Intelligent Information Management, 2017, 9, 245-254
http://www.scirp.org/journal/iim
ISSN Online: 2160-5920
ISSN Print: 2160-5912

Ordinal Semi On-Line Scheduling for Jobs with Arbitrary Release Times on Identical Parallel Machines

Sai Ji[1], Rongheng Li[1*], Yunxia Zhou[2]

[1]Key Laboratory of High Performance Computing and Stochastic Information Processing, Department of Mathematics, Hunan Normal University, Changsha, China
[2]Department of Computer, Hunan Normal University, Changsha, China
Email: *lirongheng@hunnu.edu.cn

How to cite this paper: Ji, S., Li, R.H. and Zhou, Y.X. (2017) Ordinal Semi On-Line Scheduling for Jobs with Arbitrary Release Times on Identical Parallel Machines. *Intelligent Information Management*, **9**, 245-254.
https://doi.org/10.4236/iim.2017.96014

Received: October 4, 2017
Accepted: November 12, 2017
Published: November 15, 2017

Abstract

In this paper, we investigate the problem of semi-on-line scheduling n jobs on m identical parallel machines under the assumption that the ordering of the jobs by processing time is known and the jobs have arbitrary release times. Our aim is to minimize the maximum completion time. An ordinal algorithm is investigated and its worst case ratio is analyzed.

Keywords

Schedule, Algorithm, Worst Case Ratio, Parallel Machines

1. Introduction

The problem of minimizing the maximum completion time for scheduling n jobs on m identical parallel machines (which is denoted by $P_m/\cdot/C_{\max}$) have attracted the interests of many researchers since it was proposed by Graham in 1969 [1]. The problem is defined as follows: Given a job set $L=\{J_1,J_2,\cdots,J_n\}$ of n jobs and an identical parallel machine set $\{M_1,M_2,\cdots,M_m\}$, where job J_j has non-negative processing time p_j, assign the jobs onto the machines so as to minimize the maximum completion of the m machines.

A scheduling problem is called off-line if we have complete information about the job data before constructing a schedule. In contrast, the scheduling problem is called online if the jobs appear one by one and it requires scheduling the arriving job irrevocably on a machine without knowledge of the future jobs. The processing time of next job becomes available only after the current job is scheduled. Graham [1] proposed the List Scheduling (LS) algorithm to minimize the

maximum completion time for online scheduling n jobs on m identical parallel machines.

Li and Huang [2] generalized Graham's classical on-line scheduling problem to m identical machines. They describe the requests of all jobs in terms of order. For an order of the job J_j, the scheduler is informed of a 2-tuple $\left(r_j, p_j\right)$, where r_j and p_j represent the release time and the processing time of the job J_j, respectively. The orders of request have no release time but appear online one by one at the very beginning time of the system. In this online situation, the jobs' release times are assumed to be arbitrary. If all jobs' release times are zero, then the problem in Li and Huang [2] becomes the same as the Graham's classical on-line scheduling problem.

For the classical online problem on the identical parallel machine system, we know that no algorithm can be better than algorithm LS when the number of machines is less than 4. In many applications, partial information about jobs can be made available in advance. This motivates us to study semi-online scheduling problems when different types of partial information become available [3]. He and Zhang [4], and He and Dosa [5] consider the system when the lengths of all jobs are known in $[1, r]$ with $r \geq 1$. See Cheng *et al.* [6]; Li and Huang [7]; Seiden *et al.* [8]; Li *et al.* [9] for more recent results on semi-online scheduling.

Liu *et al.* [10] firstly considered ordinal semi-online problem in which it is assumed that the values of the processing times p_j are unknown, but that the order of the jobs by non-increasing processing time is known, *i.e.*, $p_1 \geq p_2 \geq \cdots \geq p_n$. The problem can be denoted as $P_m / \text{ordinal} / C_{\max}$. They proposed an algorithm with worst case performance ratio not greater than $1+\frac{m-1}{m+\left\lceil \frac{m}{2} \right\rceil}$. Later $Q_2 / \text{ordinal} / C_{\max}$ is considered by Tan and He [11] and $P_3 / \text{ordinal} / C_{\max}$ is considered by He and Tan [12].

In this paper, we assume that we are given m identical parallel machines $\{M_1, M_2, \cdots, M_m\}$ and an ordinal job list $L = \{J_1, J_2, \cdots, J_n\}$ with arbitrary release times, *i.e.*, each J_j has a release time r_j and a processing size of p_j satisfying $p_1 \geq p_2 \geq \cdots \geq p_n$. Our aim is to minimize the maximum completion time. We denote this problem as $P_m / \left(\text{ordinal}, r_j\right) / C_{\max}$.

The rest of the paper is organized as follows. In Section 2, some definitions and the algorithm P are given. In Section 3, we analyze the algorithm P and show its the upper bound of the worst case ratio. In Section 4, we give some concluding remarks.

2. Some Definitions and the Algorithm *P*

In this section we will give some definitions and an algorithm *P*.

Definition 1. Let algorithm A be a heuristic algorithm of scheduling job list L. $C_{\max}^{A}(L)$ and $C_{\max}^{OPT}(L)$ denote the makespan of algorithm A and an optimal off-line algorithm, respectively. We define

$$R(m,A)=\sup_{L}\frac{C_{\max}^{A}(L)}{C_{\max}^{OPT}(L)}$$

as the worst case performance ratio of algorithm A.

Definition 2. Suppose that J_j is the current job with information (r_j, p_j), *i.e.*, release time r_j and size of p_j, to be scheduled on machine M_i. We say that machine M_i has an idle time interval for job J_j, if there exists a time interval $[T_1, T_2]$ satisfying the following two conditions:

1) Machine M_i is idle in interval $[T_1, T_2]$ and a job with release time T_2 has been assigned to machine M_i to start at time T_2.

2) $T_2 - \max[T_1, r_j] \geq p_j$.

It is obvious that if machine M_i has an idle time interval for job J_j then we can assign J_j to machine M_i in the idle interval.

The algorithm *P*:

Let $L=\{J_1, J_2, \cdots, J_n\}$ be a job list of problem $P_m/(\text{ordinal}, r_i)/C_{\max}$. We assign jobs of L one by one on machine $M_i (i=1,2,\cdots,m)$ according to the following rules:

If $1 \leq i \leq \left\lfloor \frac{m}{2} \right\rfloor$ holds, we assign the following jobs on machine M_i:

$$\{J_i\} \bigcup \left\{ J_{2m+1-i+k\left(m+\left\lceil \frac{m}{2} \right\rceil\right)} \mid k \geq 0 \right\}.$$

If $1+\left\lfloor \frac{m}{2} \right\rfloor \leq i \leq m$ holds, we assign the following jobs on machine M_i:

$$\{J_i\} \bigcup \left\{ J_{2m+1-i+k\left(m+\left\lceil \frac{m}{2} \right\rceil\right)} \mid k \geq 0 \right\} \bigcup \left\{ J_{3m+1-i+k\left(m+\left\lceil \frac{m}{2} \right\rceil\right)} \mid k \geq 0 \right\}.$$

Assignment example for $m=5$:

$$\begin{array}{llllll} M_1: & J_1 & J_{10} & J_{18} & \cdots \\ M_2: & J_2 & J_9 & J_{17} & \cdots \\ M_3: & J_3 & J_8 & J_{13} & J_{16} & J_{21} & \cdots \\ M_4: & J_4 & J_7 & J_{12} & J_{15} & J_{20} & \cdots \\ M_5: & J_5 & J_6 & J_{11} & J_{14} & J_{19} & \cdots \end{array}$$

Note: The above assignment just means the order of the job's assigning, not mean the order of job's processing because of the release times of the jobs. For example on machine M_1, if $r_1=6, p_1=8, r_{10}=6, p_{10}=6$, then J_{10} begin to be processed at time zero and J_1 begin to be processed at 6 even though J_1 is assigned before J_{10} because J_1 appears before J_{10}.

The following symbols will be used in the analysis of this paper later on:

1) $P_1^n = \sum_{j=1}^{n} p_j$, $P_k^n = \sum_{j=k}^{n} p_j$.

2) U_i: the total sum of the idle time on machine M_i in the schedule *P*.

3) C_i: The completion time of machine M_i in the schedule *P*. It is equal to

the total sum of the processing time of the jobs assigned on machine M_i and the idle time of machine M_i in the schedule *P*. It is easy to see that $C_{\max}^{P}(L)=\max\{C_1,C_2,\cdots,C_m\}$ holds.

1) $[h]_i$: the index of the *h*-th job assigned on machine M_i in the schedule *P*.
2) $\lceil x\rceil$: It represents the smallest integer not less than *x*.
3) $\lfloor x\rfloor$: It represents the largest integer not bigger than *x*.

3. Main Results

The following simple inequality will be referred to later on:

$$C_{\max}^{OPT}(L)\geq\max\left\{\frac{\sum_{j=1}^{n}p_j}{m},r_j+p_j,j=1,2,\cdots,n,U_i+p_i,i=1,2,\cdots,m\right\}$$

Furthermore it is easy to get

$$C_{\max}^{OPT}(L)\geq U_i,i=1,2,\cdots,m$$

Lemma 1: For any job list $L=\{J_1,J_2,\cdots,J_n\}$ from problem $P_m/(\text{ordinal},r_i)/C_{\max}$, suppose that h jobs are assigned on machine M_i by algorithm *P*. If $h\geq 2$ and $[h]_i-[1]_i=[h]_i-i\geq\dfrac{h-1}{x}$ hold, then we have

$$C_i\leq xP_{i+1}^{n}+U_i+p_i.$$

Proof: It is easy to see that $\forall h\geq 2,\ x\geq\dfrac{h-1}{[h]_i-i}$ holds. Firstly we prove the following statement by induction for h:

$$xP_{i+1}^{[h]_i}\geq p_{[2]_i}+\cdots+p_{[h]_i}+\left[x\left([h]_i-[1]_i\right)-(h-1)\right]p_{[h]_i}\tag{1}$$

For $h=2$ we have

$$\begin{aligned}xP_{i+1}^{[2]_i}&=x\left(p_{[1]_i+1}+\cdots+p_{[2]_i}\right)\geq x\left[\left([2]_i-[1]_i\right)p_{[2]_i}\right]\\&=p_{[2]_i}+\left[x\left([2]_i-[1]_i\right)-(2-1)\right]p_{[2]_i}\end{aligned}$$

That means the statement is true for $h=2$. Now suppose (1) holds for $h=k$, *i.e.*,

$$xP_{i+1}^{[k]_i}\geq p_{[2]_i}+p_{[3]_i}+\cdots+p_{[k]_i}+\left[x\left([k]_i-[1]_i\right)-(k-1)\right]p_{[k]_i}.$$

Then for $h=k+1$ we have

$$\begin{aligned}xP_{i+1}^{[k+1]_i}&\geq p_{[2]_i}+p_{[3]_i}+\cdots+p_{[k]_i}+\left[x\left([k]_i-[1]_i\right)-(k-1)\right]p_{[k]_i}\\&\quad+x\left(p_{[k]_i+1}+p_{[k]_i+2}+\cdots+p_{[k+1]_i}\right)\\&\geq p_{[2]_i}+p_{[3]_i}+\cdots+p_{[k]_i}+\left[x\left([k]_i-[1]_i\right)-(k-1)\right]p_{[k]_i}\\&\quad+\left[x\left([k+1]_i-[k]_i\right)\right]p_{[k+1]_i}\\&\geq p_{[2]_i}+p_{[3]_i}+\cdots+p_{[k]_i}+\left[x\left([k]_i-[1]_i\right)-(k-1)\right]p_{[k+1]_i}\end{aligned}$$

$$+\left[x\left(\left[k+1\right]_i-\left[k\right]_i\right)\right]p_{\left[k+1\right]_i}$$
$$=p_{\left[2\right]_i}+p_{\left[3\right]_i}+\cdots+p_{\left[k+1\right]_i}+\left[x\left(\left[k+1\right]_i-\left[1\right]_i\right)-\left(k+1-1\right)\right]p_{\left[k+1\right]_i}$$

By (1) we have

$$xP_{i+1}^n+U_i\geq xP_{i+1}^{\left[h\right]_i}+U_i\geq p_{\left[2\right]_i}+p_{\left[3\right]_i}+\cdots+p_{\left[h\right]_i}$$
$$+\left[x\left(\left[h\right]_i-\left[1\right]_i\right)-\left(h-1\right)\right]p_{\left[k\right]_i}+U_i$$
$$\geq p_{\left[2\right]_i}+p_{\left[3\right]_i}+\cdots+p_{\left[h\right]_i}+U_i=C_i-p_i$$

where the last equality results from $\left[1\right]_i=i$ by the rules of algorithm *P*. The claim is proved.

Lemma 2: For any job list $L=\left\{J_1,J_2,\cdots,J_n\right\}$ from problem $P_m/\left(\text{ordinal},r_i\right)/C_{\max}$, we have

$$C_i\leq\begin{cases}\dfrac{2p_{i+1}^n}{3\left(m-i+1\right)}+U_i+p_i & 1\leq i\leq m;\ \ m \text{ is even}\\ \dfrac{2p_{i+1}^n}{3\left(m-i+1\right)+1}+U_i+p_i & 1\leq i\leq\dfrac{m-1}{2};\ \ m \text{ is odd}\\ \dfrac{4p_{i+1}^n}{3m+1}+U_i+p_i & \dfrac{m+1}{2}\leq i\leq m;\ m \text{ is odd}\end{cases}$$

Proof: Case 1: m is even and $1\leq i\leq m$.

Case 1.1: $1\leq i\leq\dfrac{m}{2}$ and $h=k+2$.

By the rules of *P* we have

$$\frac{h-1}{\left[h\right]_i-i}=\frac{k+2-1}{2m+1-2i+\dfrac{3km}{2}}=\frac{2k+2}{4m-4i+2+3km}$$
$$=\frac{2\left(k+1\right)}{3\left(k+1\right)\left(m-i+1\right)+3k\left(i-1\right)+m-i-1}$$
$$\leq\frac{2\left(k+1\right)}{3\left(k+1\right)\left(m-i+1\right)}=\frac{2}{3\left(m-i+1\right)}$$

Case 1.2: $\dfrac{m}{2}<i\leq m$ and $h=2k+1$.

$$\frac{h-1}{\left[h\right]_i-1}=\frac{2k}{2m+1-2i+\dfrac{3km}{2}}$$
$$=\frac{2k}{3k\left(m-i+1\right)+3k\left(i-1-\dfrac{m}{2}\right)+2\left(m-i\right)+1}$$
$$\leq\frac{2}{3\left(m-i+1\right)}$$

Case 1.3: $\dfrac{m}{2}<i\leq m$ and $h=2k+2$

$$\frac{h-1}{[h]_i-1}=\frac{2k+1}{3m+1-2i+\frac{3km}{2}}$$

$$=\frac{2\left(k+\frac{1}{2}\right)}{3\left(k+\frac{1}{2}\right)(m-i+1)+3k\left(i-1-\frac{m}{2}\right)+\frac{3m-i-1}{2}}$$

$$\le\frac{2}{3(m-i+1)}$$

Let $x=\frac{2}{3(m-i+1)}$, by Lemma 1 when m is even and $1\le i\le m$ we have

$$C_i\le\frac{2P_{i+1}^n}{3(m-i+1)}+U_i+p_i$$

Case 2: m is odd. $1\le i\le\frac{m-1}{2}$ and $h=k+2$ hold.

$$\frac{h-1}{[h]_i-1}=\frac{k+2-1}{2m+1-2i+\frac{k(3m+1)}{2}}$$

$$=\frac{2(k+1)}{3(k+1)(m-i+1)+(k+1)+3k(i-1)+m-i-2}$$

$$\le\frac{2(k+1)}{3(k+1)(m-i+1)+(k+1)+3k(i-1)+i+1-2}$$

$$\le\frac{2(k+1)}{3(k+1)(m-i+1)+(k+1)}$$

$$\le\frac{2}{3(m-i+1)+1}$$

Let $x=\frac{2}{3(m-i+1)+1}$, by Lemma 1, when m is odd and $1\le i\le\frac{m-1}{2}$ we have

$$C_i\le\frac{2P_{i+1}^n}{3(m-i+1)+1}+U_i+p_i$$

Case 3: m is odd and $\frac{m+1}{2}\le i\le m$

Case 3.1: $h=2k+1$

$$\frac{h-1}{[h]_i-i}=\frac{2k}{2m+1-2i+\frac{k(3m+1)}{2}}\le\frac{4}{3m+1}$$

Case 3.2: $h=2k+2$

$$\frac{h-1}{\lceil h\rceil_i-i}=\frac{2k+1}{3m+1-2i+\frac{k(3m+2)}{2}}$$

$$= \frac{4\left(k+\frac{1}{2}\right)}{\left(k+\frac{1}{2}\right)(3m+1)+\frac{3}{2}+\frac{9m}{2}-4i}$$

$$\leq \frac{4}{3m+1}$$

Let $x=\frac{4}{3m+1}$, when m is odd and $\frac{m+1}{2} \leq i \leq m$ we have

$$C_i \leq \frac{4P_{i+1}^n}{3m+1} + U_i + p_i$$

Hence the Lemma is proved.

Theorem 3 For any job list $L=\{J_1, J_2, \cdots, J_n\}$ from problem $P_m / (\text{ordinal}, r_i) / C_{\max}$ we have:

$$R(m,P) \leq 2 + \frac{m-1}{m+\left\lceil \frac{m}{2} \right\rceil}$$

Proof: Case 1: If m is even and $1 \leq i \leq m$ holds, by Lemma 2 we have

$$\begin{aligned}
C_i &\leq p_i + \frac{2P_{i+1}^n}{3(m-i+1)} + U_i \\
&= p_i - \frac{2P_1^i}{3(m-i+1)} + \frac{2P_1^n}{3(m-i+1)} + U_i \\
&= \frac{2}{3(m-i+1)}\left(\frac{3(m-i+1)}{2}p_i - P_1^i\right) + \frac{2P_1^n}{3(m-i+1)} + U_i \\
&\leq \frac{3m-5i+3}{3(m-i+1)}p_i + \frac{2m}{3(m-i+1)} \times \frac{P_1^n}{m} + U_i \\
&\leq \frac{8m-8i+6}{3(m-i+1)} \max\left\{p_i, \frac{P_1^n}{m}, U_i\right\} \\
&\leq \frac{8m-8i+6}{3(m-i+1)} C_{\max}^{OPT}(L) \\
&\leq \frac{8m-2}{3m} C_{\max}^{OPT}(L)
\end{aligned}$$

where the last inequality results from the fact that $\frac{8m-8x+6}{3(m-x+1)}$ is a decreasing function of x in interval $x \in [1, m]$

Case 2: m is odd and $1 \leq i \leq \frac{m-1}{2}$

By Lemma 2 we have

$$\begin{aligned}
C_i &\leq p_i + \frac{2P_{i+1}^n}{3(m-i+1)+1} + U_i \\
&= p_i - \frac{2P_1^i}{3(m-i+1)+1} + \frac{2P_1^n}{3(m-i+1)+1} + U_i
\end{aligned}$$

$$
\begin{aligned}
&= \frac{2}{3(m-i+1)+1}\left(\frac{3(m-i+1)+1}{2}p_i - P_1^i\right) + \frac{2P_1^n}{3(m-i+1)+1} + U_i \\
&\le \frac{3m-5i+4}{3(m-i+1)+1}p_i + \frac{2m}{3(m-i+1)+1} \times \frac{P_1^n}{m} + U_i \\
&\le \frac{8m-8i+8}{3(m-i+1)+1}\max\left\{p_i, \frac{P_1^n}{m}, U_i\right\} \\
&\le \frac{8m-8i+8}{3(m-i+1)+1}C_{\max}^{OPT}(L) \\
&\le \frac{8m}{3m+1}C_{\max}^{OPT}(L)
\end{aligned}
$$

where the last inequality results from the fact that $\frac{8m-8x+6}{3(m-x+1)+1}$ is a decreasing function of x in interval $x \in [1, m]$.

Case 3: m is odd and $\frac{m+1}{2} \le i \le m$

By Lemma 2 we have

$$
\begin{aligned}
C_i &\le p_i + \frac{4}{3m+1}P_{i+1}^n + U_i \\
&= p_i + \frac{4}{3m+1}\left(P_1^n - P_1^i\right) + U_i \\
&= \frac{4}{3m+1}\left[\frac{(3m+1)p_i}{4} - \left(p_1 + p_2 + \cdots + p_i\right)\right] + \frac{4}{3m+1}P_1^n + U_i \\
&\le \frac{4}{3m+1}\left[\frac{(3m+1)p_i}{4} - ip_i\right] + \frac{4}{3m+1}P_1^n + U_i \\
&= \frac{3m-4i+1}{3m+1}p_i + \frac{4}{3m+1}P_1^n + U_i \\
&\le \frac{10m-4i+2}{3m+1}\max\left\{p_i, \frac{P_1^n}{m}, U_i\right\} \\
&\le \frac{10m-4i+2}{3m+1}C_{\max}^{OPT}(L) \\
&\le \frac{8m}{3m+1}C_{\max}^{OPT}(L)
\end{aligned}
$$

By the above conclusions, when m is even we have $C_i \le \frac{8m-2}{3m}C_{\max}^{OPT}(L)$ hold for $i = 1, 2, \cdots, m$. Hence we get

$$
\frac{C_{\max}^{P}(L)}{C_{\max}^{OPT}(L)} \le \frac{\frac{8m-2}{3m}C_{\max}^{OPT}(L)}{C_{\max}^{OPT}(L)} = \frac{8m-2}{3m} = 2 + \frac{m-1}{m+\left\lceil\frac{m}{2}\right\rceil}
$$

When m is odd then $C_i \le \frac{8m}{3m+1}C_{\max}^{OPT}(L)$ hold for $i = 1, 2, \cdots, m$. Hence we get

$$\frac{C_{\max}^{P}(L)}{C_{\max}^{OPT}(L)} \le \frac{\frac{8m}{3m+1}C_{\max}^{OPT}(L)}{C_{\max}^{OPT}(L)} = \frac{8m}{3m+1} = 2+\frac{m-1}{m+\left\lceil \frac{m}{2} \right\rceil}$$

Thus we get

$$R(m,P) = \sup_{L} \frac{C_{\max}^{P}(L)}{C_{\max}^{OPT}(L)} \le 2+\frac{m-1}{m+\left\lceil \frac{m}{2} \right\rceil}$$

Hence the theorem is proved.

4. Concluding Remarks

In this paper, we consider the semi-online scheduling problem $P_m/(\text{ordinal}, r_i)/C_{\max}$ in which the job list has non-increasing processing times and arbitrary release times. An algorithm is investigated and it is shown that its worse case performance ratio is bounded by $2+(m-1)\Big/\left(m+\left\lceil \frac{m}{2} \right\rceil\right)$ for all values of m. For this problem, to investigate better algorithms or give lower bound and upper bound would be worth doing. There are many other scheduling problems where ordinal algorithms could be developed. The investigations to find good algorithms for these problems would be also of interest to the scheduling community.

Acknowledgements

This work was partly supported by the Chinese National Natural Science Foundation Grant (No.11471110) and the Foundation Grant of Education Department of Hunan (No. 16A126).

References

[1] Graham, R.L. (1969) Bounds on Multiprocessing Timing Anomalies. *SIAM Journal on Applied Mathematics*, **17**, 416-429. https://doi.org/10.1137/0117039

[2] Li, R.H. and Huang, H.C. (2004) On-Line Scheduling for Jobs with Arbitrary Release Times. *Computing*, **73**, 79-97. https://doi.org/10.1007/s00607-004-0067-1

[3] Kellerer, H., Kotov, V., Speranza, M.G. and Tuza, Z. (1997) Semi On-Line Algorithms for the Partition Problem. *Operations Research Letters*, **21**, 235-242. https://doi.org/10.1016/S0167-6377(98)00005-4

[4] He, Y. and Zhang, G. (1999) Semi On-Line Scheduling on Two Identical Machines. *Computing*, **62**, 179-187. https://doi.org/10.1007/s006070050020

[5] He, Y. and Dósa, G. (2005) Semi-Online Scheduling Jobs with Tightly-Grouped Processing Times on Three Identical Machines. *Discrete Applied Mathematics*, **150**, 140-159. https://doi.org/10.1016/j.dam.2004.12.005

[6] Cheng, T.C.E., Kellerer, H. and Kotov, V. (2012) Algorithms Better Than LPT for Semi-Online Scheduling with Decreasing Processing Times. *Operations Research Letters*, **40**, 349-352. https://doi.org/10.1016/j.orl.2012.05.009

[7] Li, R.H. and Huang, H.C. (2007) List Scheduling for Jobs with Arbitrary Release

Times and Similar Lengths. *Journal of Scheduling*, **10**, 365-373. https://doi.org/10.1007/s10951-007-0042-8

[8] Seiden, S., Sgall, J. and Woeginger, G.J. (20000 Semi-Online Scheduling with Decreasing Job Sizes. *Operations Research Letters*, **27**, 215-227.

[9] Li, R.H., Cheng, X.Y. and Zhou, Y.X. (2014) On-Line Scheduling for Jobs with Non-Decreasing Release Times and Similar Lengths on Parallel Machines. *Optimization-A Journal of Mathematical Programming and Operations Research*, **63**, 867-882.

[10] Liu, W.P., Sidney, J.B. and Vliet, A. (1996) Ordinal Algorithm for Parallel Machine Scheduling. *Operations Research Letters*, **18**, 223-232. https://doi.org/10.1016/0167-6377(95)00058-5

[11] Tan, Z.Y. and He, Y. (2001) Semi Online Scheduling with Ordinal Data on Two Uniform Machines. *Operations Research Letters*, **28**, 221-231. https://doi.org/10.1016/S0167-6377(01)00071-2

[12] He, Y. and Tan, Z.Y. (2002) Ordinal-Online Scheduling for Maximizing the Minimum Machine Completion Time. *Journal of Combinatorial Optimization*, **6**, 199-206. https://doi.org/10.1023/A:1013855712183