Online Malleable Job Scheduling for $m \leq 3$

Jessen T. Havill
Department of Mathematics and Computer Science
Denison University
Granville, OH 43023
havill@denison.edu

Abstract

A *malleable* parallel job is one that may be assigned to any number of processors in a parallel computing environment. In our particular problem, we assume that the execution time of a job j with processing requirement p_j is $p_j/k_j + (k_j - 1)c$ if the job is assigned to $k_j \in \{1, 2, \ldots, m\}$ processors, where c is a constant representing overhead and m is the number of processors. We provide online algorithms for $m = 2$ and $m = 3$ with asymptotically optimal competitive ratios of 3/2 and 5/3, respectively. We also provide a similar online algorithm for the more general problem with job dependent overhead term c_j that has optimal competitive ratio $\phi = (1+\sqrt{5})/2$.

Keywords: online algorithms, scheduling

1 Introduction

A *malleable* parallel job is one that may be assigned to any number of processors in a parallel computing environment. In our particular problem, we assume that the execution time of a job with processing requirement p is $p/k + (k-1)c$ if the job is assigned to $k \in \{1, 2, \ldots, m\}$ processors, where c is a constant and m is the number of processors. The constant c represents the per-processor setup time required to partition and distribute workload to a machine and later collect results. This speedup function nicely models computation time for problems that are amenable to data parallel decomposition and has recently been empirically validated using standard benchmarks [2].

The input to our problem is a sequence σ of n jobs, where job j has processing requirement p_j. An *online* algorithm must irrevocably assign to each job j a start time $s_j \geq 0$ and a number of processors k_j, where $1 \leq k_j \leq m$, before future jobs are known. The goal is to minimize the makespan, or maximum completion time, of the schedule. For any job sequence σ, let $C_A(\sigma)$ and $C^*(\sigma)$ denote the makespan of an online algorithm A and the optimal makespan, respectively. We say that an online algorithm A is ρ-competitive if, for all σ, $C_A(\sigma) \leq \rho\, C^*(\sigma) + a$, where a is constant relative to σ.

Previously it was shown that the online algorithm SET (Shortest Execution Time) that assigns each job to the number of processors that minimizes that job's individual execution time has competitive ratio $4(m-1)/m$ and $4m/(m+1)$ for all even and odd m, respectively [6]. The algorithm ECT (Earliest Completion Time) that assigns each job to the number of processors that minimizes that job's completion time has competitive ratio 2, 9/4, and 20/9 for $m = 2$, 3, and 4, respectively and competitive ratio at least 30/13 for arbitrarily large m [7, 3]. Recently, Guo and

Kang [5] studied a more general version of this problem in which the constant c is replaced with a job-dependent overhead term c_j and jobs can arrive over time, allowing an online algorithm to delay scheduling decisions. They proved that any online algorithm for this more general problem has competitive ratio at least $\phi = (1 + \sqrt{5})/2$ for all $m \geq 2$ and provided an online algorithm for $m = 2$ with matching competitive ratio.

Based on the classical lower bound results from Faigle, et al. [4], Havill and Mao [6] showed that any online algorithm for this problem must have competitive ratio at least 3/2 and 5/3 on $m = 2$ and $m = 3$ machines, respectively. They also showed that, even if the additive constant a in the definition of ρ-competitive is allowed to be proportional to m, any online algorithm must still have competitive ratio strictly greater than 1. In this paper, we present online algorithms for $m = 2$ and 3 with competitive ratios that match these lower bounds up to small additive constants. In particular, our algorithms for $m = 2$ and 3 have makespan $C(\sigma) \leq (3/2)\, C^*(\sigma) + c$ and $C(\sigma) \leq (5/3)\, C^*(\sigma) + 6c$, respectively, for any instance σ. The additive term in each case arises only from idle time, during which only one job is executing, that may occur at the end of an arbitrarily long schedule. For the more general problem with job-dependent overhead term c_j studied by Guo and Kang [5], we provide a simpler optimal online algorithm for $m = 2$ (for the model in which delayed decisions are not allowed) with competitive ratio $\phi = (1 + \sqrt{5})/2$ (and no additive constant). Together, these results imply that this more general model is strictly harder, at least for $m = 2$, with respect to competitive ratio.

Several other variants of parallel job scheduling have been studied over the past two decades. For the problem with nonmalleable jobs (i.e., each k_j is given) that arrive online in a list, the best competitive ratio is known to be between $3/2 + \sqrt{33}/6 \approx 2.457$ [10] and $7/2 + \sqrt{10} \approx 6.6623$ [12, 8]. There have also been a number of papers that study approximation algorithms and polynomial time approximation schemes for variants of the offline malleable job scheduling problem; for some of the more recent results, see [9, 1]. Sgall [11] presented a survey of a related online malleable parallel job scheduling problem in which the lengths of the jobs are unknown until they complete.

2 Two Machines

We first present an online algorithm for the problem with constant overhead c, followed by a similar algorithm for the more general problem with job-specific overhead c_j.

2.1 Constant Overhead

Let $I(j)$ denote the idle time at the end of the schedule during which only one job is executing, from the point of view of job j as it arrives. Our algorithm assigns to a job j

$$k_j = \begin{cases} 1 & \text{if } p_j \leq 4c + I(j) \\ 2 & \text{otherwise} \end{cases}$$

and start time s_j as early as possible (on the least loaded processor if $k_j = 1$). The algorithm does not *backfill*, i.e., schedule a job on a machine before a previously scheduled job on that machine.

Theorem 1 *The competitive ratio of our online algorithm is* 3/2.

Proof Consider an arbitrary instance for the problem. Let t_1^1 and t_1^2 denote the total time in the algorithm's schedule during which only one job is executing on one processor and the total time

during which two jobs are executing concurrently, respectively. Note that t_1^1 is also the total idle time in the schedule. Let $\widehat{t_1^1}$ denote the open idle time at the end of the schedule and let $\widetilde{t_1^1} = t_1^1 - \widehat{t_1^1}$. Let J_k denote the set of jobs that the algorithm chooses to execute on k processors. Note that $\widetilde{t_1^1} = \sum_{j \in J_2} I(j)$. Let J_2^* denote the set of jobs that are assigned to two processors in the optimal schedule. Then we know that

$$C^* \geq \left(\frac{1}{2}\right) \sum_j p_j + |J_2^*|c = \left(\frac{1}{2}\right) \left(t_1^1 + 2t_1^2 + \sum_{j \in J_2} p_j \right) + |J_2^*|c \,. \tag{1}$$

From the algorithm, we know that

$$\sum_{j \in J_2} p_j > \sum_{j \in J_2} (4c + I(j)) = 4|J_2|c + \widetilde{t_1^1} \,. \tag{2}$$

Therefore, the makespan of the online algorithm is

$$\begin{aligned}
C &= \widetilde{t_1^1} + t_1^2 + \sum_{j \in J_2} \left(\frac{p_j}{2} + c \right) + \widehat{t_1^1} \\
&= \widetilde{t_1^1} + t_1^2 + \frac{1}{2} \sum_{j \in J_2} p_j + |J_2|c + \widehat{t_1^1} \\
&< \left(\frac{3}{4}\right) \widetilde{t_1^1} + t_1^2 + \left(\frac{3}{4}\right) \sum_{j \in J_2} p_j + \widehat{t_1^1} && \text{by (2)} \\
&= \left(\frac{3}{4}\right) t_1^1 + t_1^2 + \left(\frac{3}{4}\right) \sum_{j \in J_2} p_j + \left(\frac{1}{4}\right) \widehat{t_1^1} \\
&= \left(\frac{3}{4}\right) t_1^1 + \left(\frac{3}{2}\right) t_1^2 + \left(\frac{3}{4}\right) \sum_{j \in J_2} p_j + \left(\frac{1}{4}\right) \widehat{t_1^1} - \left(\frac{1}{2}\right) t_1^2 \\
&\leq \left(\frac{3}{2}\right) (C^* - |J_2^*|\, c) + \left(\frac{1}{4}\right) \widehat{t_1^1} - \left(\frac{1}{2}\right) t_1^2 && \text{by (1)} \\
&= \left(\frac{3}{2}\right) C^* - \left(\frac{3}{2}\right) |J_2^*|\, c + \left(\frac{1}{4}\right) \widehat{t_1^1} - \left(\frac{1}{2}\right) t_1^2 \,.
\end{aligned}$$

We now consider two cases. In the first case, assume $|J_2^*| = 0$, i.e., the optimal schedule assigns all jobs to one processor. Therefore, since the algorithm dictates that the idle time at the end of the schedule is at most $4c$,

$$C \leq \left(\frac{3}{2}\right) C^* + \left(\frac{1}{4}\right) (4c) = \left(\frac{3}{2}\right) C^* + c \,.$$

In the second case, $|J_2^*| \geq 1$, i.e., the optimal schedule assigns at least one job to two processors. Therefore,

$$C \leq \left(\frac{3}{2}\right) C^* - \frac{3}{2}c + \left(\frac{1}{4}\right) (4c) < \left(\frac{3}{2}\right) C^* \,.$$

The lower bound follows from the previously mentioned general lower bound of 3/2 [6]. To show that the additive constant is tight for the algorithm, consider an instance σ consisting of two jobs

with $p_1 = 4c + \epsilon$ and $p_2 = 4c$. The algorithm will choose $k_1 = 2$ and $k_2 = 1$, giving makespan $C(\sigma) = 7c + \epsilon/2$. Since the optimal makespan is $C^*(\sigma) = 4c + \epsilon$, we have $C(\sigma) = (3/2)\,C^*(\sigma) + c$ as $\epsilon \to \infty$. In addition, we note that *any* online algorithm that assigns a job with length $p + \epsilon$ to two processors followed by a job with length p to one processor will also have at least this makespan. □

2.2 Job-specific Overhead

We now consider the more general problem studied by Guo and Kang [5] in which the constant c is replaced with a variable c_j for each job j. We present a simpler alternative to their algorithm with a more straightforward upper bound proof.

Our algorithm assigns to a job j

$$k_j = \begin{cases} 1 & \text{if } p_j \le 2\phi(\phi^2 c_j + I(j)) \\ 2 & \text{otherwise} \end{cases}$$

and a start time s_j as early as possible (on the least loaded processor if $k_j = 1$). The algorithm does not backfill. We note that, if we remove the $I(j)$, this algorithm assigns k_j identically to the algorithm of Guo and Kang [5].

Theorem 2 *The competitive ratio of our online algorithm is* $\phi = (1 + \sqrt{5})/2$.

Proof Consider an arbitrary instance for the problem. We use the same notation defined in the proof of Theorem 1. First notice that, by summing over all jobs assigned to 2 processors, we have

$$\sum_{j \in J_2} c_j < \left(\frac{1}{2\phi^3}\right) \sum_{j \in J_2} (p_j - 2\phi I(j)) = \left(\frac{1}{2\phi^3}\right) \sum_{j \in J_2} p_j - \left(\frac{1}{\phi^2}\right) \widetilde{t_1^1}\,. \tag{3}$$

The makespan of the online algorithm is

$$\begin{aligned} C &= \widetilde{t_1^1} + t_1^2 + \sum_{j \in J_2} \left(\frac{p_j}{2} + c_j\right) + \widehat{t_1^1} \\ &= \widetilde{t_1^1} + t_1^2 + \frac{1}{2} \sum_{j \in J_2} p_j + \sum_{j \in J_2} c_j + \widehat{t_1^1} \\ &< \left(\frac{1}{\phi}\right) \widetilde{t_1^1} + t_1^2 + \left(\frac{1}{\phi}\right) \sum_{j \in J_2} p_j + \widehat{t_1^1} \qquad \text{by (3)} \\ &= \frac{1}{\phi} \left(t_1^1 + 2t_1^2 + \sum_{j \in J_2} p_j \right) + \left(\frac{1}{\phi^2}\right) \widehat{t_1^1} - \left(\frac{1}{\phi^3}\right) t_1^2\,. \end{aligned}$$

We now consider three cases. In the first two cases, we will use the fact that

$$C^* \ge \frac{1}{2} \sum_j p_j = \frac{1}{2} \left(t_1^1 + 2t_1^2 + \sum_{j \in J_2} p_j \right)\,. \tag{4}$$

In the first case, assume $\widehat{t_1^1} = 0$, i.e., there is no open idle time at the end of the schedule. Then, by (4),

$$C \leq \left(\frac{2}{\phi}\right) C^* + \left(\frac{1}{\phi^2}\right) \widehat{t_1^1} - \left(\frac{1}{\phi^3}\right) t_1^2 \leq \left(\frac{2}{\phi}\right) C^* < \phi\, C^* \ .$$

Let job l be the last job to to finish in the algorithm's schedule. In the second case, assume that $\widehat{t_1^1} > 0$, i.e., there is some open idle time at the end of the algorithm's schedule during which job l is executing, and the optimal schedule assigns job l to one processor. Then we know that $C^* \geq p_l$. Using this fact and (4), we have

$$C \leq \left(\frac{2}{\phi}\right) C^* + \left(\frac{1}{\phi^2}\right) \widehat{t_1^1} - \left(\frac{1}{\phi^3}\right) t_1^2 \leq \left(\frac{2}{\phi}\right) C^* + \left(\frac{1}{\phi^2}\right) p_l \leq \phi\, C^* \ .$$

In the third case, assume $\widehat{t_1^1} > 0$ and the optimal schedule assigns job J_l to two processors. Then we know that

$$C^* \geq \frac{1}{2} \sum_j p_j + c_l = \frac{1}{2} \left(t_1^1 + 2t_1^2 + \sum_{j \in J_2} p_j \right) + c_l \ . \tag{5}$$

Therefore

$$\begin{aligned}
C &\leq \left(\frac{2}{\phi}\right) (C^* - c_l) + \left(\frac{1}{\phi^2}\right) \widehat{t_1^1} - \left(\frac{1}{\phi^3}\right) t_1^2 && \text{by (5)} \\
&= \left(\frac{2}{\phi}\right) C^* + \frac{1}{\phi^2} \left(\widehat{t_1^1} - \left(\frac{1}{\phi}\right) t_1^2 - 2\phi c_l \right) \\
&\leq \left(\frac{2}{\phi}\right) C^* + \frac{1}{\phi^2} \left(p_l - I(l) - \left(\frac{1}{\phi}\right) I(l) - 2\phi c_l \right) \\
&= \left(\frac{2}{\phi}\right) C^* + \frac{1}{\phi^2} \left(p_l - \phi I(l) - 2\phi c_l \right) \\
&\leq \left(\frac{2}{\phi}\right) C^* + \frac{1}{\phi^2} \left(C^* + \frac{p_l}{2} - c_l - \phi I(l) - 2\phi c_l \right) && \text{since } C^* \geq p_l/2 + c_l \\
&\leq \phi\, C^* + \frac{1}{\phi^2} \left(\frac{2\phi(\phi^2 c_l + I(l))}{2} - \phi I(l) - \phi^3 c_l \right) && \text{since } k_l = 1 \\
&= \phi\, C^* \ .
\end{aligned}$$

□

Guo and Kang [5] proved that any online algorithm for this problem has competitive ratio at least ϕ when an algorithm is allowed to delay the scheduling of a job. Although this result does not directly apply here, their lower bound instance can be easily modified to hold for our case as well.

Theorem 3 *The competitive ratio of any online algorithm for the case with job dependent overhead is at least $\phi = (1 + \sqrt{5})/2$ when $m = 2$.*

Proof Consider an arbitrary online algorithm A. An adversary begins by issuing a job with $p_1 = 1$ and $c_1 = \phi - 3/2$. If the algorithm assigns $k_1 = 1$, the adversary stops. In this case, A has competitive ratio at least $1/(\phi - 1) = \phi$. Otherwise, the adversary issues a second job with $p_2 = 1$ and $c_2 = 1/2$. Regardless of the value chosen for k_2, the makespan of A is ϕ. Since the optimal makespan is 1, the competitive ratio of A is at least ϕ. □

3 Three Machines

We now present an optimal online algorithm for $m = 3$. Let $I_3(j)$ denote the contiguous "blocked in" idle area that would result just prior to s_j if p_j were assigned to 3 processors. Here is our algorithm:

Schedule(j)

if $p_j > 9c + I_3(j)$ then
    $k_j \leftarrow 3$
else if $p_j > 18c$ then
    $k_j \leftarrow 2$
else
    $k_j \leftarrow 1$

In other words, the algorithm assigns

$$k_j = \begin{cases} 3 & \text{if } p_j > 9c + I_3(j) \\ 2 & \text{if } 18c < p_j \leq 9c + I_3(j) \\ 1 & \text{otherwise} \end{cases}$$

and a start time s_j as early as possible (on the least loaded processor(s) if $k_j < 3$). The algorithm does not backfill.

Theorem 4 *The competitive ratio of our algorithm is* 5/3.

Proof Let t_1^1 denote the total time during which only one job is executing on one processor. Let t_1^2 and t_1^3 denote the total time during which two and three jobs, respectively, are each executing concurrently on one processor in the algorithm's schedule. Let $I_3 = \sum_{j \in J_3} I_3(j)$. Notice that

$$I_3 \geq 2\left(t_1^1 - \widehat{t_1^1}\right) \tag{6}$$

where $\widehat{t_1^1}$ is the time at the end of the algorithm's schedule during which only job l, with $k_l = 1$, is executing. Also, notice that

$$\widehat{t_1^1} \leq p_l - I_3(l)/2$$

since $k_l = 1$ and $I_3(l)/2$ is a lower bound on the time that job l might be executing concurrently with one or more other jobs. By the algorithm, we know that

$$p_l \leq 18c \text{ and } p_l \leq 9c + I_3(l)$$

which implies that

$$p_l \leq \frac{27c + I_3(l)}{2} .$$

Therefore,

$$\widehat{t_1^1} \leq 27c/2 . \tag{7}$$

Also notice that

$$\sum_{j \in J_2} p_j > 18|J_2|c \tag{8}$$

and

$$\sum_{j \in J_3} p_j > \sum_{j \in J_3} (9c + I_3(j)) = 9|J_3|c + I_3 \; . \tag{9}$$

The optimal makespan is

$$C^* \geq \frac{t_1^1 + 2t_1^2 + 3t_1^3 + \sum_{j \in J_2 \cup J_3} p_j}{3} \; . \tag{10}$$

Finally, we examine the makespan of the online schedule:

$$\begin{aligned}
C &= t_1^1 + t_1^2 + t_1^3 + \sum_{j \in J_2} \left(\frac{p_j}{2} + 1 \right) + \sum_{j \in J_3} \left(\frac{p_j}{3} + 2 \right) \\
&= t_1^1 + t_1^2 + t_1^3 + \frac{\sum_{j \in J_2} p_j}{2} + |J_2|c + \frac{\sum_{j \in J_3} p_j}{3} + 2|J_3|c \\
&\leq t_1^1 + t_1^2 + t_1^3 + \frac{\sum_{j \in J_2} p_j}{2} + \frac{\sum_{j \in J_2} p_j}{18} + \frac{\sum_{j \in J_3} p_j}{3} + 2 \left(\frac{\sum_{j \in J_3} p_j - I_3}{9} \right) && \text{by (8) and (9)} \\
&= t_1^1 + t_1^2 + t_1^3 + \frac{5}{9} \sum_{j \in J_2 \cup J_3} p_j - \frac{2I_3}{9} \\
&\leq t_1^1 + t_1^2 + t_1^3 + \frac{5}{9} \sum_{j \in J_2 \cup J_3} p_j - \frac{4 \left(t_1^1 - \widehat{t_1^1} \right)}{9} && \text{by (6)} \\
&= \frac{5}{9} t_1^1 + t_1^2 + t_1^3 + \frac{5}{9} \sum_{j \in J_2 \cup J_3} p_j + \frac{4\widehat{t_1^1}}{9} \\
&\leq \frac{5}{9} t_1^1 + t_1^2 + t_1^3 + \frac{5}{9} \sum_{j \in J_2 \cup J_3} p_j + \frac{4 \cdot 27c/2}{9} && \text{by (7)} \\
&\leq \frac{5}{3} \left(\frac{t_1^1 + 2t_1^2 + 3t_1^3 + \sum_{j \in J_2 \cup J_3} p_j}{3} \right) + 6c && \text{by (10)} \\
&\leq \frac{5}{3} C^* + 6c \; .
\end{aligned}$$

The lower bound follows from the previously mentioned general lower bound of 5/3 [6]. □

We note that, like in the $m = 2$ case, the additive term arises from schedules having a relatively long job on a single processor at the end. We suspect that this additive term is not tight. Consider an instance σ consisting of 3 jobs with $p_1 = p_2 = 9c + \epsilon$ and $p_3 = 9c$. The algorithm will assign the first two jobs to three machines and the last job to one machine, giving makespan 19c, as $\epsilon \to \infty$. Since the optimal makespan is $9c$, we have $C(\sigma) = (5/3)\, C^*(\sigma) + 4c$.

4 Conclusions

We have given optimal (up to additive constants) online algorithms for scheduling malleable parallel jobs on $m = 2$ and 3 machines when the execution time of a job j is defined to be $p_j/k_j + (k_j - 1)c$,

where p_j is the processing requirement and the job is assigned to k_j machines. These algorithms have competitive ratios 3/2 and 5/3 when $m = 2$ and 3, respectively, matching previously known lower bounds [6]. We have also given an optimal online algorithm with competitive ratio $\phi = (1 + \sqrt{5})/2$ when $m = 2$ for the problem with a job-dependent overhead term c_j. The obvious open questions concern whether similar algorithms can be extended to general m and whether the additive terms can be eliminated.

References

[1] J. Blazewicz, M. Y. Kovalyov, and D. Trystram. Preemptable malleable task scheduling problem. *IEEE Transactions on Computers*, 55(4):486–490, 2006.

[2] R. Dutton, W. Mao, J. Chen, and I. W. Watson. Parallel job scheduling with overhead: A benchmark study. In *Proceedings of the IEEE International Conference on Networks, Architecture, and Storage (NAS)*, pages 326–333, 2008.

[3] R. A. Dutton and W. Mao. Online scheduling of malleable parallel jobs. In *Proceedings of the ISATED international Conference on Parallel and Distributed Computing and Systems*, pages 1–6, 2007.

[4] U. Faigle, W. Kern, and G. Turán. On the performance of on-line algorithms for partition problems. *Acta Cybernetica*, 9(2):107–119, 1989.

[5] S. Guo and L. Kang. Online scheduling of malleable parallel jobs on two identical machines. *European Journal of Operational Research*, 2010. In press, doi:10.1016/j.ejor.2010.03.005.

[6] J. T. Havill and W. Mao. Competitive online scheduling of perfectly malleable jobs with setup times. *European Journal of Operational Research*, 187:1126–1142, 2008.

[7] J. T. Havill, W. Mao, and V. Dimitrov. Improved parallel job scheduling with overhead. In *Proceedings of the Seventh Joint Conference on Information Sciences*, pages 393–396, 2003.

[8] J. L. Hurink and J. J. Paulus. Improved online algorithms for parallel job scheduling and strip packing. *Theoretical Computer Science*, 2009 (to appear).

[9] K. Jansen. Scheduling malleable parallel tasks: An asymptotic fully polynomial time approximation scheme. *Algorithmica*, 39(1):59–81, 2004.

[10] W. Kern and J. J. Paulus. A note on the lower bound for online strip packing. Technical Report 1893, Department of Applied Mathematics, University of Twente, 2009.

[11] J. Sgall. On-line scheduling of parallel jobs. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science, LNCS 841*, pages 159–176, 1994.

[12] D. Ye, X. Han, and G. Zhang. A note on online strip packing. *Journal of Combinatorial Optimization*, 17:417–423, 2009.