A new algorithm for online uniform-machine scheduling to minimize the makespan

T.C.E. Cheng[*][†] C.T. Ng[‡] Vladimir Kotov[§]

Abstract

We consider the online scheduling problem with $m-1$, $m \geq 2$, uniform machines each with a processing speed of 1, and one machine with a speed of s, $1 \leq s \leq 2$, to minimize the makespan. The well-known list scheduling (LS) algorithm has a worst-case bound of $\frac{3m-1}{m+1}$ [1]. An algorithm with a better competitive ratio was proposed in [3]. It has a worst-case bound of 2.8795 for a big m and $s = 2$. In this note we present a 2.45-competitive algorithm for $m \geq 4$ and any s, $1 \leq s \leq 2$.

Keywords: Online algorithms; competitive ratio; multi-machine scheduling; uniform machines

1 Introduction

In the classical uniform-machine scheduling problem, there are $m \geq 2$ machines $(M_1, M_2, \ldots, M_m)$ with speeds $(s_1, s_2, \ldots, s_m)$ associated with the machines, respectively. A list of n independent jobs with nonnegative processing times $p_1, \ldots, p_n$ has to be scheduled nonpreemptively on these machines with the objective of minimizing the makespan. In the *online* version of the multi-machine scheduling problem, each job must be immediately and irrevocably assigned to one of the machines without any knowledge of the future

[*]Department of Logistics, The Hong Kong Polytechnic University, Kowloon, Hong Kong, lgtcheng@inet.polyu.edu.hk

[†]corresponding author

[‡]Department of Logistics, The Hong Kong Polytechnic University, Kowloon, Hong Kong, lgtctng@inet.polyu.edu.hk

[§]Faculty of Applied Mathematics and Computer Science, Belarusian State University, Skarina ave. 4, Minsk, 220050, Belarus, kotovVM@bsu.by

jobs. The performance of an online algorithm is measured by its *competitive ratio*, i.e., the worst-case ratio with respect to the optimal solution of the corresponding offline problem.

The online multi-machine scheduling problem was first investigated by Graham, who showed that the list scheduling (LS) algorithm has a competitive ratio of exactly $2 - 1/m$ [2]. Cho and Sahni [1] proved that for the uniform-machine system, the LS algorithm has a worst-case bound of $\frac{3m-1}{m+1}$ for $m \geq 3$. When $s_i = 1$ ($i = 1, \ldots, m-1$) and $s_m > 1$, Cho and Sahni also showed that the LS algorithm has a worst-case bound of $1 + \frac{(m-1)}{m+s-1} \min(2, s) \leq 3 - \frac{4}{m+1}$, and the bound $3 - \frac{4}{m+1}$ is achieved when $s = 2$.

Li and Shi [3] proved that the LS algorithm is the best possible one for $m \leq 3$, and proposed an algorithm that is significantly better than the LS algorithm when $s_i = 1$ ($i = 1, \ldots, m-1$) and $s_m = 2$, $m \geq 4$. The algorithm has a worst-case bound of 2.8795 for a big m.

In this note we present a 2.45-competitive algorithm for any $m \geq 4$ and any s_m, $1 \leq s_m = s \leq 2$. This is a marked improvement over the algorithm of Li and Shi because our algorithm yields a considerably lower competitive ratio while our results hold under more general conditions. The construction of the algorithm is based on simple logical reasoning, which is different from the traditional approach applied to construct the LS algorithm.

2 A 2.45-competitive algorithm for an arbitrary number of machines

Before presenting the main results, we introduce some notation.

m = the number of machines;

$V_{i,j}$ = the current load of the ith machine after assigning p_j in step j;

V_j = the minimum current load of the machines $1, \ldots, m-1$ after assigning p_j in step j;

L_j = the lower bound for the makespan in step j; $L_j = \max(\frac{p_1 + \ldots + p_j}{m+s-1}, \frac{p_{max}}{s}, p_{max2})$, where $p_{max} = \max(p_1, \ldots, p_j)$, and $p_{max2} = \max(p_1, \ldots, p_{max} - p_{max}, \ldots, p_j)$.

Property 1 $L_1 \leq L_2 \leq \ldots \leq L_j$.

Algorithm S.

Let $\alpha = 1.45$. *In step* j *we assign* p_j *to the most loaded machine* $i < m$ *with the property* $V_{i,j-1} + p_j \leq (1 + \alpha) L_j$ *if such a machine exists; otherwise, we assign* p_j *to the* m*th machine.*

Lemma 1 *In step j of algorithm S, if $V_{m,j-1} < V_{m,j}$ holds, then $p_j > p_{max2}$.*

Proof: Let $p_j \le p_{max2}$. The algorithm assigns p_j to machine m only when $V_{t,j-1} + p_j > (1+\alpha)L_j$ for all $t = 1, \dots, m-1$. From this, it follows that $V_{t,j-1} > (1+\alpha)L_j - p_j \ge \alpha L_j$ for all $t = 1, \dots, m-1$. Therefore, $L_j > L_j \frac{(m-1)\alpha+1}{m+s-1}$. But this is not possible for $m \ge 4$. ■

Lemma 2 *In step j of algorithm S, if $V_{m,j-1} = V_{m,j} > L_j$ holds, then $V_{j-1} < V_j$ only if $p_j > \alpha L_j$.*

Proof: W.l.o.g. let V_{j-1} be the load of the $(m-1)$th machine, i.e., $V_{j-1} = V_{m-1,j-1}$. It is easy to see that if $p_j \le \alpha L_j$, then the algorithm assigns p_j to machine $m-1$ only when $V_{t,j-1} + p_j > (1+\alpha)L_j$ for all $t = 1, \dots, m-2$. But this is possible in case $V_{t,j-1} > L_j$ for all $t = 1, \dots, m-3$ and $V_{t,j-1} + p_j > (1+\alpha)L_j$ for $t = m-2$.

From this and $V_{m,j-1} = V_{m,j} > L_j$, it follows that the total processing time of the jobs is larger than $(m-3+1+\alpha+s)L_j$.

Hence $L_j > L_j \frac{m-2+\alpha+s}{m+s-1}$. But this is not possible for $m \ge 4$. ■

Lemma 3 *In step j of algorithm S, if $V_{m,j-1} < V_{m,j}$ holds, then $p_j > (1+\alpha)L_j - V_{j-1}$.*

Proof: In step j, algorithm S assigns p_j to machine m because $V_{j-1} + p_j > (1+\alpha)L_j$. ■

Lemma 4 *Let $f_1, f_2, \dots, f_q$ be the jobs assigned to machine m, then $f_{l-1} < \alpha f_l$ for $l = 2, \dots, q$.*

Proof: In step j of algorithm S, let $V_{m,j-1} < V_{m,j}$ and $p_j < \alpha f_q$ hold. It is easy to see that if $p_j \le \alpha L_j$, then the algorithm assigns p_j to machine m only when $V_{t,j-1} + p_j > (1+\alpha)L_j$ for all $t = 1, \dots, m-1$. From this, it follows that $L_j > \frac{L_j(m-1+\alpha)+f_q}{m+s-1}$. But this is possible only when $L_j > f_q/(s-\alpha) > p_j/(s-\alpha)\alpha \ge L_j/(2-\alpha)\alpha$, a contradiction. ■

Theorem 2 *For $\alpha = 1.45$, in any step j of algorithm S, either $V_{j-1} + p_j \le (1+\alpha)L_j$ or $V_{m,j-1} + \frac{p_j}{s} \le (1+\alpha)L_j$.*

Proof: In step n of algorithm S, let

$$V_{n-1} + p_n > (1+\alpha)L_n, V_{m,n-1} + \frac{p_n}{s} > (1+\alpha)L_n. \tag{1}$$

This means that $V_{m,n-1} > \alpha L_n$; otherwise, we have a contradiction with $\frac{p_n}{s} \le L_n$ and $V_{m,n-1} + \frac{p_n}{s} > (1+\alpha)L_n$. Therefore, we can determine the maximum index k such that $V_{m,k-1} \le L_n$ and $V_{m,k} > L_n$. This means that in any current step j, $j = k+1, \ldots, n$, we can apply Lemma 2.

Let $Z = p_k$.

Let $\beta = 1 + \alpha - \frac{1}{\alpha}$ and let $\gamma = 1 + \alpha - \frac{1}{\alpha\beta}$.

Let $Z_1, Z_2, \ldots, Z_q = p_n$ be the jobs assigned after job $Z = p_k$ to machine m, $q \ge 1$. Let $Y_1, Y_2, \ldots, Y_t$ be the jobs assigned to machine m before Z, and after this assignment, let the current load become bigger than V_k. W.l.o.g. let Y_j be assigned before Y_{j+1}, $j = 1, \ldots, t-1$.

First, we show that

$$\beta Z < Z_1, \beta Z_1 < Z_2, \ldots, \beta Z_{q-1} < Z_q. \tag{2}$$

Indeed, during the current step j, $j = k+1, \ldots, n$, algorithm S either changes or does not change V_{j-1}. Recall that $V_j < V_{m,k}$. In the former case, let j correspond to the step of algorithm S after assigning Z_f and before assigning Z_{f+1}. From Lemma 2, it follows that $p_j > \alpha Z_f$. Therefore, for Z_{f+1}, we have $Z_{f+1} > \alpha p_j$. Hence, $Z_{f+1} > \alpha^2 Z_f > (1 + \alpha - \frac{1}{\alpha})Z_f$ for $\alpha \ge 1.45$. In the latter case, the result follows from Lemma 3, taking into consideration $V_{j-1} = V_{k-1} < Z/\alpha$.

Therefore,

$$Z + Z_1 + \ldots + Z_q < Z_q(1 + \frac{1}{\beta} + \frac{1}{\beta^2} + \ldots + \frac{1}{\beta^q}). \tag{3}$$

It should be mentioned that if job Z_2 was assigned without changing V_{k-1}, then from Lemma 3, taking into consideration $V_{j-1} = V_{k-1} < \frac{Z_1}{\alpha\beta}$, it follows that $Z_2 > \gamma Z_1$.

From Lemma 4, we have

$$Y_1 + \ldots + Y_t < Y_t(1 + \frac{1}{\alpha} + \frac{1}{\alpha^2} + \ldots + \frac{1}{\alpha^{t-1}}) < \frac{\alpha Y_t}{\alpha - 1}. \tag{4}$$

Let job Y_t be the jobs assigned to machine m during step $t1$.

Case 1. $p_n = Z_1$, i.e., only one job p_n is assigned to machine m after job p_k.

1.1. $L_k \geq V_{m,k-1}$. From Lemma 3 and the definition of k, it follows that $Z > \alpha L_k$. From Lemma 3, it follows that $p_n > \alpha Z$. Therefore, from (1), it follows that

$1 + \alpha < \frac{V_{m,k-1} + \frac{Z}{s} + \frac{p_n}{s}}{L_n} \leq \frac{L_k + \frac{Z}{s} + \frac{p_n}{s}}{\max(Z, \frac{p_n}{s})} \leq \frac{1}{\alpha} + \frac{1}{\alpha} + 1.$

For $\alpha \geq 1.45$, this is not possible.

1.2. $L_k < V_{m,k-1}$. This means that $V_k < V_{m,k-1}$.

1.2.1. $V_k < V_{n-1}$. The algorithm changes V_k by adding at least one job p_j to the machine that has the minimum load in step k. From Lemma 2, it follows that $p_j > \alpha Z$. Taking into consideration Lemmas 1 and 3, we have $p_n > \alpha p_j > \alpha^2 Z$. Therefore, from $L_n \geq V_{m,k-1}$ and (1), we obtain

$1 + \alpha < \frac{V_{m,k-1} + \frac{Z}{s} + \frac{p_n}{s}}{\max(\frac{p_n}{s}, L_n)} < 2 + \frac{1}{\alpha^2},$

which is not possible for $\alpha \geq 1.45$.

1.2.2. $V_k = V_{n-1}$. From Lemma 3, if follows that $Y_t > \alpha L_{t1}$. Taking into consideration Lemma 3 for $k = t1$, we have $p_k > Y_t(1 + \alpha - \frac{1}{\alpha}) = \beta Y_t$. Therefore, from (1) and (4), it follows that

$1 + \alpha < \frac{V_{m,k-1} + \frac{Z}{s} + \frac{p_n}{s}}{L_n} \leq \frac{V_{m,k-1} + \frac{Z}{s} + \frac{p_n}{s}}{\max(L_n, \frac{p_n}{s})} \leq \frac{\alpha}{\beta\gamma(\alpha-1)} + \frac{1}{\gamma} + 1,$

or

$1 + \alpha < \frac{V_{m,k-1} + \frac{Z}{s} + \frac{p_n}{s}}{L_n} \leq \frac{V_{m,k-1} + \frac{Z}{s} + \frac{p_n}{s}}{\max(L_n, \frac{p_n}{s})} \leq \frac{1}{\alpha\beta(\alpha-1)} + \frac{1}{\beta} + 1.$

The first possibility corresponds to the situation when $V_k = V_{t1}$.

This is not possible for $\alpha \geq 1.45$.

Case 2. $p_n = Z_q$ and $q \geq 2$.

2.1. $L_k \geq V_{m,k-1}$ From (1), (2) and (3), it follows that

$1 + \alpha < \frac{V_{m,k-1} + \frac{Z}{s} + \frac{Z_1}{s} + \ldots + \frac{Z_q}{s}}{\max(Z_1, \frac{Z_q}{s})} < \frac{1}{\alpha\beta^q} + 1 + \frac{1}{\beta} + \frac{1}{\beta^2} + \ldots + \frac{1}{\beta^q},$

which is not possible for $\alpha \geq 1.45$.

2.2. $L_k < V_{m,k-1}$.

2.2.1. $q = 2$ and $V_{t1} = V_{n-1}$.

From (1), (4) and Lemma 3, it follows that

$1 + \alpha < \frac{V_{m,k-1} + \frac{Z}{s} + \frac{Z_1}{s} + \frac{Z_2}{s}}{\max(Z_1, \frac{Z_2}{s})} < \frac{\alpha}{(\alpha-1)\beta^2\gamma} + \frac{1}{\beta\gamma} + \frac{1}{\gamma} + 1,$

which is not possible for $\alpha \geq 1.45$.

2.2.3. $q = 2$, $V_{t1} < V_{n-1}$.

From (1), (4), Lemmas 2 and 3, it follows that

$1+\alpha < \frac{V_{m,k-1}+\frac{Z}{s}+\frac{Z_1}{s}+\frac{Z_2}{s}}{\max(Z_1,\frac{Z_2}{s})} < \frac{1}{\alpha\beta^2(\alpha-1)}+\frac{1}{\alpha^2\beta}+\frac{1}{\beta}+1,$

or

$1+\alpha < \frac{V_{m,k-1}+\frac{Z}{s}+\frac{Z_1}{s}+\frac{Z_2}{s}}{\max(Z_1,\frac{Z_2}{s})} < \frac{1}{\alpha\beta\gamma(\alpha-1)}+\frac{1}{\beta\gamma}+\frac{1}{\gamma}+1,$

which is not possible for $\alpha \geq 1.45$.

2.4. $q > 2$.

From (3) and (4), it follows that

$1+\alpha < \frac{V_{m,k-1}+\frac{Z}{s}+\frac{Z_1}{s}+\ldots+\frac{Z_q}{s}}{max(Z_{q-1},\frac{Z_q}{s})} < \frac{\alpha}{(\alpha-1)\beta^{q+1}}+1+\frac{1}{\beta}+\frac{1}{\beta^2}+\ldots+\frac{1}{\beta^q},$

which is not possible for $\alpha \geq 1.45$. ■

It should be pointed out that we have only applied the basic relation (1) without taking into consideration the value of m. If m is a fixed number, we can rewrite the inequality in Lemma 3 as

$$p_j > \alpha L_j + \frac{p_j}{m+s-1} \qquad (5)$$

because the difference between V_{j-1} and L_j is at least $\frac{p_j}{m+s-1}$. Therefore, we can use the inequality $p_n > \alpha \max(Z, L_n)\frac{m+s-1}{m+s-2}$.

Hence, for a fixed m, we can repeat the above analysis and obtain an even better worst-case performance bound for our proposed algorithm.

3 Conclusions

In this note we presented a simple algorithm that yields the best known competitive ratio for online uniform-machine scheduling to minimize the makespan. In addition, we developed a new technique for analyzing the worst-case performance of our algorithm, which is quite different from the traditional approach used to analyze the LS algorithm. It is an interesting open question whether there exists such an algorithm for the case of $s > 2$.

Acknowledgments

This research was supported in part by The Hong Kong Polytechnic University under grant number G-T596. Vladimir Kotov was also supported in part by INTAS-Belarus 03-50-5975.

References

[1] Y. Cho and S. Sahni, Bounds for list schedules on uniform processors, *SIAM J. Comput.,* **9** (1980) 91–103.

[2] R.L. Graham, Bounds on multiprocessing timing anomalies, *SIAM J. Appl. Math.* **17** (1969) 263–269.

[3] R. Li and L. Shi, An on-line algorithm for some uniform processor scheduling, *SIAM J. Comput.,* **27** (1998) 414–422.