An optimal algorithm for preemptive on-line scheduling

Article *in* Operations Research Letters · October 1995

DOI: 10.1016/0167-6377(95)00039-9

3 authors, including:

Bo Chen
The University of Warwick
76 PUBLICATIONS **1,413** CITATIONS

SEE PROFILE

An Optimal Algorithm for Preemptive On-line Scheduling *

Bo Chen †
André van Vliet ‡
Gerhard J. Woeginger §

June 22, 1995

Abstract. We investigate the problem of on-line scheduling jobs on m identical parallel machines where preemption is allowed. The goal is to minimize the makespan. We derive an approximation algorithm with worst-case guarantee $m^m/(m^m-(m-1)^m)$ for every $m \geq 2$, which increasingly tends to $e/(e-1) \approx 1.58$ as $m \to \infty$. Moreover, we prove that for no $m \geq 2$ there does exist any approximation algorithm with a better worst-case guarantee.

Keywords. scheduling $*$ preemption $*$ on-line algorithms $*$ worst-case bounds

1 Introduction

We study the problem of preemptively scheduling a list $(J_1, J_2, \ldots)$ of jobs on-line on m identical parallel machines. Associated with each job J_j is its *processing time* (or *length*) $p_j = p(J_j)$. At any time, each machine can handle at most one job and each job can be processed by at most one machine. *Preemption* is permitted, which allows to split a job and spread its processing over several machines. The jobs are not known *a priori*: job J_j becomes known only when J_{j-1} has already been scheduled. As soon as job J_j appears, it must *irrevocably* be assigned to one or more time slots of one or more machines. The objective is to find a schedule that minimizes the maximum completion time, or *makespan*.

The off-line version of this problem, where all the job information, such as the total number of jobs, their arrival times and processing times, is fully known in advance, can be easily solved to optimality in polynomial time (see McNaughton [2]). On the other hand, the off-line version *without* preemption is known to be strongly NP-complete [1].

The quality of an (approximation) algorithm H, often called *heuristic*, is usually measured by its *worst-case* (*performance*) *ratio*

$$R^H(m) = \sup\{H(L)/\textsc{Opt}(L) : \ L \text{ is a list of jobs}\}, \tag{1}$$

*This article appears in *Operations Research Letters* 18 (1995), 127–131. The work was supported by the Tinbergen Institute at Erasmus University Rotterdam and by the Christian Doppler Laboratorium für Diskrete Optimierung.

†Warwick Business School, University of Warwick, Coventry, CV4 7AL, U.K.
‡Econometric Institute, Erasmus University Rotterdam, 3000 DR Rotterdam, The Netherlands.
§TU Graz, Institut für Mathematik, Kopernikusgasse 24, A-8010 Graz, Austria.

where $H(L)$ denotes the makespan of a schedule produced by heuristic H for scheduling the list L of jobs on m machines, and $\textsc{Opt}(L)$ denotes the corresponding makespan of some (off-line) optimal schedule.

In this note we will develop an approximation algorithm with worst-case ratio $m^m/(m^m - (m-1)^m)$ for on-line preemptive scheduling on m machines. As m increases, this ratio increasingly tends to $e/(e-1) \approx 1.58$, where e denotes the Eulerian constant. By deriving a lower bound on the worst-case ratio of any on-line algorithm, we will prove that our algorithm is *best possible*.

2 Preliminaries

For a set of jobs with processing times $p_1, \ldots, p_n$, there are two straightforward lower bounds for the makespan of any preemptive schedule on m machines. First, the makespan must be at least $\max_{1\le j\le n}\{p_j\}$ since no job can be processed on two machines at the same time. Secondly, the makespan must be at least $(\sum_{j=1}^n p_j)/m$, the average machine load. McNaughton [2] proved that there always exists a schedule with makespan equal to the maximum of these two bounds.

Proposition 2.1 (McNaughton) *For every set of n jobs with processing times $p_1, \ldots, p_n$ and for $m \ge 1$ machines, the length* $\textsc{Opt}$ *of the optimal preemptive schedule equals* $\max\{(\sum_{j=1}^n p_j)/m, \max_{1\le j\le n} p_j\}$. □

We will use the following notation. For $m \ge 2$, we introduce the numbers $\alpha(m) = m/(m-1)$ and

$$r(m) = \frac{\alpha^m}{\alpha^m - 1} = \frac{m^m}{m^m - (m-1)^m}.$$

To simplify the presentation, we will drop the dependence on m and always write α and r instead of $\alpha(m)$ and $r(m)$.

3 The Approximation Algorithm

Throughout this section, the machine loads at step $t \ge 0$ (i.e., immediately after the t-th job has been scheduled) will be denoted by $\{L_i^t\}$, where $L_1^t \le L_2^t \le \cdots \le L_m^t$. The corresponding machines are denoted by $M_1^t, \ldots, M_m^t$. The current optimum makespan in step t (for the job set $\{J_1, \ldots, J_t\}$) is denoted by $\textsc{Opt}^t$, and the sum $\sum_{j=1}^t p_j$ of the processing times of all known jobs is denoted by S^t.

The basic principle of our algorithm is to keep the most lightly loaded machines still lightly loaded in an anticipation of possible arrival of longer jobs. The algorithm will schedule jobs in such a way that, at any step $t \ge 0$,

(I1) $L_m^t \le r\textsc{Opt}^t$.

(I2) For any $1 \le k \le m$,

$$\sum_{i=1}^k L_i^t \le \frac{\alpha^k - 1}{\alpha^m - 1} S^t.$$

Let us formally describe how the algorithm schedules a new job J_{t+1}. First, compute the new optimum makespan OPT^{t+1} according to Proposition 2.1. If $L_m^t + p_{t+1} \le r\text{OPT}^{t+1}$, then schedule job J_{t+1} onto machine M_m^t. Otherwise, let $\ell = \min\{1 \le i \le m : L_i^t + p_{t+1} \ge r\text{OPT}^{t+1}\}$. Schedule $r\text{OPT}^{t+1} - L_\ell^t$ portion of J_{t+1} onto machine M_ℓ^t, and the remaining part of J_{t+1}, if any, onto machine $M_{\ell-1}^t$.

The following two lemmas show that the above described algorithm is well defined, and (I1) and (I2) are always fulfilled.

Lemma 3.1 *If* (I1) *and* (I2) *are fulfilled at step t, then job J_{t+1} can be successfully scheduled, and* (I1) *is still fulfilled at step $t+1$.*

Proof. If $L_m^t + p_{t+1} \le r\text{OPT}^{t+1}$ then we are done, since $L_m^{t+1} = L_m^t + p_{t+1}$. Suppose $L_m^t + p_{t+1} > r\text{OPT}^{t+1}$. Then ℓ is well defined. Suppose $\ell \ge 2$. Since

$$L_{\ell-1}^t + (p_{t+1} - (r\text{OPT}^{t+1} - L_\ell^t)) < L_\ell^t$$

according to the definition of ℓ, we conclude that, if machine $M_{\ell-1}^t$ receives any workload of J_{t+1}, then it can finish the work by L_ℓ^t, the time machine M_ℓ^t starts processing J_{t+1}. Therefore, the constraint that any job can be processed by at most one machine at a time is honored. On the other hand, we have

$$L_m^{t+1} = L_\ell^t + (r\text{OPT}^{t+1} - L_\ell^t) = r\text{OPT}^{t+1},$$

hence (I1) is fulfilled.

Finally, suppose $\ell = 1$. We will show that $L_1^t + p_{t+1} \le r\text{OPT}^{t+1}$. Hence we can conclude that $L_1^t + p_{t+1} = r\text{OPT}^{t+1}$, which simply implies what we want to prove.

Condition (I2) with $k = 1$ yields $L_1^t \le (\alpha - 1)S^t/(\alpha^m - 1)$. Let $S^t = \lambda p_{t+1}$ for some $\lambda \ge 0$. It is easy to verify that

$$\frac{(\alpha-1)\lambda}{\alpha^m - 1} + 1 \le r \max\left\{\frac{\lambda+1}{m}, 1\right\},$$

which, together with the fact that

$$\text{OPT}^{t+1} \ge \max\left\{\frac{S^t + p_{t+1}}{m}, p_{t+1}\right\} = p_{t+1}\max\left\{\frac{\lambda+1}{m}, 1\right\},$$

leads to

$$\begin{aligned} L_1^t + p_{t+1} &\le \frac{\alpha-1}{\alpha^m-1}S^t + p_{t+1} = \left(\frac{(\alpha-1)\lambda}{\alpha^m-1} + 1\right)p_{t+1} \\ &\le \left(\frac{(\alpha-1)\lambda}{\alpha^m-1} + 1\right)\left(\max\left\{\frac{\lambda+1}{m}, 1\right\}\right)^{-1}\text{OPT}^{t+1} \le r\text{OPT}^{t+1}. \end{aligned}$$

This completes our argument. □

Lemma 3.2 *If* (I1) *and* (I2) *are fulfilled at step t, then* (I2) *is still fulfilled at step $t+1$.*

Proof. We proof that for each $1 \le k \le m$ the corresponding inequality in (I2) holds. Let

$$u = \min\{1 \le i \le m : \text{machine } M_i^t \text{ receives workload of } J_{t+1}\}.$$

We will distinguish two cases depending on the machine index u.

If $k < u$, all the loads in the left-hand side of the inequality in (I2) did not increase when going from step t to step $t+1$, whereas the sum of processing times in the right-hand side of (I2) did increase by p_{t+1}. This settles the easy first case.

In the other case $k \geq u$, we will prove that

$$\sum_{i=k+1}^{m} L_i^{t+1} \geq \frac{\alpha^m - \alpha^k}{\alpha^m - 1} S^{t+1}. \tag{2}$$

Since $\sum_{i=1}^{m} L_i^{t+1} = S^{t+1}$, inequality (2) is equivalent to the desired inequality in (I2). If $k = m$ then (2) is trivial. Suppose $k < m$. Observe that by the way the algorithm schedules job J_{t+1}, we have $L_m^{t+1} = r\text{OPT}^{t+1}$ and $L_i^{t+1} = L_{i+1}^t$ for $u < k+1 \leq i \leq m-1$, which transform the expression in the left-hand side of (2) into

$$\sum_{i=k+2}^{m} L_i^t + r\text{OPT}^{t+1} \geq \frac{\alpha^m - \alpha^{k+1}}{\alpha^m - 1}(S^{t+1} - p_{t+1}) + r\text{OPT}^{t+1}, \tag{3}$$

where the inequality follows from (I2) at step t for $k+1$ and from $S^t = S^{t+1} - p_{t+1}$. Let $p_{t+1} = \mu S^{t+1}$ for some $0 \leq \mu \leq 1$. Then, noticing that $\text{OPT}^{t+1} \geq \max\{S^{t+1}/m, p_{t+1}\}$, we conclude that the right-hand side of (3) is at least

$$\frac{(\alpha^m - \alpha^{k+1})(1-\mu)}{\alpha^m - 1} S^{t+1} + r \max\Big\{\frac{1}{m}, \mu\Big\} S^{t+1}. \tag{4}$$

By direct calculation one finds that the right-hand side of (4) cannot be smaller than that of (2). □

In conclusion, we have established the following.

Theorem 3.3 *The approximation algorithm we described has a worst-case ratio at most* $r(m) = m^m/(m^m - (m-1)^m) < e/(e-1)$.

Proof. The above two lemmas, together with the trivial fact that, at step 0, both (I1) and (I2) are fulfilled, imply that (I1) and (I2) are fulfilled at any step of the algorithm. Then (I1) immediately leads to the theorem. □

4 A Matching Lower Bound

In this section, we prove that there cannot exist any on-line algorithm with worst-case ratio smaller than $r(m)$. From the problem statement we see that one has the freedom to reserve idle time for coming new jobs. However, intuition suggests that this sort of idle time is not necessary. The following observation supports this intuition.

Observation 4.1 *If there exists an approximation algorithm* H_1 *for on-line preemptive scheduling with worst-case ratio* r^**, then there exists another approximation algorithm* H_2 *with a worst-case ratio at most* r^* *that never introduces machine idle time between two jobs on the same machine.*

Proof. Given any schedule S_1 (with or without idle time). For any $0 \le i \le m$, we denote by I_i a maximal (with respect to containment) time interval during which exactly i machines are busy. Let $T_i = \{I_i\}$ and $t_i = \sum\{|I_i| : I_i \in T_i\}$ for $i = 0, \ldots, m$, where $|I_i|$ denotes the length of interval I_i. Then we see that the makespan of S_1 is at least $\sum_{i=1}^{m} t_i$. Consider the following schedule S_2: During time interval $[\sum_{j=m-i+2}^{m} t_j, \sum_{j=m-i+1}^{m} t_j]$ only machine M_i up to machine M_m are busy, $i = 1, \ldots, m$. Clearly, this simple schedule does not have machine idle time between any two jobs, and its makespan is $\sum_{i=1}^{m} t_i$. Moreover, both schedules S_1 and S_2 have the same $\{t_i\}_{i=0}^{m}$.

Now suppose both S_1 and S_2 are schedules of the same set of jobs, and one more job is added to S_1. Suppose the job is processed within time interval(s) of T_i for q_i time units, $i = 0, \ldots, m-1$. Then in S_2 we can simulate this assignment by letting machine M_{m-i} start processing q_i portion of the job at time $\sum_{j=i+1}^{m} t_j$, $i = 0, \ldots, m-1$. It is easy to see that after adding the job to S_1 and S_2 in the above way, the makespan of the new S_2 is still less than or equal to that of the new S_1, and both new schedules have the same new $\{t_i\}_{i=1}^{m}$.

It should be clear now that the required approximation algorithm H_2 can be obtained by making H_2 simulate H_1 for assigning every job in the way we just described. Both H_1 and H_2 start with the same empty schedule. □

Next, we introduce the following sequence of jobs, and consider any algorithm's behavior in assigning these jobs.

- At step 0, there arrive m jobs, all with processing times $1/m$.
- At step t, $t \ge 1$, there arrives a new job J_t with processing time $p_t = m^{t-1}/(m-1)^t$.

Observation 4.2 *At step t, immediately after the arrival of job J_t, the overall sum of processing times equals α^t and the optimum makespan equals $p_t = \alpha^t/m$.*

Proof. This follows from Proposition 2.1. □

Now suppose that there exists an approximation algorithm H with worst case ratio $r(m)-\varepsilon$ for some real $\varepsilon > 0$. By Observation 4.1 we may assume that H never introduces machine idle time. Analogously to the preceding section, we denote by $L_1^t \le L_2^t \le \cdots \le L_m^t$ the machine loads at the time immediately after algorithm H scheduled job J_t.

Claim 4.3 *For any $t \ge 0$, $L_m^t \le (r-\varepsilon)\alpha^t/m$ must hold.*

Proof. Since $L_m^t \le (r-\varepsilon)\mathrm{OPT}^t = (r-\varepsilon)\alpha^t/m$. □

Claim 4.4 *For all $1 \le i \le m-1$ and all $t \ge 0$, $L_{m-i}^{t+i} \le L_m^t$ holds.*

Proof. We use induction on i. For $i = 1$ we must prove that $L_{m-1}^{t+1} \le L_m^t$ holds. Suppose the contrary. Then at step $t+1$, two machines have loads larger than L_m^t. Since H does not introduce machine idle time, at the time point L_m^t, the job J_{t+1} was processed on at least two machines with largest and second largest load, which is a contradiction.

Now assume that we have proved the claim up to some fixed value i ($1 \le i \le m-2$), i.e., $L_{m-i}^{t+i} \le L_m^t$. In case $L_{m-i-1}^{t+i+1} > L_m^t \ge L_{m-i}^{t+i}$ would be true, we could argue as above that job J_{t+i+1} is processed on at least two machines at the same time. □

Theorem 4.5 *For all $m \geq 2$, there does not exist any on-line algorithm for preemptive scheduling on m machines with worst-case ratio strictly smaller than $r(m)$.*

Proof. We are ready to derive a contradiction to the assumption that H has worst-case ratio $r - \varepsilon$. Let us take a closer look at step $(m-1)$.

By Claim 4.4, we know that $L_i^{m-1} \leq L_m^{i-1}$ for all $1 \leq i \leq m-1$, which, together with Claim 4.3 and Observation 4.2, yields that $(r-\varepsilon)$ times the current optimum makespan

$$\begin{aligned}(r-\varepsilon)\alpha^{m-1}/m &\geq L_m^{m-1} = \alpha^{m-1} - \sum_{i=1}^{m-1} L_i^{m-1} \\ &\geq \alpha^{m-1} - \sum_{j=1}^{m-1} L_m^{j-1} \geq \alpha^{m-1} - \sum_{i=1}^{m-1} (r-\varepsilon)\alpha^{i-1}/m\end{aligned}$$

Comparing the last expression in the right-hand side to the left-hand side and doing some easy calculations, we see that $\varepsilon \leq 0$ must hold. □

Acknowledgement. The main part of this work was carried out while the first author was working at Erasmus University Rotterdam, and the third author was visiting this University.

References

[1] M.R. Garey and D.S. Johnson, "Strong NP-completeness results: Motivation, examples and implications", *J. Assoc. Comput. Mach.* **25** (1978), 499–508.

[2] R. McNaughton, "Scheduling with deadlines and loss functions", *Management Sci.* **6** (1959), 1–12.