On Preemption and Learning in Stochastic Scheduling

Nadav Merlis [*1] Hugo Richard [*12] Flore Sentenac [*1] Corentin Odic [1] Mathieu Molina [13] Vianney Perchet [12]

Abstract

We study single-machine scheduling of jobs, each belonging to a job type that determines its duration distribution. We start by analyzing the scenario where the type characteristics are known and then move to two learning scenarios where the types are unknown: non-preemptive problems, where each started job must be completed before moving to another job; and preemptive problems, where job execution can be paused in the favor of moving to a different job. In both cases, we design algorithms that achieve sublinear excess cost, compared to the performance with known types, and prove lower bounds for the non-preemptive case. Notably, we demonstrate, both theoretically and through simulations, how preemptive algorithms can greatly outperform non-preemptive ones when the durations of different job types are far from one another, a phenomenon that does not occur when the type durations are known.

1. Introduction

Single Machine Scheduling is a longstanding problem with many variants and applications (Pinedo, 2012). In this problem, a set of N jobs must be processed on one machine, each of a different 'size' – processing time required for its completion. An algorithm is a policy assigning jobs to the machine, and performance is usually measured by *flow time* – the sum of the times when jobs have finished. If one has access to the size of each job, then scheduling the jobs by increasing size is optimal (Schrage, 1968). Unfortunately, for most applications, this knowledge is unavailable; yet, oftentimes, some structure or knowledge on the jobs can still be leveraged.

In this paper, we focus on scheduling problems where jobs are grouped by types that determine their duration distribution. This model approximates many real-world scenarios. For example, when scheduling patients for surgery, patients may be grouped by expected procedure time (Magerlein & Martin, 1978). The model is also relevant in computing problems, where jobs with similar features are expected to have a similar processing time (Li et al., 2006). Lastly, in calendar learning, where an agent advises the user on how to organize its day based on the tasks to be done, similar tasks can be assumed to have a similar duration (White & Hassan Awadallah, 2019).

In practice, when encountering a new scheduling task, we usually know the type of each job, but have little-to-no information on the expected duration under each type. Then, the scheduling algorithm must *learn* the characteristics of each type to be able to utilize this information. This must be done concurrently with the scheduling of tasks, which poses an extra challenge – to be useful, learning must be done as early as possible; however, wrong scheduling allocation at the beginning delays all jobs and causes large penalties.

In this work, we show how learning can be efficiently done in scheduling problems with job types, characterized by exponential distributions, in two different settings – the non-preemptive setting, where once a job started running, it must be completed, and the preemptive setting, where jobs can be put on hold. We present two algorithms in each setting and show that the preemptive setting has a clear advantage when the type durations have to be learned. This comes in contrast to the case of known types, where under reasonable assumptions, the optimal algorithm is non-preemptive.

While our algorithms resemble classic bandit methods, the scheduling objective requires different analysis approaches. In particular, in the context of scheduling, the quality of an algorithm is measured by the *ordering* of jobs. In stark contrast, regret-minimization objectives measure the *number of plays* from each arm (job type). Indeed, in scheduling problems, the number of pulls from each job type is always the same – by the end of the interaction, we would finish all the jobs of all types. Thus, both our algorithmic design and analysis will be comparative – focus on the number of jobs evaluations from a bad type before the completion of jobs of a good type.

Our contributions are as follows. **(1)** We present the scheduling setting with unknown job types. **(2)** We analyze the

[*]Equal contribution [1]ENSAE, Paris [2]Criteo, Paris [3]Inria, France. Correspondence to: Nadav Merlis <nadav.merlis@ensae.fr>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

optimal algorithm for the case of known job types, called Follow-the-Perfect-Prediction (FTPP), and bound its competitive ratio (CR). **(3)** We present explore-then-commit (ETC) and upper confidence bound (UCB) algorithms for the preemptive and non-preemptive settings and bound their performance, compared to FTPP. In particular, our bounds show that the non-preemptive algorithms have worse dependence on the durations of the longest job types. **(4)** We complement this by proving lower bounds to the non-preemptive case. **(5)** We end by simulating our suggested algorithms and show that their empirical behavior is consistent with our theoretical findings.

2. Related Work

Scheduling problems. The scheduling literature and problem zoology are large. We focus on static scheduling on a single machine with the objective of minimizing flow time. Static scheduling (Motwani et al., 1994) means that all scheduled tasks are given in advance before the scheduling starts. Possible generalizations include dynamic scheduling where scheduled tasks arrive online (Becchetti & Leonardi, 2004), weighted flow time (Bansal & Dhamdhere, 2007) where different jobs have different weights, multiple machines (Lawler & Labetoulle, 1978)); and many more (Dürr et al., 2020; Tsung-Chyan et al., 1997). While we only tackle some versions of this problem, we believe that our approaches can be adapted or extended to other settings.

Clairvoyant and non-clairvoyant scheduling. In clairvoyant scheduling, job sizes are assumed to be known, and scheduling the shortest jobs first gives the lowest flow time (Schrage, 1968). In non-clairvoyant scheduling, job sizes are arbitrary and unknown. The Round Robin (RR) algorithm, which gives the same amount of computing time to all jobs, is the best deterministic algorithm with a competitive ratio of $2 - \frac{2}{N+1} = 2 + o(N)$ (Motwani et al., 1994). The best randomized algorithm has a competitive ratio of $2 - \frac{4}{N+3} = 2 + o(N)$ (Motwani et al., 1994).

Stochastic scheduling. Stochastic scheduling covers a middle ground where job sizes are known random variables. The field of *optimal stochastic scheduling* aims to design optimal algorithms for stochastic scheduling (see Cai et al. 2014 for a review). When distributions have a non-decreasing hazard rate, scheduling the shortest mean first is optimal (see Cai et al. 2014, Corollary 2.1).

In this work, we consider exponential job sizes (which have a non-decreasing hazard rate), as frequently assumed in the scheduling literature (Kämpke, 1989; Hamada & Glazebrook, 1993; Cunningham & Dutta, 1973; Cai & Zhou, 2000; 2005; Pinedo & Weiss, 1985; Glazebrook, 1979) and similarly for the presence of different types of jobs (Mitzenmacher, 2020; Hamada & Glazebrook, 1993; Marbán et al., 2011). Yet, in contrast to most of the literature on stochastic scheduling, the means of the exponential sizes are unknown to the scheduler and are learned as the algorithm runs. Nonetheless, we later present algorithms whose CR asymptotically converges to the optimal value, obtained in stochastic scheduling with known job means.

The problem of learning in scheduling has received some attention lately. Specifically, Levi et al. (2019) consider a setting where it is possible to 'test' jobs to learn about their attributes, which comes at a cost. In (Krishnasamy et al., 2018), the authors propose an algorithm to learn the $c\mu$ rule (a rule to balance different holding costs per job) in the context of dynamic queues. Perhaps closest to our setting, in (Lee & Vojnovic, 2021), job types are also considered, but the length of the jobs is assumed to be known, and the goal is to deal with the uncertainty on the holding costs, which are noisily observed at each iteration. In the last two papers, no explicit exploration is needed, which stands in contrast with our setting.

The problem we tackle was previously studied in a Bayesian setting (Marbán et al., 2011), under the assumption of two job types, and a Bayesian algorithm, called LSEPT, was presented. When run with an uninformative prior (the same for all job types), LSEPT is reduced to a greedy algorithm; whenever a job finishes, it runs until completion a job whose type has the lowest expected belief on its mean size (computed across jobs that have been processed so far). The author proved it has better performance in expectation than fully non-adaptive methods, but provided no other guarantee. In Appendix D, we empirically evaluate this algorithm and show it has a behavior typical of greedy algorithms: it has a very large variance, and its CR does not converge to the optimal CR, in contrast to our suggested methods.

3. Setting and Notations

We consider scheduling problems of N jobs on a single machine, each belonging to one of K job types. We assume that $N = nK$, i.e, there are n jobs of each type. The different sizes (also called processing times) of the jobs of type k are denoted $(P_i^k)_{i\in[n]}$,[1] where $P_i^k \sim \mathcal{E}(\lambda_k)$ are independent samples from an exponential variable of parameter λ_k. By extension, $\mathbb{E}[P_i^k] = \lambda_k$, and we call λ_k the mean size of type k. We assume without loss of generality that the mean sizes of the K types are in an increasing order $\lambda_1 \leq \lambda_2 \ldots \leq \lambda_K$ and denote $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)$. With slight abuse of notations, we sometimes ignore the job types and denote the job durations by P_i for $i \in [N]$.

[1]For clarity of exposition, we assume that there are exactly n jobs per type. When types have different numbers of jobs $n_1, n_2, \ldots, n_K$, all algorithms can run with $n = \max_\ell n_\ell$, and all bounds hold with this same parameter n.

Next, denote b_i^k and e_i^k the beginning and end dates of the computation of the i^{th} job of type k. We define the cost of an algorithm ALG, also called *flow time*, as the sum of all completion times: $C_{\text{ALG}} = \sum_{k=1}^{K} \sum_{i \in n} e_i^k$. Given knowledge of the job size realizations, the cost is minimized by an algorithm that computes them in increasing order, which we term as OPT.

Preemption is the operation of pausing the execution of one job in the favor of running another one. Thus, *preemptive algorithms* are ones that support preemption, while *non-preemptive algorithms* do not allow it and must run each started job until completion.

4. Benchmark: Follow The Perfect Prediction

We compare our algorithms to a baseline that completes each job by increasing expected sizes, called *Follow-The-Perfect-Prediction* (FTPP). For exponential job sizes, this strategy is optimal between all algorithms without access to job size realizations (see Cai et al. 2014, Corollary 2.1). Thus, with learning, we aim at designing algorithms approaching the performance of FTPP, whilst mitigating the cost of learning (i.e., mitigating the cost of exploration). In the rest of this section, we analyze the performance of FTPP.

First, we evaluate the performance of non-clairvoyant algorithms that do not exploit the job type structure. We then compare the performance of the best of those algorithms against that of FTPP and show the clear advantage of using the structure of job types.

4.1. Non-clairvoyant Algorithms

An algorithm A is said non-clairvoyant if it does not have any information on the job sizes, including the job type structure. Recall that RR is the algorithm that computes all unfinished jobs in parallel and is the optimal deterministic algorithm in the adversarial setting. The following proposition states that in our setting, it is the optimal algorithm among all non-clairvoyant ones.

Proposition 4.1. *For any $\boldsymbol{\lambda}$ and any (deterministic or randomized) non-clairvoyant algorithm A, there exists a job ordering such that $\mathbb{E}[C_A] \geq \mathbb{E}[C_{\text{RR}}]$.*

Proof sketch (full proof in Appendix A.3). Consider T_{ij}^A, the amount of time that job i and job j delay each other. As the algorithm is unaware of the expected job size order, its run is independent of whether the expected size of job i is smaller or greater than that of j. This holds because a non-clairvoyant algorithm has no information on job expected sizes nor on the existence of job types. As an adversary, we can therefore choose the job order so that the algorithm incurs the largest flow time. A careful analysis then provides $\mathbb{E}[T_{ij}^A] \geq 2\mathbb{E}[T_{ij}^{OPT}]$ where OPT is the optimal realization-aware algorithm. A similar reasoning is made in the case of randomized algorithms. We conclude by observing that RR achieves such delay. □

Unfortunately, even though our setting is not adversarial, the CR of RR is bounded from below (Lemma A.3):

$$\text{for any } \boldsymbol{\lambda}, \quad \frac{\mathbb{E}\left[C_{\text{RR}}\right]}{\mathbb{E}\left[C_{\text{OPT}}\right]} \geq 2 - \frac{4}{n+3}. \qquad (1)$$

4.2. Performance of FTPP

The first statement establishes that FTPP outperforms RR on any instance.

Lemma 4.2. *For any n and $\boldsymbol{\lambda}$,*

$$\mathbb{E}[C_{\text{FTPP}}] \leq \mathbb{E}[C_{\text{RR}}].$$

The proof is a straightforward computation, done in Appendix A.2.3.

This indicates that when information on the job types is available, it is always advantageous to use it. In the rest of the section, we quantify the improvement this extra information brings. More precisely, we show that on a wide variety of instances, the CR of FTPP is much smaller than that of RR. We first present such a bound when $K = 2$.

Proposition 4.3. *The CR of* FTPP *with $K = 2$ types of jobs with n jobs per type with $\lambda_1 = 1$ and $\lambda_2 = \lambda > 1$ satisfies:*

$$\frac{\mathbb{E}[C_{\text{FTPP}}]}{\mathbb{E}[C_{\text{OPT}}]} \leq 2 - 4\frac{\lambda - 1}{(1+\lambda)^2 + 4\lambda}.$$

Proof sketch (full proof in Proposition A.4). The total expected flow time of any algorithm is given by the sum of the expected time spent computing all jobs and the expected time lost waiting as jobs delay each other. In the case of OPT, the expected flow time can be computed in closed form using that job i and j delay each other by $\mathbb{E}[\min(X_i, X_j)] = \frac{\mathbb{E}[X_i]\mathbb{E}[X_j]}{\mathbb{E}[X_i]+\mathbb{E}[X_j]}$. The CR $\frac{\mathbb{E}[C_{\text{FTPP}}]}{\mathbb{E}[C_{\text{OPT}}]}$ can then be calculated and is upper bounded to yield the result. □

In the case $K = 2$, there exists values of λ for which the CR of FTPP is lower than 1.71. In the general case, Proposition A.5 in Appendix A.2.2 shows that there exist values of K and $\boldsymbol{\lambda}$ for which the CR is as low as 1.274.[2]

5. Non-Preemptive Algorithms

After establishing FTPP as the baseline for learning algorithms, we move to tackle learning in the non-preemptive

[2]An exact expression for the CR of FTPP is given at Equation (7) in the appendix and is omitted for clarity reasons.

Algorithm 1 Non-Preemptive Algorithms routine

```
1: Init: type set $\mathcal{U} = [K]$, active jobs $i_k = 1, \forall k \in [K]$
2: while $\mathcal{U}$ is not empty do
3:     Use a type selection subroutine to select a type $k \in \mathcal{U}$
4:     Run job $i_k$ until completion
5:     Set $i_k \leftarrow i_k + 1$
6:     if All jobs of type $k$ are completed then
7:         Remove type $k$ from $\mathcal{U}$
8:     end if
9: end while
```

setting, where once started, job execution cannot be stopped (see Algorithm 1). This is relevant, for example, to settings where switching tasks is very costly (e.g., in running time or memory) or even impossible (e.g., in medical applications, where treatment of a patient cannot be stopped). We show how algorithms from the bandit literature can be adapted to the scheduling setting and bound their excessive cost, compared to FTPP. Specifically, by treating each job type as an 'arm', we adapt explore-then-commit and optimism-based strategies to the scheduling setting.

5.1. Description of ETC-U and UCB-U

In the following, we describe the type selection mechanism for ETC-U and UCB-U. The full pseudo-code of both algorithms is available in Appendix B.1.

Let $\mathcal{U}$ be the set of all job types with at least one remaining job.

ETC-U type selection. While ETC-U runs, it maintains a set of types $\mathcal{A}$ that are candidates for having the lowest mean size among the incomplete types $\mathcal{U}$. At each iteration, ETC-U chooses a job of type k of the minimal number of completed jobs in $\mathcal{A}$ and executes it to completion. Then, $\mathcal{U}$ and $\mathcal{A}$ are updated and the procedure repeats until no more jobs are available in $\mathcal{U}$.

We now describe the mechanism of maintaining the candidate type set $\mathcal{A}$. At a given iteration, denote by m_k and m_ℓ, the number of jobs of type k and ℓ that have been computed up to that iteration. Letting

$$\hat{r}_{k,\ell}^{\min(m_k,m_\ell)} = \frac{\sum_{i=1}^{\min(m_k,m_\ell)} \mathbb{1}\{P_i^k < P_i^\ell\}}{\min(m_k,m_\ell)} \quad \text{and}$$

$$\delta_{k,\ell}^{\min(m_k,m_\ell)} = \sqrt{\frac{\log(2n^2K^3)}{2\min(m_k,m_\ell)}},$$

a type ℓ is excluded from $\mathcal{A}$ if there exists a type k such that

$$\hat{r}_{k,l}^{\min(m_k,m_\ell)} - \delta_{k,\ell}^{\min(m_k,m_\ell)} > 0.5. \qquad (2)$$

In the proof, we show that this condition implies w.h.p. that $\lambda_k < \lambda_\ell$. Thus, when it holds, job type ℓ is no longer a candidate for the remaining job type with the smallest expectation, and we say that type k eliminates type ℓ. Once a job type is eliminated, it remains so until $\mathcal{A}$ is empty, at which point all job types in $\mathcal{U}$ are reinstated to $\mathcal{A}$.

Finally, whenever $\mathcal{A}$ contains only one type k, all jobs of this type are run to completion, and after all jobs from type k are finished, it is removed from $\mathcal{U}$ and therefore from $\mathcal{A}$. This means that types that were eliminated by k can be candidates again.

UCB-U type selection. At every iteration, the algorithm computes an index for each job type and plays a type with the minimal index from the incomplete types $\mathcal{U}$. Specifically, if m_k jobs were completed from type k, the index of the type is defined as

$$\underline{\lambda}_k^{m_k} = \frac{2\sum_{i=1}^{m_k} X_i^k}{\chi^2_{2m_k}(1 - \frac{1}{2n^2K^2})},$$

where $\chi^2_m(\delta)$ is the δ-percentile of a χ^2 distribution with m degrees of freedom. In the proof, we show that these indices are a lower bound of the job means w.h.p., so choosing the minimal index corresponds to choosing the type with the optimistic shortest duration.

5.2. Cost Analysis

Proposition 5.1. *The following bounds hold:*

$$\mathbb{E}[C_{\text{ETC-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + \frac{1}{n}\mathbb{E}[C_{\text{OPT}}] + \sum_{k\in[K]} \left[\frac{1}{2}(k-1)(2K-k) + (K-k)^2\right]\lambda_k n\sqrt{8n\log(2n^2K^3)}$$

and

$$\mathbb{E}[C_{\text{UCB-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}] + n(K-1)\sqrt{3n\ln(2n^2K^2)}\sum_{k=1}^{K}\lambda_k.$$

Proposition 5.1 shows that both ETC-U and UCB-U have sublinear excess cost. Indeed the optimal cost and thus also the cost of FTPP is lower bounded by $\Omega(n^2\sum_k \lambda_k)$ which makes the terms in $O(n\sqrt{n})$ strictly sublinear in n compared to the optimal cost. When all parameters lie within a finite range bounded away from 0, the optimal cost scales as K^2n^2 making the excess cost linear in K (up to log terms) compared to the optimal cost.

Proof sketch (full proof in Appendix B). The above proposition is a concatenation of Propositions B.2 and B.4.

The proof of both bounds starts with the decomposition of the cost with the following Lemma (proven in Appendix B.2).

Lemma 5.2 (Cost of non-preemptive algorithms). *Any non-preemptive algorithm A has a cost*

$$\mathbb{E}[C_A] = \mathbb{E}[C_{\text{FTPP}}] + \sum_{\substack{(\ell,k)\in[K^2],k>\ell\\(i,j)\in[n]^2}} (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\{e_i^k \le b_j^\ell\}\right].$$

This lemma is obtained by computing explicitly the expected cost of algorithm FTPP and using the fact that the realized length of the jobs conditioned on their type is independent of their start date.

Then the two proofs diverge.

For ETC-U, the first step is to prove that condition (2) implies w.h.p that $\lambda_k \le \lambda_\ell$, which implies that the type in $\mathcal{U}$ with the smallest mean is never eliminated. Then, for the sake of the analysis, the run of the algorithm is divided into phases. In phase number ℓ, type ℓ is the job type remaining with the smallest mean. We then bound the total number of samples before an arm with a large mean is eliminated at phase ℓ.

The proof for UCB also starts by showing that w.h.p., arm indices lower bound the true means. Then, under the condition that the bounds hold, we upper bound the number of times an arm of type $k \ge \ell$ can be pulled while type ℓ is still active. □

The bounds in Proposition 5.1 hold for any value of the parameters. When the parameter values are far from each other, tighter bounds hold. We give here these tighter bounds for $K = 2$ when $\lambda_2 \ge 3\lambda_1$. A more general version of this bound is given in the appendix, Propositions B.2 and B.4.

Lemma 5.3. *If $K = 2$ and $\lambda_2 \ge 3\lambda_1$, the following bounds hold:*

$$\mathbb{E}[C_{\text{ETC-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + 12\lambda_2 n \log(2n^2K^3) + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}],$$

and

$$\mathbb{E}[C_{\text{UCB-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + \frac{9}{2}\lambda_2 n \ln\left(2n^2K^2\right) + \frac{4}{n}\mathbb{E}[C_{\text{OPT}}].$$

The bounds of this section seem quite discouraging – they imply that the existence of even one type of extremely large duration has grave implications on the cost of any algorithm. Unfortunately, for any non-preemptive algorithm, an extra cost w.r.t. FTPP scaling as $n\lambda_K$ is unavoidable. Indeed, in the beginning, no information on the mean types is available, and any started job will be fully computed, delaying all remaining $nK - 1$ jobs (see Appendix B.5.2 for a formal proof).

5.3. Lower bound

We end this section by analyzing lower bounds for any non-preemptive scheduling algorithm. In particular, we focus on the dependency of the excessive cost, compared to FTPP, as a function of n. We focus on lower bounds for the case of $K = 2$, providing a lower bound when λ_1 and λ_2 are close to each other and showing that in this case, the excess cost increases as $n\sqrt{n}$.

Proposition 5.4 (Dependency in n). *For any $\lambda_1 < \lambda_2$, the flow time of any (possibly randomized) non-preemptive algorithm A satisfies:*

$$\mathbb{E}[C_A] \ge \mathbb{E}[C_{\text{FTPP}}] + (\lambda_2 - \lambda_1)n^2 \frac{\exp(-n\frac{(\lambda_2-\lambda_1)^2}{\lambda_1\lambda_2})}{8}.$$

In particular, for any $\lambda_2 \le \lambda_1\left(1 + \frac{1}{\sqrt{n}}\right)$,

$$\mathbb{E}[C_A] \ge \mathbb{E}[C_{\text{FTPP}}] + (\lambda_1 + \lambda_2)n\sqrt{n}\frac{e^{-1/4}}{24}.$$

Proof sketch (full proof in Appendix B.5.1). We start with the decomposition of Lemma 5.2 and look at the event

$$E = \mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\{e_j^2 < b_i^1\} \ge n^2/2\right\}.$$

Notice that a since the algorithm is non-preemptive, a job that terminates delays any incomplete job by its full duration. Thus, the event indicates that the 'number of times' a job of type 2 delayed any job of type 1 is at least $n^2/2$. Since $\lambda_2 > \lambda_1$, this scenario would lead to a large excess cost.

Then, using standard information-theoretic tools, we lower bound the probability that either E occurs in the original scheduling problem or $\bar{E}$ occurs in a problem where the type order has been switched. In both problems, the relevant event causes an excess cost of $\Omega(n^2)$, and substituting the exact probability of this failure case concludes the proof. □

6. Preemptive Algorithms

In this section, we show how to leverage preemption to get better theoretical and practical performances.

In practice, we allow preemption by discretizing the computation time into small time slots of length Δ. Then, at every iteration, one or multiple job types are selected depending on some algorithm-specific criteria. The current running job(s) of the selected type is allocated computation time Δ instead of being run to completion. As before, we employ both an explore-then-commit strategy and an optimism-based strategy. In both cases, the only dependence of the resulting algorithm on the discretization size is due

Algorithm 2 Preemptive Algorithms routine

```
1:  Init: type set $\mathcal{U} = [K]$, active jobs $i_k = 1, \forall k \in [K]$
2:  while $\mathcal{U}$ is not empty do
3:      Use a type selection subroutine to select a type $k \in \mathcal{U}$
4:      Run job $i_k$ for $\Delta$ time units
5:      if $i_k$ was completed then
6:          Set $i_k \leftarrow i_k + 1$
7:      end if
8:      if All jobs of type $k$ are completed then
9:          Remove type $k$ from $\mathcal{U}$
10:     end if
11: end while
```

to the discretization error (the time between the end of a job and the end of a window), which decreases with the discretization step. We omit that discretization error of at most ΔN is the bounds.

Note that in practice, any implementation of RR proceeds in a similar manner. For instance, in (Motwani et al., 1994), the discretization step is assumed much smaller than the length of the jobs. In particular, when we say we run jobs in parallel, in practice, they cyclically run in a RR manner with a small discretization step.

6.1. ETC-RR and UCB-RR

ETC-RR type selection. As ETC-U, ETC-RR maintains a set of types $\mathcal{A}$ that are candidates for lowest mean size among the set $\mathcal{U}$ of types with at least one remaining jobs. The main difference is that the job type selected is the one in $\mathcal{A}$ with the lowest total run-time (not the one with the lowest number of computed jobs).

The statistics needed to construct $\mathcal{A}$ are different from the ones used in ETC-U. At a given time, $\beta_{k,\ell}$ is the number of times a job of type k has finished while ℓ and k were both active. Moreover, we define

$$\hat{r}_{k,\ell} = \frac{\beta_{k,\ell}}{\beta_{k,\ell} + \beta_{\ell,k}} \quad \text{and} \quad \delta_{k,\ell} = \sqrt{\frac{\log(2n^2K^3)}{2(\beta_{k,\ell} + \beta_{\ell,k})}}.$$

The elimination rule is the same as the one of ETC-U, using these modified statistics.

Reducing the number of algorithm updates: In practice, both the statistics and the chosen types are not updated at every iteration; active jobs run in parallel (meaning in a round-robin style), and the statistics are updated every time a job terminates. This formulation of the algorithm is the one we implement(see pseudo-code in Appendix C.1).

UCB-RR type selection. For each job type $k \in [K]$, we introduce $T_k(t)$, the number of times job type k has been chosen up to iteration t, and the random variables $(x_k^s)_s$ s.t.:

$$x_k^s = \sum_t \mathbb{1}\{a(t) = k, T_k(t) = s \text{ and the job finishes}\}.$$

It is the indicator that a job of type k is completed when this type is picked for the s^{th} time by the algorithm. We define the empirical means as:

$$\hat{\mu}_k(T) := \frac{1}{T}\sum_{s=1}^{T} x_k^s,$$

and define the index for each arm k as

$$u_k(t) = \max\left\{\tilde{\mu} \in [0,1] : d\left(\hat{\mu}_k(T_k(t)), \tilde{\mu}\right) \leq \frac{\log n^2}{T_k(t)}\right\},$$

with $d(x, y)$ the Kullback-Leibler divergence between x and y. A job type with the largest index is selected.

Reducing the number of algorithm updates: As for ETC-RR, the running jobs and statistics are not updated at every iteration. Suppose type k^* is chosen at iteration t. If k^* is the last remaining type, it is run until the end. Otherwise, let ℓ be the type with the second largest index. We define

$$\tilde{\mu}_k^{\gamma}(T) := \frac{1}{T + 2^{\gamma}}\sum_{s=1}^{T} x_k^s,$$

and

$$\tilde{u}_k^{\gamma}(t) = \max\left\{\tilde{\mu} \in [0,1] : d\left(\tilde{\mu}_k^{\gamma}(T_k(t)), \tilde{\mu}\right) \leq \frac{\log n^2}{T_k(t) + 2^{\gamma}}\right\}.$$

This would be the new index of arm k, were it to run for additional 2^{γ} iterations with no job terminating during this additional iterations. Then, we set

$$\gamma^* = \operatorname*{argmax}_{\gamma} \tilde{u}_{k^*}^{\gamma}(t) \geq u_\ell(t),$$

and type k^* is allocated 2^{γ^*} iterations with no statistics update.

6.2. Cost Analysis

Proposition 6.1. *The following bounds hold:*

$$\mathbb{E}[C_{\text{ETC-RR}}] \leq \mathbb{E}[C_{\text{FTPP}}] + \frac{12K}{n}\mathbb{E}[C^{OPT}] + 4n\sqrt{n\log(2n^2K^3)}\sum_{k=1}^{K-1}(K-k)^2\lambda_k.$$

and for any $\Delta \leq \frac{\lambda_1}{4}$ *and* $n \geq \max(20, 10\ln(K))$,

$$\mathbb{E}[C_{\text{UCB-RR}}] \leq \mathbb{E}[C_{\text{FTPP}}] + \frac{12K}{n}\mathbb{E}[C^{OPT}] + 6n\sqrt{2n\log(2n^2K^2) + 2}\sum_{k=1}^{K-1}(K-k)\lambda_k.$$

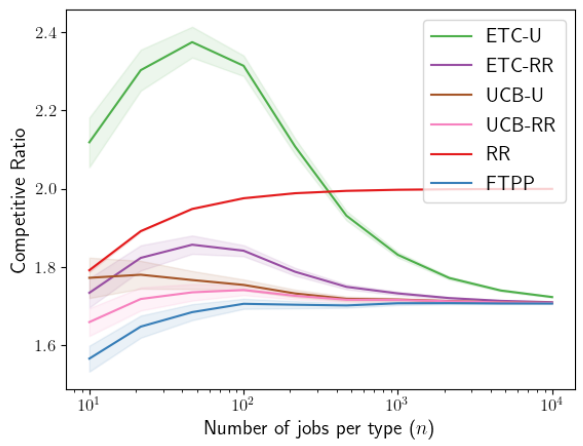

Figure 1. CR of all algorithms with varying number of jobs, $\lambda_1 = 1$, $\lambda_2 = 0.25$, averaged over 400 seeds.

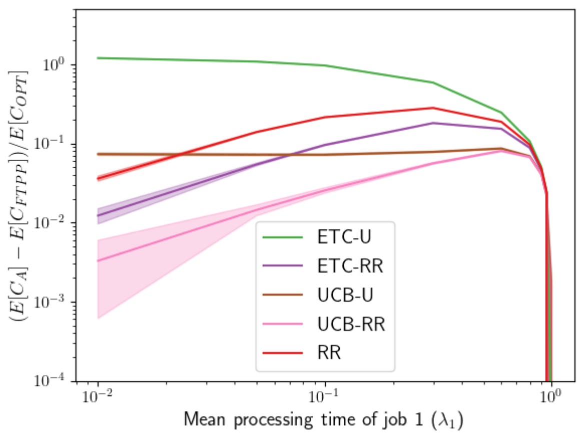

Figure 2. Normalized excess cost of all algorithms w.r.t. FTPP with varying value of λ_1, for $\lambda_2 = 1$ and $n = 50$, averaged over 5,000 seeds.

Proof sketch (full proof in Appendix C). The above proposition is a combination of Propositions C.3 and C.4.

Both algorithms belong to the following family of type-wise non-preemptive algorithms.

Definition 6.2. Recall that b_i^k and e_i^k are the beginning and end dates of the computation of the i^{th} job of type k. A **type-wise non-preemptive algorithm** is an algorithm that computes jobs of the same type one after another, i.e., $\forall i \in [n], \forall k \in [K], e_i^k \leq b_{i+1}^k$.

The following Lemma, proven in Appendix C.2 bounds the expected cost of any type-wise non-preemptive algorithms.

Lemma 6.3 (Cost of type-wise non-preemptive algorithms)**.** *Any type-wise non-preemptive algorithm A has the following upper bound on its cost:*

$$\begin{aligned}\mathbb{E}[C_A] \leq &\mathbb{E}[C_{\text{FTPP}}] \\ &+ \sum_{\substack{(\ell,k)\in[K^2],k>\ell \\ (i,j)\in[n]^2}} (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\{e_j^k < b_i^\ell\}\right] \\ &+ (K-1)n\sum_{k=1}^{K}\lambda_k.\end{aligned}$$

The proof of this Lemma again involves computing explicitly the cost of FTPP and using that the realization of a job length is independent of its start date. A first upper bound is obtained by noting that a job started before another delays the former in expectation by at most its expected length. The second element to the proof is the fact that at every job termination, at most a job of each other type is currently active. This observation leads to the upper bound on the additional cost of preemption and the last term of the expression of the Lemma. Note that this last term implies that our upper bound will include a term scaling as $n\lambda_K$, which would indicate that preemptive algorithms have an extra learning cost scaling as the highest mean type. However, we strongly believe this to be an artefact of the analysis.

Given the decomposition, the two proofs diverge.

The analysis of ETC-RR is split into phases, as the analysis of ETC-U. However, the bound on the number of 'bad' jobs computed in each phase requires more care because of independence arguments. Specifically, the upper bound is derived from concentration bounds on the computed statistics, and an additional bound on the number of successful jobs of each type when two types are run in parallel. The details on how to deal with those two non-independent events can be found in Appendix C.3.

For UCB-RR, the first step is to distinguish two types of 'failures' of the index. In the first failure case, the index deviates below the true mean. We show that this happens with probability $O(1/n^2)$ (Lemma C.7), independently of Δ. The second type of failure is when the index of a sub-optimal arm is much larger than its true mean. Here, we show that the upper bound on the number of iterations where this happens does diverge as Δ goes to zero. However, the algorithm only incurs a cost on the 'bad pull' of a type when the selected job terminates. The probability of job termination decreases as Δ decreases, which compensates for the rise in the upper bound (Equation (32)).

□

7. Experiments

In this section, we design synthetic experiments to compare ETC-U, ETC-RR, UCB-RR,UCB-U, RR and FTPP. All code is written in Python. We use matplotlib (Hunter,

2007) for plotting, and numpy (Harris et al., 2020) for array manipulations. The above libraries use open-source licenses. Computations are run on a laptop.[3]

The first experiment plots the CR of each algorithm for two types of jobs and fixed values of λ_1, λ_2 as n varies (see Figure 1). Even though all our suggested algorithms have the same asymptotic performance, their non-asymptotic behavior drastically varies. As predicted by theory, the preemptive versions of the algorithm consistently outperform the non-preemptive ones.

In the second experiment, $n = 50$ and $\lambda_2 = 1$ are fixed, while λ_1 varies in $(0, 1)$ (see Figure 2). To be able to discern performance gaps when λ_1 is small, we plot the difference between the CR of different algorithms and FTPP at a logarithmic scale. Here, for small values of λ, both preemptive methods outperform the non-preemptive ones. This corresponds with the improvement in the dominant error term of the preemptive cost upper bounds, as a function of λ_2.

7.1. Discussion

Preemptive vs. Non-Preemptive The competitive ratio of all algorithms is asymptotically the one of FTPP. Indeed, it always holds that $\mathbb{E}[C_{\mathrm{OPT}}] \geq (\lambda_1 + \lambda_2)\frac{n^2}{4}$ (Equation (6)), so by Propositions 6.1 and 5.1, for any algorithm A among ETC-U, ETC-RR, UCB-U and UCB-RR:

$$\mathrm{CR}_A = \mathrm{CR}_{\mathrm{FTPP}} + \mathcal{O}\left(\sqrt{\frac{\log(n)}{n}}\right).$$

On the one hand, the leading term in the cost is the same for all algorithms. On the other hand, the error term can be much smaller in the case of preemptive algorithms.

To illustrate this claim, let us consider the case where there are two types of jobs of expected sizes λ_1 and λ_2, respectively. Instantiating the bounds of Propositions 5.1 and 6.1 to this setting, we get:

$$\begin{aligned}\mathbb{E}[C_{\text{ETC-U}}] \leq \mathbb{E}[C_{\mathrm{FTPP}}] + n(\lambda_1 + \lambda_2)\sqrt{8n\log(2n^2K^3)} \\ + \frac{8}{n}\mathbb{E}[C_{\mathrm{OPT}}], \qquad (3)\end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[C_{\text{ETC-RR}}] \leq & \mathbb{E}[C_{\mathrm{FTPP}}] + 2n\lambda_1(\sqrt{4n\log(2n^2K^3)} + 1) \\ & + \frac{16}{n}\mathbb{E}[C_{\mathrm{OPT}}]. \qquad (4)\end{aligned}$$

If $\lambda_2 \gg \lambda_1$ the bound in Equation (3) is much larger than the bound in Equation (4), which is consistent with what we observe in Figure 2. In particular, one can observe that for small λ_1, non-preemptive algorithms converge to a strictly positive error (due to the unavoidable dependence in $\lambda_2 = 1$), while the error of the preemptive algorithms diminishes. This empirically supports our claim that the $n\lambda_K$-dependence, as appears in the preemptive cost decomposition of Lemma 6.3, is only due to a proof artefact.

Optimism-based vs. explore-then-commit. In the simulations, we see that optimism-based algorithms perform much better than their ETC counterparts. In traditional bandit settings, it is well known that the regret of ETC strategies is a constant-times larger than that of optimism-based strategies. Here, we believe that in addition to that, a second phenomenon, not reflected in the analysis, renders the optimism-based strategies better than the other ones. Because of the structure of the cost, a pull of a ‘bad job’ at the beginning is much more expensive than the same pull done later in the interaction (as it delays more jobs). Optimism-based strategies explore continuously as they run, whereas ETC strategies have all the exploration at the beginning, when it is more expensive. Again, this phenomenon stands in contrast with traditional bandits, where only the number of ‘bad pulls’ matter, and not their position.

8. Conclusion and Future Work

We designed and analyzed a family of algorithms for static scheduling on a single machine in the presence of job types. The special cost structure of this problem differs from that of traditional bandit problems, and early mistakes carry much more weight than late ones, as they delay more jobs. This modified cost directly impacts the performance of algorithms; although all suggested algorithms asymptotically have the same CR as the optimal algorithm that knows job type sizes (FTPP), their non-asymptotic performances differ.

When preemption is allowed, algorithms that explore job types with a strategy inspired by the worst-case optimal deterministic algorithm RR have a clear advantage over non-preemptive learning algorithms. Thus, because of the cost structure, the performance is impacted not only by the number of exploratory steps but also by the nature of the exploratory steps.

Due to the ubiquitousness of scheduling problems, we believe that our results could be extended to many other variants of this setting. In particular, it would be interesting to take our algorithmic principles and test them on real-world scheduling problems. Whether our current assumption on the exponential distribution of job sizes can be removed is an exciting direction for future work. We believe that many of the proofs for the non-preemptive case can be extended to other well-behaved distributions. However, in the preemptive case, our proofs do heavily rely on the properties

[3]The code to reproduce experiemnts is available at `https://github.com/hugorichard/ml4a-scheduling`.

of the exponential distribution, mainly the memoryless increments property. Without it, we can no longer average over different sections of a job evaluation to estimate its expected duration. Nonetheless, we believe that the results are still generalizable although proving the bounds would be technically much harder.

Moreover, we believe that elements from our works can be taken to other online learning settings outside the scope of scheduling. Specifically, we believe that the notion of types serves as a reasonable approximation that allows the integration of learning to many online problems. We also think that the study of cost functions that are sensitive the early exploration is of great interest.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101034255.

Nadav Merlis is partially supported by the Viterbi Fellowship, Technion.

Mathieu Molina is supported by the French National Research Agency (ANR) through grant ANR-20-CE23-0007.

Vianney Perchet acknowledges support from the French National Research Agency (ANR) under grant number (ANR-19-CE23-0026 as well as the support grant, as well as from the grant "Investissements d'Avenir" (LabEx Ecodec/ANR-11-LABX-0047).

References

Bansal, N. and Dhamdhere, K. Minimizing weighted flow time. *ACM Transactions on Algorithms (TALG)*, 3(4): 39–es, 2007.

Becchetti, L. and Leonardi, S. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *Journal of the ACM*, 51(4):517–539, July 2004. ISSN 0004-5411. doi: 10.1145/1008731.1008732.

Cai, X. and Zhou, X. Asymmetric earliness and tardiness scheduling with exponential processing times on an unreliable machine. *Annals of Operations Research*, 98(1): 313–331, 2000.

Cai, X. and Zhou, X. Single-machine scheduling with exponential processing times and general stochastic cost functions. *Journal of Global Optimization*, 31(2):317–332, 2005.

Cai, X., Wu, X., and Zhou, X. *Optimal Stochastic Scheduling*, volume 4. Springer, 2014.

Calin, O. and Udrişte, C. *Geometric modeling in probability and statistics*, volume 121. Springer, 2014.

Cunningham, A. A. and Dutta, S. K. Scheduling jobs, with exponentially distributed processing times, on two machines of a flow shop. *Naval Research Logistics Quarterly*, 20(1):69–81, 1973.

Dürr, C., Erlebach, T., Megow, N., and Meißner, J. An adversarial model for scheduling with testing. *Algorithmica*, 82(12):3630–3675, 2020.

Garivier, A., Ménard, P., and Stoltz, G. Explore first, exploit next: The true shape of regret in bandit problems. *Mathematics of Operations Research*, 44(2):377–399, 2019. doi: 10.1287/moor.2017.0928. URL `https://doi.org/10.1287/moor.2017.0928`.

Glazebrook, K. D. Scheduling tasks with exponential service times on parallel processors. *Journal of Applied Probability*, 16(3):685–689, 1979.

Hamada, T. and Glazebrook, K. D. A bayesian sequential single machine scheduling problem to minimize the expected weighted sum of flowtimes of jobs with exponential processing times. *Operations Research*, 41(5): 924–934, 1993.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Hunter, J. D. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

Kämpke, T. Optimal scheduling of jobs with exponential service times on identical parallel processors. *Operations Research*, 37(1):126–133, 1989.

Krishnasamy, S., Arapostathis, A., Johari, R., and Shakkottai, S. On learning the cμ rule in single and parallel server networks. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 153–154. IEEE, 2018.

Lattimore, T. and Szepesvári, C. *Bandit Algorithms*. Cambridge University Press, 2020. doi: 10.1017/9781108571401.

Laurent, B. and Massart, P. Adaptive estimation of a quadratic functional by model selection. *Annals of Statistics*, pp. 1302–1338, 2000.

Lawler, E. L. and Labetoulle, J. On preemptive scheduling of unrelated parallel processors by linear programming. *Journal of the ACM (JACM)*, 25(4):612–619, 1978.

Lee, D. and Vojnovic, M. Scheduling jobs with stochastic holding costs. *Advances in Neural Information Processing Systems*, 34:19375–19384, 2021.

Levi, R., Magnanti, T., and Shaposhnik, Y. Scheduling with testing. *Management Science*, 65(2):776–793, 2019.

Li, H., Chen, J., Tao, Y., Gro, D., and Wolters, L. Improving a local learning technique for queuewait time predictions. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06)*, volume 1, pp. 335–342. IEEE, 2006.

Magerlein, J. M. and Martin, J. B. Surgical demand scheduling: a review. *Health services research*, 13(4):418, 1978.

Marbán, S., Rutten, C., and Vredeveld, T. Learning in stochastic machine scheduling. In *International Workshop on Approximation and Online Algorithms*, pp. 21–34. Springer, 2011.

Mitzenmacher, M. Scheduling with predictions and the price of misprediction. In *11th Innovations in Theoretical Computer Science Conference (ITCS 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

Motwani, R., Phillips, S., and Torng, E. Nonclairvoyant scheduling. *Theoretical computer science*, 130(1):17–47, 1994.

Pinedo, M. and Weiss, G. Scheduling jobs with exponentially distributed processing times and intree precedence constraints on two parallel machines. *Operations Research*, 33(6):1381–1388, 1985.

Pinedo, M. L. *Scheduling*, volume 29. Springer, 2012.

Schrage, L. A proof of the optimality of the shortest remaining processing time discipline. *Operations Research*, 16 (3):687–690, 1968.

Tsung-Chyan, L., Sotskov, Y. N., Sotskova, N. Y., and Werner, F. Optimal makespan scheduling with given bounds of processing times. *Mathematical and Computer Modelling*, 26(3):67–86, 1997.

White, R. W. and Hassan Awadallah, A. Task duration estimation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pp. 636–644, 2019.

A. Benchmark FTPP

In this section, for all jobs $i \in [N]$, we call P_i the job size of job i. Jobs are ordered in increasing order of their expected size (Notation P_i and $P_{i \mod n}^{\lceil i/n \rceil}$ denote the same job). For any algorithm A, we note T_{ij}^A for each $(i,j) \in [N]^2$ the amount of time job i and job j delay each other under algorithm A.

A.1. Cost of OPT and FTPP, CR of RR

Let us express the expected cost of any algorithm in terms of $T_{i,j}^A$ for $k \in [K]$:

$$\mathbb{E}[C_A] = \mathbb{E}\left[\sum_{i=1}^{N} P_i + \sum_{i=1}^{N}\sum_{j=i+1}^{N} T_{i,j}^A\right] \tag{5}$$

Lemma A.1 (Cost of OPT)**.** *The cost of OPT is given by*

$$\mathbb{E}[C_{\mathrm{OPT}}] = n^2\left(\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + \frac{3n}{4}\sum_{\ell=1}^{K}\lambda_\ell.$$

Note that this lemma implies the following inequality, which will be used in other proofs:

$$\mathbb{E}[C_{\mathrm{OPT}}] \geq \frac{n^2}{4}\left(\sum_{\ell=1}^{K}\lambda_\ell\right) \tag{6}$$

Proof. We apply Equation (5) with $A = \mathrm{OPT}$. In that case, for any to jobs $(i,j) \in [N]$, $i \neq j$, as the shortest job is scheduled first, we have

$$\mathbb{E}[T_{ij}^A] = \mathbb{E}[\min(P_i, P_j)].$$

So $\mathbb{E}[T_{ij}^A] = \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}$ if job i is of type k and job j is of type ℓ.

$$\begin{aligned}
\mathbb{E}[C_{\mathrm{OPT}}] &= \mathbb{E}\left[\sum_{i=1}^{N} P_i + \sum_{i=1}^{N}\sum_{j=i+1}^{N} T_{i,j}^A\right]\\
&= \sum_{\ell=1}^{K}\sum_{i=1}^{n}\left(\lambda_\ell + \sum_{j=i+1}^{n}\frac{\lambda_\ell}{2} + \sum_{k=\ell+1}^{K}\sum_{j=1}^{n}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right)\\
&= \sum_{\ell=1}^{K}\left(n\lambda_\ell + \frac{n(n-1)}{2}\frac{\lambda_\ell}{2} + n^2\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right)\\
&= n^2\left(\sum_{\ell=1}^{K}\frac{1}{4}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + \frac{3n}{4}\sum_{k=1}^{K}\lambda_k.
\end{aligned}$$

□

Lemma A.2 (Cost of FTPP)**.** *The cost of FTPP is given by:*

$$\mathbb{E}[C_{\mathrm{FTPP}}] = n^2\left(\frac{1}{2}\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell\right) + n\left(\frac{\sum_{\ell=1}^{K}\lambda_\ell}{2}\right)$$

Proof. We apply Equation (5) with $A = \mathrm{OPT}$, so that $\mathbb{E}[T_{ij}^A] = \min(\lambda_k, \lambda_\ell)$ if job i is of type k and job j is of type ℓ

$$\begin{aligned}
\mathbb{E}[C_{\mathrm{FTPP}}] &= \mathbb{E}\left[\sum_{i=1}^N P_i + \sum_{i=1}^N \sum_{j=i+1}^N T_{i,j}^A\right] \\
&= \sum_{\ell=1}^K \sum_{i=1}^n \left(\lambda_\ell + \sum_{j=i+1}^n \lambda_\ell + \sum_{k=\ell+1}^K \sum_{j=1}^n \lambda_\ell\right) \\
&= \sum_{\ell=1}^K \left(n\lambda_\ell + \frac{n(n-1)}{2}\lambda_\ell + n^2 \sum_{k=\ell+1}^K \lambda_\ell\right) \\
&= n^2 \left(\frac{1}{2}\sum_{\ell=1}^K \lambda_\ell + \sum_{\ell=1}^K (K-\ell)\lambda_\ell\right) + n\left(\frac{\sum_{\ell=1}^K \lambda_\ell}{2}\right).
\end{aligned}$$

□

Lemma A.3 (CR of RR). *For any For any $\boldsymbol{\lambda}$, the following lower bound holds:*

$$\frac{\mathbb{E}[C_{\mathrm{RR}}]}{\mathbb{E}[C_{\mathrm{OPT}}]} \geq 2 - \frac{4}{n+3}.$$

Proof. For RR, any two jobs are run in parallel until one terminates, thus:

$$\mathbb{E}[T_{ij}^{\mathrm{RR}}] = 2\mathbb{E}[\min(P_i, P_j)].$$

Thus, by equation 5:

$$\mathbb{E}[C_{\mathrm{RR}}] = \sum_{i=1}^N \mathbb{E}[P_i] + \sum_{j=1}^N 2\mathbb{E}[\min(P_i, P_j)].$$

On the other hand:

$$\mathbb{E}[C_{\mathrm{OPT}}] = \sum_{i=1}^N \mathbb{E}[P_i] + \sum_{j=1}^N \mathbb{E}[\min(P_i, P_j)].$$

Thus:

$$\begin{aligned}
\frac{\mathbb{E}[C_{\mathrm{RR}}]}{\mathbb{E}[C_{\mathrm{OPT}}]} &= 2 - \frac{\sum_{i=1}^N P_i}{\mathbb{E}[C_{\mathrm{OPT}}]} \\
&= 2 - \frac{n\sum_{\ell=1}^K \lambda_\ell}{n^2\left(\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell + \sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + \frac{3n}{4}\sum_{\ell=1}^K \lambda_\ell} \\
&\geq 2 - \frac{n\sum_{\ell=1}^K \lambda_\ell}{n^2\left(\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell\right) + \frac{3n}{4}\sum_{\ell=1}^K \lambda_\ell} \\
&= 2 - \frac{4}{n+3}.
\end{aligned}$$

With the second line obtained by Lemma A.1. □

A.2. CR of FTPP

A.2.1. CR WITH K TYPES

Proposition A.4 (Upper bound on the CR in function of $\boldsymbol{\lambda}$). *The CR of FTPP with K types of jobs with n jobs per type satisfies:*

$$\frac{\mathbb{E}[C_{\mathrm{FTPP}}]}{\mathbb{E}[C_{\mathrm{OPT}}]} \leq 2 - f_K(\boldsymbol{\lambda})$$

where $f_K(\boldsymbol{\lambda}) = \frac{2\sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell} - \sum_{\ell=1}^K (K-\ell)\lambda_\ell}{\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell + \sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}}$

Note that instantiating this bound with $K = 2$ types of jobs, n jobs per type, $\lambda_1 = 1$ and $\lambda_2 = \lambda > 1$, we get Proposition 4.3:

$$\frac{\mathbb{E}[C_{\mathrm{FTPP}}]}{\mathbb{E}[C_{\mathrm{OPT}}]} \le 2 - 4\frac{\lambda - 1}{(1+\lambda)^2 + 4\lambda}.$$

Proof of Proposition A.4. Compute $\mathbb{E}[C_{\mathrm{OPT}}]$ using Lemma A.1, $\mathbb{E}[C_{\mathrm{FTPP}}]$ using Lemma A.2.

The competitive ratio of FTPP is given by:

$$\mathrm{CR}_{\mathrm{FTPP}} = \frac{\mathbb{E}[C_{\mathrm{FTPP}}]}{\mathbb{E}[C_{\mathrm{OPT}}]} = \frac{n^2\left(\frac{1}{2}\sum_{\ell=1}^K \lambda_\ell + \sum_{\ell=1}^K (K-\ell)\lambda_\ell\right) + n(\frac{1}{2}\sum_{\ell=1}^K \lambda_\ell)}{n^2\left(\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell + \sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}\right) + n(\frac{3}{4}\sum_{\ell=1}^K \lambda_\ell)} \tag{7}$$

For any values $a, b, c, d \in \mathbb{R}_+^4$,

$$\text{if } a > c > 0 \text{ and } d > b > 0, \text{ then } \frac{a+b}{c+d} \le \frac{a}{c}. \tag{8}$$

Now, we have $\frac{1}{2} \le \frac{3}{4}$ and

$$\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell + \sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell} \le \frac{1}{2}\sum_{\ell=1}^K \lambda_\ell + \sum_{\ell=1}^K (K-\ell)\lambda_\ell.$$

This implies:

$$\begin{aligned}\mathrm{CR}_{\mathrm{FTPP}} &\le \frac{\frac{1}{2}\sum_{\ell=1}^K \lambda_\ell + \sum_{\ell=1}^K (K-\ell)\lambda_\ell}{\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell + \sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}} \\ &= 2 - \underbrace{\frac{2\sum_{\ell=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell} - \sum_{\ell=1}^K (K-\ell)\lambda_\ell}{\sum_{\ell=1}^K \frac{1}{4}\lambda_\ell + \sum_{k=1}^K \sum_{k=\ell+1}^K \frac{\lambda_k\lambda_\ell}{\lambda_k+\lambda_\ell}}}_{f_K(\boldsymbol{\lambda})}.\end{aligned}$$

□

A.2.2. Upper bound on the CR for particular values of $\boldsymbol{\lambda}$

Proposition A.5.

$$\forall K > 1, \exists \boldsymbol{\lambda}, 0 < \lambda_1 \le \dots \le \lambda_K = 1, \mathrm{CR}_{\mathrm{FTPP}}(\boldsymbol{\lambda}) \le \frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K}$$

with $H_K = \sum_{k=1}^K \frac{1}{k}$, $B_K = \sum_{k=1}^K \frac{1}{k^2}$ *and* $A_K = \sum_{k=1}^K \sum_{\ell=1}^{k-1} \frac{1}{k^2+\ell^2}$.

Furthermore $\lim_{K\to\infty} \frac{H_K - \frac{1}{2}B_K}{\frac{1}{4}B_K + A_K} = \frac{4}{\pi} \approx 1.273$ which implies that there exists some value of K for which $\mathrm{CR}_{\mathrm{FTPP}}(\boldsymbol{\lambda}) \le 1.274$.

Proof. A way to prove such a result would be to find the minimum of $\mathrm{CR}_{\mathrm{FTPP}}$ with respect to λ. But this is difficult. We propose another point $\tilde{\boldsymbol{\lambda}}$.

$$\tilde{\lambda}_k = \frac{1}{(K-k+1)^2}. \tag{9}$$

We express the competitive ratio using $\tilde{\boldsymbol{\lambda}}$:

$$\begin{aligned}
\mathrm{CR}_{\mathrm{FTPP}}(\tilde{\boldsymbol{\lambda}}) &= \frac{\sum_{k=1}^{K}(\frac{1}{2}+K-k)\tilde{\lambda}_k}{\sum_{k=1}^{K}\left(\frac{1}{4}\tilde{\lambda}_k+\sum_{\ell=k+1}^{K}\frac{\tilde{\lambda}_k\tilde{\lambda}_\ell}{\tilde{\lambda}_k+\tilde{\lambda}_\ell}\right)} \\
&= \frac{\sum_{k=1}^{K}(\frac{1}{2}+K-k)\frac{1}{(K-k+1)^2}}{\sum_{k=1}^{K}\left(\frac{1}{4}\frac{1}{(K-k+1)^2}+\sum_{\ell=k+1}^{K}\frac{1}{(K-k+1)^2+(K-\ell+1)^2}\right)} \\
&= \frac{\sum_{k=1}^{K}(\frac{1}{2}+k-1)\frac{1}{k^2}}{\sum_{k=1}^{K}\left(\frac{1}{4}\frac{1}{k^2}+\sum_{\ell=1}^{k-1}\frac{1}{k^2+\ell^2}\right)} \\
&= \frac{H_K-\frac{1}{2}B_K}{\frac{1}{4}B_K+A_K} \\
\text{with} \quad H_K=\sum_{k=1}^{K}\frac{1}{k}, &\quad B_K=\sum_{k=1}^{K}\frac{1}{k^2}, \quad \text{and} \quad A_K=\sum_{k=1}^{K}\sum_{\ell=1}^{k-1}\frac{1}{k^2+\ell^2}.
\end{aligned}$$

This shows the first part of the lemma. The fact that $\frac{H_K-\frac{1}{2}B_K}{\frac{1}{4}B_K+A_K}\to\frac{4}{\pi}$ follows from Lemma A.6. ☐

Lemma A.6.

$$\lim_{K\to\infty}\frac{H_K-\frac{1}{2}B_K}{\frac{1}{4}B_K+A_K}=\frac{4}{\pi}$$

with $H_K=\sum_{k=1}^{K}\frac{1}{k}$, $B_K=\sum_{k=1}^{K}\frac{1}{k^2}$ *and* $A_K=\sum_{k=1}^{K}\sum_{\ell=1}^{k-1}\frac{1}{k^2+\ell^2}$

Proof. We now focus on the behavior of $\frac{H_K-\frac{1}{2}B_K}{\frac{1}{4}B_K+A_K}$ as K goes to ∞.

We know that for the harmonic number $H_K=\Theta(\log(K))$, and that for the partial sum of the Basel problem $0\le B_K\le\sum_{k=1}^{\infty}k^{-2}=\pi^2/6=\mathcal{O}(1)$. Let us bound A_k. Using the fact that for $y>0$ and $x>0$ the function $f:(x,y)\mapsto(x^2+y^2)^{-1}$ is decreasing in x, for $(k,\ell)\in[K]^2$ we have

$$\begin{aligned}
\int_{\ell}^{\ell+1}\frac{1}{k^2+t^2}dt \le &\frac{1}{k^2+\ell^2}\le\int_{\ell-1}^{\ell}\frac{1}{k^2+t^2} \\
\frac{1}{k^2}\int_{\ell}^{\ell+1}\frac{1}{(t/k)^2+1}dt \le &\frac{1}{k^2+\ell^2}\le\frac{1}{k^2}\int_{\ell-1}^{\ell}\frac{1}{(t/k)^2+1}dt \\
\frac{1}{k}(\arctan(\frac{\ell+1}{k})-\arctan(\frac{\ell}{k}))\le &\frac{1}{k^2+\ell^2}\le\frac{1}{k}(\arctan(\frac{\ell}{k})-\arctan(\frac{\ell-1}{k})).
\end{aligned}$$

Hence by summing for $1\le\ell<k\le K$:

$$\begin{aligned}
\sum_{k=1}^{K}\frac{1}{k}(\arctan(1)-\arctan(\frac{1}{k}))\le A_K\le\sum_{k=1}^{K}\frac{1}{k}(\arctan(\frac{k-1}{k})-\arctan(0)), \\
\sum_{k=1}^{K}\frac{1}{k}(\frac{\pi}{4}-\arctan(\frac{1}{k}))\le A_k\le\sum_{k=1}^{K}\frac{1}{k}\arctan(\frac{k-1}{k}).
\end{aligned}$$

For the right-hand-side we use that $\arctan$ is increasing, thus

$$A_K\le\sum_{k=1}^{K}\frac{1}{k}\arctan(\frac{k-1}{k})\le\frac{\pi}{4}H_K.$$

Using that $\arctan(x)\le x$ for $x\ge 0$, we have

$$A_K\ge\sum_{k=1}^{K}\frac{1}{k}(\frac{\pi}{4}-\frac{1}{k})=\frac{\pi}{4}H_K-B_K.$$

Combining everything we obtain the following inequality:

$$\frac{H_K - \frac{1}{2}B_K}{\frac{\pi}{4}H_K + \frac{1}{4}B_K} \leq \mathrm{CR}_{\mathrm{FTPP}}(\tilde{\boldsymbol{\lambda}}) \leq \frac{H_K - \frac{1}{2}B_K}{\frac{\pi}{4}H_K - \frac{3}{4}B_K}.$$

Therefore

$$\lim_{K\to\infty} \mathrm{CR}_{\mathrm{FTPP}}(\tilde{\boldsymbol{\lambda}}) = \frac{4}{\pi} \approx 1.273.$$

□

A.2.3. The cost of FTPP is lower than the cost of RR

Let us order all jobs $i \in [N]$ in order of their increasing expected size, and denote P_i, the size of job i. An alternative notation to P_i is $P_{i \mod n}^{\lceil i/n \rceil}$, where the first is used in this proof for convenience. We consider here the most general setting where $K = N$.

We have

Lemma A.7.

$$\mathbb{E}[C_{\mathrm{FTPP}}] \leq \mathbb{E}[C_{\mathrm{RR}}]$$

Proof. The cost of FTPP with $K = N$ and $n = 1$ is given by

$$\begin{aligned}\mathbb{E}[C_{\mathrm{FTPP}}] &= \sum_{i=1}^{N} \mathbb{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathbb{E}\left[T_{ij}^{\mathrm{FTPP}}\right] \\ &= \sum_{i=1}^{N} \mathbb{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \min(\lambda_i, \lambda_j)\end{aligned}$$

where T_{ij}^{FTPP} is the amount of time job i and job j delay each other in FTPP which verifies

$\mathbb{E}\left[T_{ij}^{\mathrm{FTPP}}\right] = \min(\lambda_i, \lambda_j)$

Similarly, using $T_{ij}^{\mathrm{RR}} = 2\min(P_i, P_j)$ which implies $\mathbb{E}[T_{ij}^{\mathrm{RR}}] = 2\frac{\lambda_i\lambda_j}{\lambda_i+\lambda_j}$, we get

$$\begin{aligned}\mathbb{E}[C_{\mathrm{RR}}] &= \sum_{i=1}^{N} \mathbb{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathbb{E}\left[T_{ij}^{\mathrm{RR}}\right] \\ &= \sum_{i=1}^{N} \mathbb{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} 2\frac{\lambda_i\lambda_j}{\lambda_i + \lambda_j}\end{aligned}$$

Then we write

$$\begin{aligned}2\frac{\lambda_i\lambda_j}{\lambda_i + \lambda_j} &= \frac{2}{\frac{1}{\lambda_i} + \frac{1}{\lambda_j}} \\ &\geq \frac{2}{\frac{1}{\min(\lambda_i,\lambda_j)} + \frac{1}{\min(\lambda_i,\lambda_j)}} \\ &\geq \min(\lambda_i, \lambda_j)\end{aligned}$$

We conclude that $C_{\mathrm{FTPP}} \leq C_{\mathrm{RR}}$. □

A.3. Lower bound: Proof of Proposition 4.1

Let us order all jobs $i \in [N]$ in order of their increasing expected size, and denote P_i, the size of job i. An alternative notation to P_i is $P_{i \mod n}^{\lceil i/n \rceil}$, where the first is used in this proof for convenience. We consider here the most general setting where $K = N$. Any algorithm has a cost:

$$\mathbb{E}[C^A] = \sum_{i=1}^{N} \mathbb{E}[P_i] + \sum_{i=1}^{N} \sum_{j=i+1}^{N} \mathbb{E}[T_{ij}^A]$$

where $T_{ij}^A = D_{ij}^A + D_{ji}^A$ where D_{ij}^A is the amount of time job i delay job j.

Lemma A.8. *Consider $K = N$ jobs where job $i \in [N]$ has mean size λ_i and $\lambda_1 \leq \cdots \leq \lambda_N$. Consider any algorithm A and let T_{ij}^A the total amount of time spent by A on i or j while both jobs are alive.*

$$\mathbb{E}[T_{ij}^A] \geq 2\mathbb{E}[T_{ij}^{OPT}]$$

where OPT *is the optimal offline algorithm*

Proof of Lemma A.8. Let us first prove our proposition for any deterministic algorithm A. We denote $i(t)$ amount of time that A allocates to job i after a time $t < T_{ij}^A$ is allocated to job i or j.

$$\begin{aligned}
&\mathbb{E}[T_{ij}^A] \\
&= \int_{t=0}^{+\infty} \mathbb{P}(T_{ij}^A \geq t)dt \\
&= \int_{t=0}^{+\infty} \mathbb{P}\left(P_i \geq i(t)\right)\mathbb{P}\left(P_j \geq t - i(t)\right)dt \\
&= \int_{t=0}^{+\infty} \exp\left(-\frac{i(t)}{\lambda_i}\right)\exp\left(-\frac{t - i(t)}{\lambda_j}\right)dt \\
&= \int_{t=0}^{+\infty} \exp\left(-\frac{i(t) + t/2 - t/2}{\lambda_i}\right)\exp\left(-\frac{t - (i(t) + t/2 - t/2)}{\lambda_j}\right)dt \\
&= \int_{t=0}^{+\infty} \exp\left(-\left(\frac{1}{\lambda_i} + \frac{1}{\lambda_j}\right)\frac{t}{2}\right)\exp\left(-\left(\frac{1}{\lambda_i} - \frac{1}{\lambda_j}\right)\left(i(t) - \frac{t}{2}\right)\right)dt.
\end{aligned}$$

Calling $f(t) = \exp\left(-\left(\frac{1}{\lambda_i} + \frac{1}{\lambda_j}\right)\frac{t}{2}\right)$ and $g(t) = |\frac{1}{\lambda_i} - \frac{1}{\lambda_j}|\left(i(t) - \frac{t}{2}\right)$ it holds that either

$$\int_{t=0}^{\infty} f(t)\exp(-g(t))dt \geq \int_{t=0}^{\infty} f(t)dt$$

or

$$\int_{t=0}^{\infty} f(t)\exp(g(t))dt \geq \int_{t=0}^{\infty} f(t)dt.$$

Otherwise, we would have

$$\int_{t=0}^{\infty} f(t)\frac{1}{2}(\exp(-g(t)) + \exp(g(t)))dt < \int_{t=0}^{\infty} f(t)dt$$

which cannot be true since $\forall t$, $\frac{1}{2}(\exp(-t) + \exp(t)) \geq 1$.

Therefore an adversary knowing $i(t)$ can always chose the order of λ_i and λ_j such that

$$\mathbb{E}[T_{ij}^A] \geq \int_{t=0}^{+\infty} \exp(-(\frac{1}{\lambda_i} + \frac{1}{\lambda_j})\frac{t}{2})dt = 2\frac{\lambda_i\lambda_j}{\lambda_i + \lambda_j}$$

The optimal delay is

$$\mathbb{E}[T_{ij}^{OPT}] = \mathbb{E}[\min(P_i, P_j)] = \frac{\lambda_i \lambda_j}{\lambda_i + \lambda_j}$$

so our Lemma is proven for any deterministic algorithm A.

Consider a randomized algorithm R which can be seen as a probabilistic distribution over the set of deterministic algorithms. Therefore A, $i(t)$ and $g(t)$ are now seen as random variables. By the tower rule, the amount of time job i and j delay each other in R is such that:

$$\begin{aligned}\mathbb{E}[T_{ij}^{R}] &= \mathbb{E}[\mathbb{E}[T_{ij}^{A}|A]] \\ &= \mathbb{E}[\int_{t=0}^{+\infty} f(t)\exp(\text{sign}(\lambda_i - \lambda_j) g(t)) dt]\end{aligned}$$

By the same argument as in the deterministic case, it holds that either

$$\mathbb{E}[\int_{t=0}^{\infty} f(t)\exp(-g(t))dt] \geq \int_{t=0}^{\infty} f(t)dt$$

or

$$\mathbb{E}[\int_{t=0}^{\infty} f(t)\exp(g(t))dt] \geq \int_{t=0}^{\infty} f(t)dt$$

Otherwise, we would have

$$\mathbb{E}[\int_{t=0}^{\infty} f(t)\frac{1}{2}(\exp(-g(t)) + \exp(g(t)))dt] < \int_{t=0}^{\infty} f(t)dt$$

which implies that there exists a deterministic function g such that

$$\int_{t=0}^{\infty} f(t)\frac{1}{2}(\exp(-g(t)) + \exp(g(t)))dt < \int_{t=0}^{\infty} f(t)dt$$

which cannot be true as shown in the deterministic case. The rest of the argument is the same as in the deterministic case and therefore omitted. □

Now we are ready to prove Proposition 4.1.

Proof of Proposition 4.1. Take any algorithm A

$$\mathbb{E}[C_A] = \sum_{i=1}^{N} \mathbb{E}[P_i] + \sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathbb{E}[T_{ij}^{A}] \quad (10)$$

$$\geq \sum_{i=1}^{N} \lambda_i + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathbb{E}[T_{ij}^{OPT}] \quad (11)$$

where (11) comes from Lemma A.8.

Observe that applying RR on the same data would yield an expected completion time:

$$\begin{aligned}\mathbb{E}[C_{\text{RR}}] &= \sum_{i=1}^{N} \mathbb{E}[P_i] + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathbb{E}[\min(P_i, P_j)] \\ &= \sum_{i=1}^{N} \mathbb{E}[P_i] + 2\sum_{i=1}^{N}\sum_{j=i+1}^{N} \mathbb{E}[T_{ij}^{OPT}] \\ &\leq \mathbb{E}[C_A]\end{aligned}$$

which concludes the proof. □

B. Analysis of Non-Preemptive Learning algorithms

B.1. Full Algorithmic Details

In this appendix, we present a full description of ETC-U and UCB-U.

Algorithm 3 Explore-Then-Commit Uniform (ETC-U)]

```
1: **Input :** $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types)
2: For all pairs of different types $k, \ell$ initialize $\delta_{k,\ell} = 0$, $\hat{r}_{k,\ell} = 0$ and $h_{k,\ell} = 0$
3: For all types $k$, set $m_k = 0$
4: **repeat**
5:   $\mathcal{U}$ is the set of types with at least one remaining job
6:   **if** $\mathcal{A}$ is empty **then**
7:     $\mathcal{A} = \{\ell \in \mathcal{U}, \forall k \in \mathcal{U}, k \neq \ell,\ \hat{r}_{k,\ell} - \delta_{k,\ell} \leq 0.5\}$
8:   **end if**
9:   Select the type $\ell$ with the lowest number of finished jobs $\ell = \mathrm{argmin}_{k\in\mathcal{A}} m_k$ and run one job of type $\ell$ yielding a
     size $P^{\ell}_{m_\ell+1}$.
10:  $m_\ell = m_\ell + 1$
11:  **for** $k, \ell$ in $\mathcal{A}$, $k \neq \ell$ **do**
12:    $h_{k,\ell} = \sum_{i=1}^{\min(m_k,m_\ell)} \mathbb{1}\{P_i^k < P_i^\ell\}$
13:    $\delta_{k,\ell} = \sqrt{\frac{\log(2n^2K^4)}{2\min(m_k,m_\ell)}}$
14:    $\hat{r}_{k,\ell} = \frac{h_{k,\ell}}{\min(m_k,m_\ell)}$
15:    **if** $\hat{r}_{k,\ell} - \delta_{k,\ell} \geq 0.5$ or $m_\ell = n$ **then**
16:      Remove $\ell$ from $\mathcal{A}$
17:    **end if**
18:  **end for**
19: **until** $\mathcal{U}$ is not empty
```

Algorithm 4 Upper-Confidence-Bound-Uniform (UCB-U)

```
1: **Input :** $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types)
2: For all types $k \in [K]$, set $m_k = 0$
3: Set $\mathcal{U} = [K]$
4: For all types $k \in [K]$, compute the lower bound $\underline{\lambda}_k^{m_k}$ using Equation (16)
5: **repeat**
6:   Select $k^* = \mathrm{argmin}_{k\in\mathcal{U}} \underline{\lambda}_k^{m_k}$
7:   Set $m_{k^*} = m_{k^*} + 1$
8:   Compute a job of type $k^*$ until completion and record its size $P_{k^*}^{m_{k^*}}$
9:   Update the lower bound $\underline{\lambda}_{k^*}^{m_{k^*}}$ using again Equation (16)
10:  If $m_{k^*} = n$, remove $k^*$ from $\mathcal{U}$
11: **until** $\mathcal{U}$ is empty
```

B.2. Cost Decomposition

In this appendix, we analyze the non-preemptive learning algorithms presented in our paper - ETC-U and UCB-U. We start by presenting a general cost decomposition result that relates the cost of any non-preemptive algorithm to the one of FTPP. We will use this result to derive the bounds of both our suggested algorithms.

Lemma B.1 (Cost of non-preemptive algorithms). *Any non-preemptive algorithm A has a cost*

$$\mathbb{E}[C_A] = \mathbb{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right]$$

Proof. Denote P_i, the size of the job i, and $T_{ij}^A = D_{ij}^A + D_{ji}^A$, where D_{ij}^A is the amount of time a job i delays job j. For any algorithm, we have:

$$C_A = \sum_{a=1}^{N} P_a + \sum_{a=1}^{N} \sum_{b=a+1}^{N} T_{ab}^A$$

For non-preemptive algorithms, $T_{ab}^A = P_a$ if job a is scheduled before b and P_b otherwise so that we can write

$$C_A = \sum_{a=1}^{N} P_a + \sum_{a=1}^{N} \sum_{b=a+1}^{N} \left(P_a \mathbb{1}\{P_a \text{ computed before } P_b\} + P_b \mathbb{1}\{P_b \text{ computed before } P_a\}\right)$$

Now assume w.l.o.g. that $(P_a)_{a\in[N]}$ are in the order chosen by FTPP, i.e., P_a is the a^{th} executed task by FTPP and if $a \le b$ then $\mathbb{E}[P_a] \le \mathbb{E}[P_b]$. Under this convention, we get:

$$C_{\text{FTPP}} = \sum_{a=1}^{N} P_a + \sum_{a=1}^{N} \sum_{b=a+1}^{N} P_a$$

and recalling that

$$\mathbb{1}\{P_a \text{ computed before } P_b\} = 1 - \mathbb{1}\{P_b \text{ computed before } P_a\}$$

we have

$$C_A = C_{\text{FTPP}} + \sum_{a=1}^{N} \sum_{b=a+1}^{N} (P_b - P_a)\mathbb{1}\{P_b \text{ computed before } P_a\}$$

Reindexing the job without changing the order, where P_i^k is now the i-th job of type k, we have:

$$C_A = C_{\text{FTPP}} + \sum_{\ell=1}^{K}\sum_{j=1}^{n}\sum_{k=\ell+1}^{K}\sum_{i=1}^{n}(P_i^k - P_j^\ell)\mathbb{1}\{P_i^k \text{ computed before } P_j^\ell\}$$

Taking the expectation finishes the proof. □

B.3. Upper bound for ETC-U

Proposition B.2. *The following upper bounds hold:*

$$\mathbb{E}[C_{\text{ETC-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + \frac{1}{n}\mathbb{E}[C_{\text{OPT}}] + \sum_{k\in[K]} \left[\frac{1}{2}(k-1)(2K-k) + (K-k)^2\right] \lambda_k n \sqrt{8n\log(2n^2K^3)}.$$

and

$$\mathbb{E}[C_{\text{ETC-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + \frac{1}{n}\mathbb{E}[C_{\text{OPT}}] + \sum_{k\in[K]}\sum_{\ell=1}^{k-1}(K-\ell)\frac{(\lambda_k+\lambda_\ell)^2}{(\lambda_k-\lambda_\ell)} n 8\log(2n^2K^3).$$

We start with the following technical lemma, isolated to be reused in other proofs. Pick some $\alpha \in \mathbb{N}$. Let $(X_i^1)_{i\in[\alpha n]}$ and $(X_i^2)_{i\in[\alpha n]}$ be independent exponential variables of parameters λ_1 and λ_2 respectively. Define for any $m \in [\alpha n]$:

$$\hat{r}^m = \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}_{X_i^1 < X_i^2}$$

and

$$\delta^{(m,n)} = \sqrt{\frac{\log(2n^2K^3)}{2m}}.$$

Let r denote the expectation $r := \mathbb{E}\left[\mathbb{1}_{X_i^1<X_i^2}\right] = \frac{\lambda_2}{\lambda_1+\lambda_2}$.

Lemma B.3. *For any $m \in [\alpha n]$, the estimator $\hat{r}^m$ is within $\delta^{(m,n)}$ of its expectation w.h.p:*

$$\mathbb{P}\left(\exists m \in [\alpha n] \text{ s.t. } |\hat{r}^m - r| \geq \delta^{(m,n)}\right) \leq \frac{\alpha}{nK^3}.$$

Proof. By Hoeffding's inequality:

$$\forall m \in [\alpha n],\ \mathbb{P}\left(|\hat{r}^m - r| \geq \sqrt{\frac{\log(2n^2K^3)}{2m}}\right) \leq \frac{1}{n^2K^3}$$

The lemma is then obtained by a union bound over the αn possible values of m. □

We are now ready to prove Proposition B.2.

Proof. Recall that we assumed without loss of generality that $\lambda_1 \leq \cdots \leq \lambda_K$. Recall also the definition for any $(k,\ell) \in [K]^2$, for any $(m_\ell, m_k) \in [n]^2$, of:

$$\hat{r}_{k,\ell}^{\min(m_k,m_\ell)} = \frac{1}{\min(m_k,m_\ell)} \sum_{i=1}^{\min(m_k,m_\ell)} \mathbb{1}_{P_i^k < P_i^\ell}.$$

Let us define the good event $\mathcal{E}$ as:

$$\mathcal{E} := \left\{\forall (k,\ell) \in [K]^2, \forall m \in [n], |\hat{r}_{k,\ell}^m - \mathbb{E}[r_{k,\ell}^m]| < \delta^{(m,n)}\right\}$$

By Lemma B.3 applied with $\alpha = 1$, for any couple (ℓ, k) it holds that :

$$\mathbb{P}\left(\exists m \in [n] \text{ s.t. } |\hat{r}_{k,\ell}^m - \mathbb{E}[r_{k,\ell}^m]| > \delta^{(m,n)}\right) \leq \frac{1}{n}.$$

A union bound over the $\frac{K(K-1)}{2}$ possible pairs gives the following bound:

$$\mathbb{P}\left(\overline{\mathcal{E}}\right) \leq \frac{1}{2nK}. \tag{12}$$

With the help of Lemma B.1, the cost of ETC-U can be decomposed using the event $\mathcal{E}$ as follows:

$$\begin{aligned}
\mathbb{E}[C_{\text{ETC-U}}] &= \mathbb{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right] && \text{(Lemma B.1)} \\
&\leq \mathbb{E}[C_{\text{FTPP}}] + \underbrace{\sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\} |\mathcal{E}\right]}_{(i)} \\
&\quad + \underbrace{\sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\} |\overline{\mathcal{E}}\right] \mathbb{P}\left(\overline{\mathcal{E}}\right)}_{(ii)}. && (13)
\end{aligned}$$

Bounding (ii)**.** Recall that by assumption, if $k \geq \ell$, then $\lambda_k \geq \lambda_\ell$. Therefore, we have that

$$\begin{aligned}
(ii) &= \mathbb{P}\left(\overline{\mathcal{E}}\right) \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} \underbrace{(\lambda_k - \lambda_\ell)}_{\geq 0} \mathbb{E}\left[\underbrace{\mathbb{1}\{P_i^k \text{ computed before } P_j^\ell\}}_{\leq 1} | \overline{\mathcal{E}}\right] \\
&\leq \mathbb{P}\left(\overline{\mathcal{E}}\right) \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} (\lambda_k - \lambda_\ell) \\
&= n^2 \mathbb{P}\left(\overline{\mathcal{E}}\right) \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \\
&= n^2 \mathbb{P}\left(\overline{\mathcal{E}}\right) \left(\sum_{k=1}^{K} (k-1)\lambda_k - \sum_{\ell=1}^{K} (K-\ell)\lambda_\ell \right) \\
&\leq n^2 K \mathbb{P}\left(\overline{\mathcal{E}}\right) \sum_{k=1}^{K} \lambda_k \\
&\leq 4K \mathbb{E}[C_{\mathrm{OPT}}] \mathbb{P}\left(\overline{\mathcal{E}}\right) && \text{(Equation (6))} \\
&\leq \frac{2}{n} \mathbb{E}[C_{\mathrm{OPT}}], && (14)
\end{aligned}$$

where the last inequality is by Equation (12).

Bounding (i)**.** Consider any couple $(k, \ell) \in [K]^2$ s.t. $\ell \leq k$. Let $m^*_{\ell,k}$ be the number of comparisons performed between jobs of type ℓ and k before the algorithm detects that $\lambda_\ell \leq \lambda_k$. A first obvious upper bound is $m^*_{\ell,k} \leq n$. A second upper is obtained by noting that $m^*_{\ell,k}$ is smaller than any m' s.t.

$$\delta^{(m',n)} < \frac{1}{2}\left|\frac{\lambda_k}{\lambda_k + \lambda_\ell} - 0.5\right|.$$

For this value of $\delta^{(m',n)}$, the event $\mathcal{E}$ ensures that if $\lambda_k \geq \lambda_\ell$, then

$$\hat{r}^{m'}_{\ell,k} - \delta^{(m',n)} \overset{\text{Under } \mathcal{E}}{\geq} \mathbb{E}[r^{m'}_{\ell,k}] - 2\delta^{(m',n)} > \frac{\lambda_k}{\lambda_k + \lambda_\ell} - \left(\frac{\lambda_k}{\lambda_k + \lambda_\ell} - \frac{1}{2}\right) = \frac{1}{2},$$

and type k would be eliminated. This implies the following upper bound on $m^*_{\ell,k}$:

$$m^*_{\ell,k} \leq \min\left(n, 8\left(\frac{\lambda_k + \lambda_\ell}{\lambda_k - \lambda_\ell}\right)^2 \log\left(2n^2K^3\right)\right). \tag{15}$$

On the other hand, notice that under the good event $\mathcal{E}$, a type ℓ will never be eliminated due to a type k of greater expected duration $\lambda_k \geq \lambda_\ell$, since

$$\hat{r}^{m'}_{k,\ell} - \delta^{(m',n)} \overset{\text{Under } \mathcal{E}}{\leq} \left(\mathbb{E}[r^{m'}_{\ell,k}] + \delta^{(m',n)}\right) - \delta^{(m',n)} = \frac{\lambda_\ell}{\lambda_k + \lambda_\ell} \leq \frac{1}{2}.$$

We decompose the run of the algorithm into (up to K) phases. For each $\ell \in [K]$, we call phase ℓ the iterations at which jobs of type ℓ are the jobs with the smallest mean still not terminated. Note that during phase ℓ, job type ℓ is always active, as the contrary would mean event $\mathcal{E}$ does not hold. This implies that the number of jobs of any type $k > \ell$ computed during phase ℓ is lower than $m^*_{\ell,k}$. We have the following bound:

$$
\begin{aligned}
(i) &= \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}|\mathcal{E}\right]\\
&\overset{(1)}{\leq} \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before phase } \ell+1\right\}|\mathcal{E}\right]\\
&\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{i\in[n]}n(\lambda_k-\lambda_\ell)\mathbb{E}\left[\sum_{o=1}^{\ell}\mathbb{1}\left\{P_i^k \text{ computed during phase } o\right\}|\mathcal{E}\right]\\
&\overset{(2)}{\leq} \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{o=1}^{\ell}\mathbb{E}[m^*_{o,k}|\mathcal{E}]n(\lambda_k-\lambda_\ell)\\
&\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{o=1}^{\ell}\mathbb{E}[m^*_{o,k}|\mathcal{E}]n(\lambda_k-\lambda_o)\\
&\overset{(3)}{=} \sum_{k=1}^{K}\sum_{o=1}^{k-1}\sum_{l=o}^{k-1}\mathbb{E}[m^*_{o,k}|\mathcal{E}]n(\lambda_k-\lambda_o)\\
&= \sum_{k=1}^{K}\sum_{o=1}^{k-1}\mathbb{E}[m^*_{o,k}|\mathcal{E}]n(k-o)(\lambda_k-\lambda_o)\\
&\overset{(4)}{\leq} \sum_{k=1}^{K}\sum_{\ell=1}^{k-1}\mathbb{E}[m^*_{\ell,k}|\mathcal{E}]n(K-\ell)(\lambda_k-\lambda_\ell)\\
&\overset{(5)}{\leq} \sum_{k=1}^{K}\sum_{\ell=1}^{k-1}\min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2\log\left(2n^2K^3\right)\right)n(K-\ell)(\lambda_k-\lambda_\ell).
\end{aligned}
$$

(1) is since by the beginning of phase $\ell+1$, all jobs of type ℓ were completed. (2) is since during phase o, the o^{th} type was not eliminated, so there cannot be more than $m^*_{o,k}$ jobs of type k in this phase. In (3), we changed the summation order and in (4), we replaced $o\to\ell$. Finally, (5) is due to the bound of Equation (15), which holds under $\mathcal{E}$.

Next, for any $\lambda_k\geq\lambda_\ell$, we have:

$$
(\lambda_k-\lambda_\ell)\min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2\log\left(2n^2K^3\right)\right)\leq(\lambda_k+\lambda_\ell)\sqrt{8n\log(2n^2K^3)},
$$

since $\min\{a,b\}\leq\sqrt{ab}$ for any $a,b\geq 0$. This implies that

$$
\begin{aligned}
(i) &\leq \sum_{k=1}^{K}\sum_{\ell=1}^{k-1}n(K-\ell)(\lambda_k+\lambda_\ell)\sqrt{8n\log(2n^2K^3)}\\
&= \sum_{k=1}^{K}\left[\frac{1}{2}(k-1)(2K-k)+(K-k)^2\right]\lambda_k n\sqrt{8n\log(2n^2K^3)}.
\end{aligned}
$$

Substituting this and the bound of Equation (14) into the decomposition of Equation (13) gives the first bound of the proposition.

The second bound is obtained by upper bounding:

$$
(\lambda_k-\lambda_\ell)\min\left(n, 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2\log\left(2n^2K^3\right)\right)\leq 8\frac{(\lambda_k+\lambda_\ell)^2}{\lambda_k-\lambda_\ell}\log\left(2n^2K^3\right),
$$

□

B.4. Upper bound for UCB-U

Proposition B.4. *The expected cost of* UCB-U *is upper bounded by:*

$$\mathbb{E}[C_{\text{UCB-U}}] \leq \mathbb{E}[C_{\text{FTPP}}] + n(K-1)\sqrt{3n\ln(2n^2K^2)}\sum_{k=1}^{K}\lambda_k + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}],$$

and:

$$\mathbb{E}[C_{\text{UCB-U}}] \leq \mathbb{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\frac{(\lambda_k+\lambda_\ell)^2}{\lambda_k-\lambda_\ell}3n\ln\left(2n^2K^2\right) + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}].$$

Concentration of exponential distribution If $X \sim \mathcal{E}(\lambda)$, then $\frac{2}{\lambda}X \sim \mathcal{E}(2) = \chi^2_2$ (χ^2 with 2 degrees of freedom). It follows that if $\forall i \in [m], X_i \sim \mathcal{E}(\lambda)$, then $\frac{2}{\lambda}\sum_{i=1}^m X_i \sim \chi^2_{2m}$. Denote $\chi^2_{2m}(\alpha)$ the α-th percentile, we have with probability $1-\delta$ that

$$\frac{2\sum_i X_i}{\chi^2_{2m}(1-\delta/2)} \leq \lambda \leq \frac{2\sum_i X_i}{\chi^2_{2m}(\delta/2)}$$

Setting $\delta = \frac{1}{n^2K^2}$, we get the following formula for a lower bound:

$$\underline{\lambda}_k^m = \frac{2\sum_{i=1}^m X_i^k}{\chi^2_{2m}(1-\frac{1}{2n^2K^2})} \tag{16}$$

and another formula for the upper bound

$$\overline{\lambda}_k^m = \frac{2\sum_{i=1}^m X_i^k}{\chi^2_{2m}(\frac{1}{2n^2K^2})}$$

If a job k is wrongly scheduled before a job of type ℓ, then the decision rule is misleading meaning that:

$$\underline{\lambda}_k^{m_k} = \frac{2\sum_{i=1}^{n_k} X_i^k}{\chi^2_{2n_k}(1-\frac{1}{2n^2K^2})} < \frac{2\sum_{i=1}^{n_\ell} X_i^\ell}{\chi^2_{2n_\ell}(1-\frac{1}{2n^2K^2})} = \underline{\lambda}_\ell^{m_\ell}$$

even though $\lambda_\ell < \lambda_k$.

Bounding the cost From Lemma B.1, the cost of any non preemptive algorithm A writes

$$\mathbb{E}[C_A] = \mathbb{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right] \tag{17}$$

$$\tag{18}$$

Let us then introduce the GOOD event which is:

$$\mathcal{E} = \{\forall i \in [n], \forall k \in [K], \underline{\lambda}_k^i \leq \lambda_k \leq \overline{\lambda}_k^i\}$$

With a union bound, it is easy to show that $\mathcal{E}$ holds with probability $1-\frac{1}{nK}$ and that the contradictory event $\overline{\mathcal{E}}$ happens with probability $\frac{1}{nK}$.

Using the same method as in the proof of Proposition B.2 (the decomposition using $\mathcal{E}$ and $\overline{\mathcal{E}}$ as done in Equation (13) and the derivation of Equation (14)), we can upper bound the cost of UCB-U as:

$$\mathbb{E}[C_{UCB-U}] \leq \mathbb{E}[C_{\text{FTPP}}] + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}|\mathcal{E}\right]}_{(i)}$$
$$+ \mathbb{E}[C_{OPT}]\underbrace{4KP(\overline{\mathcal{E}})}_{4/n}$$

Furthermore, P_i^k computed before P_j^ℓ implies that $\underline{\lambda}_k^i < \underline{\lambda}_\ell^j$ and therefore

$$(i) \le \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} (\lambda_k - \lambda_\ell)\mathbb{P}(\underline{\lambda}_k^i < \underline{\lambda}_\ell^j | \mathcal{E})$$

Under $\mathcal{E}$, we have $\underline{\lambda}_\ell^j \le \lambda_\ell$. Moreover, it holds that $\overline{\lambda}_k^i \ge \lambda_k$, and by the definition of $\underline{\lambda}_k^i$, and $\overline{\lambda}_k^i$,

$$\underline{\lambda}_k^i = \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})}\overline{\lambda}_k^i \ge \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})}\lambda_k.$$

Combined, under $\mathcal{E}$ we can bound $\left\{\underline{\lambda}_k^i < \underline{\lambda}_\ell^j\right\} \subseteq \left\{\lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}$ and therefore write

$$\begin{aligned}(i) &\le \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} \mathbb{1}\left\{\lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}(\lambda_k - \lambda_\ell)\\ &= \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n \max\left\{i\in[n], \lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}(\lambda_k - \lambda_\ell)\end{aligned}$$

So finally we have

$$\mathbb{E}[C_{\text{UCB-U}}] \le \mathbb{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n \max\left\{i\in[n], \lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\}(\lambda_k - \lambda_\ell) + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}] \tag{19}$$

Bounding the ratio. We now focus on bounding the maximum term in Equation (19). By Lemma 1 of (Laurent & Massart, 2000), if $U \sim \chi^2_D$, then

$$\mathbb{P}\left(U \ge D + 2\sqrt{Dx} + 2x\right) \le \exp(-x), \qquad \text{and} \qquad \mathbb{P}\left(U \le D - 2\sqrt{Dx}\right) \le \exp(-x), \tag{20}$$

and in particular,

$$\chi^2_D(\delta) \ge D - 2\sqrt{D\ln\frac{1}{\delta}}, \qquad \text{and} \qquad \chi^2_D(1-\delta) \le D + 2\sqrt{D\ln\frac{1}{\delta}} + 2\ln\frac{1}{\delta}. \tag{21}$$

Thus, for any $i \in [n]$, a necessary condition to the inequality $\lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell$ is

$$\begin{aligned}&\frac{2i - 2\sqrt{2i\ln(2n^2K^2)}}{2i + 2\sqrt{2i\ln(2n^2K^2)} + 2\ln(2n^2K^2)} < \frac{\lambda_\ell}{\lambda_k}\\ &\Rightarrow \left(1 - \frac{\lambda_\ell}{\lambda_k}\right)i - \sqrt{2\ln(2n^2K^2)}\left(1 + \frac{\lambda_\ell}{\lambda_k}\right)\sqrt{i} - \frac{\lambda_\ell}{\lambda_k}\ln\left(2n^2K^2\right) < 0\\ &\Rightarrow (\lambda_k - \lambda_\ell)\,i - \sqrt{2\ln(2n^2K^2)}\,(\lambda_k + \lambda_\ell)\sqrt{i} - \lambda_\ell \ln\left(2n^2K^2\right) < 0\\ &\Rightarrow \sqrt{i} < \frac{\sqrt{2\ln(2n^2K^2)}\,(\lambda_k + \lambda_\ell) + \sqrt{2\ln(2n^2K^2)\,(\lambda_k + \lambda_\ell)^2 + 4\ln(2n^2K^2)\,\lambda_\ell\,(\lambda_k - \lambda_\ell)}}{2\,(\lambda_k - \lambda_\ell)}\\ &\Rightarrow \sqrt{i} < \sqrt{2\ln(2n^2K^2)}\frac{(\lambda_k + \lambda_\ell) + \sqrt{(\lambda_k + \lambda_\ell)^2 + 2\lambda_\ell\,(\lambda_k - \lambda_\ell)}}{2\,(\lambda_k - \lambda_\ell)}\end{aligned}$$

Now, using the fact that $2\lambda_\ell \le \lambda_\ell + \lambda_k$, we get the simplified bound

$$\sqrt{i} < \sqrt{2\ln(2n^2K^2)}\frac{(\lambda_k + \lambda_\ell) + \sqrt{2\lambda_k\,(\lambda_k + \lambda_\ell)}}{2\,(\lambda_k - \lambda_\ell)} \le \sqrt{2\ln(2n^2K^2)}\left(1 + \sqrt{2}\right)\frac{\lambda_k + \lambda_\ell}{2\,(\lambda_k - \lambda_\ell)}\ , \tag{22}$$

or $i \leq 3\ln\left(2n^2K^2\right)\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2$. Since we also know that $i \in [n]$, we can write

$$\begin{aligned}\max\left\{i \in [n], \lambda_k \frac{\chi^2_{2i}(\frac{1}{2n^2K^2})}{\chi^2_{2i}(1-\frac{1}{2n^2K^2})} < \lambda_\ell\right\} &\leq \min\left\{3\ln\left(2n^2K^2\right)\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2, n\right\} \\ &\leq \sqrt{3n\ln\left(2n^2K^2\right)}\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell},\end{aligned}$$

where the second inequality is since $\min\{a,b\} \leq \sqrt{ab}$ for $a, b > 0$. Substituting back into Equation (19), we get the first bound in the proposition:

$$\begin{aligned}\mathbb{E}[C_{\text{UCB-U}}] &\leq \mathbb{E}[C_{\text{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n\sqrt{3n\ln\left(2n^2K^2\right)}\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}(\lambda_k-\lambda_\ell) + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}] \\ &= \mathbb{E}[C_{\text{FTPP}}] + n\sqrt{3n\ln\left(2n^2K^2\right)}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k+\lambda_\ell) + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}] \\ &= \mathbb{E}[C_{\text{FTPP}}] + n\sqrt{3n\ln\left(2n^2K^2\right)}\left(\sum_{k=1}^{K}(k-1)\lambda_k + \sum_{\ell=1}^{K}(K-\ell)\lambda_\ell\right) + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}] \\ &= \mathbb{E}[C_{\text{FTPP}}] + n(K-1)\sqrt{3n\ln\left(2n^2K^2\right)}\sum_{k=1}^{K}\lambda_k + \frac{2}{n}\mathbb{E}[C_{\text{OPT}}].\end{aligned}$$

The second bound is obtained through the upper bound:

$$(\lambda_k-\lambda_\ell)\min\left\{3\ln\left(2n^2K^2\right)\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2, n\right\} \leq 3\ln\left(2n^2K^2\right)\left(\frac{(\lambda_k+\lambda_\ell)^2}{\lambda_k-\lambda_\ell}\right).$$

B.5. Lower bounds for Non-Preemptive Algorithms

B.5.1. Small Differences (Proposition 5.4)

Proof of Proposition 5.4. Assume $K = 2$ and take any non-preemptive algorithm A. Call P_i^1 the i-th job of type 1 and P_j^2 the j-th job of type 2. According to Lemma B.1, A has a cost

$$\mathbb{E}[C_A] = \mathbb{E}[C_{\text{FTPP}}] + (\lambda_2-\lambda_1)\mathbb{E}\left[\sum_{(i,j)\in[n]^2}\mathbb{1}\left\{P_j^2 \text{ computed before } P_i^1\right\}\right]$$

if $\lambda_2 > \lambda_1$ (the role of λ_2 and λ_1 are reversed if $\lambda_2 < \lambda_1$).

We then follow the same approach as in chapter 15 in (Lattimore & Szepesvári, 2020). Consider situation 1 where $\lambda_1 = a, \lambda_2 = b$ and situation 2 where $\lambda_1 = b$ and $\lambda_2 = a$ with $a < b$ and assumes that the adversary chooses the situation based on the algorithm A. Intuitively, if A tends to complete more of jobs of type 1 before jobs of type 2, the adversary will decide that $\lambda_1 > \lambda_2$ (situation 2) otherwise, it will choose $\lambda_2 > \lambda_1$ (situation 1). Call $\mathbb{P}_{\nu_1}$ the joint probability over the scheduling decisions and job sizes in situation 1 following the policy prescribed by algorithm A and $\mathbb{P}_{\nu_2}$ the same probability in situation 2. Call $P_{a_t}(x_t)$ the probability that the job of type a_t chosen at time t is of size x_t. Calling KL the KL divergence, we have following the Lemma 15.1 in (Lattimore & Szepesvári, 2020): $KL(\mathbb{P}_{\nu_1}, \mathbb{P}_{\nu_2}) = n(KL(X_a, X_b) + KL(X_b, X_a))$ where X_a is an exponential random variable of expectation a and X_b is an exponential random variable of expectation b.

Note right away that $KL(X_a, X_b) = \frac{a}{b} - 1 - \log(\frac{a}{b})$ (*e.g.,* Calin & Udrişte, 2014, Example 4.2.1), therefore $KL(X_a, X_b) + KL(X_b, X_a) = \frac{a}{b} - 2 + \frac{b}{a} = \frac{(b-a)^2}{ab}$ so

$$KL(\mathbb{P}_{\nu_1}, \mathbb{P}_{\nu_2}) \leq n\frac{(b-a)^2}{ab}.$$

The cost of algorithm A in situation 1 is lower bounded as:

$$\mathbb{E}_{\nu_1}[C_A] \geq \mathbb{E}_{\nu_1}[C_{\text{FTPP}}] + (b-a)\mathbb{E}_{\nu_1}\left[\mathbb{1}\left\{\sum_{(i,j)\in[n]^2}\mathbb{1}\left\{P_j^2 \text{ computed before } P_i^1\right\} \geq n^2/2\right\}\right] n^2/2.$$

The cost of algorithm A in situation 2 is lower bounded as:

$$\mathbb{E}_{\nu_1}[C_A] \geq \mathbb{E}_{\nu_2}[C_{\mathrm{FTPP}}] + (b-a)\mathbb{E}_{\nu_2}\left[\mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\{P_i^1 \text{ computed before } P_j^2\} > n^2/2\right\}\right] n^2/2.$$

Introduce the event $E = \mathbb{1}\left\{\sum_{(i,j)\in[n]^2} \mathbb{1}\{P_j^2 \text{ computed before } P_i^1\} \geq n^2/2\right\}$, we have that

$$\begin{aligned}\mathbb{E}[C_A] &= \max_{\nu\in\{\nu_1,\nu_2\}} \mathbb{E}_\nu[C_A] \\ &\geq \frac{\mathbb{E}_{\nu_1}[C_A] + \mathbb{E}_{\nu_2}[C_A]}{2} \\ &\geq \frac{\mathbb{E}_{\nu_1}[C_{\mathrm{FTPP}}] + \mathbb{E}_{\nu_2}[C_{\mathrm{FTPP}}]}{2} + (b-a)n^2/2\frac{\mathbb{P}_{\nu_1}(E) + \mathbb{P}_{\nu_2}(\overline{E})}{2}\end{aligned}$$

First, let us notice that

$$\mathbb{E}[C_{\mathrm{FTPP}}] = \frac{\mathbb{E}_{\nu_1}[C_{\mathrm{FTPP}}] + \mathbb{E}_{\nu_2}[C_{\mathrm{FTPP}}]}{2}$$

Then, using Bretagnolle–Huber inequality (Th 14.2 in (Lattimore & Szepesvári, 2020)), we get $\mathbb{P}_{\nu_1}(E) + \mathbb{P}_{\nu_2}(\overline{E}) \geq \frac{1}{2}\exp(-KL(\mathbb{P}_{\nu_1}, \mathbb{P}_{\nu_2}))$ and since $KL(\mathbb{P}_{\nu_1}, \mathbb{P}_{\nu_2}) \leq n\frac{(\lambda_2-\lambda_1)^2}{\lambda_1\lambda_2}$, we have

$$\mathbb{E}[C_A] \geq \mathbb{E}[C_{\mathrm{FTPP}}] + (b-a)n^2/2\frac{\exp(-n\frac{(b-a)^2}{ab})}{4}$$

At this stage, we can rewrite the equation assuming $\lambda_2 \geq \lambda_1$ and so that we get

$$\mathbb{E}[C_A] \geq \mathbb{E}[C_{\mathrm{FTPP}}] + (\lambda_2-\lambda_1)n^2\frac{\exp(-n\frac{(\lambda_2-\lambda_1)^2}{\lambda_1\lambda_2})}{8},$$

which proves the first result of the proposition. In particular, taking $\lambda_2 \leq \lambda_1\left(1+\frac{1}{\sqrt{n}}\right)$ gives its second result

$$\begin{aligned}\mathbb{E}[C_A] - \mathbb{E}[C_{\mathrm{FTPP}}] &\geq \lambda_1 n\sqrt{n}\frac{\exp(-n\frac{1/n}{(1+1/\sqrt{n})^2})}{8} \\ &\geq \lambda_1 n\sqrt{n}\frac{e^{-1/4}}{8} \\ &\geq (\lambda_1+\lambda_2)n\sqrt{n}\frac{e^{-1/4}}{24}.\end{aligned}$$

□

B.5.2. Large Differences

Proposition B.5. *For any non-preemptive algorithm, there exists a problem instance with expected type durations of $\lambda_1 \leq \lambda_2 \cdots \leq \lambda_K$ such that*

$$\mathbb{E}[C_A] \geq \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{n}{K}\sum_{k=1}^{K}(2k-K-1)\lambda_k.$$

In particular, for $K = 2$ and $\lambda_2 \geq 3\lambda_1$, it holds that

$$\mathbb{E}[C_A] \geq \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{n}{4}(\lambda_1+\lambda_2),$$

Let p_k be the probability that a non-preemptive algorithm completes a job of type k at its first round. Notice that this distribution cannot depend on the expected duration of any of the types, since no data was gathered. Thus, types can be arbitrarily ordered without affecting this distribution. In particular, we assume w.l.o.g. that $p_1 \le p_2 \le \dots p_K$ and choose a problem instance where job types are ordered in an increasing duration $\lambda_1 \le \lambda_2 \le \dots \lambda_K$. Then, the expected cost of the algorithm can be bounded according to Lemma B.1, by

$$\begin{aligned}
\mathbb{E}[C_A] &= \mathbb{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_i^k \text{ computed before } P_j^\ell\right\}\right] \\
&\ge \mathbb{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{j\in[n]}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_1^k \text{ computed before } P_j^\ell\right\}\right] \\
&\ge \mathbb{E}[C_{\mathrm{FTPP}}] + n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{P_1^k \text{ was the first job}\right\}\right] \\
&= \mathbb{E}[C_{\mathrm{FTPP}}] + n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)p_k \\
&= \mathbb{E}[C_{\mathrm{FTPP}}] + n\sum_{k=1}^{K}p_k\sum_{\ell=1}^{k-1}(\lambda_k-\lambda_\ell).
\end{aligned}$$

Now, one can be easily convinced that between all probability vectors with non-decreasing components, this bound is minimized by the uniform distribution $p_k = 1/K$. To see this, observe that the sum $\sum_{\ell=1}^{k-1}(\lambda_k-\lambda_\ell)$ increases with k. Therefore, if p is non-uniform, then $p_K > 1/K$, and there would exist a coordinate $k < K$ to which we could move weight from p_K, which would decrease the bound.

Substituting $p_k = 1/K$ we then get

$$\begin{aligned}
\mathbb{E}[C_A] &\ge \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{n}{K}\sum_{k=1}^{K}\sum_{\ell=1}^{k-1}(\lambda_k-\lambda_\ell) \\
&= \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{n}{K}\sum_{k=1}^{K}(2k-K-1)\lambda_k.
\end{aligned}$$

In particular, if $K = 2$, we get

$$\mathbb{E}[C_A] \ge \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{n}{2}(\lambda_2-\lambda_1)$$

and, for $\lambda_2 \ge 3\lambda_1$, we have $\lambda_2 - \lambda_1 \ge \frac{\lambda_1+\lambda_2}{2}$ and thus

$$\mathbb{E}[C_A] \ge \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{n}{4}(\lambda_1+\lambda_2).$$

C. Analysis of Preemptive Learning algorithms

C.1. Full Algorithmic Details

In this appendix, we present a full description of ETC-RR and UCB-RR.

Algorithm 5 Explore-Then-Commit-Round-Robin (ETC-RR)

```
1: Input : $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types)
2: For all pairs of different types $k, \ell$ initialize $\delta_{k,\ell} = 0$, $\hat{r}_{k,\ell} = 0$ and $h_{k,\ell} = 0$
3: For all types $k$, set $c_k = 0$
4: repeat
5:     $\mathcal{U}$ is the set of types with at least one remaining job
6:     if $\mathcal{A}$ is empty then
7:         $\mathcal{A} = \{\ell \in \mathcal{U}, \forall k \in \mathcal{U}, k \neq \ell,\ \hat{r}_{k,\ell} - \delta_{k,\ell} \leq 0.5\}$
8:     end if
9:     Run jobs $(P^k_{c_k+1})_{k\in\mathcal{A}}$ in parallel until a job finishes and denote $\ell$ the type of this job
10:    $c_\ell = c_\ell + 1$
11:    for $k \in \mathcal{A}, k \neq \ell$ do
12:        $\beta_{\ell,k} = \beta_{\ell,k} + 1$
13:        $\delta_{\ell,k} = \delta_{k,\ell} = \sqrt{\frac{\log(2n^2K^4)}{2(\beta_{\ell,k}+\beta_{k,\ell})}}$
14:        $\hat{r}_{\ell,k} = \frac{\beta_{\ell,k}}{\beta_{k,\ell}+\beta_{\ell,k}}$
15:        $\hat{r}_{k,\ell} = \frac{\beta_{k,\ell}}{\beta_{k,\ell}+\beta_{\ell,k}}$
16:        if $\hat{r}_{\ell,k} - \delta_{\ell,k} \geq 0.5$ then
17:            Remove $k$ from $\mathcal{A}$
18:        end if
19:        if $\hat{r}_{k,\ell} - \delta_{k,\ell} \geq 0.5$ or $c_\ell = n$ then
20:            Remove $\ell$ from $\mathcal{A}$
21:        end if
22:    end for
23: until $\mathcal{U}$ is empty
```

Algorithm 6 Upper-Confidence-Bound-Round-Robin (UCB-RR)

```
1: Input : $n \geq 1$ (number of jobs of each type), $K \geq 2$ (number of types), discretization step $\Delta$
2: repeat
3:     $\mathcal{U}$ is the set of types with at least one remaining job
4:     Calculate type indices $u_k$ for all jobs $k \in \mathcal{U}$ according to Equation (29)
5:     Choose type $\ell \in \operatorname{argmax}_{\ell\in\mathcal{U}} u_\ell$
6:     Run a job of type $\ell$ for $\Delta$ time units
7: until $\mathcal{U}$ is empty
```

C.2. Cost Decomposition

We start with a cost decomposition, which relates the performance of preemptive algorithms to the one of FTPP. We limit ourselves to the natural family of preemptive algorithms that do not simultaneously run two tasks of the same type and is formally defined as follows.

Definition C.1. Denote b_i^ℓ and e_i^ℓ the beginning and end dates of the computation of the i^{th} job of type ℓ. A **type-wise non-preemptive algorithm** is an algorithm that computes jobs of the same type one after another, i.e., $\forall i \in [n], \forall k \in [K], e_i^\ell \leq b_{i+1}^\ell$.

This property is very natural, as for exponential durations without knowledge of the real execution times, there is no advantage in simultaneously running two tasks of the same type. Specifically, all of our suggested algorithms fall under this definition.

For such algorithms, the cumulative cost can be compared to FTPP using the following lemma.

Lemma C.2 (Cost of type-wise non-preemptive algorithms)**.** *Any type-wise non-preemptive algorithm A has the following upper bound on its cost:*

$$\mathbb{E}[C_A] \le \mathbb{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \sum_{(i,j)\in[n]^2} (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\{e_j^k < b_i^\ell\}\right] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell.$$

Proof. Recall that if D_{ij}^A is the amount of time a job i delays job j when running algorithm A, then we can write the cost of algorithm A as

$$C_A = \sum_{j=1}^{N} P_j + \sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(D_{ij}^A + D_{ji}^A\right).$$

Moreover, if b_i,e_i are the start (end) time of job i, it always holds that $D_{ij}^A \le P_i\mathbb{1}\{b_i < e_j\}$. Using this inequality and dividing the summation into types, we get

$$\begin{aligned}\mathbb{E}[C_A] \le & \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\mathbb{E}\left[P_i^\ell\mathbb{1}\{b_i^\ell < e_j^k\}\right] + \mathbb{E}\left[P_j^k\mathbb{1}\{b_j^k < e_i^\ell\}\right]\right) \\ & + \sum_{\ell=1}^{K}\left(\sum_{i=1}^{n}\mathbb{E}[P_i^\ell] + \sum_{j=i+1}^{n}\mathbb{E}[P_i^\ell\mathbb{1}\{b_i^\ell < e_j^\ell\}] + \mathbb{E}[P_j^\ell\mathbb{1}\{b_j^\ell < e_i^\ell\}]\right).\end{aligned}$$

Since jobs are independent, the expected duration of a job of type ℓ is λ_ℓ, independently of its start time. Also, as the algorithm is type-wise non-preemptive, for all $\ell \in [K], j > i$, we have $\mathbb{1}\{b_j^\ell < e_i^\ell]\} = 0$ and $\mathbb{1}\{b_i^\ell \le e_j^\ell]\} = 1$. Thus,

$$\begin{aligned}\mathbb{E}[C_A] \le & \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell\mathbb{E}\left[\mathbb{1}\{b_i^\ell < e_j^k\}\right] + \lambda_k\mathbb{E}\left[\mathbb{1}\{b_j^k < e_i^\ell\}\right]\right) \\ & + \sum_{\ell=1}^{K}\left(\sum_{i=1}^{n}\lambda_\ell + \sum_{j=i+1}^{n}\lambda_\ell\right) \\ = & \sum_{\ell=1}^{K}\lambda_\ell\frac{n(n+1)}{2} + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell\mathbb{E}\left[\mathbb{1}\{b_i^\ell < e_j^k\}\right] + \lambda_k\mathbb{E}\left[\mathbb{1}\{b_j^k < e_i^\ell\}\right]\right)}_{(i)}.\end{aligned} \tag{23}$$

We can now decompose the event that job i of type ℓ started before job j of type k finished:

$$\begin{aligned}\mathbb{1}\{b_i^\ell < e_j^k\} &= \mathbb{1}\{b_i^\ell < e_j^k \le e_i^\ell\} + \mathbb{1}\{e_i^\ell < e_j^k\} \\ &= \mathbb{1}\{b_i^\ell < e_j^k \le e_i^\ell\} + \mathbb{1}\{e_i^\ell < b_j^k\} + \mathbb{1}\{b_j^k \le e_i^\ell < e_j^k\}\end{aligned}$$

$\{e_i^\ell < b_j^k\}$ is the event that job i of type ℓ was fully computed before job j of type k started. $\{b_i^\ell < e_j^k \le e_i^\ell\}$ is the event that job i of type ℓ was running when job j of type k finished, and reciprocally for $\{b_j^k \le e_i^\ell < e_j^k\}$. This gives the following decomposition:

$$
\begin{aligned}
(i) = &\underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell \mathbb{E}\left[\mathbb{1}\left\{e_i^\ell < b_j^k\right\}\right] + \lambda_k \mathbb{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right]\right)}_{(ii)} \\
&+ \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\lambda_\ell \mathbb{E}\left[\mathbb{1}\left\{b_i^\ell < e_j^k \le e_i^\ell\right\}\right] + \lambda_k \mathbb{1}\left\{b_j^k \le e_i^\ell < e_j^k\right\}}_{(iii)} \\
&+ \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\lambda_k \mathbb{E}\left[\mathbb{1}\left\{b_j^k < e_i^\ell \le e_j^k\right\} + \lambda_k \mathbb{1}\left\{b_i^\ell \le e_j^k < e_i^\ell\right\}\right]}_{(iv)}
\end{aligned}
$$

Since the algorithm is type-wise non-preemptive, a single job of each type may run at a given time. This implies that any job of type ℓ cannot be in a middle of two different jobs of type k, namely

$$
\forall(\ell,k)\in[K]^2,\ \ell\neq k,\ \forall(i,j)\in[n]^2,\ \sum_{j=1}^{n}\mathbb{1}\left\{b_j^k \le e_i^\ell < e_j^k\right\} \le 1.
$$

The same conclusion similarly holds for all other sums in terms (iii) and (iv), and therefore implies the following bound:

$$
(iii) + (iv) \le n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_\ell + \lambda_k) = (K-1)n\sum_{\ell=1}^{K}\lambda_\ell.
$$

We also have $\mathbb{1}\left\{e_j^k < b_i^\ell\right\} \le 1 - \mathbb{1}\left\{b_i^\ell < e_j^k\right\}$, thus

$$
\begin{aligned}
(ii) &\le \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}\left(\lambda_\ell + (\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right]\right) \\
&= n^2\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(K-\ell)\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right].
\end{aligned}
$$

Combining everything into Equation (23), we get:

$$
\begin{aligned}
\mathbb{E}[C_A] &\le \sum_{\ell=1}^{K}\lambda_\ell\frac{n(n+1)}{2} + n^2\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(K-\ell)\lambda_\ell \\
&\quad+ \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell \\
&= \mathbb{E}[C_{\mathrm{FTPP}}] + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{e_j^k < b_i^\ell\right\}\right] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell,
\end{aligned}
$$

where the last equality is by Lemma A.2. □

C.3. Upper bound for ETC-RR

Proposition C.3. *The following bound holds:*

$$
\mathbb{E}[C_{\text{ETC-RR}}] \le \mathbb{E}[C_{\mathrm{FTPP}}] + \frac{12K}{n}\mathbb{E}[C^{OPT}] + 4n\sqrt{n\log(2n^2K^3)}\sum_{\ell=1}^{K}(K-\ell)^2\lambda_\ell. \tag{24}
$$

Good Event. For any couple (k, ℓ), if at some iteration $\beta_{k,\ell} + \beta_{\ell,k} = m$, we define the more precise notation for $\hat{r}_{\ell,k}$ at that iteration as $\hat{r}^m_{\ell,k}$. Notice that m represents the number of times that jobs of either type were completed while both were active. Therefore, m can have $2n$ different values, where the extreme case $m = 2n - 1$ is, for example, when $n - 1$ jobs of type k are first completed and only then all n jobs of type ℓ are completed.

Let us start by showing that the estimators $\hat{r}_{\ell,k}$ well-concentrate around their expectations. Exponential random variables are memory-less, i.e., if $X_i \sim Exp(\lambda_i)$, the law of X_i conditionally on it being larger than a constant is unchanged. In particular, 'resetting' (replacing by an independent copy) an exponential random variable at any time that precedes its activation does not affect its distribution. We employ reset when one of the types is completed, or when either of the types is removed from $\mathcal{A}$ (and then we discard the comparison between types k, ℓ), and say that way a comparison is triggered, it is taken from an i.i.d. sequence of comparisons. Specifically, given a deterministic number of comparisons m between types k, ℓ, we write

$$\hat{r}^m_{\ell,k} \overset{\mathcal{L}}{=} \frac{1}{m}\sum_{i=1}^{m} \mathbb{1}\{X^\ell_i < X^k_i\},$$

with $(X^\ell_i)_{i\in[m]}$ and $(X^k_i)_{i\in[m]}$ independent exponential variables of parameters λ_ℓ and λ_k respectively.

Finally, $\mathcal{E}$ be the event that all comparisons between k, ℓ are well-concentrated, namely,

$$\mathcal{E} = \left\{\forall (k,\ell) \in [K]^2, \forall m \in [2n], \left|\hat{r}^m_{\ell,k} - \frac{\lambda_k}{\lambda_\ell + \lambda_k}\right| < \delta^{(m,n)}\right\}$$

with $\delta^{(m,n)} = \sqrt{\frac{\log(2n^2K^3)}{2m}}$, and recall that for any $m \in [2n]$, $\mathbb{E}[\hat{r}^m_{\ell,k}] = \frac{\lambda_k}{\lambda_\ell+\lambda_k}$. By Lemma B.3 applied with $\alpha = 2$, and a union bound over the $\frac{K(K-1)}{2}$ possible pairs, we have:

$$\mathbb{P}(\overline{\mathcal{E}}) \leq \frac{1}{nK}. \tag{25}$$

Notice that under $\mathcal{E}$ a type ℓ will never be eliminated by a type k with $\lambda_k \geq \lambda_\ell$, since

$$\hat{r}_{k,\ell} - \delta_{\ell,k} < \left(\frac{\lambda_\ell}{\lambda_k + \lambda_\ell} + \delta_{\ell,k}\right) - \delta_{\ell,k} = \frac{\lambda_\ell}{\lambda_k + \lambda_\ell} \leq \frac{1}{2},$$

so if k is the minimal type in $\mathcal{A}$, then under the good event, it will never be eliminated. Moreover, type k can only be compared to a type ℓ with $\lambda_\ell \leq \lambda_k$ at most

$$m^{\max}_{\ell,k} \leq \min\left\{2n, 8\left(\frac{\lambda_k + \lambda_\ell}{\lambda_k - \lambda_\ell}\right)^2 \log(2n^2K^3)\right\} := m^*_{\ell,k} \tag{26}$$

since clearly $m \leq 2n$ and if $m \geq 8\left(\frac{\lambda_k+\lambda_\ell}{\lambda_k-\lambda_\ell}\right)^2 \log(2n^2K^3)$, then $\delta_{\ell,k} \leq \frac{1}{4}\frac{\lambda_k-\lambda_\ell}{\lambda_k+\lambda_\ell}$ and

$$\hat{r}_{\ell,k} - \delta_{\ell,k} > \left(\frac{\lambda_k}{\lambda_k + \lambda_\ell} - \delta_{\ell,k}\right) - \delta_{\ell,k} \geq \frac{\lambda_k}{\lambda_k + \lambda_\ell} - 2\cdot\frac{1}{4}\frac{\lambda_k - \lambda_\ell}{\lambda_k + \lambda_\ell} = \frac{1}{2}.$$

Cost Analysis. Assume that the active type set $\mathcal{A}$ can only change at discrete times $t \in \{0, \Delta, 2\Delta, \dots\} \triangleq \mathcal{T}$ for some $\Delta > 0$. We will later take the limit $\Delta \to 0$, which coincides with the following Algorithm 5. In the following, we denote by $\mathcal{A}(t)$ the active type set at time interval $[t, t+\Delta)$, and $\mathcal{U}(t)$ incomplete type set at $[t, t+\Delta)$. Also, let $b^\ell_i \in \mathcal{T}$ be the start time of the i^{th} job of the ℓ^{th} type and $e^\ell_i \in \mathcal{T}$ be its end-time (w.l.o.g., if a task ended at a time $t \notin \mathcal{T}$, we delay its ending to $\lceil \frac{t}{\Delta} \rceil \Delta$).

Starting from the cost decomposition of Lemma C.2, we have

$$\begin{aligned}\mathbb{E}[C_A] &\leq \mathbb{E}[C_{\text{FTPP}}] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\{e^k_j < b^\ell_i\}\right] \\ &\leq \mathbb{E}[C_{\text{FTPP}}] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\underbrace{\sum_{j=1}^{n}\mathbb{E}\left[\mathbb{1}\{e^k_j < b^\ell_n\}\right]}_{(*)}.\end{aligned} \tag{27}$$

We focus our attention on bounding term $(*)$. For any $t \in \mathcal{T}$ and every $o \in [K]$, define the events

$$\mathcal{F}_o(t) = \{o \in \mathcal{A}(t), \forall p < o : p \notin \mathcal{A}(t)\}, \qquad \bar{\mathcal{F}}_o(t) = \{\forall p \le o : p \notin \mathcal{A}(t)\}$$

These events capture the notion of *phases*, namely, when $\mathcal{F}_o$ is active, the o is the type of the smallest mean that has not been finished. Then, we can write

$$\begin{aligned}
(*) &= \sum_{j=1}^{n} \mathbb{E}\left[\mathbb{1}\{e_j^k < b_n^\ell\}\right] \\
&= \sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, e_j^k < b_n^\ell\}\right] \\
&\overset{(1)}{\le} \sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \ell \in \mathcal{U}(t)\}\right] \\
&\le \sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \mathcal{F}_o(t)\}\right] + \sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \ell \in \mathcal{U}(t), \bar{\mathcal{F}}_\ell(t)\}\right] \\
&\overset{(2)}{\le} \sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \mathcal{F}_o(t)\}\right] + \sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \overline{\mathcal{E}}\}\right] \\
&\overset{(3)}{\le} \underbrace{\sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \mathcal{F}_o(t)\}\right]}_{(i)} + n\mathbb{P}(\overline{\mathcal{E}}).
\end{aligned}$$

(1) is true since the event $\{e_j^k = t+\Delta, e_j^k < b_n^\ell\}$ implies that $b_n^\ell > t$, so type ℓ was not completed at time t. (2) holds since under $\mathcal{E}$, the type of the minimal duration expected is never eliminated, so if $\ell \in \mathcal{U}(t)$, it is impossible that $p \notin \mathcal{A}(t)$ for all $p \le \ell$ (either there is a type $p < \ell, p \in \mathcal{U}(t)$ that should not have been eliminated, or type ℓ should not have been eliminated since it is still incomplete). Finally, (3) is since every job can only end at one time point.

We now further continue to bound term (i). To do so, observe that if $e_j^k = t+\Delta$, then $k \in \mathcal{A}(t)$ and the task j of this type was completed at the interval $[t, t+\Delta)$. Moreover, since the job durations are exponential, the completion of any job in $\mathcal{A}(t)$ at interval $[t, t+\Delta)$ is independent of the events that occurred until time t. Taking into consideration that only one job of any type can run in every interval, the two following equalities hold:

$$\begin{aligned}
&\mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta, \mathcal{F}_o(t)\}\right] = (1-\exp(-\Delta/\lambda_k))\,\mathbb{E}\left[\mathbb{1}\{\mathcal{F}_o(t), k \in \mathcal{A}(t)\}\right] \\
&\mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta \text{ or } e_j^o = t+\Delta, \mathcal{F}_o(t), k\in\mathcal{A}(t)\}\right] \\
&\ = (1-\exp(-\Delta/\lambda_k - \Delta/\lambda_o))\,\mathbb{E}\left[\mathbb{1}\{\mathcal{F}_o(t), k \in \mathcal{A}(t)\}\right].
\end{aligned}$$

Thus, (i) can be written as

$$\begin{aligned}
(i) &= \sum_{o=1}^{\ell} \frac{(1-\exp(-\Delta/\lambda_k))}{(1-\exp(-\Delta/\lambda_k - \Delta/\lambda_o))} \sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta \text{ or } e_j^o = t+\Delta, \mathcal{F}_o(t), k\in\mathcal{A}(t)\}\right] \\
&\le \underbrace{\sum_{o=1}^{\ell} \frac{(1-\exp(-\Delta/\lambda_k))}{(1-\exp(-\Delta/\lambda_k - \Delta/\lambda_o))} \mathbb{E}\left[\sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{1}\{e_j^k = t+\Delta \text{ or } e_j^o = t+\Delta, \mathcal{F}_o(t), k\in\mathcal{A}(t), \mathcal{E}\}\right]}_{(ii)} \\
&\quad + \underbrace{\sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}} \mathbb{E}\left[\mathbb{1}\{e_j^k = t+\Delta \text{ or } e_j^o = t+\Delta, \mathcal{F}_o(t), \overline{\mathcal{E}}\}\right]}_{(iii)}
\end{aligned}$$

The bad-event term can be bounded by

$$\begin{aligned}(iii) &\le \mathbb{E}\left[\mathbb{1}\{\overline{\mathcal{E}}\}\sum_{o=1}^{\ell}\sum_{j=1}^{n}\sum_{t\in\mathcal{T}}\mathbb{1}\{e_j^k = t+\Delta \text{ or } e_j^o = t+\Delta, \mathcal{F}_o(t)\}\right]\\ &\le (\ell+1)n\mathbb{P}(\overline{\mathcal{E}}).\end{aligned}$$

For the second term, recall that the ℓ^{th} phase, in which type ℓ is the smallest one that was not completed, is represented by the event $\mathcal{F}_\ell(t)$. Furthermore, given the good event, the smallest type is never eliminated. so once $\mathcal{F}_\ell(t)$ becomes active, it would end only when all jobs of type ℓ are completed. In other words, the time indices in which $\mathcal{F}_\ell(t)$ hold form a (possibly empty) interval $\mathcal{I}_\Delta(\ell)$, which represents the ℓ^{th} phase. Thus, term (ii) counts the expected number of times that jobs of either type k or o could finish in this interval while type k is still active.

Now, we take the limit $\Delta \to 0$ (and denoting the limit interval $\mathcal{I}_\Delta(\ell) \to \mathcal{I}(\ell)$). Notice that the limit and expectation are interchangeable by the bounded convergence theorem, as the number of times a job of either type k or o can be completed is bounded by $2n$.

$$\begin{aligned}(ii) &\underset{\Delta\to 0}{\to} \sum_{o=1}^{\ell}\frac{\lambda_o}{\lambda_k+\lambda_o}\mathbb{E}\left[\sum_{t\in\mathcal{I}(o)}\sum_{j=1}^{n}\mathbb{1}\{e_j^k\in\mathcal{I}(o) \text{ or } e_j^o\in\mathcal{I}(o), k\in\mathcal{A}(t),\mathcal{E}\}\right]\\ &\le \sum_{o=1}^{\ell}\frac{\lambda_o}{\lambda_k+\lambda_o}m_{o,k}^*.\end{aligned}$$

The inequality holds since under the good event, at any interval where both types k and o with $\lambda_o \le \lambda_k$ are active, there can be at most $m_{o,k}^*$ comparisons. Substituting (ii) and (iii) back into (i), we get

$$(i) \le \sum_{o=1}^{\ell}\frac{\lambda_o}{\lambda_k+\lambda_o}m_{o,k}^* + (\ell+1)n\mathbb{P}(\overline{\mathcal{E}}),$$

and yet again, substituting this, through $(*)$, back into Equation (27), yields

$$\begin{aligned}&\mathbb{E}[C_A] - \mathbb{E}[C_{\text{FTPP}}]\\ &\le (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}n(\lambda_k-\lambda_\ell)\left(\sum_{o=1}^{\ell}\frac{\lambda_o}{\lambda_k+\lambda_o}m_{o,k}^* + (\ell+2)n\mathbb{P}(\overline{\mathcal{E}})\right)\\ &\le (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + 2Kn^2\mathbb{P}(\overline{\mathcal{E}})\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\\ &\quad+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}n(\lambda_k-\lambda_\ell)\sum_{o=1}^{\ell}\frac{\lambda_o}{\lambda_k+\lambda_o}\min\left\{2n, 8\left(\frac{\lambda_k+\lambda_o}{\lambda_k-\lambda_o}\right)^2\log(2n^2K^3)\right\} && \text{(Equation (26))}\\ &\le (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + 2K^2n^2\mathbb{P}(\overline{\mathcal{E}})\sum_{\ell=1}^{K}\lambda_\ell\\ &\quad+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{o=1}^{\ell}n(\lambda_k-\lambda_\ell)\frac{\lambda_o}{\lambda_k+\lambda_o}4\sqrt{n\log(2n^2K^3)}\frac{\lambda_k+\lambda_o}{\lambda_k-\lambda_o} && (\min\{a,b\}\le\sqrt{ab}, \forall a,b\ge 0)\\ &\le \left(2\frac{K-1}{n}+\frac{4K}{n}\right)\mathbb{E}[C^{OPT}] && \text{(Equation (25) and (6))}\\ &\quad+4n\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{o=1}^{\ell}\lambda_o\sqrt{n\log(2n^2K^3)} && (\lambda_o\le\lambda_\ell)\\ &\le \frac{6K}{n}\mathbb{E}[C^{OPT}] + 4n\sqrt{n\log(2n^2K^3)}\sum_{\ell=1}^{K}\sum_{o=1}^{\ell}(K-\ell)\lambda_o\\ &\le \frac{6K}{n}\mathbb{E}[C^{OPT}] + 4n\sqrt{n\log(2n^2K^3)}\sum_{\ell=1}^{K}(K-\ell)^2\lambda_\ell\end{aligned}$$

□

C.4. Analysis of UCB-RR

Proposition C.4. *The following bound holds for any* $\Delta \leq \frac{\lambda_1}{4}$ *and* $n \geq \max(20, 10\ln(K))$ *:*

$$\mathbb{E}[C_{\text{UCB-RR}}] \leq \mathbb{E}[C_{\text{FTPP}}] + \frac{12K}{n}\mathbb{E}[C^{OPT}] + 6n\sqrt{2n\log(2n^2K^2)}\sum_{\ell=1}^{K}(K-\ell)\lambda_\ell. \tag{28}$$

Assume discretization of the time to units of Δ, as was done in the analysis of ETC-RR. Specifically, assume that the active job only changes at times $t \in \{0, \Delta, 2\Delta, \dots\} \triangleq \mathcal{T}$ for some $\Delta > 0$. We then denote the index of the discretization step by $h = \frac{t}{\Delta} + 1 \in \{1, 2, \dots\}$.

For each job type $\ell \in [K]$, we introduce $T_\ell(h)$, the number of times job type ℓ has been chosen up to iteration h. Due to the fact that job durations are exponential, their increments are independent, and increments of length Δ of jobs of type ℓ have a termination probability of $\mu_\ell = 1 - e^{-\frac{\Delta}{\lambda_k}}$. Leveraging this, let $(x_\ell^s)_{s\geq 1}$ be sequences of i.i.d Bernoulli random variables of mean μ_ℓ,. We then fix our probability space for the analysis s.t. when choosing a job of type ℓ for the s^{th} time, it is terminated if $x_\ell^s = 1$. Notice that while we allow the sequence $(x_\ell^s)_{s\geq1}$ to have more than n job terminations, it is of no consequence of the analysis, as the algorithm will never choose a job type after its n^{th} job was terminated.

Next, define the empirical means after running m discretized intervals of type-ℓ jobs as $\hat{\mu}_\ell(m) := \frac{1}{m}\sum_{s=1}^{m} x_\ell^s$, and the index at iteration h as

$$u_\ell(h) = \max\left\{\tilde{\mu} \in [0,1] : d\left(\hat{\mu}_\ell(T_\ell(t-1)), \tilde{\mu}\right) \leq \frac{\log\left(n^2K^2\right)}{T_\ell(t-1)}\right\}. \tag{29}$$

Starting from the cost decomposition of Lemma C.2, we have

$$\begin{aligned}\mathbb{E}[C_A] &\leq \mathbb{E}[C_{\text{FTPP}}] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}\sum_{(i,j)\in[n]^2}(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\{e_j^k < b_i^\ell\}\right] \\ &\leq \mathbb{E}[C_{\text{FTPP}}] + (K-1)n\sum_{\ell=1}^{K}\lambda_\ell + \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\sum_{j=1}^{n}\mathbb{E}\left[\mathbb{1}\{e_j^k < e_n^\ell\}\right]}_{(*)}.\end{aligned} \tag{30}$$

Denote $a(h) \in [K]$, the type of job chosen at iteration h, and let $\varepsilon_{\ell,k} > 0$ be some constant that will be determined later in the proof. Notice that if $e_j^k < e_n^\ell$, then there must be an iteration where type ℓ was not completed and the j^{th} job of type k were played and completed:

$$\begin{aligned}(*) \leq& \sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{a(h) = k,\ \ell \in \mathcal{U}(h), x_k^{T_k(h)} = 1\right\}\right] \\ =& \sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{a(h) = k,\ \ell \in \mathcal{U}(h), u_\ell(h) \leq u_k(h), x_k^{T_k(h)} = 1\right\}\right] \\ \leq& \sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{a(h) = k,\ \ell \in \mathcal{U}(h), u_k(h) \geq u_\ell(h) \geq \mu_\ell - \varepsilon_{\ell,k}, x_k^{T_k(h)} = 1\right\}\right] \\ &+ \sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)\mathbb{E}\left[\mathbb{1}\left\{a(h) = k,\ \ell \in \mathcal{U}(h), u_\ell(h) \leq \mu_\ell - \varepsilon_{\ell,k}, x_k^{T_k(h)} = 1\right\}\right] \\ \overset{(1)}{\leq}& \underbrace{\sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)(1 - e^{-\frac{\Delta}{\lambda_k}})\mathbb{E}\left[\mathbb{1}\{a(h) = k, u_k(h) \geq \mu_\ell - \varepsilon_\ell\}\right]}_{(i)} \\ &+ \underbrace{\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n^2(\lambda_k - \lambda_\ell)\mathbb{P}\left(\exists h \text{ s.t. } u_\ell(h) \leq \mu_\ell - \varepsilon_\ell\right)}_{(ii)}\end{aligned} \tag{31}$$

In (1), we get the first line by the memoryless property of exponential random variables, noting that all the events inside the indicator are determined before the beginning of the h^{th} iteration. The second line of this relation uses the fact that all tasks will eventually be completed, so $\sum_{h=1}^{\infty} \mathbb{E}\left[\mathbb{1}\left\{a(h)=k, x_k^{T_k(h)}=1\right\}\right]=n$.

Bounding term (i). We now bound the first term of the decomposition in Equation (31).

$$\begin{aligned}(i) :=& \sum_{h=1}^{\infty}\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\mathbb{E}\left[\mathbb{1}\{a(h)=k, u_k(h)\geq \mu_\ell-\varepsilon_\ell\}\right]\\ =& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\sum_{h=1}^{\infty}\mathbb{E}\left[\mathbb{1}\{a(h)=k, u_k(h)\geq \mu_\ell-\varepsilon_\ell\}\right].\end{aligned}$$

Denoting $\underline{d}(p,q)=d(p,q)\mathbb{1}\{p\leq q\}$, we have

$$\{\mu_\ell-\varepsilon_{\ell,k}\leq u_k(h)\} \implies \{\hat{\mu}_k(T_k(h))\leq \mu_\ell-\varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_k(T_k(h)),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\} \text{ or } \{\hat{\mu}_k(T_k(h))\geq \mu_\ell-\varepsilon_{\ell,k}\},$$

which is equivalent to $\left\{\underline{d}(\hat{\mu}_k(T_k(h)),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\right\}$. Thus, we can bound

$$\begin{aligned}(i) \leq& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\sum_{h=1}^{\infty}\mathbb{E}\left[\mathbb{1}\left\{a(h)=k, \underline{d}(\hat{\mu}_k(T_k(h)),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\right\}\right]\\ \leq& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)(1-e^{-\frac{\Delta}{\lambda_k}})\sum_{s=1}^{\infty}\mathbb{E}\left[\underline{d}(\hat{\mu}_k(s),\mu_\ell-\varepsilon_{\ell,k})\leq \frac{\log\left(n^2K^2\right)}{T_k(h)}\right],\end{aligned}$$

where the second inequality is since $T_k(h)$ increases by 1 every time that $a(h)=k$. This can be naturally bounded using the following lemma.

Lemma C.5. *Let $X_1, X_2, \ldots$ be a sequence of Bernoulli independent random variables with mean μ, and let $\hat{\mu}_s=\frac{1}{s}\sum_{t=1}^{s}X_t$ be the sample mean. Further, let $a>0, \mu'>\mu$ and define $\kappa=\sum_{s=1}^{\infty}\mathbb{1}\left\{\underline{d}(\hat{\mu}_s,\mu')\leq\frac{a}{s}\right\}$. Then,*

$$\mathbb{E}[\kappa]\leq \inf_{\varepsilon\in(0,\mu'-\mu)}\left(\frac{a}{d(\mu+\varepsilon,\mu')}+\frac{1}{d\left(\mu+\varepsilon,\mu\right)}\right).$$

Proof. The proof closely follows the one of (Lattimore & Szepesvári, 2020, Lemma 10.8). For completeness, we now state the well-known Chernoff bound.

Lemma C.6 (Chernoff's bound, e.g., Lattimore & Szepesvári (2020), Lemma 10.3)**.** *Let $X_1, X_2, \ldots, X_T$ be a sequence of Bernoulli independent random variables with mean μ, and let $\hat{\mu}=\frac{1}{T}\sum_{t=1}^{T}X_t$ be the sample mean. Then, for $a\geq 0$:*

$$\mathbb{P}(d(\hat{\mu},\mu)\geq a, \hat{\mu}\leq\mu)\leq\exp(-Ta).$$

Let $\epsilon \in (0, \mu' - \mu)$ and $u = \frac{a}{d(\mu+\varepsilon,\mu')}$. Then, it holds that

$$\begin{aligned}
\mathbb{E}[\kappa] =& \sum_{s=1}^{\infty} \mathbb{P}\left\{\underline{d}(\hat{\mu}_s, \mu') \le \frac{a}{s}\right\} \\
=& \sum_{s=1}^{\infty} \mathbb{P}\left\{\hat{\mu}_s \ge \mu' \text{ or } d(\hat{\mu}_s, \mu') \le \frac{a}{s}\right\} \\
\le& \sum_{s=1}^{\infty} \mathbb{P}\left\{\hat{\mu}_s \ge \mu + \varepsilon \text{ or } d(\hat{\mu}_s, \mu') \le \frac{a}{s}\right\} && (\mu' > \mu + \epsilon) \\
\le& \sum_{s=1}^{\infty} \mathbb{P}\left\{\hat{\mu}_s \ge \mu + \varepsilon \text{ or } d(\mu + \varepsilon, \mu') \le \frac{a}{s}\right\} && (d(\cdot, \mu') \text{ is decreasing in } [0, \mu']) \\
\le& u + \sum_{s=1}^{\infty} \mathbb{P}\left\{\hat{\mu}_s \ge \mu + \varepsilon\right\} \\
\le& u + \sum_{s=1}^{\infty}\sum_{s=1}^{\infty} \exp\left(-sd(\mu+\epsilon, \mu)\right) && \text{(Chernoff's bound)} \\
\le& \frac{a}{d(\mu+\varepsilon, \mu')} + \frac{1}{d\left(\mu+\varepsilon, \mu\right)},
\end{aligned}$$

and the proof is concluded by taking the infimum over all $\varepsilon \in (0, \mu' - \mu)$. □

Now, assume w.l.o.g. that $\lambda_k > \lambda_\ell$ (or, equivalently, $\mu_\ell < \mu_k$) for all $k > \ell$; otherwise, terms where $\lambda_k = \lambda_\ell$ in (i) will be equal to 0. Then, letting $\kappa_{k,\ell} = \sum_{s=1}^{\infty} \mathbb{1}\left\{\underline{d}(\hat{\mu}_k(s), \mu_\ell - \varepsilon_{\ell,k}) \le \frac{\log\left(n^2K^2\right)}{s}\right\}$, the last lemma implies that

$$\begin{aligned}
(i) \le& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)(1 - e^{-\frac{\Delta}{\lambda_k}})\mathbb{E}[\kappa_{k,\ell}] \\
\le& \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell)(1 - e^{-\frac{\Delta}{\lambda_k}})\left(\frac{\log\left(n^2K^2\right)}{d(\mu_k + \varepsilon_{k,\ell}, \mu_\ell - \varepsilon_\ell)} + \frac{1}{d\left(\mu_k + \varepsilon_{k,\ell}, \mu_k\right)}\right) \qquad (32)
\end{aligned}$$

for some $\varepsilon_{k,\ell} \in (0, \mu_\ell - \varepsilon_{\ell,k} - \mu_k)$ that will be determined later.

Bounding term (ii). We next focus on bounding the probabilities at the second summation of Equation (31). To do so, we prove the following lemma, which bounds each of the summands of term (ii).

Lemma C.7. *The following bound holds:* $\mathbb{P}\left(\exists h \text{ s.t. } u_\ell(h) \le \mu_\ell - \varepsilon_{\ell,k}\right) \le \frac{\mu_\ell}{n^2K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}$.

Proof. Define $S_\ell^{\max} := \sum_{h=1}^{\infty} \mathbb{1}\{a(h) = \ell\}$, the number of iterations ℓ is picked by the algorithm. We have:

$$\begin{aligned}
\mathbb{P}\left(\exists h \text{ s.t. } u_\ell(h) \le \mu_\ell - \varepsilon_{\ell,k}\right) =& \mathbb{P}\left(\exists h \text{ s.t. } \hat{\mu}_\ell(T_\ell(h)) \le \mu_\ell - \varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_\ell(T_\ell(h)), \mu_\ell - \varepsilon_{\ell,k}) \ge \frac{\log\left(n^2K^2\right)}{T_\ell(h)}\right) \\
=& \mathbb{P}\left(\exists s \le S_\ell^{\max} \text{ s.t. } \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}) \ge \frac{\log\left(n^2K^2\right)}{s}\right) \\
\le& \mathbb{P}\left(\exists 1 \le s < \infty \text{ s.t. } \hat{\mu}_\ell(s) \le \mu_\ell - \varepsilon_{\ell,k} \text{ and } d(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}) \ge \frac{\log\left(n^2K^2\right)}{s}\right).
\end{aligned}$$

Now, observe that the empirical means $\hat{\mu}_\ell$ decrease in intervals without successes. Namely, if $a < b$ are time indices such that $x_\ell^a = 1, x_\ell^b = 1$ and for all $s \in [a+1, b-1]$, $x_\ell^s = 0$, then for any $s \in [a, b-1]$, it holds that $\hat{\mu}_\ell(s) \ge \hat{\mu}_\ell(b-1)$. We thus have:

$$\mathbb{P}\left(\exists 1 \leq s < \infty : d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s},\ \hat{\mu}_\ell(s) \leq \mu_\ell - \varepsilon_{\ell,k}\right)$$
$$= \mathbb{P}\left(\exists 1 \leq s < \infty : d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s},\ \hat{\mu}_\ell(s) \leq \mu_\ell - \varepsilon_{\ell,k},\ x_\ell^{s+1} = 1\right).$$

Using the union bound, this implies

$$\begin{aligned}\mathbb{P}\left(\exists h \text{ s.t. } u_\ell(h) \leq \mu_\ell - \varepsilon_{\ell,k}\right) &\leq \sum_{s=1}^{\infty} \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s}, \hat{\mu}_\ell(s) \leq \mu_\ell - \varepsilon_{\ell,k},\ \text{and } x_\ell^{s+1} = 1\right)\\ &= \sum_{s=1}^{\infty} \mathbb{P}\left(x_\ell^{s+1} = 1\right) \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s}, \hat{\mu}_\ell(s) \leq \mu_\ell - \varepsilon_{\ell,k} | x_\ell^{s+1} = 1\right)\\ &= \sum_{s=1}^{\infty} \mu_\ell \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu_\ell - \varepsilon_{\ell,k}\right) > \frac{\log\left(n^2K^2\right)}{s}, \hat{\mu}_\ell(s) < \mu_\ell - \varepsilon_{\ell,k}\right)\\ &\leq \sum_{s=1}^{\infty} \mu_\ell \mathbb{P}\left(d\left(\hat{\mu}_\ell(s), \mu\right) > \frac{\log\left(n^2K^2\right)}{s} + d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell), \hat{\mu}_\ell(s) < \mu_\ell\right),\end{aligned}$$

where we used the fact that the sequence x_ℓ^s is independent and the last inequality is by (Lattimore & Szepesvári, 2020, Lemma 10.2, (c)). Next, using Chernoff's bound (Lemma C.6), we get

$$\begin{aligned}\mathbb{P}\left(\exists h \text{ s.t. } u_\ell(h) \leq \mu_\ell - \varepsilon_{\ell,k}\right) &\leq \mu_\ell \sum_{s=1}^{\infty} \exp\left(-s\left(d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell) + \frac{\log\left(n^2K^2\right)}{s}\right)\right)\\ &\leq \frac{\mu_\ell}{n^2K^2} \sum_{s=1}^{\infty} \exp\left(-sd(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)\right)\\ &\leq \frac{\mu_\ell}{n^2K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)},\end{aligned}$$

which concludes the proof of Lemma C.7. □

Finally, substituting back into (ii) leads to the bound

$$\begin{aligned}(ii) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n^2(\lambda_k - \lambda_\ell) \frac{\mu_\ell}{n^2K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}\\ =& \frac{1}{K^2} \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} \frac{\mu_\ell(\lambda_k - \lambda_\ell)}{d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}.\end{aligned} \tag{33}$$

Combining both bounds.

$$\begin{aligned}(*) \leq& \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} n(\lambda_k - \lambda_\ell) \left(\frac{\mu_k \log\left(n^2K^2\right)}{d(\mu_k + \varepsilon_{k,\ell}, \mu_\ell - \varepsilon_{\ell,k})} + \frac{\mu_k}{d\left(\mu_k + \varepsilon_{k,\ell}, \mu_k\right)}\right)\\ &+ \sum_{\ell=1}^{K} \sum_{k=\ell+1}^{K} (\lambda_k - \lambda_\ell) \frac{\mu_\ell}{K^2 d(\mu_\ell - \varepsilon_{\ell,k}, \mu_\ell)}.\end{aligned}$$

We now use a local refinement of Pinsker's inequality (Garivier et al., 2019):

$$d(p, q) \geq \frac{1}{2\max(p, q)}(p - q)^2.$$

This implies:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\left(\frac{2\mu_k\left(\mu_\ell-\varepsilon_{\ell,k}\right)\log\left(n^2K^2\right)}{(\mu_\ell-\varepsilon_{\ell,k}-\mu_k-\varepsilon_{k,\ell})^2}+\frac{2\mu_k(\mu_k+\varepsilon_{k,\ell})}{\varepsilon_{k,\ell}^2}\right)$$
$$+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\frac{2\mu_\ell^2}{K^2\varepsilon_{\ell,k}^2}.$$

Case 1: Assume $\mu_\ell \geq 5\mu_k$. Setting $\varepsilon_{k,\ell}=\mu_k$, we obtain,

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\left(\frac{2\mu_k(\mu_\ell-\varepsilon_{\ell,k})\log\left(n^2K^3\right)}{(\mu_\ell-\varepsilon_{\ell,k}-2\mu_k)^2}+4\right)+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\frac{2\mu_\ell^2}{K^2\varepsilon_{\ell,k}^2}$$
$$\leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\left(\frac{2\mu_k(\mu_\ell-\varepsilon_{\ell,k})\log\left(n^2K^3\right)}{(\frac{3}{5}\mu_\ell-\varepsilon_{\ell,k})^2}+4\right)+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\frac{2\mu_\ell^2}{K^2\varepsilon_{\ell,k}^2}$$

Setting $\varepsilon_{\ell,k}=\frac{1}{5}\mu_\ell$, we get:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\left(\frac{10\mu_k\log\left(n^2K^2\right)}{\mu_\ell}+4\right)+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\frac{50}{K^2}.$$

We have $\mu_k = 1-e^{-\frac{\Delta}{\lambda_k}} \leq \frac{\Delta}{\lambda_k}$, and if $\Delta \leq \frac{1}{4}\lambda_\ell$, $\frac{1}{\mu_\ell} \leq 1.13\frac{\lambda_\ell}{\Delta}$, this implies:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n\lambda_\ell 11.3\log\left(n^2K^2\right)+\sum_{\ell=1}^{K}\lambda_\ell\left(\frac{50}{K}+4nK\right).$$

Since $K \geq 2$, this implies:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n\lambda_\ell 11.3\log\left(n^2K^2\right)+\sum_{\ell=1}^{K}\lambda_\ell\left(12.5+4n\right)K.$$

Case 2: Assume $\mu_\ell \leq 5\mu_k$. Setting $\varepsilon_{k,\ell}=(\mu_\ell-\mu_k)/4$ and $\varepsilon_{\ell,k}=(\mu_\ell-\mu_k)/4$, we obtain,

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\frac{\mu_k^2}{(\mu_\ell-\mu_k)^2}\left(32\log\left(n^2K^2\right)+64\right)+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\frac{32\mu_\ell^2}{K^2(\mu_\ell-\mu_k)^2}.$$

It also holds that $(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n^2(\lambda_k-\lambda_\ell)$. Thus:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} n(\lambda_k-\lambda_\ell)\frac{4\mu_k}{(\mu_\ell-\mu_k)}\sqrt{2n\left(\log\left(n^2K^3\right)+4\right)}+\sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K}(\lambda_k-\lambda_\ell)\frac{4\sqrt{2}\mu_\ell n}{K(\mu_\ell-\mu_k)}.$$

If $\Delta \leq \frac{1}{4}\lambda_\ell$, we have:

$$\frac{1}{\mu_\ell-\mu_k} \leq 1.46\frac{\lambda_k\lambda_\ell}{(\lambda_k-\lambda_\ell)}.$$

We thus obtain:

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} 6n\lambda_\ell\sqrt{2n\left(\log\left(n^2K^2\right)+4\right)}+\sum_{\ell=1}^{K}9n\lambda_\ell.$$

For any $n \geq \max(10, 10\log(K))$, we have:

$$\ln(n^2K^2) \leq \frac{1}{2}n$$

which implies $6n\lambda_\ell\sqrt{2n\left(\log\left(n^2K^2\right)+4\right)} \geq 11.3\log\left(n^2K^2\right)$. Thus for any $n \geq \max(10, 10\log(K))$ and $\Delta \leq \frac{1}{4}\lambda_\ell$,

$$(*) \leq \sum_{\ell=1}^{K}\sum_{k=\ell+1}^{K} 6n\lambda_\ell\sqrt{2n\left(\log\left(n^2K^2\right)+4\right)}+\frac{10K}{n}\mathbb{E}[C_{OPT}].$$

D. Additional experiments

We implemented the Bayesian approach of (Marbán et al., 2011) that we call LSEPT. We used an uninformative prior $\alpha = 1$, $w = 0$ (the same for all job types). LSEPT is then in essence a greedy algorithm. Whenever a job finishes, it runs until completion a job whose type has the lowest empirical mean size (computed across jobs that have been processed so far).

We ran all algorithms with $K = 2$, where jobs of type 1 have a mean size $\lambda_1 = 0.8$ and jobs of type 2 have a mean $\lambda_2 = 1$.

As can be seen in Figure 3, LSEPT has better mean performance than RR, a non-adaptive method. However, it has a large variance and its performance does not improve with n. This is typical of the performance of greedy algorithms: since the algorithm commits very early, it can either get very good or very bad performances. We plot the mean over 200 seeds.

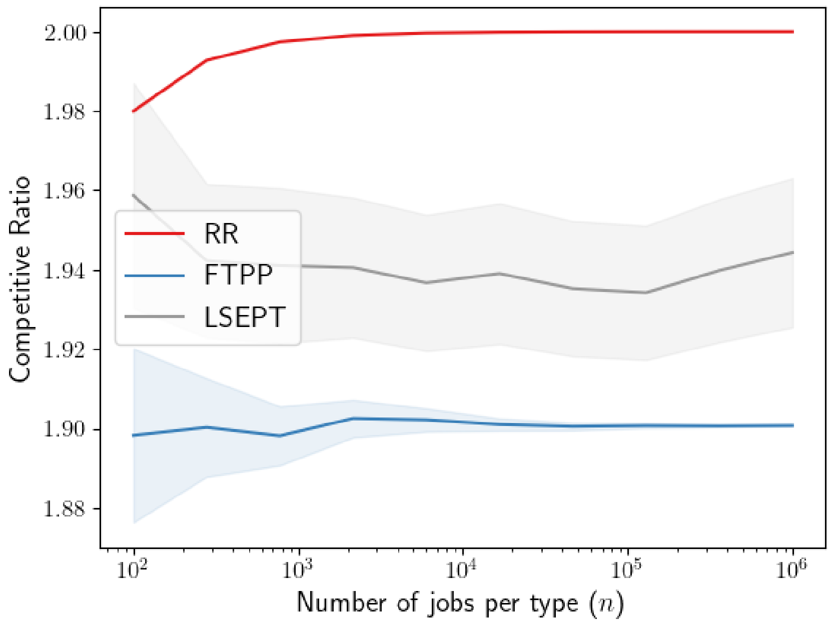

Figure 3. **CR on jobs with 2 different types**. $K = 2$, $\lambda_2 = 1$ and $\lambda_1 = 0.8$, n takes a grid of values.