Abstract number: 025-0407

Single-machine Scheduling with Periodic Maintenance and both Preemptive and Non-preemptive jobs in Remanufacturing System[1]

Liu Biyu Chen Weida

（School of Economics and Management, Southeast University, 211189 Nanjing, China）

Abstract： This paper considers a single-machine scheduling with preventive periodic maintenance activities in remanufacturing system. Although the scheduling problem with maintenance has attracted researchers' attention, most of past studies consider only non-preemptive jobs. In fact, there exist both preemptive and non-preemptive jobs due to their different processing technic in remanufacturing workshop. Our objective is to find a schedule to minimize the makespan and subject to periodic maintenance and non-preemptive jobs. In this paper, an algorithm（we denote it as LPT-LS algorithm）is proposed. Its steps are as follows: 1) Firstly, scheduled the non-preemptive jobs in machine by the LPT（the Longest Processing Time）rule; 2)Then, scheduled the preemptive jobs into the machine's remaining time by the LS（the List Scheduling）rule. And more importantly we discuss the worst-case ratios of this algorithm in three different cases in terms of the value of the total processing time of the preemptive jobs（We denoted it as S_2）: 1) When S_2 is longer than the remaining time of the machine after assigned the non-preemptive jobs by the LPT rule, it is equal to 1; 2) When S_2 falls in between the remaining time of the machine by the

[1]**Biographies:** Liu Biyu（1981-）, Female, doctoral student, liubiyu2000@163.com; Chen Weida（1965-）, male, doctor, professor, cwd@seu.edu.cn.

LPT rule and the OPT rule, it is less than 2; 3) When S_2 is less than the remaining time of the machine by the OPT rule, it is less than 2.

Key words: Single-machine scheduling; Preventive periodic maintenance; Preemptive jobs and Non-preemptive jobs; LPT-LS algorithm

1. Introduction

Most literatures on scheduling theory assume that there are only non-preemptive jobs. In fact, this assumption may not be valid in a real production situation such as in the reprocessing workshop of remanufacturing industry. There also exist preemptive jobs. In remanufacturing industry, in order to reassemble finished products, new components are required since the recovery rate of return components can never reach 100%. In general, a disassembly order is always released first and then the disassembly result determines whether a purchasing order is needed[1]. And it is necessary to reprocess the disassembling cores as early as possible in order to determine whether purchase new components or not. Our objective is to find a schedule to minimize the makespan. For simpleness, we only discuss off-line cores and assume that the reprocessing time have been determined by workers' experience. It is common to observe in practice that before machines are maintained it has some spare time but there is not a non-preemptive job whose processing time exactly is equal to the machine's spare time and can't be scheduled into the machine in the same period. In order to making the best of machine capacity, we can schedule the preemptive jobs after processing the non-preemptive jobs（In remanufacturing

industry, we assume there is zero setup time）. Therefore, scheduling non-preemptive and preemptive jobs has gradually become a common practice in many remanufacturing enterprise. With proper scheduling rule, the workshop can improve production efficiency and resulting in increased productivity and high customers satisfaction degree[2].

Some literatures have researched the single-machine scheduling problem with single maintenance and non-preemptive jobs. Liao and Chen [3] have considered a scheduling problem for the objective of minimizing the maximum tardiness. They propose a branch-and-bound algorithm to derive the optimal schedule and a heuristic algorithm for large-sized problems. They also provide computational results to demonstrate the efficiency of their heuristic. Lee[4] shows that the longest processing time (LPT) rule has a tight worst-case ratio of 4/3 for the objective of minimizing the makespan, and he also presents heuristics for other objectives, such as minimizing the maximum lateness, the number of tardy jobs, and the total weighted completion time, etc. Lee and Liman [5] prove that the worst-case ratio of the shortest processing time (SPT) rule is 9/7 for the objective of minimizing the total completion time. Graves and Lee[6] extend the problem to consider semi-preemptive jobs. Ma et al. [7] discuss a single-machine scheduling problem with an unavailable period in semipreemptive case to minimize makespan and present a LPT-based heuristic. Chen[8] considers a single-machine scheduling problem with periodic maintenance to find a schedule that minimizes the number of tardy jobs subject to periodic maintenance and nonpreemptive jobs. An effective heuristic and a branch-and-bound algorithm are

proposed to find the optimal schedule. Wu and Lee[9] discuss a special problem where jobs' processing time is a decreasing function of its position in the schedule; when jobs are preemptive, it is shown that the SPT rule provides the optimal schedule, and for the nonpreemptive case, a mixed integer programming technique is proposed. Yang and Yang[10] studied a single-machine scheduling with the job-dependent aging effects under multiple maintenance activities and variable maintenance duration considerations simultaneously to minimize the makespan. Ji et al.[11] consider a single machine scheduling problem with periodic maintenance for the objective of minimizing the makespan. They show that the worst-case ratio of the classical Longest Processing Time first algorithm is 2.

In this paper we considered the scheduling problem with periodic maintenance to minimize the makespan for both non-preemptive and preemptive jobs. Firstly, we proposed an algorithm by which the jobs are scheduled. Then we discussed the worst-case ratios of this algorithm in three different cases in terms of the value of the total processing time of the preemptive jobs.

2. Problem description and notation

Formally, the considered problem could be described as follows:

We assume there were n independent jobs $J = \{J_{11}, J_{12}, \cdots, J_{1n_1}, \cdots, J_{21}, J_{22}, \cdots, J_{2n_2}\}$ (Assume all jobs' arrival time is zero.) for reprocessing including non-preemptive and preemptive ones, which were processed on a single machine. Here non-preemptive means that if a job can't be finished before maintenance activity, it has to restart; preemptive means that if a job can't be finished before maintenance activity, it can be

continued to processed based on the previous process. The number of the former is n_1 and the latter is n_2（Obviously, n_1 plus n_2 is equal to n）. They are defined group 1 and group 2, respectively.

The processing time of job J_{ij} is p_{ij} ($i=1,2$; when $i=1$, then $j=1,2,...n_1$, when $i=2$, then $j=1,2,...n_2$). We assumed that $T \geq p_{1j}$ for every $j=1,2,...n_1$, for otherwise there is trivially no feasible schedule. Let C_{ij} be the completion time of job J_{ij}, then the objective is to minimize the makespan, which is defined as $C_{\max} = \max_{i=1,2;\, j=1,2,...n_i} C_j^{(i)}$. Using the three-field notation of Graham et al. [12], we denoted this scheduling problem as $1/(np-p)-pm/C_{\max}$. It can easily be shown that this problem is strongly NP-hard [4], but no approximation algorithm has been provided and analyzed in the literature.

We thought of each interval between two consecutive maintenance activities as a batch with a capacity T. Thus, a schedule π can be denoted as $\pi=(B_1,M_1,B_2,M_2,...,M_{L-1},B_L)$, where M_i is the *i*th maintenance activity, L is the number of batches, and B_i is the *i*th batch of jobs. An illustration of the considered problem in the form of a Gantt chart is given in Fig.1.

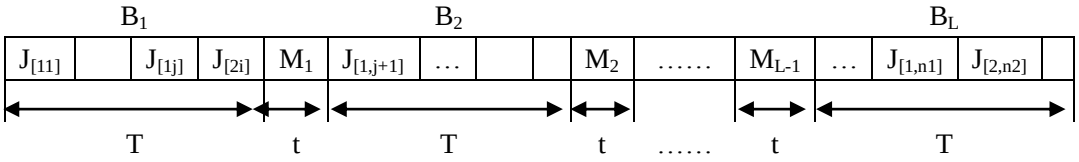

Fig.1. An illustration of the problem under consideration, where $J_{[1j]}$ denotes the non-preemptive job placed in the jth position of the given schedule.

We use the worst-case ratio to measure the quality of an approximation algorithm.

Specifically, for the makespan problem, let C_P denote the makespan produced by an approximation algorithm P, and C_{OPT} denote the makespan produced by an optimal off-line algorithm. Then the worst-case ratio of algorithm P is defined as the smallest number c. （i.e., $\frac{C_P}{C_{OPT}} \le c$ ）

Some other notations are described as follows:

T —The length of time interval between two consecutive maintenance periods

t —The amount of time to perform each maintenance activity

C_{OPT} —The makespan（correspond to the load of bin packing problem） produced by an optimal off-line algorithm

n_{OPT} —The number of machine working periods（correspond to the bin numbers of bin packing problem） by an optimal off-line algorithm

L_{OPT} —The machine working time's length in the last period by an optimal off-line algorithm（ $0 \le L_{OPT} \le T$ ）

C_{LPT} —The makespan produced by LPT algorithm (i.e., the completion time of job J_{1j})

n_{LPT} —The number of machine working periods by LPT algorithm

L_{LPT} —The machine working time's length in the last period by LPT algorithm （ $0 \le L_{LPT} \le T$ ）

S_1 —The total processing time of Group 1 jobs

S_2 —The total processing time of Group 2 jobs

3. The algorithm and its worst-case ratio

In this section we analyze the algorithm (we define it as "LPT-LS" algorithm) and

its worst-case ratio. Before analyzing the LPT-LS algorithm, we first define two kinds of classical algorithms, which are LPT and LS algorithm, respectively.

The LPT rule is a classical heuristic for solving scheduling problems. It can be formally described as follows.

Algorithm LPT. Re-order all the jobs such that $p_{11} \geq p_{12} \geq \ldots \geq p_{1n_1}$; then process the jobs consecutively as early as possible.

The LS algorithm is a classical polynomial time approximation algorithm for solving scheduling problems. It can be described as follows.

Algorithm LS (Graham (1966)). When a job is available (ties are broken arbitrarily), assign it to the processing interval of the machine where it can be finished as early as possible.

Here we define "LPT-LS" algorithm as follows:

Algorithm LPT-LS. The first step is to schedule the non-preemptive jobs by the LPT rule; the second step is to schedule the preemptive jobs by the LS rule for this considered scheduling problem.

We let C_{LPT-LS} be the makespan of all being scheduled jobs（i.e., including non-preemptive and preemptive jobs）.

The makespan of optimal schedule is

$$C_{OPT} = (n_{OPT} - 1)(T + t) + L_{OPT} \qquad (1)$$

While the makespan of the LPT schedule is

$$C_{LPT} = (n_{LPT} - 1)(T + t) + L_{LPT}. \qquad (2)$$

(3) subtracts (2)， we obtain

$$C_{LPT} - C_{OPT} = (n_{LPT} - n_{OPT})(T + t) + L_{LPT} - L_{OPT} \quad (3)$$

Lemma 1. (See Ji et al. (2007))The worst-case ratio of the LPT algorithm for periodic maintenance is 2. (i.e., $\frac{C_{LPT}}{C_{OPT}} = 2.$)

This first step of the scheduling is similar to the problem that Ji et al. (2007) have researched. They prove that the worst-case ratio of the classical LPT algorithm is 2. And they show that there is no polynomial time approximation algorithm with a worst-case ratio less than 2 unless P = NP, which implies that the LPT algorithm is the best possible. So, for non-preemptive jobs the LPT algorithm is also the best possible.

For the problem $1/(np-p)-pm/C_{\max}$, the worst-case ratio of the LPT –LS algorithm exists three cases in terms of the value of S_2, the conclusions are proved in the following theorems.

Firstly, we define a function: $f(x) = k$, when $k < x \le k+1$, k is a positive integer. Obviously, $f(\frac{S_i}{T})$ is the frequency of machine maintenance, where S_i is the total processing time of all being scheduled jobs. And it is easy to obtain,

$$n_i - 1 \le 2 f(\frac{S_i}{T}). \quad (4)$$

In the following we discuss the worst-case ratio when S2 takes different values.

Theorem 3.1. When $S_2 \ge C_{LPT} - S_1 - (n_{LPT} - 1)t$, $\frac{C_{LPT-LS}}{C_{OPT}} = 1.$

Proof. It means the total processing time of preemptive jobs is longer than the machine spare time after scheduling non-preemptive jobs. Obviously, in this situation, the preemptive jobs can be assigned into the machine's spare time so that the machine is always processing jobs unless maintenance. We can obtain that,

$$C_{LPT-LS} = (S_1 + S_2) + f(\frac{S_1 + S_2}{T})t. \quad (5)$$

It is the optimal algorithm. That is $\frac{C_{LPT-LS}}{C_{OPT}} = 1.$

Instance1. We assume there are 7 jobs in the reprocessing workshop. $T = 15, t = 0.5, n = 7, n_1 = 3, n_2 = 4, p_{11} = 6, p_{12} = 8, p_{13} = 5, p_{21} = 2$

So, we can obtain

$S_1 = 19, C_{LPT} = 20.5, n_{LPT} = 2, C_{LPT} - S_1 - (n_{LPT} - 1)t = 1, S_2 = 11, S_2 > C_{LPT} - S_1 - (n_{LPT} - 1)t$

$\frac{C_{LPT-LS}}{C_{OPT}} = \frac{30.5}{30.5} = 1.$

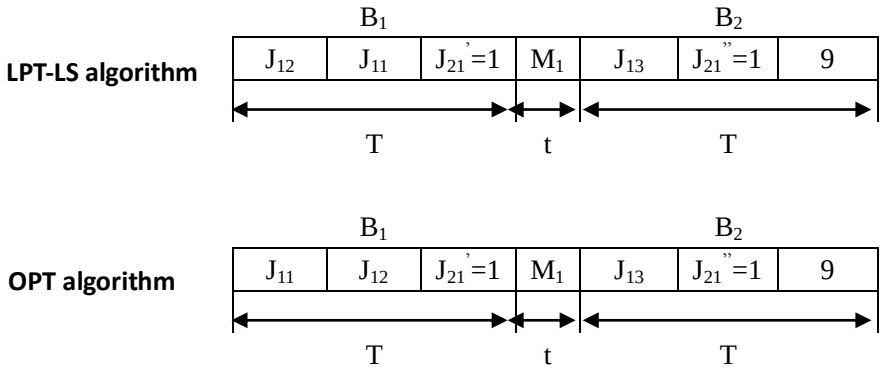

Fig.2. An illustration of the considered problem in case 1.

Lemma 2. While $n_{LPT} \geq 2$ periods are used in the LPT schedule, $S_1 > \frac{n_{LPT}T}{2}$.

Proof. Note that, the total processing time of the jobs in any two pairwise used bins is strictly larger than *T* by LPT rule. And in all periods at most one period in which the total processing time of the jobs are less than *T*/2. Regardless of which period total processing time is less than T/2, it plus any other one period processing time, the result must larger than T. We discuss as follows:

Case 1: If n_{LPT} is even, then,

$$S_1 = \sum_{k=1}^{n_{LPT}} B_k = \sum_{j=1}^{n_1} p_{1j} > \frac{bT}{2} \quad (6)$$

Case 2: If n_{LPT} is odd, we can choose ($n_{LPT}-1$) bins including the bin in which the jobs' processing time is less than T/2 and separate them into $\frac{n_{OPT}-1}{2}$ groups with two pairwise. The total processing time of jobs in the remained bin （note it B_x）is larger than T/2.

Then,

$$S_1 = \sum_{k=1}^{n_{LPT}} B_k = \sum_{j=1}^{n_1} p_{1j} = \frac{(b-1)\ T}{2} + t\ (B_x) > \frac{bT}{2} \quad (7)$$

Hence, we obtain,

$$S_1 > \frac{bT}{2} \quad (8)$$

Theorem 3.2. When $C_{OPT} - S_1 - (n_{OPT}-1)t \le S_2 < C_{LPT} - S_1 - (n_{LPT}-1)t$,

$\frac{C_{LPT-LS}}{C_{OPT}} < 2.$

Proof. It means the total processing time of preemptive jobs is shorter than the machine spare time after scheduling non-preemptive jobs by LPT rule and longer than the optimal algorithm. It is obviously that,

$$C_{LPT} \le n_{LPT}T + (n_{LPT}-1)t \quad (9)$$

Note that，

$$n_{LPT} - 1 \le 2f(\frac{S_1}{T}) \quad (10)$$

Combining（8），（10）and Lemma 3, we can obtain,

$$\frac{C_{LPT}}{S_1 + f(\frac{S_1}{T})t} \le \frac{n_{LPT}T + (n_{LPT}-1)t}{S_1 + f(\frac{S_1}{T})t} < \frac{n_{LPT}T + (n_{LPT}-1)t}{\frac{n_{LPT}T}{2} + f(\frac{S_1}{T})t} \le \frac{n_{LPT}T + (n_{LPT}-1)t}{\frac{n_{LPT}T}{2} + \frac{(n_{LPT}-1)}{2}t} = 2 \quad (11)$$

Instance 2. We assume there are 8 jobs in the reprocessing workshop. $T=20, t=0.5, n=8, n_1=7, n_2=1, p_{11}=14, p_{12}=12, p_{13}=10, p_{14}=p_{15}=6, p_{16}=4, p_{17}=3, p_{21}=2$

So, we can obtain

$$S_1=55, C_{LPT}=58, n_{LPT}=3, C_{LPT}-S_1-(n_{LPT}-1)t=2, C_{OPT}-S_1-(n_{OPT}-1)t=1, S_2=1.5$$

$$C_{OPT}-S_1-(n_{LPT}-1)t<S_2<C_{LPT}-S_1-(n_{LPT}-1)t$$

$$\frac{C_{LPT-LS}}{C_{OPT}}=\frac{C_{LPT}}{C_{OPT}}=\frac{58}{57.5}<2.$$

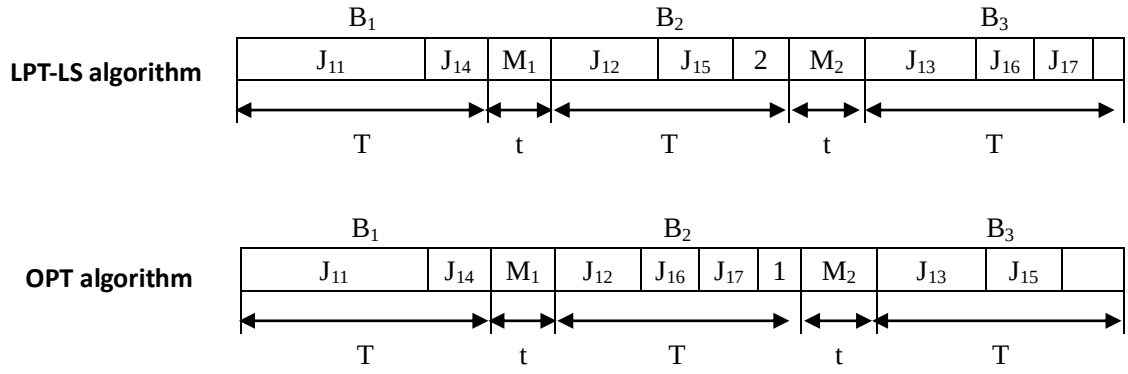

Fig.3. An illustration of the considered problem in case 2.

Theorem 3.3. When $S_2<C_{OPT}-S_1-(n_{OPT}-1)t$, $\frac{C_{LPT-LS}}{C_{OPT}}<2.$

Proof. It means the total processing time of preemptive jobs is shorter than the machine spare time after scheduling non-preemptive jobs by the optimal algorithm. The preemptive jobs can be scheduled into the machine's spare time. i.e., excluding the last bin, the other bins can be loaded full without any spare volume. So it's similar to the scheduling problem which Ji et al. (2007) has researched.

Therefore, the worst-case ratio of the LPT-LS algorithm for the problem $1/(nr-r)-pm/C_{\max}$ in terms of the value of S_2 are as follows:

1）When $S_2 \geq C_{LPT}-S_1-(n_{LPT}-1)t$, $\frac{C_{LPT-LS}}{C_{OPT}}=1$

2） When $C_{OPT} - S_1 - (n_{OPT} - 1)t \le S_2 < C_{LPT} - S_1 - (n_{LPT} - 1)t$, $\frac{C_{LPT-LS}}{C_{OPT}} < 2$

3） When $S_2 < C_{OPT} - S_1 - (n_{OPT} - 1)t$, $\frac{C_{LPT-LS}}{C_{OPT}} < 2.$

4. Conclusions

For the problem $1/(np-p)-pm/C_{\max}$, this paper proposed a LPT-LS algorithm and discussed the worst-case ratios of this algorithm in three different cases in terms of the value of the total processing time of the preemptive jobs which have been listed as Section 3. In this research, we assume the processing time is fixed and off-line. In future research, it is worth considering the processing time is random and on-line. It is also worth considering the problem with respect to other objectives and in parallel-machine systems.

References

[1] Tang O, Grubbstr O M R W, Zanoni S., *Planned lead time determination in a make-to-order remanufacturing system* [J]. International Journal of Production Economics 2007, 108(1-2): 426-435.

[2] R.H.P.M. Arts, Gerald M. Knapp, Lawrence Mann Jr, *Some aspects of measuring maintenance in the process industry* [J]. Journal of Quality in Maintenance Engineering 1998, 4(1):6–11.

[3] Liao C.J., Chen W.J., *Single-machine scheduling with periodic maintenance and non-resumable jobs* [J]. Computers and Operations Research 2003, 30(9):1335–1347.

[4] Lee CY, *Machine scheduling with an availability constraint* [J]. Journal of Global

Optimization 1996, 9(3-4):395–416.

[5] Lee CY, Liman SD. *Single machine flow-time scheduling with scheduled maintenance* [J]. Acta Informatica, 1992, 29(4):375–382.

[6] Graves GH, Lee CY. *Scheduling maintenance and semiresumable jobs on a single machine* [J]. Naval Research Logistics, 1999, 46(7):845–863.

[7] Ma Y, Yang S.L., Chu C.B., *Minimizing makespan in semiresumable case of single-machine scheduling with an availability constraint* [J]. Systems Engineering — Theory & Practice, 2009, 29(4): 128–134.

[8] Chen W.J., *Minimizing number of tardy jobs on a single machine subject to periodic maintenance* [J]. Omega, 2009,37:591–599.

[9] Wu C.C., Lee W.C.，*A note on single-machine scheduling with learning effect and an availability constraint* [J]. The International Journal of Advanced Manufacturing Technology, 2007, 33(5-6): 540–544.

[10] Yang S.J., Yang D.L., Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities, Omega，2010，38 ：528-533.

[11] Ji M, He Y, Cheng T.C.E., *Single-machine scheduling with periodic maintenance to minimize makespan* [J]. Computers & Operations Research, 2007, 34(6): 1764-1770.

[12] Graham RL, Lawler EL, Lenstra JK, Rinnooy Kan AHG. *Optimization and approximation in deterministic sequencing and scheduling: a survey* [M]. Annals of Discrete Mathematics, 1979, 5:287–326.