{2}L_i - \frac{1}{4}H_i)^+$ is the best possible. So, to improve the upon the 2.8-competitive Algorithm 3M one needs to find new arguments in bounding the optimal solution or a new design for the online algorithm.

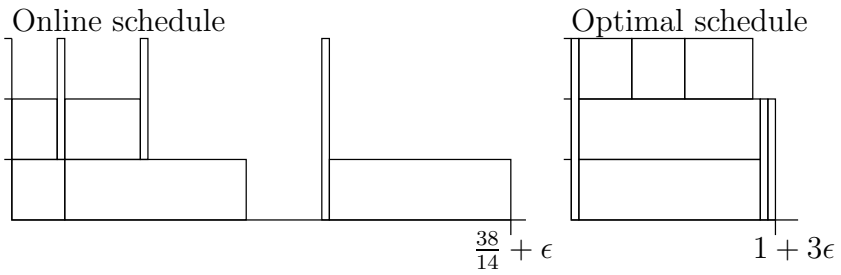

Figure 6: Lower bound on performance of Algorithm 3M

4 Semi-online: Non-increasing m_j

In this section we consider the semi-online case of $P|online - list, m_j|C_{max}$ where the jobs arrive in non-increasing order of machine requirement m_j. The best known lower bound on the competitive ratio is 1.88 [12] from classical parallel machine scheduling, i.e. the construction consist only of jobs with $m_j = 1$. Scheduling the jobs greedy is 2.75-competitive [14] (and no better than 2.5 competitive). In the following an improved algorithm is presented and some special cases are studied.

For the presented algorithm, the jobs are classified in two categories depending on their machine requirement. We call a job j *big* if $m_j > \frac{m}{3}$ and *small* if $m_j \leq \frac{m}{3}$, where m denotes the total number of machines available. Let B be the set of big jobs and S be the set of small jobs. The presented algorithm is a modified version of the greedy algorithm in the way that it first schedules the big jobs consecutively and than the small jobs in a greedy manner:

Modified Greedy (MG**):**

1. Schedule the big jobs one after the other.
2. Schedule the small jobs in a greedy fashion.

Since the jobs appear in non-increasing order of m_j, the two steps of the algorithm are always executed in the given order. Figure 7 illustrates the structure of online schedules created by the Algorithm MG. The shaded area indicates where small jobs are scheduled.

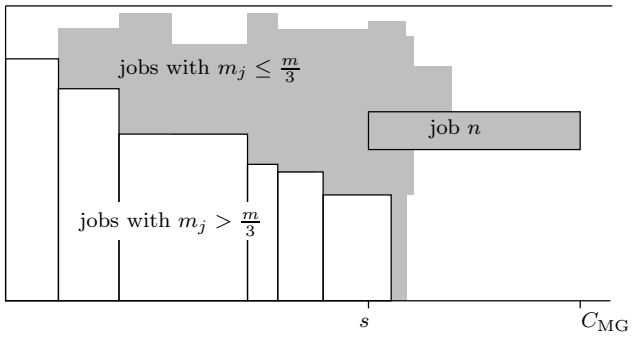

Figure 7: Structure of the online schedule created by MG.

Theorem 3 *Algorithm MG is* $\left(\frac{5}{2} - \frac{3}{2m}\right)$*-competitive.*

Proof: As long as only big jobs arrive, the algorithm is 2-competitive, since in the optimal offline schedule no more than two jobs with $m_j > \frac{m}{3}$ can be scheduled in parallel. Therefore, we only have to analyze the algorithm if at least one small job has been scheduled. Furthermore, we only have to consider a situation, where the last scheduled job determines the makespan of MG. Let this last job be job n, and let s denote the starting time of job n (see Figure 7).

The makespan of the online schedule is given by $C_{\text{MG}} = s + p_n$. Since no jobs with $m_j > \frac{m}{3}$ are scheduled in parallel and the jobs with $m_j \leq \frac{m}{3}$ are scheduled in a greedy fashion, there are at least $\frac{2m}{3}$ machines busy at each time in $[0, s)$. The load in $[0, s)$ is, therefore, at least $\frac{2m}{3}s$. Since job n starts at s, this implies $s \leq \frac{3}{2m}\sum_{i=1}^{n-1} m_i p_i$. Using the lower bounds (1) and (2), we get

$$\begin{aligned} C_{\text{MG}} &= s + p_n \leq \frac{3}{2m}\sum_{i=1}^{n-1} m_i p_i + p_n \\ &= \frac{3}{2m}\sum_{i=1}^{n} m_i p_i + \left(1 - \frac{3m_n}{2m}\right) p_n \\ &\leq \frac{3}{2}C^* + \left(1 - \frac{3m_n}{2m}\right) C^* \leq \left(\frac{5}{2} - \frac{3}{2m}\right) C^* \ , \end{aligned}$$

proving the theorem. □

For a small number of machines, Theorem 3 gives competitive ratios smaller than 2.5. However, for a large number of machines the competitive ratio given in Theorem 3 still tends to 2.5, which is the best known lower bound on the competitive ratio of the greedy algorithm. In the following we prove that Algorithm MG has for every number of machines a competitive ratio strictly less than 2.5. For this, let r be the start time of the job that is the first job Algorithm MG has scheduled parallel to another job (first in execution of Algorithm MG not necessary first on the time axis). Furthermore, let t be the start time of the job that is the first job starting after r that is scheduled parallel to two other jobs. Let h be this jobs starting at t. A sketch of a possible online schedule constructed upto h is given in Figure 8. In the schedule of Figure 8 no small job starts before r, but this is not excluded.

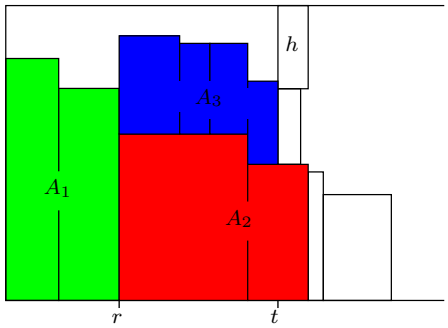

Figure 8: Sketch of an online schedule upto job h.

Lemma 2 *A lower bound on the optimal makespan* C^* *is given by* $\frac{3}{4}t$*, i.e.* $C^* \geq \frac{3}{4}t$*.*

Proof: We can assume that h is the last job in the list of jobs, because the addition of more jobs will only increase the value of C^* and not of t. Since h is scheduled in parallel to two other jobs, we have $m_h \leq \frac{m}{3}$. Furthermore, since h is the first such job, at each point in time before t at least one job from B is scheduled, i.e. $t \leq \sum_{j \in B} p_j$.

Let A_1 be the subset of B which contains all jobs with $m_j > \frac{2m}{3}$ which are scheduled to start in $[0, r)$. Due to the definition of r one job $j \in A_1$ has to complete at r (or $A_1 = \emptyset$ and $r = 0$). Let A_2 (A_3) be the subset of B (S) which contain the jobs scheduled to start in $[r, t)$ (see Figure 8).

To bound the makespan of the optimal schedule observe the following. The jobs in A_2 cannot be scheduled in parallel to jobs in A_1 but possibly can be scheduled in parallel with other jobs in A_2. Let $|A_i|$ denote the total processing time of the jobs in A_i. Since the total processing time of the jobs in A_2 is at least $t - r$ we have

$$\begin{aligned} C^* &\geq |A_1| + \frac{1}{2}|A_2| \\ &\geq r + \frac{1}{2}(t - r) = \frac{1}{2}(r + t) \ . \end{aligned} \tag{11}$$

On the other hand, due to the definition of A_i, jobs from the sets A_1, A_2 and A_3 which can be scheduled in parallel with two other jobs from these sets, are the jobs belonging to A_3. Thus, the most compact way to schedule the jobs from A_1, A_2 and A_3 are as indicated in Figure 9. Formally, this yields

$$\begin{aligned} C^* &\geq |A_1| + \frac{1}{2}|A_2| + \frac{1}{3}(|A_3| - |A_1|) \\ &\geq r + \frac{1}{2}(t - r) + \frac{1}{3}(t - 2r) = \frac{5}{6}t - \frac{1}{6}r \ . \end{aligned} \tag{12}$$

If $r \geq \frac{1}{2}t$ we have by (11) that

$$C^* \geq \frac{1}{2}\left(\frac{1}{2}t + t\right) = \frac{3}{4}t \ ,$$

and if $r \leq \frac{1}{2}t$ we have by (12) that

$$C^* \geq \frac{5}{6}t - \frac{1}{12}t = \frac{3}{4}t \ .$$

This proves the lemma. □

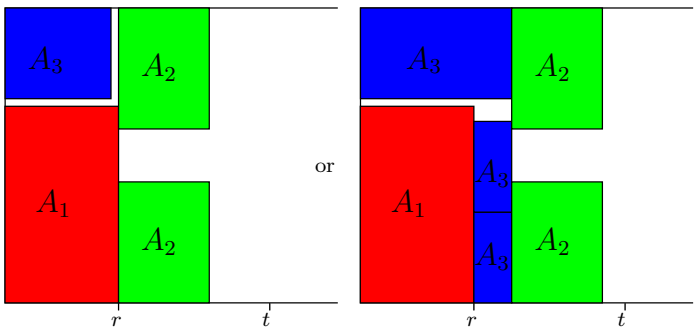

Figure 9: Sketch of the optimal solution

Lemma 2 gives a new lower bound on the optimum makespan C^*, enabling us to strengthen the upper bound on the competitive ratio of MG.

Theorem 4 *Algorithm MG is* $\frac{67}{27}$ (≈ 2.4815)*-competitive.*

Proof: W.l.o.g., we can assume that job n is the last job in the list of jobs. Suppose there are no three jobs scheduled in parallel to each other after r, i.e. job h doesn't exist and point t is not defined. When job n is a big job, the online makespan is within a factor of two of the optimal makespan. When job n is small, we get by the same reasoning as in the proof of Lemma 2 (substitute t by s) that $C^* \geq \frac{3}{4}s$, implying

$$C_{\mathrm{MG}} = s + p_n \leq \frac{4}{3}C^* + C^* = \frac{7}{3}C^* \ .$$

It remains to consider the case, that there are three jobs scheduled in parallel to each other after r. This implies that job n is a small job. If $s \leq t$, we get

$$C_{\mathrm{MG}} = s + p_n \leq t + p_n \leq \frac{4}{3}C^* + C^* = \frac{7}{3}C^* \ .$$

Thus, we only have to consider the case $s > t$. Let αt be the length of the interval $[t, s)$, i.e. $\alpha t = s - t$ and

$$C_{\mathrm{MG}} \quad = \quad (1+\alpha)t + p_n \ . \tag{13}$$

Case 1:

The online schedule contains a point after t where at least 4 jobs are scheduled in parallel. The job k, which is the first job scheduled in parallel with three other jobs, has $m_k \leq \frac{m}{4}$. Since jobs appear with non-increasing m_j, the number of machines in use between t and the start of job k is non-increasing. The load in $[t, s)$ is therefore at least $\frac{3m}{4}\alpha t$. Together with the fact that the load in $[0, t)$ is at least $\frac{2m}{3}t$, we get $C^* \geq (\frac{2}{3} + \frac{3}{4}\alpha)t$. Incorporating this bound in (13) yields

$$C_{\mathrm{MG}} \quad \leq \quad \left(\frac{1+\alpha}{\frac{2}{3}+\frac{3}{4}\alpha} + 1\right) C^* \ . \tag{14}$$

On the other hand, incorporating Lemma 2 in (13) yields

$$C_{\mathrm{MG}} \quad \leq \quad \left((1+\alpha)\frac{4}{3} + 1\right) C^* \ . \tag{15}$$

If $\alpha \geq \frac{1}{9}$ then by (14) we have

$$C_{\mathrm{MG}} \leq \left(\frac{1+\frac{1}{9}}{\frac{2}{3}+\frac{3}{4}\frac{1}{9}} + 1\right) C^* = \frac{67}{27}C^* \ ,$$

and if $\alpha \leq \frac{1}{9}$ then by (15) we have

$$C_{\mathrm{MG}} \leq \left((1+\frac{1}{9})\frac{4}{3} + 1\right) C^* = \frac{67}{27}C^* \ .$$

Case 2:

In the online schedule there are no 4 jobs scheduled in parallel after t. In this case we use a load argument, where we take the load of p_n into account as well. The load in $[0, t)$ is

at least $\frac{2m}{3}t$, and in $[t,s)$ the load is at least $(m-m_n)(s-t)$, since the machine usage after t is non-increasing. In $[s,C_{\mathrm{MG}})$ the load is at least $m_n p_n$. Thus,

$$C^* \geq \frac{2}{3}t + \frac{m-m_n}{m}(s-t) + \frac{m_n}{m}p_n \tag{16}$$

If $s-t \leq p_n$ the bound (16) becomes $C^* \geq s - \frac{1}{3}t$. Thus,

$$\begin{aligned} C_{\mathrm{MG}} &= s+p_n = s - \frac{1}{3}t + \frac{1}{3}t + p_n \\ &\leq 2C^* + \frac{1}{3}t \leq_{(Lem.\ 2)} \frac{22}{9}C^* \end{aligned}$$

Now consider the case that $s-t \geq p_n$. Since job n is small, the load in $[0, s-p_n)$ is at least $\frac{2m}{3}(s-p_n)$, and the load in $[s-p_n)$ plus the load of job n is at least mp_n. Therefore, we get

$$C^* \geq \frac{2}{3}(s-p_n) + p_n \geq \frac{2}{3}s + \frac{1}{3}p_n \tag{17}$$

Thus,

$$C_{\mathrm{MG}} = s+p_n \leq_{(17)} \frac{3}{2}C^* + \frac{1}{2}p_n \leq 2C^*$$

Therefore the algorithm MG is $\frac{67}{27}$-competitive. □

The follow theorem shows that the algorithm MG is not much better than $2\frac{13}{27}$.

Theorem 5 *The competitive ratio of MG is at least* $2\frac{5}{12}(\approx 2.4167)$.

Proof: Let k be a sufficiently large integer and ϵ and δ sufficiently small. Consider the following input sequence of jobs.

1. k jobs $a_1, ..., a_k$, where a_i has length $\frac{1}{k}$ and height $\frac{2}{3} + (k+1-i)\delta$. These are scheduled one after the other and occupy a length of 1.
2. a job d_1 of length ϵ and height $\frac{2}{3}$, which is scheduled immediately after the a_i jobs.
3. 8 jobs $b_1, ..., b_8$, where b_i has length $\frac{1}{8}$ and height $\frac{1}{3} + (2k+11-i)\delta$, which are scheduled after d_1 and occupy a length of 1.
4. a job d_2 of length $\frac{1}{k}$ and height $\frac{1}{3} + 2(k+1)\delta$, which is scheduled immediately after the b_i jobs.
5. a job of length ϵ and height $\frac{1}{3}$. This job goes parallel to d_1, making all machines occupied in $[1, 1+\epsilon)$.
6. k jobs $\tilde{a}_1, ..., \tilde{a}_k$, where $\tilde{a}_i$ has length ϵ and height $\frac{1}{3} - i\delta$. Job $\tilde{a}_i$ goes parallel to a_{k+1-i}, together they have a machine requirement of m.
7. a job c of length 1 and height $\frac{1}{3} - (k+1)\delta$. This job goes parallel to the jobs $b_1, ..., b_8$.
8. 2 jobs of length $\frac{1}{k}$ and height $\frac{1}{3} - (k+1)\delta$. Both jobs go parallel to D_2, with a total machine requirement of m.
9. 8 jobs $\tilde{b}_1, ..., \tilde{b}_8$, where $\tilde{b}_i$ has length $\frac{1}{k}$ and height $\frac{1}{3} - (k+1+i)\delta$. Job $\tilde{b}_i$ goes parallel to b_{9-i} and c, filling all machines in the corresponding interval.
10. 4 jobs $e_1, ..., e_4$, where e_i has length $\frac{1}{8}$ and height $\frac{1}{4}$. These four go parallel to each other after job d_2.

11. a job of length $\frac{3}{2}$ and height δ, which is scheduled at the end.

Figure 10 illustrates the schedule created by MG. It has makespan $3\frac{5}{8} + \frac{1}{k} + \epsilon$. In the optimal offline schedule all jobs a_i, b_i, c and e_i can be scheduled in parallel to the last job, i.e. by scheduling jobs a_i and c in parallel and scheduling half of the b_i jobs parallel to the other half and parallel to the e_i jobs. The remaining jobs are scheduled after $[0, \frac{3}{2}]$, but the length of it goes to 0 if k grows large and ϵ goes to 0.

Thus, as ϵ goes to 0 and k grows large we get:

$$\frac{C_{\mathrm{MG}}}{C^*} \to \frac{3\frac{5}{8}}{1\frac{1}{2}} = \frac{29}{12} \approx 2.4167 \ .$$

□

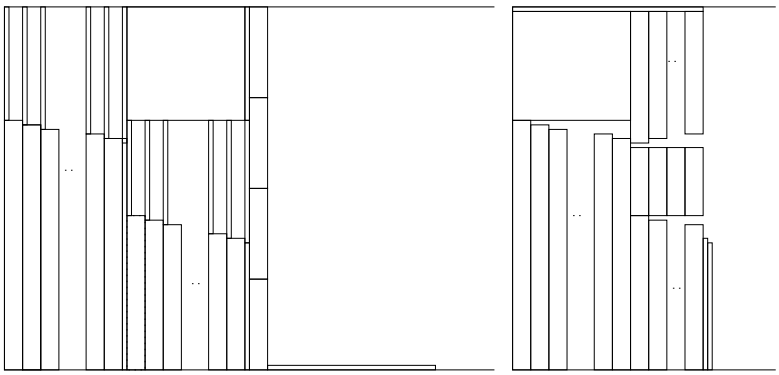

Figure 10: Lower bound construction on MG.

In the following we consider the semi-online case of $P|\mathrm{online}-\mathrm{list}, m_j|C_{\max}$ where jobs appear in non-increasing order of machine requirement for 2, 3, 4 and 5 machines.

Theorem 6 *For the* 2 *and the* 3*-machine problem the greedy algorithm is* $2-\frac{1}{m}$*-competitive if jobs appear in non-increasing order of machine requirement, and this is the best possible.*

Proof: For the 2-machine problem, no worst case example can contain a job with $m_j = 2$, since removing it decreases the online and optimal makespan by the same amount. If no jobs with $m_j = 2$ appear, the greedy algorithm is equal to the list scheduling algorithm for the classical parallel machine scheduling problem, which is $(2 - \frac{1}{m})$-competitive.

For the 3-machine problem, no worst case example contains jobs with $m_j = 3$. All jobs with $m_j = 2$ are scheduled one after the other on two machines. Afterwards, jobs with $m_j = 1$ are first scheduled parallel to the jobs with $m_j = 2$. Thus, there exists a worst case example consisting only of jobs with $m_j = 1$. Again, the greedy algorithm is equal to the list scheduling algorithm for the classical parallel machine scheduling problem, which is $(2 - \frac{1}{m})$-competitive.

The greedy algorithm is also best possible, since list scheduling is $2 - \frac{1}{m}$-competitive and best possible for the classical parallel machine schedule with $m \leq 3$. □

Theorems 6 shows that if $m \leq 3$ the parallelism of jobs is of no importance when jobs appear in non-increasing order of m_j. For more than 3 machines the machine requirements do play a role, i.e. the greedy algorithm has competitive ratio larger than $2 - \frac{1}{m}$.

Theorem 7 *If $m \geq 4$ and jobs appear in non-increasing order of machine requirement, then the greedy algorithm has competitive ratio at least ≥ 2.*

Proof: Suppose the greedy algorithm has ratio $2 - \epsilon$. Consider the following sequence of 4 jobs. Job 1 has $p_1 = 1$ and $m_1 = m - 1$, job 2 has $p_2 = \frac{\epsilon}{2}$ and $m_2 = \lceil \frac{m}{2} \rceil$, job 3 has $p_3 = \frac{\epsilon}{2}$ and $m_3 = \lfloor \frac{m}{2} \rfloor$, and job 4 has $p_4 = 1 + \frac{\epsilon}{2}$ and $m_4 = 1$. The greedy algorithm schedules job 2 and 3 in parallel, where in the optimum jobs 1 and 4 are in parallel (see Figure 11 for the case $m = 4$).

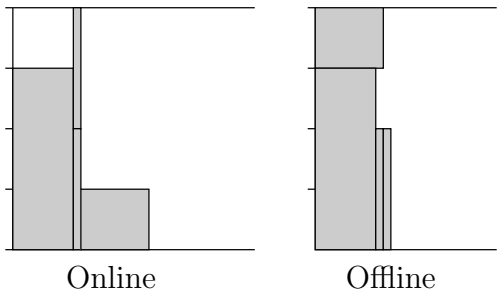

Figure 11: Counter example for $(2 - \epsilon)$-competitiveness.

Therefore, $C_{\text{Greedy}} = 2 + \epsilon$ and $C^* = 1 + \epsilon$, and the resulting competitive ratio is:

$$\frac{C_{\text{Greedy}}}{C^*} = \frac{2 + \epsilon}{1 + \epsilon} = 2 - \frac{\epsilon}{1 + \epsilon} > 2 - \epsilon \ ,$$

yielding the required contradiction. □

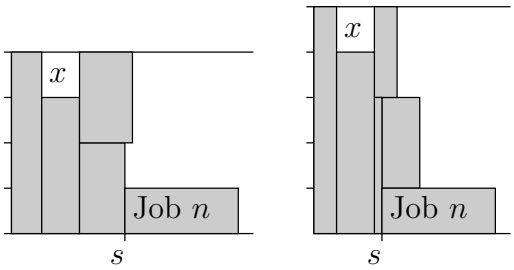

Figure 12: Greedy for $m = 4$ and $m = 5$.

Theorem 8 *For the 4 and 5-machine problem, the greedy algorithm is 2-competitive if jobs appear in non-increasing order of machine requirement.*

Proof: For $m = 4$, the schedule created by Greedy contains first the jobs with $m_j = 4$, then jobs with $m_j = 3$ followed by jobs with $m_j = 2$ in parallel. Denote the length of this interval, where jobs with $m_j = 3$ are scheduled, without any job with $m_j = 1$ in parallel, by x (see Figure 12). Let jobs n be the job determining the makespan of the online schedule, and s its start time. We only have to consider the case with $m_n = 1$, because if $m_n = 2$ it results in a competitive ratio of $\frac{3}{2}$. By definition, x is smaller than all jobs with $m_j = 1$ scheduled after x, i.e. $x \leq p_n$. Therefore, the total load of all jobs is at least $4s$, implying that $s \leq C^*$. This leads to

$$C_{\text{Greedy}} = s + p_n \leq 2C^* \ ,$$

proving the theorem for $m = 4$.

Similar for $m = 5$, there is one interval with machine usage 4 before the start of job n, and this interval is shorter than p_n (see Figure 12). □

Theorem 9 *Algorithm MG is* 2*-competitive if* $m \leq 5$ *and jobs appear in non-increasing order of machine requirement.*

Proof: Since the only jobs with $m_j \leq \frac{m}{3}$ are the jobs with $m_j = 1$, only these are scheduled in a greedy fashion. All other jobs are scheduled in series. Before any $m_j = 1$ job appears, MG is 2-competitive since no more than 2 jobs can be scheduled in parallel in the optimum. As soon as $m_j = 1$ jobs appear the schedule has at least load $m \cdot s$ in $[0, s]$, where s is the start time of the last job, implying $(2 - \frac{1}{m})$-competitiveness. □

5 Concluding remarks

In this paper we have presented and analyzed new algorithms for two special cases of online parallel job scheduling. By finding structural properties of the schedules we have improved the bounds on the optimal solution. The state of the research on online parallel job scheduling is summarized in Table 1. There remains a lot of work to be done to close the gaps between the lower and upper bounds on the competitive ratios. The presented approaches are a step in this direction.

References

[1] B.S. Baker and J.S. Schwarz. Shelf algorithms for two-dimensional packing problems. *SIAM Journal on Computing*, 12(3):508–525, 1983.

[2] D.J. Brown, B.S. Baker, and H.P. Katseff. Lower bounds for on-line two-dimensional packing algorithms. *Acta Informatica*, 18(2):207–225, 1982.

[3] W.T. Chan, F.Y.L. Chin, D. Ye, G. Zhang, and Y. Zhang. On-line scheduling of parallel jobs on two machines. *Journal of Discrete Algorithms (to appear), doi:10.1016/j.jda.2006.07.005*, 2007.

[4] U. Faigle, W. Kern, and G. Turàn. On the performance of online algorithms for partition problems. *Acta Cybernetica*, 9:107–119, 1989.

[5] R.L. Graham, E.L. Lawler, Lenstra J.K., and A.H.G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[6] J.L. Hurink and J.J. Paulus. Online algorithm for parallel job scheduling and strip packing. *Lecture Notes in Computer Science (WAOA 2007) to appear*, 2008.

[7] J.L. Hurink and J.J. Paulus. Online scheduling of parallel jobs on two machines is 2-competitive. *Operations Research Letters (to appear), doi:10.1016/j.orl.2007.06.001*, 2008.

[8] B. Johannes. Scheduling parallel jobs to minimize the makespan. *Journal of Scheduling*, 9(5):433–452, 2006.

[9] D.S. Johnson, A. Demers, J.D. Ullman, M.R. Garey, and R.L. Graham. Worst-case performance founds for simple one-dimensional packing algorithms. *SIAM Journal on Computing*, 3:299–325, 1974.

[10] C.-L. Li, X. Cai, and C.-Y. Lee. Scheduling with multiple-job-on-one-processor pattern. *IIE Transactions*, 30:433–445, 1998.

[11] K. Pruhs, J. Sgall, and E. Torng. Online scheduling. In Joseph Y-T. Leung, editor, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, chapter 15, pages 15–1–15–41. CRC Press, 2004.

[12] J.F. Rudin III. *Improved Bound for the Online Scheduling Problem*. PhD thesis, The University of Texas at Dallas, 2001.

[13] D. Ye, X. Han, and G. Zhang. A note on online strip packing. *Manuscript*, 2007.

[14] D. Ye and G. Zhang. On-line scheduling of parallel jobs. *Lecture Notes in Computer Science (SIROCCO 2004)*, 3104:279–290, 2004.

[15] D. Ye and G. Zhang. On-line scheduling of parallel jobs in a list. *Journal of Scheduling*, 10(6):407–413, 2007.

[16] G. Zhang, X. Cai, and C. Wong. Some results on resource constrained scheduling. *IIE Transactions*, 36(1):1–9, 2004.