Alcom-FT Technical Report Series
ALCOMFT-TR-01-46

Non-Clairvoyant Scheduling to Minimize the Average Flow Time on Single and Parallel Machines

Luca Becchetti* Stefano Leonardi *

November 15, 2000

Abstract

Scheduling a sequence of jobs released over time when the processing time of a job is only known at its completion is a classical problem in CPU scheduling in time sharing operating systems. A widely used measure for the responsiveness of the system is the average flow time of the jobs, i.e. the average time spent by jobs in the system between release and completion.

The Windows NT and the Unix operating system scheduling policies are based on the Multi-level Feedback algorithm [19, 21]. In this paper we prove that a randomized version of the Multi-level Feedback algorithm is competitive for single and parallel machine systems, in our opinion providing one theoretical validation of the goodness of an idea that has been very effective in practice along the last two decades.

The randomized Multi-level Feedback algorithm (RMLF) was first proposed by Kalyanasundaram and Pruhs [12] for a single machine achieving an $O(\log n \log \log n)$ competitive ratio to minimize the average flow time against the on-line adaptive adversary, where n is the number of jobs that are released. We present a version of RMLF working for any number m of parallel machines. We show for RMLF a first $O(\log n \log \frac{n}{m})$ competitiveness result against the oblivious adversary on parallel machines. We also show that the same RMLF algorithm surprisingly achieves a tight $O(\log n)$ competitive ratio against the oblivious adversary on a single machine, therefore matching the lower bound of [17].

*Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", via Salaria 113, 00198-Roma, Italia.
Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT) and IST-2000-14084 (APPOL), and by the MURST Projects "Algorithms for Large Data Sets: Science and Engineering" and "Resource Allocation in Computer Networks". E-mail: {becchett,leon}@dis.uniroma1.it

1 Introduction

In this paper we study non-clairvoyant algorithms to schedule a stream of jobs released over time on single and parallel machine systems. Every job j is assigned with a release time $r_j \geq 0$ and a processing time p_j. Job j must be globally scheduled for p_j time units on m parallel identical machines before its completion. Job preemption is allowed, the execution of a job can be stopped and resumed later on the same or on a different machine. The completion time of job j is denoted by C_j. A *non-clairvoyant* scheduling algorithm works under pretty tough conditions. It knows very little about the input instance: The existence of a job is only known at the release time of the job; The processing time of a job is only known when the job is completed.

This problem has a number of motivating applications. The most classical one is multiprocessor scheduling in a time sharing multitasking operating system, in which scheduling decisions must be taken without knowledge of the time a job needs to be executed. The obvious goal is to provide a fast response to users. Job preemption is widely recognized as a key factor to improve the responsiveness of the system. In multiprocessor computer systems, preemption requires a context switching at a processor but this cost is reasonably small [1].

A widely accepted measure of the quality of service provided to users is the average response time of the system. The response time, or flow time, of every job is the time spent by the job in the system between release and completion, i.e. $C_j - r_j$ for job j. We measure the performance of a randomized non-clairvoyant scheduling algorithm by its competitive ratio [23, 5], the worst case ratio between the expected average flow time of the algorithm and the optimal average flow time of an *oblivious* adversary that generates the input sequence without knowledge of the random choices of the algorithm.

This problem has been addressed for a couple of decades in the design of time sharing operating systems. Windows NT [19] and Unix [21] have at the very basis of their scheduling policies the Multi-level Feedback (MLF) algorithm. The basic idea of MLF is to organize jobs into a set of queues. Each job is processed for 2^i time units when in queue Q_i, before it is promoted to queue Q_{i+1} if not completed. At any time, the machines process the jobs in the lowest queues, in each queue giving priority to jobs at the front. While this algorithm turns out to be very effective in practice, it behaves very poorly from a worst case analysis point of view as explained in the following.

A good rule of thumb for flow time minimization is given by the Shortest Remaining Processing Time (SRPT) first rule. SRPT prescribes the preemption of a job on execution when a job with shorter remaining processing time is released. SRPT is indeed an optimal algorithm for a single machine [3] and provides the best known approximation for parallel machines [16]. However, a non-clairvoyant scheduling algorithm cannot stick to the SRPT rule since it has no knowledge of the processing time of the jobs before they are completed. For what concerns MLF, it behaves on some instances very differently from the SRPT rule in that it may preempt jobs in a high queue that are nearly completed to process newly released jobs of large processing time in lower queues. This may force jobs with small remaining processing to spend a long time in the system while other long jobs are processed. It has actually been shown that no deterministic non-clairvoyant algorithm can be competitive at all against a worst case adversary [17].

To circumvent these difficulties, a randomized version of MLF, called RMLF, was proposed for a single machine by Kalyanasundaram and Pruhs in [12]. The idea is to try to approximately behave like SRPT by having jobs with a remaining processing time that is still a large share of the initial processing time when they enter the queue of completion. Of course, this requirement can only be satisfied with some probability. The authors show that RMLF is $O(\log n \log \log n)$ competitive for minimizing the total flow time on a single machine against the on-line adaptive adversary [5]. The on-

[1]This in particular holds in systems that support threads [7, 11, 18, 22].

line adaptive adversary may decide its strategy at time t based on the random choices of the algorithm until time t. However, the processing time of a job must be fixed by the adversary at the release time of the job.

In this paper we present a randomized version of the Multi-level Feedback algorithm working for any number m of parallel machines. The algorithm we present is an evolution of the one presented in [12] for a single machine, for this reason we also call RMLF our algorithm. We present two main results for RMLF:

1. We show that RMLF has competitive ratio $O(\min\{\log n \log \frac{n}{m}, \log n \log P\})$ against an oblivious adversary, where P is the ratio between the largest and the shortest processing time of a job. This is the first $o(n)$-competitive result for non-clairvoyant average flow time minimization on parallel machines. This compares with the $\Omega(\log \frac{n}{m})$ and $\Omega(\log P)$ lower bounds against the oblivious adversary given in [16] for the case in which the processing time of a job is known at time of release.

2. For the case of a single machine, we show that the our version of RMLF surprisingly matches the $\Omega(\log n)$ lower bound against an oblivious adversary of Motwani, Phillips and Torng [17].

The two results are obtained as an outcome of a unified analysis of the RMLF algorithm. These results are shown with a simpler probabilistic analysis of RMLF not requiring any high probability argument. The analysis also requires a considerable strengthening and simplification of the tools of analysis previously developed in the study of algorithms for minimizing the average flow time [16, 2].

Related results. Non-clairvoyant scheduling to minimize average flow time has been first studied by Motwani, Phillips and Torng [17]. The authors prove that any deterministic non-clairvoyant algorithm is $\Omega(n^{1/3})$-competitive where n is the number of jobs released, and that every randomized non-clairvoyant algorithm is $\Omega(\log n)$-competitive for the case of a single machine. The authors also present competitive algorithms for the static case where all jobs are released at time 0. Kalyanasundaram and Pruhs [12] give the first $o(n)$ competitive non-clairvoyant scheduling result for a single machine. They prove that a randomized version of MLF is $O(\log n \log \log n)$ competitive against an on-line adaptive adversary [5]. They also claim an $\Omega(P)$ randomized lower bound for the problem, where P is the ratio between the maximum and the minimum processing time of a job, on an instance with a number of jobs that is exponential in P. In a previous paper [13] Kalyanasundaram and Pruhs study a different model in which the non-clairvoyant algorithm is equipped with a faster processor than the adversary. They prove in this case that shortest elapsed time first is a constant competitive algorithm for a single machine.

Related results are also concerned with the more classical average flow time minimization problem when the processing time of a job is known at time of release. Also in this case, if preemption is not allowed, achieving reasonable on-line performances is not possible. If preemption is allowed, the problem can be optimally solved on 1 machine by the well known SRPT (Shortest Remaining Processing Time) heuristic [3]. Leonardi and Raz [16] proved that SRPT is $O(\log(\min\{n/m, P\}))$-competitive for $m \geq 2$ identical parallel machines. No better approximation, even off-line, is known for this problem. They also prove an $\Omega(\log \frac{n}{m})$ and an $\Omega(\log P)$ lower bound on the competitive ratio of any randomized algorithm for the problem. These lower bounds clearly extend to the non-clairvoyant scheduling problem on $m \geq 2$ parallel machines. Other related work is concerned with the issues of using preemption without allowing job migration [2] and with related objective functions like the stretch metric [1, 6, 9, 4].

Our work. We prove that RMLF achieves an $O(\log n \log \frac{n}{m})$ and an $O(\log n \log P)$ randomized competitive ratio against the oblivious adversary for any number m of parallel identical machines. We recall that P is the ratio between the largest and the shortest processing time of a job. Observe that one of the two bounds may be strictly better than the other depending on the values of P, n and m. We do not contradict the $\Omega(P)$ randomized lower bound claimed in [12] since it is obtained with a number of jobs exponential in P. For a single machine we show an $O(\log n)$ bound, which is tight. Our algorithm requires the knowledge of the minimum processing time of a job that is released. This knowledge is essential for an MLF-like algorithm to establish how long to execute a job when it enters the system in the lowest queue. A wrong estimation of the minimum processing time of a job may leave many small jobs wait for a time much bigger than their processing time. This argument can be extended to show that in absence of this information it is not possible to be competitive at all. However, we do not need any knowledge of the maximum processing time of a job that is released.

Our analysis of RMLF contains a set of new tools of analysis. We mentioned in the introduction that RMLF tries to approximately follow SRPT with the goal of avoiding the preemption of jobs that are nearly finished. We prove that (Lemmas 6 and 7) every job, apart from a logarithmic number, with constant probability, has a remaining processing time that is still a large share of the initial processing time when it enters the queue in which it will be completed. This allows, roughly speaking, to concentrate only on those jobs that are called "big". We bound at any time t the difference between the number of big jobs of almost equal size that are uncompleted in RMLF and in the optimum, by limiting the difference in the volume of these jobs, that is their overall remaining processing time. We then prove (Lemma 12) that at any time t, the expected number of jobs not completed by RMLF is at most $O(\log n)$ times a constant fraction of the jobs uncompleted by the adversary plus an additive term $O(\log n(m\ k(t))$, where $k(t)$ is related to the number of non-empty queues at time t in RMLF. The $O(\log n \log P)$ result for parallel machines then follows since there are at most $O(\log P)$ queues in the system.

The proof of $O(\log n \log \frac{n}{m})$ competitiveness for parallel machines and of $O(\log n)$ competitiveness for a single machine requires to bound in different terms the additional flow time collected over time due to the $O(\log n(mk(t))$ contribution. This contribution is partly originated by the fact that on $m \geq 2$ machines RMLF may keep machines idle for more than the optimum. We will bound this contribution (section 4.3) by at most $O(\log n \log \frac{n}{m})$ times the optimum for $m \geq 2$ machines while for a single machine we will be able to limit this contribution to $O(\log n)$ times the optimum. The two results are however obtained as consequences of a unified analysis of RMLF.

Structure of the paper. In Section 2 we formally define our scheduling problem. Section 3 presents the algorithm RMLF. In Section 4 we provide the analysis of the behaviour of RMLF. In particular, in Subsection 4.1 we present some tools and preliminary results. Subsection 4.2 is about the $O(\log n \log P)$ competitive ratio, while Subsection 4.3 about the $O(\log n \log(n/m))$ competitive ratio of RMLF and the $O(\log n)$ upper bound for the single machine case. Open problems are discussed in Section 5.

2 The Problem

We are given a set J of n jobs and a set of m identical machines. Each job j is assigned with a pair (r_j, p_j) where r_j is the release time of the job and p_j is its processing time. Order the jobs by increasing release time and assume the first job released at time $r_1 = 0$. In the preemptive model a job that is running can be preempted and continued later on any machine. The algorithm decides which of the uncompleted jobs should be executed at each time. A job cannot be processed before its release time. At any time a job can be processed by a single machine, while a machine can process only a single job. For a given schedule define C_j to be the completion time of job j in that schedule. The flow time of job j is $F_j = C_j - r_j$. The total flow time for the input instance J is $F(J) = \sum_{j \in J} F_j$. The goal of the scheduling algorithm is to minimize the total flow time for each given instance of the problem, or equivalently the average flow time.

The arrival of a job is unknown to the *nonclairvoyant* algorithm until the release time of the job. The processing time of a job is unknown to the nonclairvoyant algorithm until the completion time of the job. We assume the minimum processing time of a job that is released known in advance to the algorithm, and the convention $\min_{j \in J} p_j = 1$, while we do not need any knowledge of the maximum processing time of a job.

We compare the randomized nonclairvoyant algorithm with an oblivious adversary that decides in advance, without knowledge of the random choices of the algorithm, the number of jobs n of the sequence together with the release time and the processing time of each job. The adversary is charged with the optimal flow time for the sequence. Denote by A the randomized on-line algorithm, and denote by OPT the optimal adversary. The generic (deterministic) execution of A over an input sequence J is defined by a set of random choices σ. We denote by $\Pr(\sigma)$ the probability of sequence σ. A randomized on-line algorithm A is c-competitive against an oblivious adversary if for any input J,

$$\mathrm{E}_\sigma[\mathrm{F}^{\mathrm{A}}_\sigma(\mathrm{J})] \leq \mathrm{c}\mathrm{F}^{\mathrm{OPT}}(\mathrm{J}),$$

where the expectation is taken over any possible outcome of the random choices σ of the algorithm for the input instance J. The instance J will be omitted in the following when clear from the context.

3 The RMLF Algorithm

We present our randomized version of the Multi-level Feedback algorithm for parallel machines, called RMLF in the following. A job is said *active* or *alive* at time t if released but not completed. Denote by $x_j(t)$ the time job j has been processed until time t. Denote by $y_j(t) = p_j - x_j(t)$ the remaining processing time of job j at time t. $\beta_{j,i}$ is a random variable with distribution $\Pr[\beta_{j,i} \leq x] = 1 - e^{-\gamma \frac{x}{2^i} \ln j}$. In our algorithm we choose $\gamma = 4/3$.

Active jobs are partitioned into a set of *Priority Queues* $Q_0, Q_1, \ldots$.. Within each queue, the priority is determined by the Earliest Release Time rule. For any two queues Q_i, Q_k, Q_i is said lower than Q_k if $i < k$. Job j is assigned with a target processing time $T_{j,i}$ when it enters queue Q_i. At any time t, RMLF behaves as follows:

1. Schedule alive jobs on the m machines proceeding from the lowest to the highest queue, within any queue in order of priority.
2. Job j released at time t enters queue Q_0 with $T_{j,0} = \max\{1, 2 - \beta_{j,0}\}$.
3. For a job j in a queue Q_{i-1} at time t, if $x^A_j(t) = T_{j,i-1} < p_j$, job j enters Q_i with $T_{j,i} = \max\{2^i, 2^{i+1} - \beta_{j,i}\}$.
4. For a job j in a queue Q_i at time t, if $x^A_j(t) = p_j \leq T_{j,i}$, assign $C^A_j(t) = t$ and discard j from Q_i.

The Earliest Release Time rule assigns a static priority to jobs, rather than following the dynamic priority defined by the time of arrival in a queue. The reason of this choice is to eliminate dependencies between the relative priority between jobs and the specific execution of the algorithm. We will comment more on this point in the analysis of the algorithm.

A very natural question at this point of the exposition is: why the exponential distribution rather than the uniform distribution [20]? We will sketch an example of why a uniform distribution of target $T_{j,i} \in [2^i, 2^{i+1})$ in queue i does not work.

Consider a single machine and n jobs released at time 0 with processing time $2^i(1 + \frac{2}{\sqrt{n}})$, with $n = 2^i$. With probability at least $\frac{1}{\sqrt{n}}$ every job will be completed in queue Q_{i+1} with a remaining processing time of at least $\sqrt{n}$. At some time $t = n2^i(1+\frac{2}{\sqrt{n}}) - cn$ for a constant c, the algorithm will have with high probability $O(\sqrt{n})$ uncompleted jobs in queue Q_{i+1}, while the optimal solution will have terminated all jobs with the exception of a constant number for a total flow time of $O(n^3)$. Starting a time t, release a job of size 1 for each of n^3 time units. The expected flow time of the algorithm will be of $\Omega(n^3\sqrt{n})$, while the optimum is $O(n^3)$.

4 The Analysis

In the analysis we use the following notation. We denote by $\delta(t)$ the number of active jobs at time t. The volume $V(t)$ at time t is the sum of the remaining processing times of the active jobs at time t. $L(t)$ denotes the total work done prior to time t, that is the overall time the m machines have been processing jobs until time t. For a generic function f (δ, V and L), we denote by $f^A_\sigma(t)$ and $f^{OPT}(t)$ the value of function f at time t respectively for a generic outcome σ of the random choices of RMLF and for the optimal solution OPT. By $\Delta_\sigma f(t) = f^A_\sigma(t) - f^{OPT}(t)$ we denote their difference.

In the analysis, w.l.o.g we restrict to sequences where $r_1 = 0$ and $r_j \leq \sum_{i=1}^{j-1} p_i$, $j > 1$. Every input instance can be split into a set of input instances of this form that we separately analyze.

In our analysis, we classify active jobs in classes according to their processing times. A job j is of *class* i, $i \geq 0$, if its processing time is in $[2^i, 2^{i+1})$. For a generic function f (V, ΔV, δ, $\Delta\ \delta$, L or ΔL) the notation $f_{=k}(t)$ will denote the value of function f at time t when restricted to jobs of class exactly k; we use $f_{\geq h, \leq k}(t)$ to denote the value of f at time t when restricted to jobs of classes between h and k.

Denote by $P = \frac{\max_{j\in J} p_j}{\min_{j\in J} p_j}$ the ratio between the maximum and the minimum processing time of a job. The maximum class of a job is $\lfloor \log P \rfloor + 1$. Moreover:

Fact 1 *During the execution of the algorithm at most $\lceil \log P \rceil$ queues are created.*

We start by observing that the total flow time is the integral over time of the number of active jobs (for example, see [16]):

Fact 2 $F = \sum_{j\in J} F_j = \int_t \delta(t)dt$.

Moreover we observe the following basic lower bound over the total flow time:

Fact 3 $F = \sum_{j\in J} F_j \geq \sum_{j\in J} p_j$.

The following fact holds for the algorithm:

Fact 4 *A job j of class i ends either in queue Q_i or in queue Q_{i+1}.*

Proof. Job j with $p_j \in [2^i, 2^{i+1})$ is processed for less than 2^i in Q_{i-1} while it is processed for at least 2^{i+1} in queue Q_{i+1}. □

Definition 1 *Job j of class i is* unlucky *in a given execution of the algorithm if it has processing time $p_j \leq 2^i + 2^{i-2}$ and it ends in queue Q_{i+1}. A job that is not unlucky is said lucky.*

We denote by $\delta^l_\sigma(t)$ and by $\delta^u_\sigma(t)$ respectively the number of lucky and unlucky jobs that are active at time t for a specific execution σ of RMLF.

Definition 2 *At any time t, a lucky job j in queue Q_i is said big if it entered Q_i at time $\bar{t} \leq t$ and $y_j(\bar{t}) \geq \frac{p_j}{\ln j}$, it is said* small *otherwise.*

We denote by $\delta^b_\sigma(t)$ the number of big (lucky) jobs that are alive at time t for a specific outcome σ of the random choices of RMLF.

The following gives a basic property of the RMLF algorithm:

Fact 5 *At any time t and for any queue Q_i, at most m jobs in Q_i have been executed but have neither been completed nor promoted to Q_{i+1}.*

Proof. Let $j_1, j_2, ...$ be the jobs in Q_i at some time t considered in order of priority. Let k be the largest integer such that j_k has received some processing time in Q_i. Consider the last time $t' \leq t$ such that, at time t', j_k was processed in Q_i. At time t' every job j_h, $h < k$, is also executed as well. Two cases are possible: (i.) if j_h was in queue Q_i at time t', it would be on execution since it precedes job j_k in the order; (ii.) if j_h was at time t' in a queue Q_l, $l < j$, it would also be on execution since a job in Q_i, i.e. j_k, was executed at that time. The claim of the Lemma then follows, since at most m jobs can be on execution at each time. □

We first present the probabilistic statements we use in the analysis. Then we give the analysis of the competitive ratio for RMLF.

4.1 Probabilistic analysis

We start by showing that a job of class i with processing time close to 2^i will be completed with high probability in queue i, therefore overcoming the counterexample for the uniform distribution described in Section 3. This is also shown in Figure 4.1 a), where one can observe that the area below the probability density of the exponential distribution in the interval $[2^i, 2^i + 2^{i-2})$ is bounded by $1/j$.

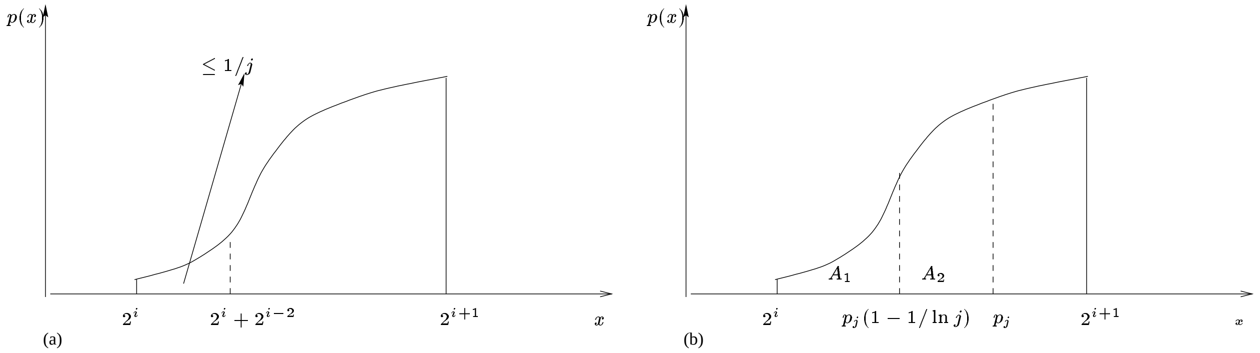

Figure 1: Function $p(x) = d\Pr[T_{j,i} \leq x]/dx$ is the probability density of $T_{j,i}$. a) The probability that a job j of class i is unlucky is at most $1/j$. b) Areas A_1 and A_2 represent the probabilities that a job is respectively big or small. They differ for at most a constant factor.

We therefore obtain the following Lemma:

Lemma 6 *The expected number of unlucky jobs along the execution of the algorithm is H_n.*

Proof. Assume job j has class i. We show that if $p_j \leq 2^i + 2^{i-2}$, then the job enters queue Q_{i+1} with probability at most $1/j$. This follows from:

$$\begin{aligned}\Pr[X_j^l = 0] &= \Pr[T_{j,i} < p_j] \le \Pr[T_{j,i} < 2^i + 2^{i-2}] \\ &= \Pr[\beta_{j,i} > 32^{i-2}] = e^{-\ln j} = \frac{1}{j}\end{aligned}$$

since $\gamma = 4/3$. The expected number of unlucky jobs is then bounded by $\sum_{j=1}^{n} \frac{1}{j} = H_n$. $\square$

In the remaining part of this section we present a second fundamental property of the exponential probability distribution, that is if a job is lucky and alive at a time t when all machines are busy then with constant probability the job is big at time t. This property will be used later to reduce the analysis to limiting the difference between the number of big jobs of RMLF and the overall numer of jobs of the optimum.

This statement is easily proved if a job j is at time t in a queue Q_k, $k < i$, while it takes some work in case job j is in queue Q_i or Q_{i+1}. Assume job j in queue Q_{i+1}, the fact that job j is big when it enters queue Q_{i+1} depends on the random choice for $T_{j,i}$. In Figure 4.1 b) it is shown that for a lucky job, i.e. a job with $p_j \ge 2^i + 2^{i-2}$, the area below the probability density in the interval $[2^i, p_j(1-\frac{1}{j}))$ is related to the area below the probability density in the interval $[p_j(1-\frac{1}{j}), p_j)$ by a constant factor. This is the only interval we are interested since job j is not completed in queue Q_i.

However, we are interested in the probability that a job is big conditioned to the fact that the job j is lucky and alive at time t and all the machines are busy. The favorable cases are clearly those for which $T_{j,i}$ is small, leaving a large processing time to be executed in queue Q_{i+1}. We therefore need to show that the whole favorable part of the probability space is actually available. This is proved by showing that if for a given execution σ with an assignment $T_{j,i} = T$ job j is lucky and alive at time t when all machines are idle, so it is for any other execution that differs from σ only for variable $T_{j,i}$ that is assigned to a value $T' < T$.

In proving the statement above it is required to argue about the time a job is processed in different executions of the algorithm. Having a static definition of the priority of a job, independently from the specific execution, allows to state that as long as $T_{j,i}$ decreases job j will not be processed up to completion by time t. An analogous argument is required for the case of j in queue Q_i. These arguments lead to the following statement:

Lemma 7 *There exists a constant α such that, for every job $j \ge \alpha$ at time t it holds* $\Pr[X_j^b(t) = 1 | \delta^A(t) \ge m] \ge e^{-4\gamma} \Pr[X_j^l(t) = 1 | \delta^A(t) \ge m]$.

Proof. The following Lemma is used in the proof of the Lemma.

Lemma 8 *Consider a job j of class i and a specific assignment $\mathcal{X}$ of all random variables with the exception of $T_{j,i-1}$ ($T_{j,i}$). Consider the set $\Sigma_j^i(t)$ of all sequences with assignment $\mathcal{X}$ ($\Sigma_j^{i+1}(t)$) where job j is in queue Q_i (Q_{i+1}) at time t, such that $X_{j,\sigma}^l(t) = 1$ and $\delta_\sigma^A(t) \ge m$. Then, there exists a value $x \in [2^{i-1}, 2^i)$ ($x \in [2^i, 2^{i+1})$) such that for every $T_{j,i-1} \in [2^{i-1}, x)$ ($T_{j,i} \in [2^i, x)$) the corresponding sequence is in $\Sigma_j^i(t)$ ($\sigma \in \Sigma_j^{i+1}(t)$), while it is not in $\Sigma_j^i(t)$ for every $T_{j,i-1} \in [x, 2^i)$ ($T_{j,i} \in [x, 2^{i+1})$).*

Proof. Consider a set of sequences $\Sigma_j^i(t)$. Consider two possible values T and $T' > T$ for $T_{j,i-1}$ and let σ and σ' be the two corresponding sequences. The claim follows if we show that $\sigma' \in \Sigma_j^i(t)$ implies $\sigma \in \Sigma_j^i(t)$.

Let t' denote the time at which j is processed for T units in queue Q_{i-1} according to σ. The schedules for σ and σ' are identical up to time t'. We start by proving that for every $\hat{t} \in (t', t]$ and for every $h \neq j$, either j is completed or $y_{h,\sigma'}(\hat{t}) \ge y_{h,\sigma}(\hat{t})$.

The proof is by contradiction. Let $t' < \bar{t} < t$ be the last time when $y_{h,\sigma'}(\bar{t}) \geq y_{h,\sigma}(\bar{t})$ for all $h \neq j$. There is a job $h \neq j$ such that $y_{h,\sigma'}(\bar{t}) = y_{h,\sigma}(\bar{t})$ while $y_{h,\sigma'}(\bar{t}+\epsilon) < y_{h,\sigma}(\bar{t}+\epsilon)$ for every ϵ smaller than a given value. The contradiction follows by proving that the set of jobs that have priority over j in σ' at time $\bar{t}$ contains the set of jobs that have priority over j in σ at time $\bar{t}$. Recall that a job j' has priority over a job j'' if j' is in a lower queue than j'' or if they are in the same queue but j' has been released earlier.

We will distinguish between j and all jobs different from j and h. According to execution σ, job j is at time $\bar{t}$ in queue Q_i, while at the same time it is either in queue Q_{i-1} or Q_i in execution σ'. On the other side, since $y_{h,\sigma'}(\bar{t}) = y_{h,\sigma}(\bar{t})$, at $\bar{t}$ h is in the same queue in the two executions. Therefore, if j has priority over h in σ, so it does in σ'. For all jobs other than h and j, we observe that they have in σ' at time $\bar{t}$ a remaining processing time bigger or equal than in σ. Therefore they are in the same queue or in a lower queue, for which they do not lose their priority on h moving from execution σ to execution σ'.

As a consequence we have that (Fact a.) $\delta^A_{\sigma'}(\hat{t}) \geq \delta^A_{\sigma}(\hat{t})$ for any $\hat{t} \geq t'$ such that job j is not completed yet in sequence σ', since job j is not completed and all other jobs have in σ' a remaining processing time that is not less than in σ.

Consider the smallest value x for variable $T_{j,i-1}$ such that for the corresponding execution σ we have $X^l_{j,\sigma}(t) = 0$. From Fact a. we immediately derive that (i.) For every $T < x$, the sequence σ with $T_{j,i-1} = T$ has $\delta^A_{\sigma}(t) \geq m$, $X^l_{j,\sigma}(t) = 1$ and therefore it is in $\Sigma^i_j(t)$. From Fact a. we also derive that the overall work done by any time $\hat{t} \geq t'$ by the machines in sequence σ' until job j is completed is not less than the work done in σ. Since all jobs other than j have a higher remaining processing time in σ', job j cannot have a higher remaining processing time in σ' thus proving (ii.) For every $T \geq x$, the sequence σ with $T_{j,i-1} = T$ has $X^l_{j,\sigma}(t) = 0$ and therefore it is not in $\Sigma^i_j(t)$; Statements (i.) and (ii.) together prove the claim of the Lemma. □

Consider a job $j \geq e^5$ and consider all sequences $\Sigma_j(t)$ for which $X^l_{j,\sigma}(t) = 1$ and $\delta^A_{\sigma}(t) \geq m$. For all sequences in which j is in queue Q_k, $k \leq i-1$, j is big, since $T_{j,i-2} < 2^{i-1}$ and then $y_j(t) > p_j - 2^{i-1} \geq p_j/2 \geq p_j/\ln j$, for $j \geq 8$. As a consequence, the only possibility for a lucky job of class i to be small is to enter queue Q_i or queue Q_{i+1} at time $\bar{t} \leq t$, with $y_j(\bar{t}) < \frac{p_j}{\ln j}$.

First consider any subset $\Sigma^i_j(t)$ of $\Sigma_j(t)$ in which job j is in queue Q_i at time t and all sequences in $\Sigma^i_j(t)$ differ only for $T_{j,i-1}$. Lemma 8 ensures that there exists a value x such that there exists a sequence $\sigma \in \Sigma^i_j(t)$ for every $T_{j,i-1} \in [2^{i-1}, x)$. If we prove a bound on the probability that job j is big when conditioned to sequences $\sigma \in \Sigma^i_j(t)$ for all random variables with the exception of $T_{j,i-1}$, then the claim is proved for j in Q_i at time t. We proceed as follows:

$$\begin{aligned}
\Pr[T_{j,i-1} \leq p_j - \frac{p_j}{\ln j} | \sigma \in \Sigma^i_j(t)] &= \Pr[T_{j,i-1} \leq p_j(1 - \frac{1}{\ln j}) | T_{j,i-1} \leq x] \\
&\geq \Pr[T_{j,i-1} \leq p_j(1 - \frac{1}{\ln j}) | T_{j,i-1} < 2^i] \\
&\geq \frac{\Pr[T_{j,i-1} \leq p_j(1 - \frac{1}{\ln j})]}{\Pr[T_{j,i-1} < 2^i]} \\
&\geq \Pr[\beta_{j,i-1} \geq 2^i - p_j(1 - \frac{1}{\ln j})] \\
&\geq e^{-\gamma \frac{2^i - p_j(1 - \frac{1}{\ln j})}{2^{i-1}} \ln j} \\
&= e^{-\gamma \frac{2^i - p_j}{2^{i-1}} \ln j - \gamma \frac{p_j}{2^{i-1}}}
\end{aligned}$$

$$\geq \quad e^{-\gamma\frac{p_j}{2^{i-1}}} \geq e^{-4\gamma},$$

where the second inequality follows since $x < 2^i$, while the last inequality from $p_j \leq 2^{i+1}$.

We then prove the claim in case job j is in queue Q_{i+1} at time t. Since job j is lucky, $p_j > 2^i + 2^{i-2}$. We show that job j is big when it enters Q_{i+1} with probability at least $e^{-4\gamma}$. Consider any subset $\Sigma_j^{i+1}(t)$ of $\Sigma_j(t)$ in which job j is in queue Q_{i+1} at time t and all sequences in $\Sigma_j^{i+1}(t)$ differ only for $T_{j,i}$. Lemma 8 ensures that there exists a value x such that there exists a sequence $\sigma \in \Sigma_j^{i+1}(t)$ for every $T_{j,i} \in [2^i, x)$ and no such sequence for $T_{j,i} \geq x$. If we prove a bound on the probability that job j is big when conditioned to sequences $\sigma \in \Sigma_j^{i+1}(t)$ for all random variables with the exception of $T_{j,i}$, then the claim is proved for j in Q_{i+1} at time t. We proceed as follows:

$$\begin{aligned}
\Pr[T_{j,i} \leq p_j - \frac{p_j}{\ln j} | \sigma \in \Sigma_j^{i+1}(t)] &= \Pr[T_{j,i} \leq p_j(1 - \frac{1}{\ln j}) | T_{j,i} < x] \\
&\geq \Pr[T_{j,i} \leq p_j(1 - \frac{1}{\ln j}) | T_{j,i} < p_j] \\
&= \frac{\Pr[T_{j,i} \leq p_j(1 - \frac{1}{\ln j})]}{\Pr[T_{j,i} < p_j]} \\
&= \frac{\Pr[\beta_{j,i} \geq 2^{i+1} - p_j(1 - \frac{1}{\ln j})]}{\Pr[\beta_{j,i} > 2^{i+1} - p_j]} \\
&\geq \frac{e^{-\gamma\frac{2^{i+1} - p_j(1-\frac{1}{\ln j})}{2^i}\ln j}}{e^{-\gamma\frac{2^{i+1}-p_j}{2^i}\ln j}} \\
&= \frac{e^{-\gamma\frac{2^{i+1}-p_j}{2^i}\ln j - \gamma\frac{p_j}{2^i}}}{e^{-\gamma\frac{2^{i+1}-p_j}{2^i}\ln j}} \\
&\geq e^{-\gamma\frac{p_j}{2^i}} \geq e^{-2\gamma}.
\end{aligned}$$

Here, the fourth inequality holds since $T_{j,i,} = \max\{2^i, 2^{i+1} - \beta_{j,i}\}$, since $p_j \geq 2^i + 2^{i-2}$ and since $p_j(1 - 1/\ln j) \geq 2^i$ when $j \geq e^5$. The last inequality follows since $p_j \leq 2^{i+1}$. Taking $\alpha = \lceil e^5 \rceil$ the claim of the Lemma is then proved. □

4.2 The $O(\log n \log P)$-competitive ratio

In this section we prove that $RMLF$ is $O(\log n \log P)$ competitive. The total flow time of $RMLF$ can be expressed as:

$$\begin{aligned}
\mathrm{E}_\sigma[\mathrm{F}_\sigma^{\mathrm{A}}] &= \mathrm{E}_\sigma\left[\int_{\mathrm{t}\geq 0} \delta_\sigma^{\mathrm{A}}(\mathrm{t})\mathrm{dt}\right] = \mathrm{E}_\sigma\left[\int_{\mathrm{t}:\delta_\sigma^{\mathrm{A}}(\mathrm{t})<\mathrm{m}} \delta_\sigma^{\mathrm{A}}(\mathrm{t})\mathrm{dt}\right] + \mathrm{E}_\sigma\left[\int_{\mathrm{t}:\delta_\sigma^{\mathrm{A}}(\mathrm{t})\geq\mathrm{m}} \delta_\sigma^{\mathrm{A}}(\mathrm{t})\mathrm{dt}\right] \\
&= \mathrm{E}_\sigma\left[\int_{\mathrm{t}:\delta_\sigma^{\mathrm{A}}(\mathrm{t})<\mathrm{m}} \delta_\sigma^{\mathrm{A}}(\mathrm{t})\mathrm{dt}\right] + \int_{\mathrm{t}\geq 0} \mathrm{E}_\sigma[\delta_\sigma^{\mathrm{u}}(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}] \Pr[\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\mathrm{dt} \\
&\quad + \int_{t\geq 0} \mathrm{E}_\sigma[\delta_\sigma^{\mathrm{l}}(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}] \Pr[\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\mathrm{dt} \qquad (1)
\end{aligned}$$

The second equality follows by partitioning for every possible set σ of random choices of RMLF the interval $\{t \geq 0\}$ into the set of time instants where $\delta_\sigma^A(t) < m$ and the set of time instants where

$\delta^A_\sigma(t) \geq m$. The third equality follows by the linearity of expectation and by partitioning the set of alive jobs into the set of lucky and unlucky jobs.

In the remainder of this section we need to bound the three contributions to equation (1). The first contribution is bounded since in any execution of the algorithm we restrict to those time instants where there are at most m jobs in the system. The following Lemma, based on the claim of Lemma 6 will help us in bounding the second contribution to the total flow time.

Lemma 9 *At any time t it holds* $\mathrm{E}_\sigma[\delta^{\mathrm{u}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\Pr[\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}] = \mathrm{O}(\log \mathrm{n})$.

Proof. The claim of the Lemma follows from:

$$\begin{aligned}
\mathrm{E}_\sigma[\delta^{\mathrm{u}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\Pr[\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}] &= E_\sigma[\delta^u_\sigma(t)] - \mathrm{E}_\sigma[\delta^{\mathrm{u}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) < \mathrm{m}]\Pr[\delta^{\mathrm{A}}(\mathrm{t}) < \mathrm{m}] \\
&\leq \sum_{j=1}^{n} \Pr[X^l_j = 0 \cap X_j(t) = 1] \leq \sum_{j=1}^{n} \Pr[X^l_j = 0] \\
&= O(\log n),
\end{aligned}$$

where the last inequality follows from Lemma 9. □

The next Lemma that uses the Claim of Lemma 7 will relate the expected number of lucky jobs alive at time t to the expected number of big jobs alive at time time t.

Lemma 10 *At any time t, the number of lucky jobs in the system satisfies the following relation:*

$$\mathrm{E}_\sigma[\delta^{\mathrm{l}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}] \leq \alpha + \mathrm{e}^{4\gamma}\mathrm{E}_\sigma[\delta^{\mathrm{b}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}].$$

Proof. Set $\alpha = \lceil e^5 \rceil$. We have the following equations:

$$\begin{aligned}
\mathrm{E}_\sigma[\delta^{\mathrm{l}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}] &= \sum_{j=1}^{n} \Pr[X^l_j(t) = 1|\delta^A(t) \geq m] \\
&\leq \alpha + \sum_{j=\alpha+1}^{n} \Pr[X^l_j(t) = 1|\delta^A(t) \geq m] \\
&\leq \alpha + e^{4\gamma} \sum_{j=\alpha+1}^{n} \Pr[X^l_b(t) = 1|\delta^A(t) \geq m] \\
&\leq \alpha + e^{4\gamma}\mathrm{E}_\sigma[\delta^{\mathrm{b}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}],
\end{aligned}$$

where the third inequality follows from the Claim of Lemma 7. □

The next Lemma bounds the expected time spent by the algorithm while all machines are busy.

Lemma 11 $\int_{t\geq 0} \Pr[\delta^A(t) \geq m]\ dt \leq \frac{1}{m}\sum_{j\in J} p_j$.

Proof. The claim is derived from the following relations:

$$\begin{aligned}
\int_{t\geq 0} \Pr[\delta^A(t) \geq m]\ dt &= \int_{t\geq 0} \sum_{\sigma:\delta^A_\sigma(t)\geq m} \Pr[\sigma]\ dt \\
&= \sum_{\sigma} \Pr[\sigma] \int_{t:\delta^A_\sigma(t)\geq m} dt
\end{aligned}$$

$$\leq \frac{1}{m}\sum_{j\in J} p_j \sum_{\sigma} \Pr[\sigma]$$
$$= \frac{1}{m}\sum_{j\in J} p_j,$$

where the third inequality follows since for every σ $\int_{t:\delta^A_\sigma(t)\geq m} dt$ is bounded by $\frac{1}{m}\sum_{j\in J} p_j$. $\square$

The expected cost of RMLF of expression (1) is then rewritten as follows:

$$\begin{aligned}
\mathrm{E}_\sigma[\mathrm{F}^{\mathrm{A}}_\sigma] &= \mathrm{E}_\sigma\left[\int_{\mathrm{t}:\delta^{\mathrm{A}}_\sigma(\mathrm{t})<\mathrm{m}} \delta^{\mathrm{A}}_\sigma(\mathrm{t})\mathrm{dt}\right] + \int_{\mathrm{t}\geq 0} \mathrm{E}_\sigma[\delta^{\mathrm{u}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t})\geq \mathrm{m}]\Pr[\delta^{\mathrm{A}}(\mathrm{t})\geq \mathrm{m}]\mathrm{dt} \\
&\quad + \int_{t\geq 0} \mathrm{E}_\sigma[\delta^{\mathrm{l}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t})\geq \mathrm{m}]\Pr[\delta^{\mathrm{A}}(\mathrm{t})\geq \mathrm{m}]\mathrm{dt} \\
&= \sum_{j\in J} p_j + O(\log n)\sum_j p_j + \int_{t\geq 0} (\alpha + e^{4\gamma}\mathrm{E}_\sigma[\delta^{\mathrm{b}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t})\geq \mathrm{m}])\Pr[\delta^{\mathrm{A}}(\mathrm{t})\geq \mathrm{m}]\mathrm{dt} \qquad (2)
\end{aligned}$$

The inequality follows by observing that (i.) for every set of random choices σ, $\int_{t:\delta^A_\sigma(t)<m} \delta^A_\sigma(t)dt \leq \sum_{j\in J} p_j$, since the overall work done by the m machines in any possible execution of the algorithm is bounded by the sum of the processing times of the n jobs; (ii.) by applying the claim of Lemma 9 that bounds the expected number of unlucky jobs in the whole sequence at any time t and by observing that $\delta^u_\sigma(t) = 0$ for every $t \geq \sum_j p_j$; (iii.) by applying the claim of Lemma 10.

We finally present the critical part of the proof in which we bound the number of big jobs at time t in the generic execution of RMLF determined by a set of random choices of the algorithm.

Lemma 12 *For any outcome σ of the random choices of the algorithm, for any time $t : \delta^A_\sigma(t) \geq m$, $\delta^b_{\sigma,\geq k_1,\leq k_2}(t) \leq 2\ln n\left(2(m-1)(k_2-k_1+1) + 3\delta^{OPT}(t)\right) + m(k_2-k_1+2)$.*

Proof. In the following we omit σ when clear from the context. For $\delta^b_{\geq k_1,\leq k_2}(t)$ we write the following relations:

$$\delta^b_{\geq k_1,\leq k_2}(t) \leq m(k_2-k_1+2) + \ln n \sum_{i=k_1}^{k_2} \frac{V^A_{=i}(t)}{2^i}. \qquad (3)$$

The bound follows since every big job of class i has remaining processing time at least $2^i/\ln n$. We should additionally consider those big jobs of class i, $i = k_1, .., k_2$, with a processing time smaller than $2^i/\ln n$ since they have been processed after entering the queue they are in. For a job of class i this may happen in queues $i-1$, i or $i+1$. However, for every queue there are at most m jobs that have been processed after entering the queue (Fact 5), for which the total number of such jobs is bounded by $m(k_2-k_1+2)$. We continue with:

$$\begin{aligned}
\sum_{i=k_1}^{k_2} \frac{V^A_{=i}(t)}{2^i} &= \sum_{i=k_1}^{k_2} \frac{V^{OPT}_{=i}(t) + \Delta V_{=i}(t)}{2^i} \\
&\leq 2\delta^{OPT}_{\geq k_1,\leq k_2}(t) + \sum_{i=k_1}^{k_2} \frac{\Delta V_{=i}(t)}{2^i} = 2\delta^{OPT}_{\geq k_1,\leq k_2}(t) + \sum_{i=k_1}^{k_2} \frac{\Delta V_{\leq i}(t) - \Delta V_{\leq i-1}(t)}{2^i}
\end{aligned}$$

$$
\begin{aligned}
&= 2\delta^{OPT}_{\geq k_1, \leq k_2}(t) + \frac{\Delta V_{\leq k_2}(t)}{2^{k_2}} + \sum_{i=k_1}^{k_2-1} \frac{\Delta V_{\leq i}(t)}{2^{i+1}} - \frac{\Delta V_{\leq k_1-1}(t)}{2^{k_1}} \\
&\leq 2\delta^{OPT}_{\geq k_1, \leq k_2}(t) + \delta^{OPT}_{\leq k_1-1}(t) + 2\sum_{i=k_1}^{k_2} \frac{\Delta V_{\leq i}(t)}{2^{i+1}} \\
&\leq 2\delta^{OPT}_{\leq k_2}(t) + 2\sum_{i=k_1}^{k_2} \frac{\Delta V_{\leq i}(t)}{2^{i+1}}, \qquad (4)
\end{aligned}
$$

where the second inequality follows since a job of class i has size less than 2^{i+1}, while the fifth inequality follows since $-\frac{\Delta V_{\leq k_1-1}(t)}{2^{k_1}} \leq \frac{V^{OPT}_{\leq k_1-1}}{2^{k_1}} \leq \delta^{OPT}_{\leq k_1-1}$.

We are left to study the term $\sum_{i=k_1}^{k_2} \frac{\Delta V_{\leq i}(t)}{2^{i+1}}$. For any $t_1 \leq t_2 \leq t$, for a generic function f, denote by $f^{[t_1,t_2]}(t)$ the value of function f at time t when restricted to jobs released between t_1 and t_2, e.g. $L^{[t_1,t_2]}_{\leq i}(t)$ is the work done by time t on jobs of class at most i released between time t_1 and t_2. Let $t_0(t)$ be the last time prior to time t in the execution determined by the set of random choices σ, such that $\delta^A_\sigma(t_0) < m$. Denote by $t_i < t$ the maximum between $t_0(t)$ and the last time prior to time t in which a job was processed in queue Q_{i+1} or higher in this specific execution of RMLF. Observe that, for $i = k_1, \ldots, k_2$, $[t_{i+1}, t) \supseteq [t_i, t)$.

At time t_{i+1} either the algorithm was processing a job in queue Q_{i+2} or higher, or $t_{i+1} = t_0(t)$, for which at time t_{i+1} at most $m-1$ jobs were in queues $Q_0, \ldots, Q_{i+1}$. At time t we have:

$$
\begin{aligned}
\Delta V_{\leq i}(t) &\leq V^{A[0,t_{i+1}]}_{\leq i}(t) + \Delta V^{(t_{i+1},t]}_{\leq i}(t) \\
&\leq V^{A[0,t_{i+1}]}_{\leq i}(t) + V^{A[0,t_{i+1}]}_{> i}(t) - V^{A[0,t_{i+1}]}_{> i}(t_{i+1}) + L^{A(t_{i+1},t]}_{> i}(t) - L^{OPT(t_{i+1},t]}_{> i}(t) \\
&\leq (m-1)2^{i+2} + L^{A(t_{i+1},t]}_{> i}(t) - L^{OPT(t_{i+1},t]}_{> i}(t).
\end{aligned}
$$

The first inequality follows since $\Delta V_{\leq i}(t)$ is bounded by the sum of a first contribution equal to the volume of the jobs released until time t_{i+1} in RMLF's schedule and a contribution equal to the volume difference of jobs released after time t_{i+1}. The second inequality states that the volume difference is bounded by the time spent in the interval $(t_{i+1}, t]$ by RMLF processing jobs of class bigger than i minus the time spent by the optimal solution processing jobs of class bigger than i released in the interval $(t_{i+1}, t]$. The third inequality follows from $V^{A,[0,t_{i+1}]}_{\leq i}(t) + V^{A,[0,t_{i+1}]}_{> i}(t) - V^{A,[0,t_{i+1}]}_{> i}(t_{i+1}) \leq (m-1)2^{i+2}$. This relation holds since there are at most $m-1$ jobs in queues $Q_0, .., Q_{i+1}$ at time t_{i+1}. Assume there are x_1 such jobs of class $\leq i$ and x_2 such jobs of class $> i$. The remaining volume of the x_1 jobs of class $\leq i$ at time t_{i+1} is at most $x_1 2^{i+2}$, while the x_2 jobs of class $> i$ are processed in $[t_{i+1}, t)$ for at most $x_2 2^{i+2}$ time units. From $x_1 + x_2 \leq m-1$, the equation follows.

The fact that at most $m-1$ jobs are in the system in queues $Q_0, ..., Q_{i+1}$ at time t_{i+1} is crucial in proving the tightness result for $m = 1$. In fact for $m = 1$, this contribution to the flow time will disappear.

In the following we adopt the convention $t_{k_1-1} = t$. From the inequality above we write:

$$
\begin{aligned}
\sum_{i=k_1}^{k_2} \frac{\Delta V_{\leq i}(t)}{2^{i+1}} &\leq \sum_{i=k_1}^{k_2} \frac{(m-1)2^{i+2} + \Delta L^{(t_{i+1},t]}_{> i}(t)}{2^{i+1}} \leq 2(m-1)(k_2 - k_1 + 1) \\
&\quad + \sum_{i=k_1}^{k_2} \frac{\Delta L^{(t_{i+1},t]}_{> i}(t)}{2^{i+1}} \qquad (5)
\end{aligned}
$$

We then concentrate on the term $\sum_{i=k_1}^{k_2} \frac{\Delta L_{>i}^{(t_{i+1},t]}(t)}{2^{i+1}}$ for which we have:

$$\begin{aligned}
\sum_{i=k_1}^{k_2} \frac{\Delta L_{>i}^{(t_{i+1},t]}(t)}{2^{i+1}} &= \sum_{i=k_1}^{k_2} \frac{L_{>i}^{A(t_{i+1},t]} - L_{>i}^{OPT(t_{i+1},t]}}{2^{i+1}} = \sum_{i=k_1}^{k_2} \sum_{j=k_1-1}^{i} \frac{L_{>i}^{A(t_{j+1},t_j]} - L_{>i}^{OPT(t_{j+1},t_j]}}{2^{i+1}} \\
&\leq \sum_{j=k_1-1}^{k_2} \sum_{i=j}^{k_2} \frac{L_{>i}^{A(t_{j+1},t_j]} - L_{>i}^{OPT(t_{j+1},t_j]}}{2^{i+1}},
\end{aligned}$$

where the second equality follows by partitioning the work done on the jobs released in the interval $(t_{i+1}, t]$ into the work done on the jobs released in the intervals $(t_{j+1}, t_j]$, $j = k_1 - 1, .., i$.

Let $\bar{i}(j) \in [j, .., k_2]$ be the index that maximizes $L_{>i}^{A(t_{j+1},t_j]} - L_{>i}^{OPT(t_{j+1},t_j]}$. We then have:

$$\begin{aligned}
\sum_{i=k_1}^{k_2} \frac{\Delta L_{>i}^{(t_{i+1},t]}(t)}{2^{i+1}} &\leq \sum_{j=k_1-1}^{k_2} \sum_{i=j}^{k_2} \frac{L_{>\bar{i}(j)}^{A(t_{j+1},t_j]} - L_{>\bar{i}(j)}^{OPT(t_{j+1},t_j]}}{2^{i+1}} \leq \sum_{j=k_1-1}^{k_2} \frac{L_{>\bar{i}(j)}^{A(t_{j+1},t_j]} - L_{>\bar{i}(j)}^{OPT(t_{j+1},t_j]}}{2^{j}} \\
&\leq 2 \sum_{j=k_1-1}^{k_2} \delta_{>\bar{i}(j)}^{OPT(t_{j+1},t_j]}(t) \leq 2\delta_{\geq k_1}^{OPT(t_{j+1},t]}(t) \\
&\leq 2\delta_{\geq k_1}^{OPT}(t). \qquad (6)
\end{aligned}$$

To prove the third inequality observe that every job of class bigger than $\bar{i}(j) \geq j$ released in the time interval $(t_{j+1}, t_j]$ is processed by RMLF in the interval $(t_{j+1}, t]$ for at most 2^{j+2} time units. Order the jobs of this specific set by increasing $x_j(t)$. Now, observe that each of these jobs has remaining processing time at least 2^{i+1} and we give to the optimum the further advantage that it finishes every such job when processed for an amount $x_j(t) \leq 2^{j+2}$. To maximize the number of finished jobs the optimum places the work $L_{>\bar{i}(j)}^{OPT(t_{j+1},t_j]}$ on the jobs with smaller $x_j(t)$. The optimum is then left at time t with a number of jobs $\delta_{>\bar{i}(j)}^{OPT(t_{j+1},t_j]} \geq \frac{L_{>\bar{i}(j)}^{A(t_{j+1},t_j]} - L_{>\bar{i}(j)}^{OPT(t_{j+1},t_j]}}{2^{j+1}}$, for which the third inequality holds.

Altogether, from (3), (4), (5) and (6) we obtain:

$$\begin{aligned}
\delta_{\geq k_1, \leq k_2}^{b}(t) &\leq m(k_2 - k_1 + 2) + 2\ln n(\delta_{\leq k_2}^{OPT}(t) + 2(m-1)(k_2 - k_1 + 1) + 2\delta_{\geq k_1}^{OPT}(t)) \\
&\leq m(k_2 - k_1 + 2) + 2\ln n\left(2(m-1)(k_2 - k_1 + 1) + 3\delta^{OPT}(t)\right),
\end{aligned}$$

for which the claim of the Lemma is proved. □

We then plug the claim of Lemma 12 in the expression (2) of the total flow time of RMLF to achieve our first result:

$$\begin{aligned}
\mathrm{E}_\sigma[\mathrm{F}_\sigma^{\mathrm{A}}] &= \sum_{j \in J} p_j + O(\log n) \sum_j p_j + \int_{t \geq 0} \left(\alpha + e^{4\gamma} \mathrm{E}_\sigma[\delta_\sigma^{\mathrm{b}}(\mathrm{t}) | \delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\right) \Pr[\delta^A(t) \geq m] dt \\
&\leq \sum_{j \in J} p_j(1 + O(\log n)) + \alpha \int_{t \geq 0} \Pr[\delta^A(t) \geq m] dt + O(\log P) \int_{t \geq 0} m \Pr[\delta^A(t) \geq m] dt \\
&\quad + \int_{t \geq 0} e^{4\gamma} \left(2\ln n(2(m-1)(\log P + 2) + 3\delta^{OPT}(t))\right) \Pr[\delta^A(t) \geq m] dt \\
&= \sum_{j \in J} p_j(\alpha + O(\log n)) + O(\log n \log P) \int_{t \geq 0} m \Pr[\delta^A(t) \geq m] dt
\end{aligned}$$

$$+O(\log n)\int_{t\geq 0}\delta^{OPT}(t)dt = O(\log n)\sum_{j\in J} p_j + O(\log n\log P)\sum_j p_j + O(\log n)F^{OPT}$$
$$= O(\log n\log P)F^{OPT},$$

where in the fourth equation we apply the claim of Lemma 11. We therefore state our first theorem:

Theorem 13 *RMLF is an $O(\log n\log P)$ competitive nonclairvoyant randomized algorithm for minimizing the total flow time on parallel machines.*

4.3 The $O(\log n)$ competitive ratio for $m = 1$, $O(\log n\log\frac{n}{m})$ competitive ratio for $m \geq 2$

In this section we strengthen the analysis to obtain an $O(\log n)$ tight bound for a single machine and an $O(\log n\log\frac{n}{m})$ bound for $m \geq 2$ parallel machines. Let $\overline{k}$ be the lowest class such that less than m jobs of class $\geq \overline{k}$ are released. let $T(\sigma) = \{t : \delta^A_\sigma(t) \geq m\}$ be the set of time instants where all machines are busy in the execution of RMLF defined by σ. Let $T_j(\sigma) \subseteq T(\sigma)$, $j = 0,..,\overline{k}$, be the set of time instants where at least one machine is busy with jobs in queue Q_j and no machine is busy with jobs in queues higher than Q_j in the execution of RMLF defined by σ. Let $T_{\overline{k}+1}(\sigma) \subseteq T(\sigma)$ be the set of time instants when at least one machine is processing a job in queue $Q_{\overline{k}+1}$ or higher and all machines are busy. Observe that for each σ, $\{T_0(\sigma), T_1(\sigma), \ldots, T_{\overline{k}+1}(\sigma)\}$ defines a partition of $T(\sigma)$. For the total flow time of RMLF, the last term of Equation (2) can be rewritten using the claim of Lemma 12 as follows:

$$\int_{t\geq 0}\mathrm{E}_\sigma[\delta^{\mathrm{b}}_\sigma(\mathrm{t})|\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\Pr[\delta^{\mathrm{A}}(\mathrm{t}) \geq \mathrm{m}]\mathrm{dt} = \sum_\sigma \Pr(\sigma)\int_{\mathrm{t}\in\mathrm{T}(\sigma)}\delta^{\mathrm{b}}_\sigma(\mathrm{t})\mathrm{dt}$$
$$\leq \sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}\int_{t\in T_j(\sigma)}(2m + \delta^b_{\sigma,\geq j,\leq \overline{k}-1}(t))dt + \sum_\sigma \Pr(\sigma)\int_{t\in T_{\overline{k}+1}(\sigma)}\delta^b_\sigma(t)dt$$
$$\leq \sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}\int_{t\in T_j(\sigma)}\left[2m + 2\ln n\left(2(m-1)(\overline{k}-j) + 3\delta^{OPT}(t)\right) + m(\overline{k}-j+1)\right]dt$$
$$+\sum_\sigma \Pr(\sigma)2m\int_{t\in T_{\overline{k}+1}(\sigma)}dt$$
$$= \sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}\int_{t\in T_j(\sigma)}4\ln n(m-1)(\overline{k}-j)dt$$
$$+\sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}\int_{t\in T_j(\sigma)}6\ln n\delta^{OPT}(t)dt + \sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}\int_{t\in T_j(\sigma)}m(\overline{k}-j+3)dt + 2\sum_j p_j$$
$$= O(\log n)\sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}(m-1)(\overline{k}-j)T_j(\sigma) + \sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}m(\overline{k}-j)T_j(\sigma)$$
$$+3m\sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}T_j(\sigma) + O(\log n)\sum_\sigma \Pr(\sigma)\int_{t\in T_j(\sigma)}\delta^{OPT}(t)dt + 2\sum_j p_j$$
$$= O(\log n)\sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}(m-1)(\overline{k}-j)T_j(\sigma) + \sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}m(\overline{k}-j)T_j(\sigma)$$
$$+O(\log n)F^{OPT}$$

The second inequality follows by partitioning for every σ the interval $\{t \geq 0 : \delta^A_\sigma(t) \geq m\}$ into the $T_j(\sigma)$ and by observing that at any time $t \in T_j(\sigma)$ there are at most m jobs of class less than j and $m-1$ jobs of class bigger or equal than $\overline{k}$. The third inequality follows by the claim of Lemma 12 and by observing that for any time $t \in T_{\overline{k}+1}(\sigma)$ at most $2m$ jobs are in the system since (i.) a machine is processing a job in a queue $\overline{k}+1$ or higher and hence at most $m-1$ jobs of class less than $\overline{k}$ are in the system; (ii.) at most $m-1$ jobs of class $\geq \overline{k}$ have been released. The other inequalities follow by rearranging the terms and observing that i) $\sum_\sigma Pr(\sigma)\int_{t\in T_j(\sigma)} \delta^{OPT}(t)dt \leq F^{OPT}$, ii) $\sum_j p_j \leq F^{OPT}$, iii) by applying the claim of Lemma 11.

As a consequence of the inequalities above and from equation (2), we can write:

$$\begin{aligned}\mathrm{E}_\sigma[\mathrm{F}^{\mathrm{A}}_\sigma] &= O(\log n)F^{OPT}\\ &+e^{4\gamma}\left((m-1)O(\log n)\sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}}(\overline{k}-j)T_j(\sigma)+\sum_\sigma \Pr(\sigma)\sum_{j=0}^{\overline{k}} m(\overline{k}-j)T_j(\sigma)\right). \quad (7)\end{aligned}$$

We are left to bound for any σ the term $F_\sigma(n) = \sum_{j=0}^{\overline{k}} m(\overline{k}-j)T_j(\sigma)$. We show this in the following Lemma:

Lemma 14 *For any outcome of the random choices σ of the algorithm it holds $F_\sigma(n) = \sum_{j=1}^{\overline{k}} m(\overline{k}-j)T_j(\sigma) = O(\log \frac{n}{m})F^{OPT}$.*

Proof. In the following we omit σ when clear from the context. We define $T^l_j(\sigma) \subseteq T(\sigma)/T_{\overline{k}+1}(\sigma)$ to be the set of time instants where machine l, $l = 1,..,m$, is processing a job of queue j, $j = 0,\ldots,\overline{k}$, in the execution defined by the random choices σ. Observe that, for each l, $\{T^l_0(\sigma),\ldots T^l_{\overline{k}}\}$ defines a partition of $T(\sigma)/T_{\overline{k}+1}(\sigma)$. Let n^l_j be the number of jobs finished by machine l in queue j in this specific execution of RMLF. We have the following inequalities:

$$F(n) \;=\; \sum_{j=0}^{\overline{k}} m(\overline{k}-j)T_j \leq \sum_{j=0}^{\overline{k}}\sum_{l=1}^{m}(\overline{k}-j)T^l_j \leq \sum_{j=0}^{\overline{k}}\sum_{i=0}^{j}\sum_{l=1}^{m} n^l_j(\overline{k}-i)2^{i+1}$$

The second inequality follows since at any time $t \in T_j$ every machine is working on a job of queue at most j while the contribution given in the second inequality for any time $t \in T_j$ by any machine l is bigger that $(\overline{k}-j)$. The third inequality follows since any job that is completed in queue Q_j has been processed in any queue Q_i, $i \leq j$, for at most 2^{i+1}, therefore giving a contribution of $(\overline{k}-i)2^{i+1}$.

Since $\sum_{i=0}^{j} 2^{i+1}(\overline{k}-i) \leq 2^{j+2}(\overline{k}-j+1)$, it follows

$$F(n) \;=\; 4\sum_{j=0}^{\overline{k}}\sum_{l=1}^{m} n^l_j(\overline{k}-j+1)2^j$$

Let $I_j = 2^j \sum_{l=1}^{m} n^l_j$. We show that $\sum_{j=0}^{\overline{k}}(\overline{k}-j+1)I_j = O(\log \frac{n}{m})F^{OPT}$. Let $\mathcal{I} = \max\{2^{\overline{k}}, \frac{1}{m}\sum_{j=0}^{\overline{k}} I_j\}$. Since there are more than m jobs of class at least $\overline{k}-1$ we have $m2^{\overline{k}-1} \leq F^{OPT}$. Since every job ending in a queue j has size at least 2^{j-1}, we have $\sum_{j=0}^{\overline{k}} I_j = \sum_{j=0}^{\overline{k}} 2^j \sum_{l=1}^{m} n_j^l \leq 2F^{OPT}$. Therefore we have $m\mathcal{I} \leq 2F^{OPT}$.

To prove that $F(n) = O(\sum_{j=0}^{\overline{k}} (\overline{k} - j)I_j + 2F^{OPT}) = O(\log \frac{n}{m})F^{OPT}$ we study the following optimization problem:

$$\begin{aligned}
\max_{\{I_0,\ldots,I_{\overline{k}}\}} F(n) &= \sum_{j=0}^{\overline{k}} (\overline{k} - j)I_j \\
n &\geq \sum_{j=0}^{\overline{k}} \frac{I_j}{2^j} \\
\mathcal{I} &\geq \frac{1}{m}\sum_{j=0}^{\overline{k}} I_j,
\end{aligned}$$

where the first constraint holds since $\frac{I_j}{2^j} = \sum_{l=1}^{m} n_j^l \leq n$.

We rewrite the problem using variables $Y_j = \sum_{i \leq j} I_i$, $j \geq 0$, with the convention $Y_{-1} = 0$:

$$\begin{aligned}
\max_{\{Y_0,\ldots,Y_{\overline{k}}\}} F(n) &= \sum_{j=0}^{\overline{k}} Y_j \\
n &\geq \sum_{j=0}^{\overline{k}} \frac{Y_j - Y_{j-1}}{2^j} \geq \sum_{j=0}^{\overline{k}} \frac{Y_j}{2^{j+1}} \\
m\mathcal{I} &\geq Y_{\overline{k}} \geq Y_{\overline{k}-1} \geq \ldots
\end{aligned}$$

The objective function is maximized assigning $m\mathcal{I} = Y_{\overline{k}} = Y_{\overline{k}-1} = Y_{\overline{k}-2} = \ldots. = Y_{\overline{k}-l}$, and $0 = Y_{\overline{k}-l-1} = \ldots. = Y_0$ with l being the minimum integer such that the second constraint is tight or violated, namely the minimum integer such that $\frac{m\mathcal{I}}{2^{\overline{k}-l-1}} \geq n$.

We then compute a value l such that $2^{l+1} = \frac{n}{m}\frac{2^{\overline{k}}}{\mathcal{I}}$. Since $2^{\overline{k}} \leq \mathcal{I}$ we have $l = O(\log \frac{n}{m})$, thus yielding $F(n) = O(\log \frac{n}{m})m\mathcal{I} = O(\log \frac{n}{m})F^{OPT}$ that completes the proof. □

The bound obtained from Claim 14 holds for any σ. We therefore obtain by inserting the claim in Equation (3) our first Theorem:

Theorem 15 *RMLF is a $O(\log n \log \frac{n}{m})$ competitive non-clairvoyant randomized algorithm for minimizing the total flow time on parallel machines.*

Considering Equation (3) with $m = 1$ we obtain our tight result for single machine:

Theorem 16 *RMLF is a $O(\log n)$ competitive non-clairvoyant randomized algorithm for minimizing the total flow time on single machine.*

5 Open problems

There is still a logarithmic gap between the competitive result on parallel machines and the $\Omega(\log \frac{n}{m})$ randomized lower bound for the case in which the processing time of a job is known at time of release [16]. However, our conjecture is that like in the single machine, the lack of knowledge about the processing times of the jobs leads to a logarithmic overhead.

The fact that a randomized version of MLF achieves optimal or almost optimal performances from a worst case point of view may be a good indication that MLF is also very efficient on input sequences drawn from specific probability distributions like the uniform or the exponential distribution. This might be a further validation of the goodness of MLF in practice.

We finally mention non-clairvoyant minimization of other flow time related metrics, like the average stretch, i.e. $\sum_{j \in J} \frac{F_j}{p_j}$. Constant competitive algorithms for the clairvoyant case have recently been proposed [4, 9].

Acknowledgments. We are immensely grateful to Kirk Pruhs for his help in a delicate phase of this work. We would also like to thank A. Marchetti-Spaccamela and S. Muthukrishnan for several discussions on these topics.

References

[1] S. Acharya, S. Muthukrishnan and G. Sundaram. Scheduling data delivery over multiple channels. *Bell Labs Technical Memo*, 1999. Available from `acharya@research.bell-labs.com`.

[2] B. Awerbuch, Y. Azar, S. Leonardi, and O. Regev. Minimizing the Flow Time without Migration. *In Proc. ACM Symposium on the Theory of Computing (STOC '99)*, 1999.

[3] K.R. Baker. *Introduction to Sequencing and Scheduling.* Wiley, 1974.

[4] L. Becchetti, S. Leonardi and S. Muthukrishnan. Scheduling to minimize Average Stretch without Migration. *In Proc. of Annual Symposium on Discrete Algorithms (SODA '00)*, 2000.

[5] S. Ben-David, A. Borodin, R. M. Karp, G. Tardos and A. Widgerson". "On the power of randomization in on-line algorithms". In *Proc. ACM Symposium on Theory of Computing (STOC'90)*, pp. 379-386, 1990.

[6] M. Bender, S. Chakrabarti, and S. Muthukrishnan. Flow and Stretch Metrics for Scheduling Continuous Job Streams. *In Proc. of Annual Symposium on Discrete Algorithms (SODA '99)*, 1999.

[7] T. W. Doeppner. Threads, a System for the Support of Concurrent Programming. *Technical Report CS-87-11*, Brown University, 1987.

[8] J. Du, J. Y. T. Leung, and G. H. Young. Minimizing mean flow time with release time constraint. *Theoretical Computer Science*, 75(3):347–355, 1990.

[9] J. E. Gehrke, S. Muthukrishnan, R. Rajaraman, and A. Shaheen. Scheduling to Minimize Average Stretch Online. *Proc. Annual Symposium on Foundations of Computer Science*, 1999.

[10] L. Hall, D. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line algorithms. In *Proc. of 7th ACM-SIAM Symposium on Discrete Algorithms*, pages 142–151, 1996.

[11] Institute for Electrical and Electronic Engineers. *POSIX P1003.4a, Threads Extensions for Portable Operating Systems*, 1994.

[12] B. Kalyanasundaram, and K. Pruhs. Minimizing Flow Time Nonclairvoyantly. *In Proc. of IEEE Symposium on Foundations of Computer Science (FOCS '97)*, 1997, pages 345-352.

[13] B. Kalyanasundaram, and K. Pruhs. Speed is as powerful as clairvoyance. *In Proc. of IEEE Symposium on Foundations of Computer Science (FOCS '95)*, 1995, pages 214–221.

[14] B. Kalyanasundaram, and K. Pruhs. Eliminating Migration in Multi-Processor Scheduling. *In Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA '99)*, 1999.

[15] H. Kellerer, T. Tautenhahn and G.J. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. *Proc. of the 28th Annual ACM Symposium on the Theory of Computing*, pp. 418-426, 1996.

[16] S. Leonardi and D. Raz. Approximating total flow time on parallel machines. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 110–119, El Paso, Texas, 1997.

[17] R. Motwani, S. Phillips and E. Torng. Non-Clairvoyant Scheduling. *Theoretical Computer Science, special issue on dynamic and on-line Algorithms*, 130, 1994, pages 17–47.

[18] F. Mueller. A Library Implementation of POSIX Threads under UNIX. *In Proc. of the Winter 1993 USENIX Technical Conference*, pp. 29-41, Jan. 1993.

[19] Gary Nutt. *Operating System Projects using Windows NT.* Addison Wesley, 1999.

[20] Kirk Pruhs. Personal Communication, 2000.

[21] A. S. Tanenbaum. *Modern Operating Systems.* Prentice-Hall Inc. 1992.

[22] Sun Microsystems. *SunOS 5.3 System Services*, Nov. 1993.

[23] D. Sleator and R.E. Tarjan. Amortized Efficiency of List Update and Paging Rules. *Communications of ACM* 28, 1985.