Competitive Online Scheduling with Fixed Number of Queues

Richard Bryann L. Chua
Department of Physical Sciences and Mathematics
University of the Philippines Manila
Padre Faura St., Ermita, Manila
(632)5265858
rlchua@up.edu.ph

Jaime D.L. Caro
Department of Computer Science
University of the Philippines Diliman
Diliman, Quezon City
(632)9202080
jaime.caro@up.edu.ph

ABSTRACT

One of the complex parts of an operating system design is CPU scheduling, where the OS schedules a sequence of arriving jobs to use the CPU, without knowledge of the time and number of arriving jobs and their execution times. One of the measures of performance of a scheduling algorithm is the average flow time. For a long time, most major operating systems, like Windows and UNIX used a scheduling algorithm based on the Multilevel Feedback scheduling algorithm. In this research, we present the Randomized Multilevel Feedback 2 (RMLF2) scheduling algorithm, which is a version of the RMLF algorithm proposed by Kalyanasundaram and Pruhs, and show that it has a competitive ratio of $O(\ln n)$ in terms of minimizing flow time against an online adaptive adversary. Since this obtains the $\Omega(\log n)$ lower bound for any randomized scheduling algorithm, it has a tight competitive ratio of $\Theta(\ln n)$.

Keywords: randomized multilevel feedback, scheduling, competitive analysis, online algorithm, randomized algorithm

1. INTRODUCTION

CPU scheduling is essential in any multiprogramming systems. In such system, processes arrive over time and the processor should decide which process to run, in ways that meet system objectives, such as response time and throughput [1].

In [13], Motwani, et. al. complained that many early researches in scheduling have been concerned with clairvoyant scheduling, where the characteristics of a job, like its release time and execution time, are known beforehand. Many scheduling algorithms were created from these researches, including the shortest remaining processing time (SRPT), which is considered the most optimal scheduling algorithm that minimizes the total flow time (see [5, 9]). The clairvoyant approach is not applicable in reality because the nature of the scheduling problems encountered is nonclairvoyant, meaning it is impossible to have a priori knowledge of the running time of processes and the time that they will arrive. Since the operating systems does not know the running time of the processes, it is not possible to obtain an optimal average flow time. Hence, in nonclairvoyant scheduling analysis, one uses the method of competitive analysis. In competitive analysis, the performance of a nonclairvoyant scheduler is compared to that of an optimal clairvoyant scheduler for each set of input processes. The nonclairvoyant scheduler is compared to that of an optimal clairvoyant scheduler for each set of input processes. The nonclairvoyant algorithm is measured in terms of its competitive ratio, c_n, which is defined as

$$c_n = \max_J \frac{C_A(J)}{C_{OPT}(J)}$$

where $C_A(J)$ denotes the cost of the schedule produced by the nonclairvoyant algorithm A on input J, $C_{OPT}(J)$ denotes the cost of the optimal schedule produced by the optimal clairvoyant algorithm *OPT*, and the maximum is over all inputs J with n processes. In [8], Kalyanasundaram and Pruhs interpreted the competitive ratio as the payoff to a game played between an online nonclairvoyant algorithm and an all-powerful malevolent adversary *OPT* that specifies the input J, and schedules J optimally.

In [9], Kalyanasundaram and Pruhs proposed the Randomized Multilevel Feedback (RMLF) scheduling algorithm, which is a variant of the Multilevel Feedback (MLF) scheduling algorithm used in Windows and UNIX, and showed that it has a competitive ratio of $O(\log n \log\log n)$, where n is the number of jobs. Their result have been obtained with an RMLF algorithm where the highest queue is determined by the length of the longest job. However, in most operating systems, the number of queues is fixed. In this research, we propose a version of RMLF, where the number of queues is fixed, and determine its competitive ratio.

2. RELATED RESULTS

In [16], Pruhs et. al. classified scheduling algorithms into online and offline algorithm. In an *online algorithm*, the algorithm does not have access to the entire input sequence, as it makes its decision. On the other hand, an offline algorithm has access to the entire input sequence. Another classification used by Pruhs et. al. in [16] is whether an algorithm is clairvoyant or nonclairvoyant. A *nonclairvoyant algorithm* has no knowledge of the characteristics of the running jobs while a *clairvoyant algorithm* can gain knowledge of the characteristics of the running job. The most widely accepted measure of the performance of a scheduling algorithm is the flow time – the time spent by the job in the system between its release and completion, that is

$$F_i = C_i - r_i$$

where F_i is the flow time of job J_i, C_i is the completion time, and r_i is its release time. Flow time is also called wait time or latency [16].

Research on nonclairvoyant scheduling was started by Motwani et. al. in [13]. They have obtained an $\Omega(2-2/(n+1))$ competitive ratio for any static deterministic nonclairvoyant scheduling algorithm and an $\Omega(n^{1/3})$ competitive ratio for any dynamic deterministic algorithm. In static scheduling, all jobs are released at time 0, while in dynamic scheduling, jobs have arbitrary and nonnegative release times.

Randomization can be done to a scheduling algorithm to improve its performance. Motwani et. al. have shown in [13] that a static randomized scheduling algorithm has a competitive ratio of $\Omega(2-4/(n+3))$, which is the same as the deterministic one, but a dynamic randomized scheduling algorithm has a competitive ratio that improves to $\Omega(\log n)$.

Kalyanasundaram and Pruhs proposed the Randomized Multilevel Feedback (RMLF) scheduling algorithm in [9]. RMLF is similar to the multilevel feedback scheduling used in UNIX systems described by Bach in [1]. The idea behind RMLF is to try to approximately behave like SRPT [2]. They have showed that RMLF has a competitive ratio of $O(\log n \log\log n)$ against an adaptive adversary. In an adaptive adversary, the adversary knows all the actions taken by RMLF for servicing the input instance revealed to RMLF up to time t, and may makes its decision based on this knowledge [4]. Becchetti and Leonardi in [2] performed a competitive analysis against an oblivious adversary and found it to have a competitive ratio of $O(\log n)$. In an oblivious adversary, the input sequence is constructed in advance and the adversary pays the optimal cost [4]. Becchetti et al. later in [3] used smoothed competitive analysis with partial bit randomization smoothening model to analyze the multilevel feedback scheduling algorithm and found it to have a smoothed competitive ratio $O\left((2^k/\sigma)^3 + (2^k/\sigma)^2 2^{K-k}\right)$, where 2^K is the maximum processing time, k is a constant in the smoothed processing time, and σ is the standard deviation of the distribution.

3. SCHEDULING PROBLEM DEFINITION

We are given a set J of n jobs and these are to be run on a single processor machine. Each job J_j, $1 \le j \le n$, is characterized by a release time r_j and an execution or processing time x_j, $x_j > 0$. We order the jobs in J by increasing release times, that is, given two jobs J_i and J_j, if $i < j$, then $r_i < r_j$.

RMLF2 is an online nonclairvoyant scheduling algorithm which means it does not know the arrival time of a job until the job is released. The RMLF2 also does not know the execution time of a job until the completion of a job. To be more convenient, we will assume that the length of the shortest job is 2 and this is known by the algorithm a priori.

Definition 1 Define the following quantities

1. Let $J = \{J_1, J_2, \ldots, J_n\}$ be the set of jobs.
2. Let r_j be the release time of a job J_j.
3. Let x_j be the processing time of job J_j.
4. Let $w_j(t)$ be the amount of time that J_j has been run before time t.
5. Let $y_j(t) = x_j - w_j(t)$ be the remaining time that J_j needs to be processed at time t.
6. Let τ be a constant and is set to 12.
7. Let β_j, $3 \le j \le n$, be an exponentially distributed random variable with probability distribution function $\Pr[\beta_j \le x] = 1 - \exp(-\tau x \ln j) = 1 - j^{-\tau x}$.
8. Let l be the index of the highest queue.
9. For all i, $0 \le i \le l$ and for all j, $1 \le j \le n$, define a target $T_{i,j} = 2^i \max(1, 2-\beta_j)$. $T_{i,j}$ is the maximum amount of time job J_j is run in the machine as it reaches Q_i. Randomization is performed on the target in order to improve its performance.
10. For all i, $0 \le i \le l$ and for all j, $1 \le j \le n$, define the quantum as
$$Q_{i,j} = \begin{cases} 2^{i-1}\max(1, 2-\beta_j) & \text{if } i \ge 1 \\ \max(1, 2-\beta_j) & \text{if } i = 0 \end{cases}$$
Quantum is the maximum amount of time job J_j spent running in Q_i.
11. Let C_j be the completion time of job J_j.
12. Let $F_j = C_j - r_j$ be the flow time of a job J_j. The total flow time of the set J is $F(J) = \sum_{J_j \in J} F_j$. The goal of any scheduling algorithm is to minimize the total flow time.

We compare RMLF2 against an online adaptive adversary. We measure the cost of RMLF2 and the adversary by the total flow time. The adversary is charged with the optimal flow time. We can say that any randomized algorithm A has a competitive ratio of c against an adaptive adversary if for any input J,
$$c = \max_J \frac{E[F_A(J)]}{F_{OPT}(J)}$$
where the expectation if taken over any possible J.

4. RANDOMIZED MULTILEVEL FEEDBACK ALGORITHM WITH FIXED NUMBER OF QUEUES (RMLF2)

RMLF2 has priority queues $Q_0, Q_1, \ldots, Q_l$. We say that Q_i is lower than Q_j if $i < j$.

Algorithm 2 At any time t, RMLF2 behaves as follows:

1. When a job J_h is released at time r_h, it is placed on Q_0 and the target $T_{0,h}$ is set to $\max(1, 2-\beta_h)$. If, just prior to r_h, Q_0 was empty and J_j is currently running, J_j is preempted and J_h is run.
2. Let Q_{i-1} be the lowest nonempty queue. RMLF2 always run the job at the front of Q_{i-1}. Let $J_j \in Q_{i-1}$ be the job that is being run. If J_j has been run for a time $Q_{i-1,j}$,
   a. If $i - 1 \ne l$
      i. Job J_j is removed from Q_{i-1} and placed on Q_i.
      ii. The target $T_{i,j}$ is set to $2T_{i-1,j} = 2^i \max(1, 2-\beta_j)$.
      iii. The job at the front of Q_{i-1} is run. If Q_{i-1} is empty, the job at the front of the new lowest nonempty queue is run. If none is found, the algorithm terminates.
   b. If $i - 1 = l$

i. Job J_j is removed from Q_l and placed on Q_l.
ii. The job at the front of Q_l is run. This process continues until Q_l becomes empty.

3. When a job J_j that is being run is completed, it is removed from the queue and the first job of the lowest nonempty queue is run. If there are no more jobs, the algorithm terminates.

5. RMLF2 ALGORITHM ANALYSIS

Definition 3 Define the following quantities

1. Let *RMLF*2 be the RMLF2 scheduler.
2. Let $U_{RMLF2}(t)$ be the set of jobs that are released by time t, but not completed by the *RMLF*2 by time t.
3. Let *ADV* be the adversarial scheduler.
4. Job J_j is short for Q_i if $2^{i-1} \le x_j \le 2^{i-1} + 2^{i-2}$. Job J_j is long for Q_i if $x_j > 2^{i-1} + 2^{i-2}$.
5. Job J_j is unlucky if it is short for some Q_i, and is promoted to Q_i at some time. Mathematically, this is $2^{i-1} \le x_j \le 2^{i-1} + 2^{i-2}$, in terms of processing time. In terms of quantum, this is $0 < y_j \le 2^{i-2}$.
6. Let *BL* be the set of jobs $J_j \in U_{RMLF2}(t)$ such that J_j is not in the front of a queue at time t, and such that if $J_j \in Q_h(t)$ then J_j is long for Q_h.
7. A queue Q_i is small at some time t if there is a contiguous subqueue C of $BL \cap Q_i(t)$ with

$$\sum_{J_j \in C} y_j(t) = \frac{|C| 2^{i-3}}{\tau \ln n}, \quad 3 \le i \le n$$

5.1 HIGH PROBABILITY ARGUMENTS

There are two events that let an adversary significantly defeat RMLF2, namely, the presence of unlucky jobs and small queues [9]. We prove that these two events happen with low probability, therefore, we can ignore them in our analysis.

5.1.1 UNLUCKY JOBS

We prove in this section that the probability of the presence of unlucky jobs even for the usual MLF algorithm is low.

Lemma 4 The probability that the number of unlucky jobs is greater than $(c + 1) \ln n / \ln \ln n$ is at most $1 / n^c$, $c \ge 2$.

Proof. Let J_j be a job that is short for Q_i. Note that

$$T_{i-1,j} = 2^{i-1} \max(1, 2 - \beta_j) \ge 2^{i-1} + 2^{i-1}(1 - \beta_j)$$

The first term is the total time that job J_j has been run while in Q_i by the usual MLF, and the second term, $2^{i-1}(1 - \beta_j)$, is its remaining processing time.

In the case that a job is moved to Q_i from Q_{i-1}, $y_j = 2^{i-1}(1 - \beta_j)$. When a job J_j is removed from Q_l and placed again in Q_l, $y_j = 2^{l-1}(1 - \beta_j)$ since $x_j = a2^{l-1} + 2^{l-1}(1 - \beta_j)$. In any case, for a job to be unlucky, we must have $y_j \le 2^{i-2}$. Therefore, $2^{i-1}(1 - \beta_j) \le 2^{i-2}$, which means $\beta_j \ge ½$. The probability that a job is unlucky is $\Pr[\beta_j > 1/2] = 1 - \Pr[\beta_j \le 1/2] = j^{-\tau/2} = j^{-6}$.

Let $v_1, v_2, \ldots, v_n$ be Bernoulli trials with the probability that v_j is unlucky is j^{-6}. From [14], the expectation of v_j is $E[v_j] = j^{-6}$. Let X be a random variable that denotes the total number of success (unlucky jobs). The expectation of X is $E[x] = \mu = \sum_{j=1}^{n} j^{-6}$. The right tail bound from [6], is

$$\Pr[X - \mu \ge x] \le \left(\frac{\mu e}{x}\right)^x$$

To get a value of x such that $(\mu e/x)^x \le 1 / n^c$, we have

$$x(\ln x - \ln \mu - 1) \ge c \ln n$$

Let $x = d \ln n / \ln \ln n$, then

$$\frac{d \ln n}{\ln \ln n}\left(\ln\left(\frac{d \ln n}{\ln \ln n}\right) - \ln \mu - 1\right) \ge c \ln n$$

$$d \ln n + \frac{d \ln n}{\ln \ln n}(\ln d - \ln \ln \ln n - \ln \mu - 1) \ge c \ln n$$

If we let $d = c + 1$, this inequality is true. ■

5.1.2 SMALL QUEUES

RMLF2 prevents the presence of small queues with high probability. We prove in this section that the presence of small queues happens with low probability.

Let C be a contiguous subqueue of jobs. Assume that we have a particular subqueue C that makes Q_i small. Number the jobs in C as $J_1, J_2, \ldots, J_k$ according to increasing order of release times. Let t be a time that C makes Q_i small.

We first prove that the event that $\sum_{j=1}^{k} \beta_j$ is too small happens with low probability.

Lemma 5

$$\Pr\left[\sum_{j=1}^{k} \beta_j \le \frac{k}{2\tau \ln n}\right] \le \frac{1}{2^k}, \quad k \ge 2$$

Proof. From our definition of β_j, $\Pr[\beta_j \le x] = 1 - j^{-\tau x} \le 1 - n^{-\tau x}$. Let β'_j be a random variable such that $\Pr[\beta'_j \le x] = 1 - n^{-\tau x}$. For any x,

$$\Pr[\beta_j \le x] \le \Pr[\beta'_j \le x]$$

$$\Pr\left[\sum_{j=1}^{k} \beta_j \le x\right] \le \Pr\left[\sum_{j=1}^{k} \beta'_j \le x\right]$$

To prove this lemma, it suffices to show that

$$\Pr\left[\sum_{j=1}^{k} \beta'_j \le x\right] \le \frac{1}{2^k}$$

Let $X_1, X_2, \ldots, X_k$ be independent exponential random variables with probability distribution $1 - \exp(-n/b)$, respectively. The sum $S = X_1 + X_2 + \ldots + X_k$ has the Erlang form of the gamma distribution [7, 10],

$$\Pr[S \le x] = 1 - \exp\left(-\frac{x}{b}\right) \sum_{j=0}^{k-1} \frac{(x/b)^j}{j!}$$

where $S = \sum_{j=1}^{k} \beta'_j$ and $b = 1/(\tau \ln n)$.

The Maclaurin series of

$$\exp\left(\frac{x}{b}\right) = \sum_{h=0}^{\infty} \frac{(x/b)^h}{h!}$$

Thus,

$$\Pr[S \le x] = 1 - \exp\left(-\frac{x}{b}\right)\sum_{j=0}^{k-1}\frac{(x/b)^h}{h!}$$
$$\le \sum_{h=k}^{\infty}\frac{(x/b)^h}{h!}$$
$$\le \frac{(x/b)^k}{k!}\left(1 + \left(\frac{x}{b}\right) + \left(\frac{x}{b}\right)^2 + \dots\right)$$

The second factor is a geometric series and it converges if $x\tau \ln n < 1$, thus, $x < 1/(\tau \ln n)$.

Let $x = 1 / (2\tau \ln n)$, we have

$$\Pr\left[\sum_{j=1}^{k}\beta'_j \le \frac{1}{2\tau \ln n}\right] \le \frac{1}{2^k k!}\left(1 + \frac{1}{2} + \frac{1}{4} + \dots\right)$$
$$\le \frac{1}{2^k}$$

■

We now bound the probability that a particular $y_j(t)$ is small by the probability that a β_j is small.

Lemma 6 For every $J_j \in C$, and for all $\gamma < 2^{i-2}$, $i < l$, $\Pr[y_j(t) \le \gamma] \le \Pr[2^{i-1}\beta_j \le \gamma]$.

Proof. Consider a job $J_j \in C$. By Lemma 4, we can ignore unlucky jobs. Assume J_j is long for Q_i, which means $x_j \ge 2^{i-1} + 2^{i-2}$. We divide our proof into three cases. Since our variables are of common time, we will drop our reference on t.

Case 1. $2^{i-1} + 2^{i-2} \le x_j \le 2^i$

The remaining processing time of J_j is $y_j(t) = x_j(t) - T_{i-1,j} \ge x_j - 2^{i-1}(2 - \beta_j)$. Since the job is promoted from Q_{i-1} to Q_i, we have

$$x_j \quad \ge T_{i-1,j}$$
$$2^i\beta_j \ge 2^i - x_j$$

We have assumed that $\gamma < 2^{i-2}$, therefore

$$\Pr[y_j(t) \le \gamma]$$
$$\le \Pr[x_j(t) - 2^{i-1}(2-\beta_j) \le \gamma \,|\, 2^i\beta_j \ge 2^i - x_j]$$
$$= \Pr[2^i\beta_j \le \gamma + 2^i - x_j \,|\, 2^i\beta_j \ge 2^i - x_j]$$
$$= \Pr[2^i\beta_j \le \gamma]$$

The equation above is true because an exponential distribution is memoryless [12].

Case 2. $2^i \le x_j \le T_{l,j}$

The remaining processing time of J_j is

$$y_j(t) = x_j(t) - T_{i-1,j}$$
$$= x_j - 2^i + 2^i - 2^{i-1}\max(1, 2 - \beta_j)$$

Assume that $y_j(t) \le \gamma + (x_j(t) - 2^i)$. This is equivalent to $2^{i-1}\beta_j \le \gamma$. Therefore,

$$\Pr[y_j(t) \le \gamma + (x_j(t) - 2^i)] = \Pr[2^i\beta_j \le \gamma]$$

It is trivial that

$$\Pr[y_j(t) \le \gamma] \le \Pr[y_j \le \gamma + (x_j(t) - 2^i)] = \Pr[2^i\beta_j \le \gamma]$$

Case 3. $x_j > T_{l,j}$

The remaining processing time J_j is $y_j(t) = x_j(t) - T_{l,j} - aQ_{l,j} \ge x_j - 2^{l+1} + 2^l\beta_j - a2^l + 2^{l-1}a\beta_j$. Since J_j is moved from Q_l to Q_l,

$$x_j \quad \ge T_{l,j} + aQ_{l,j}$$
$$2^{l-1}\beta_j \ge (2^{l+1} - 2^l\beta_j + a2^l - x_j)\frac{1}{a}$$

This means that

$$\Pr[y_j(t) \le \gamma]$$
$$\le \Pr\left[x_j(t) - 2^{l+1} + 2^l\beta_j - a2^l + 2^{l-1}a\beta_j \le \gamma \,\middle|\, 2^{l-1}\beta_j \ge \frac{2^{l+1} - 2^l\beta_j - x_j}{a} + 2^l\right]$$
$$= \Pr\left[2^{l-1}\beta_j \le \frac{\gamma}{a} + \frac{2^{l+1} - 2^l\beta_j - x_j}{a} + 2^l \,\middle|\, 2^{l-1}\beta_j \ge \frac{2^{l+1} - 2^l\beta_j - x_j}{a} + 2^l\right]$$
$$= \Pr\left[2^{l-1}\beta_j \le \frac{\gamma}{a}\right]$$
$$\le \Pr[2^{l-1}\beta_j \le \gamma]$$

■

We now bound the $\Pr\left[\sum_{j=1}^{k} y_j(t) \le \sigma\right]$ by relating this to $\Pr\left[\sum_{j=1}^{k}\beta_j \le \sigma\right]$.

Lemma 7 Let $\sigma = |C|2^{i-2}/(2\tau \ln n)$ then

$$\Pr\left[\sum_{J_j \in C} y_j(t) \le \sigma\right] \le \Pr\left[\sum_{j=1}^{|C|/2} 2^{i-1}\beta_j \le \sigma\right]$$

Proof. We can observe that for all $C' \subseteq C$,

$$\Pr\left[\sum_{J_j \in C} y_j(t) \le \sigma\right] \le \Pr\left[\sum_{J_j \in C'} y_j(t) \le \sigma\right]$$

Let D be the set of jobs in C such that $y_j(t) < 2^{i-2}$. We can see that

$$\Pr\left[\sum_{J_j \in C} y_j(t) \le \sigma\right] \le \Pr\left[\sum_{J_j \in D} y_j(t) \le \sigma\right]$$

If $|D| < |C|/2$, then there are at least $|C|/2$ jobs in C with remaining processing time that are at least 2^{i-2}. Hence,

$$\sum_{J_j \in C} y_j(t) \geq \frac{|C|2^{i-2}}{2} \geq \frac{|C|2^{i-2}}{2\tau \ln n} = \sigma$$

We assume that $|D| \geq |C|/2$. By Lemma 6,

$$\Pr\left[y_j(t) \leq \frac{\sigma}{|D|} \right] \leq \Pr\left[2^{i-1}\beta_j \leq \frac{\sigma}{|D|} \right]$$

Since $y_j \geq 0$ and $2^{i-1}\beta_j \geq 0$, for every j,

$$\Pr\left[\sum_{J_j \in D} y_j(t) \leq \frac{|D|\sigma}{D} \right] \leq \Pr\left[\sum_{J_j \in D} 2^{i-1}\beta_j \leq \frac{|D|\sigma}{D} \right]$$

$$\Pr\left[\sum_{J_j \in D} y_j(t) \leq \sigma \right] \leq \Pr\left[\sum_{J_j \in D} 2^{i-1}\beta_j \leq \sigma \right]$$

$$\leq \Pr\left[\sum_{j=1}^{|C|/2} 2^{i-1}\beta_j \leq \sigma \right]$$

■

We now conclude that the probability that RMLF2 has a small queue is low.

Lemma 8 The probability that RMLF2 has a small queue is at most $1 / 2^{|C|/2}$, where C is a contiguous subqueue of $BL \cap Q_i(t)$.

Proof. From Lemma 7,

$$\Pr\left[\sum_{J_j \in C} y_j(t) \leq \sigma \right] \leq \Pr\left[\sum_{j=1}^{|C|/2} 2^{i-1}\beta_j \leq \sigma \right]$$

$$= \Pr\left[2^{i-1}\sum_{j=1}^{|C|/2} \beta_j \leq \frac{|C|2^{i-1}}{2 \cdot 2\tau \ln n} \right]$$

$$= \Pr\left[\sum_{j=1}^{|C|/2} \beta_j \leq \frac{|C|}{2 \cdot 2\tau \ln n} \right]$$

$$\leq \frac{1}{2^{|C|/2}} \quad \text{from Lemma 5}$$

■

5.2 DETERMINISTIC ANALYSIS

The cost of a scheduling algorithm is its total flow time, which can be obtained by counting the number of unfinished jobs over time [2, 9, 11]. Since this is a cost minimization problem, we can use Yao's technique for cost minimization problem [15]. We have already proven in lemmas 4 and 8 that the events of having unlucky jobs and small queues happen with low probability, therefore, we can ignore them in our analysis. We now assume a deterministic algorithm DMLF2, which is an RMLF2 that never encounters a small queue or more than $(c+1)\ln n / \ln \ln n$ unlucky jobs.

DMLF2 always runs the job that is at the front of the lowest nonempty queue. If a new job arrives, it is placed in Q_0. If DMLF2 is currently running a job in a queue that is higher than Q_0, the running job is preempted to give priority to the new job that is at Q_0. This might result to little remaining processing time for those jobs that are at the front of the queues. But since each queue can have at most one job at its front at any time and there are l queues, the maximum number of jobs at the front of a queue is l. Since l is comparatively smaller than n, we can ignore the jobs that are at the front of a queue in our analysis.

5.2.1 JOB PARTITIONING

We now use the technique of Kalyanasundaram and Pruhs in [9]. We now partition the unfinished jobs, which will serve as the basis of what DMLF2 and the adversary will run.

Definition 9 Let t be the time being considered

1. Order the jobs in BL from highest queue to lowest queue and from the front of a queue to the back of the queue. Let $\mathcal{P} = \{P^1, \ldots, P^f\}$, where P^i is a set of $\lceil 128\tau \ln n \rceil$ jobs in $BL - \bigcup_{j=1}^{i-1} P^j$. The jobs in the lowest queues are not included in a partition if they are not enough to form a partition.
2. Let $Q_{s(h)}$ be the lowest queue that the jobs in P^h belong.
3. Let $Q_{d(h)}$ be the highest queue that the jobs in P^h belong.
4. Let $Y_{i,h} = \sum_{J_j \in P^h \cap Q_i} y_j$ be the total remaining processing time of the jobs in P^h that are in Q_i.
5. Let $n_{i,h} = |P^h \cap Q_i|$ be the number of jobs in P^h that are in Q_i.

We now bound the sum of the remaining processing time of the jobs in each P^h.

Lemma 10 For large n and for every $P^h \in \mathcal{P}$, $\sum_{J_j \in P^h} y_j > 2^{s(h)}$.

Proof. We drop our reference to h. If any $n_i \geq 8\tau \ln n$, since a queue is never small, $\sum_{J_j \in P^h \cap Q_i} y_j > 8\tau \ln n 2^{i-2} / (2\tau \ln n) = 2^i$. Since $2^i \geq 2^s$, this proves our lemma for this case.

Assume that all $n_i < 8\tau \ln n$ and $\sum_{J_j \in P} y_j \leq 2^s$. We will do proof by contradiction. Since a queue is never bad,

$$Y_i \geq \frac{n_i 2^{i-2}}{2\tau \ln n}$$

Consider the optimization problem:

$$\min \sum_{i=s}^{d} \frac{n_i 2^{i-2}}{2\tau \ln n}$$

$$\text{subject to} \sum_{i=s}^{d} n_i = 128\tau \ln n$$

The functions $f(n) = n_i 2^{i-2} / (2\tau \ln n)$ and $f(i) = 2^{i-2}$ are both increasing functions. If $n_i < n_{i+1}$, we cannot find a solution to the minimization problem. To see this, substitute n_i's to the minimization problem, where $n_i < n_{i+1}$ and consider the sum of two consecutive terms, $n_i 2^{i-2} + n_{i+1} 2^{i-1}$. If we increase n_i by a small value and decrease n_{i+1} by a small value, $(n_i + 0.01)2^{i-2} + (n_{i+1} - 0.01)2^{i-1}$, we get a much smaller value for the sum of the two consecutive terms. Therefore, to have an optimal solution, we must have $n_i \geq n_{i+1}$.

We should also have this condition in order to have an optimal solution:

$$n_i 2^{i-2} < (n_s + n_i) 2^s \qquad (1)$$

Otherwise, we could get a smaller sum by giving the value meant for n_i to n_s. From inequality 1, we get

$$
\begin{aligned}
n_i 2^{i-2} &< (n_s + n_i) 2^s \\
&< 16\tau \ln n \cdot 2^s \\
n_i &< \frac{64\tau \ln n \cdot 2^s}{2^i} \\
\sum_{i=s}^{d} n_i &< 64\tau \ln n \sum_{i=s}^{d} \frac{2^s}{2^i} \\
&< 64\tau \ln n \sum_{i=s}^{\infty} \frac{2^s}{2^i} \\
&= 128\tau \ln n
\end{aligned}
$$

This is a contradiction to the defined size of P^h, which means that $\sum_{J_j \in P^h} y_j$ is not minimized with size $128\tau \ln n$, which then proves our lemma. ■

5.2.2 ADVERSARIAL BORROWING

We now describe what an adversary can to defeat RMLF2 and then prove that RMLF2 prevents such strategy of the adversary from defeating it.

Definition 11 Let DMLF2' be a scheduling algorithm that is similar to DMLF2, except that the job remains in the queue even if they are already completed. In DMLF2', if a job J_j is completed, it is moved to the back of the next queue. If the queue is already Q_l, J_j is placed at the back of Q_l. If a completed job J_j is at the front of the lowest nonempty queue, J_j is moved to the back of the next queue. If the queue is already in Q_l, J_j is placed at the back of Q_l.

We use DMLF2' to define the last queue value and in constructing the borrowing graph.

Definition 12 Let t' be some time after t. Consider DMLF2'. If a job $J_j \in Q_i(t')$, define the last queue value of J_j at time t' as

$$
lq_j(t') = \begin{cases} i & Q_0, \ldots Q_{i-1} \text{ are empty}, r_j \le t' \\ i-1 & \text{otherwise} \end{cases}
$$

Definition 13 Define a graph with these properties: For each job, create a vertex in the graph. For every directed edge from J_h to J_j, assign a nonnegative cost $f_{h,j}$, such that

$$
f_{h,j} = \begin{cases} s & \text{for } s \text{ amount of time before time } t \\ & \text{DMLF2 was running } J_h \\ & \text{and the adversary was running } J_j \\ 0 & \text{otherwise} \end{cases}
$$

We can interpret the edge with positive cost of the graph in Definition 13, $f_{h,j}$, as the adversary borrowing $f_{h,j}$ amount of time from J_h to give to J_j. This means that the adversary runs J_j, instead of J_h which is the one run by DMLF2, with the objective that running J_j first will result to an optimal schedule. Consider the subgraph of this graph with all the edges having weight $f_{h,j}$ = 0 removed. We will refer to this subgraph as a borrow graph. Let $F_{i,j}$ be the sum of the weights of the directed edges along the directed path from J_i to J_j. We can interpret $F_{i,j}$ as the amount of time borrowed by the adversary from J_i to give to J_j. The sources here are the jobs in U_{ADV} and the sinks are the jobs in $U_{DMLF2} - U_{ADV}$. If a vertex J_j is neither a source nor sink, then the sum of the weights of the edges going to J_j is the same as the sum of the weights of the edges going out of J_j.

We use this graph to prove that DMLF2 restricts the adversary in its borrowing strategy.

Lemma 14 If there is a directed path from J_h to J_j in the borrow graph, then $lq_h(t) \le lq_j(t)$.

Proof. We now consider DMLF2'.

Base Case: Consider a time t' that DMLF2' was running J_h and the adversary was running Jj. J_j can never be in a lower queue that J_h since DMLF2' always runs a job at the lowest non-empty queue.

If J_j is in a higher queue than J_h, by the way DMLF2' moves jobs along the queues, J_j will always be moved to a higher queue first than J_h, which means $lq_h(t') \le lq_j(t')$.

If J_h and J_j are on the same queue, J_h is at the front since this is the job run by DMLF2'. Although J_h is moved to a higher queue first than J_j, the lq_h will be incremented only if the lower queue is already empty, which means J_j has already been moved to the higher queue, thus $lq_h(t') = lq_j(t')$.

Inductive hypothesis: Assume that if there is directed path from J_h to J_{j-1} of $k-1$ number of directed edges, then $lq_h(t) \le lq_j(t)$.

Inductive step: Consider a directed path from J_h to J_{i-1} and a directed edge from J_{j-1} to J_j. By the inductive hypothesis, $lq_h(t) \le lq_{j-1}(t)$. By the base case, $lq_{j-1}(t) \le lq_j(t)$. Therefore, $lq_h(t) \le lq_j(t)$. ■

We now find a lower bound for the number of uncompleted jobs of the adversary.

Lemma 15

$$
|U_{ADV}(t)| \ge \frac{f}{2}
$$

Proof. Allow the adversary to be the most powerful form of adversary by letting it borrow time from job J_i to complete job J_j, if $lq_i(t) \le lq_j(t)$, based on Lemma 14. Let $\mathcal{P} = \{P^{\alpha(1)}, P^{\alpha(2)}, \ldots, P^{\alpha(k)}\}$ be a subset of $\mathcal{P}$ where all the jobs in $P^{\alpha(i)}$ are already completed by the adversary. Mathematically, the partitions in $\mathcal{P}$ have empty intersection with U_{ADV}. Let us number the partitions such that if $i < j$, then $\alpha(i) < \alpha(j)$. Let us also number the jobs in U_{ADV} in non-decreasing order of lq values. That is, let $U_{ADV} = \{J_{\rho(i)} \mid 1 \le \rho(i) \le |U_{ADV}|\}$, if $\rho(a) \le \rho(b)$, then $lq_{\rho(a)} \le lq_{\rho(b)}$.

We use mathematical induction to show that for every i, $1 \le i \le k$, U_{ADV} contains at least i jobs, and for every h, $1 \le h \le i$, $lq_{\rho(h)} \le s(\alpha(h))$ (i.e. the smallest lq value is not greater than the lowest queue in $P^{\alpha(h)}$).

Base case: $i = 1$. Since DMLF2 always runs a job at any time, U_{ADV} is not empty at any time. If $lq_{\rho(1)} \ge s(\alpha(1))$, then for every J_j, $1 < j < |U_{ADV}|$, $lq_j > s(\alpha(1))$, since the jobs in U_{ADV} are ordered in non-decreasing order of lq values. This means that U_{ADV} contains no job that has last queue value that is less than or equal to $s(\alpha(1))$. Thus, the adversary cannot borrow time from any job to finish the job in $P^{\alpha(1)}$, based on Lemma 14. This is a contradiction to our definition of $P^{\alpha(1)}$, which is one of the set of jobs completed by the adversary. Therefore, $lq_{\rho(1)} \le s(\alpha(1))$.

We now have a general i, $1 < i < k$. From Lemma 10, $P_{\alpha(h)}$, $1 \le h \le i-1$, is completed by the adversary if it borrows, at least, $2^{s(\alpha(h))}$ time. We can now have our inductive hypothesis that $lq_{\rho(h)} \le s(\alpha(h))$. We now look at $P^{\alpha(i)}$. Note that $|P^{\alpha(i)} \cap Q_{s(\alpha(i))}|$ is at least 1. Let that job be J_c. Let $S = \{J_c\} + \bigcup_{h=1}^{i-1} P^{\alpha(h)}$. To finish the jobs in S, the adversary needs to borrow at least $y_c + \sum_{h=1}^{i-1} 2^{s(\alpha(h))}$. Assume that $lq_{\rho(i)} > s(\alpha(i))$ and $|U_{ADV}| = i - 1$.

We now consider all jobs $J_{\rho(h)}$, $1 \le h \le i-1$. From the inductive hypothesis, $lq_{\rho(h)} \le s(\alpha(h))$. The adversary can borrow time only from this set of jobs. Note that if $lq_{\rho(h)} \le s(\alpha(h))$, the highest queue where $J_{\rho(h)}$ can be located is $Q_{s(\alpha(h))+1}$, so $w_{\rho(h)} \le 2^{s(\alpha(h))}$. This means that the maximum amount of time that can be borrowed is $\sum_{h=1}^{i-1} 2^{s(\alpha(h))}$, which falls short by y_c. This contradicts our assumption that $|U_{ADV}| = i - 1$ because we need to borrow time from more than $i - 1$ jobs, thus $|U_{ADV}| \ge i$. And for us to be able to borrow from $J_{\rho(i)}$, $lq_{\rho(i)} \le s(\alpha(i))$, which contradicts our assumption.

We have shown that $|U_{ADV}| \ge k$. By the definition of k, $f - k$ partitions are not yet completed by the adversary. Each of these partitions can have at least one job, thus $|U_{ADV}(t)| \ge f - k$. This means that $f - k = k$, hence, $k = f / 2$. Therefore, $|U_{ADV}(t)| \ge f / 2$.
■

We can now obtain the competitive ratio for DMLF2.

Lemma 16 For all late times t,

$$\frac{|U_{DMLF2}(t)|}{|U_{ADV}(t)|} \le O(\ln n)$$

Proof. The number of jobs in $U_{DMLF2}(t)$ that are not BL is l, because these are the jobs that are at the front of a queue. Since l is small relative to n, we can assume that $|U_{DMLF2}(t)| = BL = n$. By Lemma 15, $|U_{ADV}| \ge f / 2$. Since each P^i has $\lceil 128\tau \ln n \rceil$ jobs, $f = n / \lceil 128\tau \ln n \rceil$. Hence,

$$|U_{ADV}(t)| \ge \frac{n}{2\lceil 128\tau \ln n \rceil}$$

$$\frac{n}{|U_{ADV}(t)|} \le 2\lceil 128\tau \ln n \rceil$$

$$\frac{|U_{DMLF2}(t)|}{|U_{ADV}(t)|} \le O(\ln n)$$

■

We can now get the competitive ratio of RMLF2.

Theorem 17 The competitive ratio of RMLF2 is $O(\ln n)$.

Proof. We have already obtained in Lemma 16 an $O(\ln n)$ competitive ratio for DMLF2. We have assumed that DMLF2 does not encounter more than $(c + 1)\ln n / \ln \ln n$ unlucky jobs and small queues. If ever RMLF2 encounters these two events separately, it will have a competitive ratio n and $n / (n - |C|)$, respectively, since the adversary will just complete those unlucky jobs and jobs comprising the small queue rather than postponing their execution to a higher queue. Therefore, $1 \le c \le \sum_{j=1}^{n} j^{-6}$, $|C| > 2$,

$$\begin{aligned}\frac{U_{RMLF2}(t)}{U_{ADV}(t)} &= \frac{|U_{DMLF2}(t)|}{|U_{ADV}(t)|} + E[\text{unlucky jobs}] + E[\text{small queues}]\\ &= O(\ln n) + n\Pr[\text{unlucky jobs}] + \frac{n}{n-|C|}\Pr[\text{small queues}]\\ &= O(\ln n) + n\left(\frac{1}{n^c}\right) + \frac{n}{n-|C|}\left(\frac{1}{2^{|C|/2}}\right)\\ &= O(\ln n)\end{aligned}$$

■

REFERENCES

[1] Maurice Bach. *The Design of the UNIX Operating Systems*. Prentice-Hall Inc., 1990.

[2] Luca Becchetti and Stefano Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *Journal of the ACM*, 51:517-539, July 2004.

[3] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, Guido Schäfer, and Tjark Vredeveld. Average case and smoothed competitive analysis of the multi-level feedback scheduling algorithm. *Mathematics for Operations Research*, 31:85-108, February 2006.

[4] Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.

[5] Peter Brucker. *Scheduling Algorithms*. Springer-Verlag, 1995.

[6] Thomas Cormen, Charles Lieserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms Second Edition*. MIT Press, 2001.

[7] James Johnson. *Probability and Statistics for Computer Science*. Wiley, 2003.

[8] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM*, 47:617-643, July 2000.

[9] Bala Kalyanasundaram and Kirk Pruhs. Minimizing flow time nonclairvoyantly. *Journal of the ACM*, 50:551-567, 2003.

[10] Averil Law and W. David Kelton. *Simulation Modeling and Analysis Third Edition*. McGraw-Hill, 2000.

[11] Stefano Leonardi and Danny Raz. Approximating flow time on parallel machines. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 110-119, 1997.

[12] Michael Mitzenmacher and Eli Upfal. *Probability and Computing*, Cambridge University Press, 2005.

[13] Rajeev Motwani, Steven Philips and Eric Torng. Non-clairvoyant scheduling. *Theoretical Computer Science*, 130:17-47, 1994.

[14] Rajeev Motwani and Prabakar Raghavan. *Randomized Algorithms*. Cambridge University Press. 1995.

[15] Kirk Pruhs. *Competitive Online Scheduling for Server Systems*.

[16] Kirk Pruhs, Jiří Sgall, and Eric Torng. Online scheduling. In Joseph Leung, editor, *Handbook of Scheduling: Algorithms, Models and Performance Analysis*, chapter 15, pages 15-1 – 15-41. CRC Press, 2004.

[17] William Stallings. *Operating Systems Internals and Desing Principles Fourth Edition*. Prentice-Hall Inc., 2001.