Online Scheduling on Uniformly Related Machines in the List Model

Xiao Xin[*,1], Guohua Mu[1], Min Mou[1]

[1] College of Foreign Studies, Shandong Institute of Business and Technology, Yantai, 264005, China

***Abstract*— The problem of scheduling jobs online on uniformly related machines in the list model is considered. Each job becomes known at its release time, at which point its processing requirement is completely specified. Each of the uniformly related machines is characterized by its speed. If a job with processing time *p* is assigned to a machine of speed *s* it requires *p*/*s* units of time. The list model orders the machines according to their capability, and each job specifies the least capable machine that can process it. The objective is to find a non-preemptive schedule that minimizes the maximum flow time, where the flow time of a job is defined to be its completion time minus its release time. A constant competitive ratio algorithm for the problem is presented.**

Keywords- Online scheduling; uniformly related machines; list model; maximum flow time; constant competitive ratio

I. INTRODUCTION

Parallel machines scheduling is a basic problem in computer science, with an almost limitless number of variants [1, 2, 3]. There are n jobs to be processed on m parallel machines. Each job j is associated with a release time r_j before which it cannot be processed, and a processing time p_j. Each machine can process at most one job at a time, and each job must be processed on one of the machines.

There are three basic types of parallel machines models. In the *unrelated* machines model, job j takes p_j / s_{ij} time units when processed by machine i, where s_{ij} is the speed of machines i on job j. If the machines are *uniformly related*, then machine i runs at a given speed s_i on all jobs. For *identical machines*, all of the machines run at speed 1 on all jobs.

Let C_j denote the *completion time* of job j under some schedule. The two most common objective functions are *makespan* (that is, $\max_j C_j$) and *total completion time* (that is, $\sum_j C_j$). However, neither is a suitable measure when jobs arrive in continuous streams [4]. Let $F_j = C_j - r_j$ denote the flow time of job j under the schedule. A suitable metric to optimize for continuous job arrivals is the *total flow time* (that is, $\sum_j F_j$). Unfortunately, inducing starvation is an inherent property of this metric. That is, some jobs may be delayed to an unbounded extent. In order to avoid starvation of any job, Bender et al. [4] initiated the study of the *maximum flow time*, $F_{\max} = \max_j F_j$.

There are different settings to consider: offline (a complete specification of the instance is available ahead of time), online (jobs become known at their release times, and the scheduler must decide only based on already arrived jobs), non-preemptive (jobs must be processed without interruption), and preemptive (jobs may be suspended and resumed later without penalty). Online algorithms are usually analyzed using the notion of *competitive ratio* which compares the solution obtained by the algorithm to that obtained by an online adversary for the worst possible input sequence.

There has been considerable work on scheduling with the objective of minimizing the maximum flow time. For non-preemptive scheduling on identical machines, Bender et al. [4] showed that the First In First Out (FIFO) strategy is an online algorithm having a competitive ratio of $3-2/m$ and this bound is tight. When preemption is allowed, M. Mastrolilli [5] observed that the offline scheduling on related and unrelated machines can be solved optimally in polynomial time by modifying the algorithms presented in [6, 7]. For online preemptive scheduling on identical machines, Ambühl and Mastrolilli [8] gave a simple 2-competitive algorithm. When m is fixed, Mastrolilli [5] presented the first fully polynomial time approximation scheme (FPTAS) for the offline non-preemptive scheduling on unrelated machines.

For online preemptive scheduling to minimize maximum weighted flow time, Chekuri and Moseley [9] and Anand et al. [10] exploited the popular resource augmentation model introduced in [11] where the algorithm is given extra speed over the adversary. An algorithm is said to be *s-speed c-competitive* if each machine can run up to s times faster than the machines in the adversary's schedule and the algorithm achieves a competitive ratio of c. Chekuri and Moseley [9] obtained a $(1+\varepsilon)$ -speed $O(1/\varepsilon)$ -competitive algorithm for identical machines, where $\varepsilon > 0$ can be made arbitrarily small. Anand et al. [10] obtained a $(1+\varepsilon)$ -speed $O(1/\varepsilon^3)$ -competitive algorithm for related machines. They also gave a $(1+\varepsilon)$ -speed

$O(1/\varepsilon)$-competitive algorithm for unrelated machines to minimize maximum flow time.

Recently, N. Bansal and B. Cloostermans [12] obtained the first constant competitive ratio algorithm for the problem of online non-preemptive scheduling to minimize maximum flow time on uniformly related machines.

In this paper, we consider the problem of scheduling jobs online on uniformly related machines in the list model. The problem is in the restricted assignment setting where each job can be assigned only to a subset of machines. More precisely, the list model [13, 14] orders the machines according to their capability, and each job specifies the least capable machine that can process it. That is, let the related machines be indexed by non-increasing order of speeds $s_1 \geq s_2 \geq \cdots \geq s_m$. Denote by $[1:i]$ the machines indexed $1,2,\ldots,i$. If job j specifies machine i, then j can be assigned to any of the machines $[1:i]$, and the machines $[1:i]$ are eligible machines for job j. For the objective of minimizing maximum flow time, we present a constant competitive ratio algorithm by modifying the algorithm presented in [12].

The remainder of this paper is organized as follows. In Section 2 we give the algorithm. In Section 3 we analyze the algorithm's competitive ratio. We conclude this paper in Section 4.

II. ALGORITHM TWOSTAGESCHEDULE

Denote by *Opt* the objective value of an optimal offline schedule. A technique of [12] allows us to estimate the value of *Opt*. We will describe this technique at the end of the next section to make sure that $F_{opt} \in [Opt,\ 1.5Opt]$.

Divide the time horizontal line into intervals I_k of equal lengths as $I_k = [3(k-1)F_{opt},\ 3kF_{opt})$. We call the time point $3kF_{opt}$ the k-th *epoch*. Let $P_i(t)$ be the sum of the remaining processing times of jobs processed on machine i at time t. Then machine i has load $l_i(t) = P_i(t)/s_i$ at time t. Recall that $[1:i]$ denotes the machines $1,2,\ldots,i$. Denote by JI_k the set of the jobs arriving during I_k at epoch k, $k = 1,2,\ldots$.

We are now ready to describe the algorithm TwoStageSchedule. It has two stages for scheduling the jobs in JI_k, $k = 1,2,\ldots$.

Algorithm TwoStageSchedule for the epoch *k*:

Step 1. Partition the jobs in JI_k into classes $J_1,\ldots,J_m$, where job j is in class J_i if machine i is the slowest eligible machine for job j which satisfies $p_j \leq s_i \cdot F_{opt}$.

Step 2. For $i = 1,2,\ldots,m$, assign the jobs j in J_i as follows:

(**Phase I**:) **If** some machine in $[1:i]$ is loaded below $3F_{opt}$ schedule j on the slowest such machine.

(**Phase II**:) **Else** schedule j on the slowest machine in $[1:i]$ such that its load stays below $6F_{opt}$.

Step 3. If no such machine exists return FAIL.

The crucial analysis in the next section is to prove the following theorem.

Theorem 1. *If $F_{opt} \geq Opt$, then TwoStageSchedule never fails.*

Based on it, we can prove TwoStageSchedule's competitiveness.

Theorem 2. *TwoStageSchedule is a 13.5-competitive algorithm for the problem of online scheduling uniformly related machines to minimize maximum flow time.*

Proof. In the schedule generate by TwoStageSchedule, any job waited at most $3F_{opt}$ to be assigned to a machine, and waited at most another $6F_{opt}$ to be finished on that machine. Hence the flow time of any job is no more than $3F_{opt} + 6F_{opt} = 9F_{opt}$. Since $F_{opt} \in [Opt,\ 1.5Opt]$, it follows that the competitive ratio of TwoStageSchedule is 13.5. □

III. ANALYZING THE COMPETITIVENESS

We are going to prove Theorem 1 in this section. Throughout this section we will compare the schedule generated by TwoStageSchedule with a restricted schedule. This restricted schedule is transformed from a optimal offline schedule by delaying each job no more than $3F_{opt}$ time such that all the jobs in JI_k are scheduled at epoch k, just as TwoStageSchedule. Note that if $F_{opt} \geq Opt$, then this restricted schedule has objective value no more than $3F_{opt} + Opt \leq 4F_{opt}$.

Fix an epoch k. Let $A_i(k)$ be the sum of the remaining processing times of jobs processed on machines $[1:i]$ at time $3kF_{opt}$ in TwoStageSchedule just before the jobs in JI_k are scheduled. That is, $A_i(k) = \sum_{i'=1}^{i} P_{i'}(3kF_{opt})$. Let $B_i(k)$ be the sum of the remaining processing times of jobs processed on machines $[1:i]$ at time $3kF_{opt}$ in

TwoStageSchedule just after all the jobs in JI_k are scheduled. Let $A_i^{ropt}(k)$ be the sum of the remaining processing times of jobs processed on machines $[1:i]$ at time $3kF_{opt}$ in the restricted schedule just before the jobs in JI_k are scheduled. Let L_i^{ropt} be the load on machine i at time $3kF_{opt}$ in the restricted schedule just after all the jobs in JI_k are scheduled. Let $B_i^{ropt}(k) = \sum_{i'=1}^{i} \max\{L_{i'}^{ropt} \cdot s_{i'}, 3F_{opt} \cdot s_{i'}\}$. We will prove that the following two invariants hold at every epoch k for all $1 \le i \le m$.

$$A_i(k) \le A_i^{ropt}(k) + F_{opt} \sum_{i'=1}^{i} s_{i'} \quad (1)$$

$$B_i(k) \le B_i^{ropt}(k) + F_{opt} \sum_{i'=1}^{i} s_{i'} \quad (2)$$

In order to prove (1) and (2), we need the following two lemmas.

Lemma 1. *If at epoch k, (1) holds for $1 \le i \le m$, then (2) holds for $1 \le i \le m$.*

Lemma 2. *If at epoch k, (2) holds for $1 \le i \le m$, then (1) holds for $1 \le i \le m$ at epoch $k+1$.*

Note that $A_i(0) = A_i^{ropt}(0)$ for all $1 \le i \le m$. It follows that (1) holds for $k = 0$. After finishing the proof of Lemmas 1 and 2, we can prove (1) and (2) for all k easily by alternately using the two lemmas.

A. Proof of Lemma 1

Lemma 3. *Let $i_1 > i_2$. If TwoStageSchedule schedules a job j in J_{i_1} on machine i_2 in Phase I, then it also schedules all jobs in J_i for $i_2 \le i < i_1$ in Phase I.*

Proof. Since TwoStageSchedule schedules j on machine i_2 in Phase I, before j is scheduled i_2 must have load less than $3F_{opt}$. TwoStageSchedule schedules the jobs in J_i for $i_2 \le i < i_1$ before J_{i_1}, therefore i_2 has load less than $3F_{opt}$ after J_i is considered. As a result, TwoStageSchedule schedules all jobs in J_i for $i_2 \le i < i_1$ in Phase I. □

Definition 1. Let $i_1 > i_2$. If TwoStageSchedule assigns no jobs in classes $[i_1 : m]$ on machines $[1 : i_2]$ at epoch k, then machines i_1 and i_2 are *separated* at epoch k.

Lemma 4. *If machines $i+1$ and i are separated at epoch k, and (1) holds for machine i, then (2) also holds for machine i .*

Proof. By the definition of the separated machines, we know that TwoStageSchedule assigns no jobs in classes $[i+1:m]$ on machines $[1:i]$ at epoch k. Hence we get $B_i(k) = A_i(k) + \sum_{i'=1}^{i} |J_{i'}|$, where $|J_{i'}|$ denotes the total processing time of jobs in $J_{i'}$.

On the other hand, any optimal schedule cannot assign jobs in $J_{i'}$ *for* $1 \le i' \le i$ on machines $[i+1:m]$. It is also true for the restricted schedule. Thus we have $B_i^{ropt}(k) \ge A_i^{ropt}(k) + \sum_{i'=1}^{i} |J_{i'}|$.

Since $A_i(k) \le A_i^{ropt}(k) + F_{opt} \sum_{i'=1}^{i} s_{i'}$, we get $B_i(k) \le A_i(k) + B_i^{ropt}(k) - A_i^{ropt}(k) \le B_i^{ropt}(k) + F_{opt} \sum_{i'=1}^{i} s_{i'}$. That is, (2) holds for machine i. □

Lemma 1 will be proved by the mathematical induction over i.

Since $A_0(k) = A_0^{ropt}(k) = 0$, $B_0(k) = B_0^{ropt}(k) = 0$, (1) and (2) holds for $i = 0$. It means that the base case is trivially true.

There are two different cases that TwoStageSchedule assigns the jobs in classes $[i+1:m]$ on machines $[1:i]$ at epoch k.

Case 1. At epoch k, no jobs in classes $[i+1:m]$ are assigned onto machines $[1:i]$ in Phase II.

If at epoch k there are no jobs at all in classes $[i+1:m]$ are assigned onto machines $[1:i]$, then machines $i+1$ and i are separated. Hence (2) holds by Lemma 4. Otherwise, we know that the jobs in classes $[i+1:m]$ are assigned onto machines $[1:i]$ only in Phase I.

Find the largest index $i_{\max} \le i$ such that machine s $i+1$ and $i_{\max} - 1 \ge 0$ are separated. Since (2) holds for $i_{\max} - 1$, we have $B_{i_{\max}-1}(k) \le B_{i_{\max}-1}^{ropt}(k) + F_{opt} \sum_{i'=1}^{i_{\max}-1} s_{i'}$. Since jobs in classes $[i+1:m]$ are assigned onto machines $[i_{\max} : i]$ only in Phase I, the jobs from classes $[i_{\max} : i]$ are also assigned in Phase I (Lemma 3). This means that the

load of any machine in $[i_{\max} : i]$ is less than $4F_{opt}$. Hence we get

$$\begin{aligned} B_i(k) &\le 4F_{opt}\sum_{i'=i_{\max}}^{i} s_{i'} + B_{i_{\max}-1}(k) \\ &\le 4F_{opt}\sum_{i'=i_{\max}}^{i} s_{i'} + B^{ropt}_{i_{\max}-1}(k) + F_{opt}\sum_{i'=1}^{i_{\max}-1} s_{i'} \\ &\le 3F_{opt}\sum_{i'=i_{\max}}^{i} s_{i'} + B^{ropt}_{i_{\max}-1}(k) + F_{opt}\sum_{i'=1}^{i} s_{i'} \\ &\le B^{ropt}_{i}(k) + F_{opt}\sum_{i'=1}^{i} s_{i'}. \end{aligned}$$

Case 2. At epoch k, some job in classes $[i+1:m]$ is assigned onto machines $[1:i]$ in Phase II.

The load of machine $i+1$ must be larger than $5F_{opt}$. Find the index $i_{\min} \ge i+1$ such that machines $[i+1:i_{\min}]$ have load more than $5F_{opt}$ and machine $i_{\min}+1$ has load at most $5F_{opt}$. It is possible that $i_{\min} = m$ which means that all of the machines $[i+1:m]$ have load more than $5F_{opt}$. We claim that machines $i_{\min}+1$ and $i_{\min}$ must be seperated. Otherwise, suppose that TwoStageSchedule assigns job j in classes $[i_{\min}+1:m]$ onto machines $[1:i_{\min}]$. Job j cannot be assigned in Phase II, because the load of machine $i_{\min}+1$ is at most $5F_{opt}$. It cannot be assigned in Phase I either, because the load of machine $i_{\min}$ is more than $5F_{opt}$.

Applying Lemma 4 to $i_{\min}$, we get

$$B_{i_{\min}}(k) \le B^{ropt}_{i_{\min}}(k) + F_{opt}\sum_{i'=1}^{i_{\min}} s_{i'} \tag{3}$$

Since the load of any machine in $[i+1:i_{\min}]$ is larger than $5F_{opt}$, we get

$$B_i(k) \le B_{i_{\min}}(k) - 5F_{opt}\sum_{i'=i+1}^{i_{\min}} s_{i'} \tag{4}$$

Since in the restricted schedule the load of any machine is at most $4F_{opt}$, we get

$$B^{ropt}_{i_{\min}}(k) \le B^{ropt}_{i}(k) + 4F_{opt}\sum_{i'=i+1}^{i_{\min}} s_{i'} \tag{5}$$

Combining the inequalities (3), (4) and (5), we get:

$$\begin{aligned} B_i(k) &\le B^{ropt}_{i_{\min}}(k) + F_{opt}\sum_{i'=1}^{i_{\min}} s_{i'} - 5F_{opt}\sum_{i'=i+1}^{i_{\min}} s_{i'} \\ &\le B^{ropt}_{i}(k) + F_{opt}\sum_{i'=1}^{i_{\min}} s_{i'} - F_{opt}\sum_{i'=i+1}^{i_{\min}} s_{i'} \\ &= B^{ropt}_{i}(k) + F_{opt}\sum_{i'=1}^{i} s_{i'}, \end{aligned}$$

which completes the proof of Lemma 1. □

B. Proof of Lemma 2

Lemma 2 will be proved by the mathematical induction over i.

Since $B_0(k) = B^{ropt}_0(k) = 0$, $A_0(k+1) = A^{ropt}_0(k+1) = 0$, the base case is trivially true.

There are two different cases depending on the value of $B_i(k) - B_{i-1}(k)$.

Case 1. $B_i(k) - B_{i-1}(k) \le 4F_{opt} \cdot s_i$.

We have $A_i(k+1) - A_{i-1}(k+1) \le F_{opt} \cdot s_i$. Using the inductive hypothesis, we get $A_{i-1}(k+1) \le A^{ropt}_{i-1}(k+1) + F_{opt}\sum_{i'=1}^{i-1} s_{i'}$. Hence we get

$$\begin{aligned} A_i(k+1) &\le A_{i-1}(k+1) + F_{opt} \cdot s_i \\ &\le A^{ropt}_{i-1}(k+1) + F_{opt}\sum_{i'=1}^{i-1} s_{i'} + F_{opt} \cdot s_i \\ &\le A^{ropt}_{i-1}(k+1) + F_{opt}\sum_{i'=1}^{i} s_{i'} \\ &\le A^{ropt}_{i}(k+1) + F_{opt}\sum_{i'=1}^{i-1} s_{i'}. \end{aligned}$$

Case 2. $B_i(k) - B_{i-1}(k) > 4F_{opt} \cdot s_i$.

TwoStageSchedule must have assigned a job onto machine i in Phase II. It means that the load of any machine in $[1:i]$ is larger than $3F_{opt}$. Thus we get

$$A_i(k+1) = B_i(k) - 3F_{opt}\sum_{i'=1}^{i} s_{i'}.$$

Recall that at the beginning of this section we define $B^{ropt}_i(k)$ as $\sum_{i'=1}^{i} \max\{L^{ropt}_{i'} \cdot s_{i'}, 3F_{opt} \cdot s_{i'}\}$. It follows that $A^{ropt}_i(k+1) = B^{ropt}_i(k) - 3F_{opt}\sum_{i'=1}^{i} s_{i'}$.

We then get

$$A_i(k+1) = B_i(k) - 3F_{opt}\sum_{i'=1}^{i} s_{i'}$$
$$\le B_i^{ropt}(k) + F_{opt}\sum_{i'=1}^{i} s_{i'} - 3F_{opt}\sum_{i'=1}^{i} s_{i'}$$
$$= A_i^{ropt}(k+1) + F_{opt}\sum_{i'=1}^{i} s_{i'},$$

which means (1) holds for i at epoch $k+1$, thus completing the proof of Lemma 2. □

C. *Proof of Theorem 1*

The restricted schedule obtained at the beginning of this section assigns the jobs in JI_k at epoch k, just as TwoStageSchedule. If $F_{opt} \ge Opt$, its objective value is no more than $4F_{opt}$, which means $B_i^{ropt}(k) \le 4F_{opt}\sum_{i'=1}^{i} s_{i'}$. Hence by (2) we get $B_i(k) \le 5F_{opt}\sum_{i'=1}^{i} s_{i'}$. Choosing $i=1$, we see that if $F_{opt} \ge Opt$, the load of machine 1 is always no more than $5F_{opt}$, and thus TwoStageSchedule never fails. □

D. *Estimating the optimal value*

According to Theorem 1, if TwoStageSchedule fails at epoch k, it must be true that $F_{opt} < Opt$. We have to abandon the assignment of the jobs in JI_k. Set $F'_{opt} = 1.5F_{opt}$ and define the new k-th epoch to be the time $(k-1)F_{opt} + 3F'_{opt}$. TwoStageSchedule then assigns the jobs in JI_k at the new epoch k. The load of any machine at the next epoch will be no more than $6F_{opt} - 3F'_{opt} = F'_{opt}$. Hence (1) still holds for all i, i.e., $A_i(k) \le F'_{opt}\sum_{i'=1}^{i} s_{i'} \le A_i^{ropt}(k) + F'_{opt}\sum_{i'=1}^{i} s_{i'}$. If $F'_{opt} \ge Opt$, then (2) holds for all i and TwoStageSchedule proceeds as normal.

IV. CONCLUSION

In this paper we investigated the problem of scheduling jobs online on uniformly related machines in the list model. For the objective of minimizing maximum flow time, we presented a constant competitive ratio algorithm. A natural extension of the list model is the interval model [14]: each job specifies both the least and the most capable machines. The algorithm presented in this paper fails to work for the interval model. It would be interesting to consider the problem of online scheduling in the interval model.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China (Nos. 61272430 and 61472227).

REFERENCES

[1] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," Annals of discrete mathematics, vol. 5, pp. 287-326, 1979.

[2] J. Y. Leung, Handbook of scheduling: algorithms, models, and performance analysis: CRC Press, 2004.

[3] Y. Robert and F. Vivien, Introduction to scheduling: CRC Press, 2009.

[4] M. A. Bender, S. Chakrabarti, and S. Muthukrishnan, "Flow and stretch metrics for scheduling continuous job streams," in SODA, 1998, pp. 270-279.

[5] M. Mastrolilli, "Scheduling to minimize max flow time: Off-line and on-line algorithms," International Journal of Foundations of Computer Science, vol. 15, pp. 385-401, 2004.

[6] J. Labetoulle, E. L. Lawler, J. K. Lenstra, and A. Rinnooy Kan, "Preemptive scheduling of uniform machines subject to release dates," Stichting Mathematisch Centrum. Mathematische Besliskunde, pp. 1-20, 1982.

[7] E. L. Lawler and J. Labetoulle, "On preemptive scheduling of unrelated parallel processors by linear programming," Journal of the ACM (JACM), vol. 25, pp. 612-619, 1978.

[8] C. Ambühl and M. Mastrolilli, "On-line scheduling to minimize max flow time: an optimal preemptive algorithm," Operations Research Letters, vol. 33, pp. 597-602, 2005.

[9] C. Chekuri, S. Im, and B. Moseley, "Online scheduling to minimize maximum response time and maximum delay factor," Theory of Computing, vol. 8, pp. 165-195, 2012.

[10] S. Anand, K. Bringmann, T. Friedrich, N. Garg, and A. Kumar, "Minimizing maximum (weighted) flow-time on related and unrelated machines," in Automata, Languages, and Programming, ed: Springer, 2013, pp. 13-24.

[11] B. Kalyanasundaram and K. Pruhs, "Speed is as powerful as clairvoyance," Journal of the ACM (JACM), vol. 47, pp. 617-643, 2000.

[12] N. Bansal and B. Cloostermans, "Minimizing maximum flow-time on related machines," in LIPIcs-Leibniz International Proceedings in Informatics, 2015.

[13] A. Bar-Noy, A. Freund, and J. Naor, "On-line load balancing in a hierarchical server topology," SIAM Journal on Computing, vol. 31, pp. 527-549, 2001.

[14] T.-W. Lam, H.-F. Ting, K.-K. To, and W.-H. Wong, "On-line load balancing of temporary tasks revisited," Theoretical Computer Science, vol. 270, pp. 325-340, 2002.