Minimizing Completion Times for Stochastic Jobs via Batched Free Times

Anupam Gupta
Carnegie Mellon
anupamg@cs.cmu.edu

Benjamin Moseley
Carnegie Mellon
moseleyb@andrew.cmu.edu

Rudy Zhou
Carnegie Mellon
rbz@andrew.cmu.edu

Abstract

We study the classic problem of minimizing the expected total completion time of jobs on m identical machines in the setting where the sizes of the jobs are stochastic. Specifically, the size of each job is a random variable whose distribution is known to the algorithm, but whose realization is revealed only after the job is scheduled. While minimizing the total completion time is easy in the deterministic setting, the stochastic problem has long been notorious: all known algorithms have approximation ratios that either depend on the variances, or depend linearly on the number of machines.

We give an $\widetilde{O}(\sqrt{m})$-approximation for stochastic jobs which have Bernoulli processing times. This is the first approximation for this problem that is both independent of the variance in the job sizes, and is sublinear in the number of machines m. Our algorithm is based on a novel reduction from minimizing the total completion time to a natural makespan-like objective, which we call the *weighted free time*. We hope this free time objective will be useful in further improvements to this problem, as well as other stochastic scheduling problems.

1 Introduction

Consider the problem of scheduling n jobs on m identical machines to minimize the *total completion time* of the jobs. If we assume the job lengths are known, we can solve the problem optimally using the *shortest processing time* (SPT) algorithm [BJS74]. But what if the jobs durations are not known precisely? In practical scheduling settings, the job sizes are typically unknown. However, we can often give good stochastic predictions based on jobs features and past data. In this work, we consider the setting where the job are *stochastic*, so the processing time of each job j is an independent random variable X_j which is distributed according to a *known* probability distribution π_j, but whose *realized value we observe only after scheduling it* irrevocably on some machine. Now the completion time C_j of job j is a random variable, which depends on the random job sizes (and on any random decisions our algorithm may make). Our goal is to minimize $\sum_j \mathbb{E}[C_j]$, the sum of expected completion times C_j of the jobs (or equivalently, their average).

Since the randomness in the job sizes is revealed as they are scheduled, the decision of which job to schedule next (and on which machine) can depend on the outcomes of already-scheduled jobs. Such scheduling policies are called *adaptive*. Formally, for each idle machine, the *adaptive scheduling policy* must choose which job to schedule next on this machine—or it may choose to idle the machine for some time period. In making this decision, it is allowed to use any information it has gained from previously-scheduled jobs. In particular, the policy knows the sizes X_j of all completed jobs j, and if a job j has currently been run for τ time the policy knows that the jobs size is distributed as $(X_j \mid X_j \geq \tau)$.

In this work we want to find near-optimal adaptive schedules, ones that result in the total expected completion time being close to that achieved by the optimal adaptive schedule. Note this is an apples-to-apples comparison where we relate the performance of our solution to the best solution of the same type, and not to a clairvoyant optimum that knows the future. This problem has been of significant interest in both the theoretical computer science and operations research communities for almost three decades now [WP80, WVW86, MSU99, SU05, JS18, Sch08, SSU16, GMUX20, IMP15, EFMM19].

While the deterministic problem can be solved optimally (using the shortest processing time policy), the stochastic setting is significantly more challenging. Early results for stochastic completion time minimization focused on giving optimal policies only for restricted classes of instances, e.g., the case where all job distributions were exponentials, or where the jobs could be stochastically ordered [WP80, WVW86]. Then, starting with the ground-breaking work of Möhring, Schulz, and Uetz [MSU99], approximation algorithms were given that worked

for all stochastic instances. However, almost all such algorithms have approximation ratios with at least linear dependence on the *squared coefficient of variation* $\Delta := \max_j \frac{\mathrm{Var}(X_j)}{(\mathbb{E}X_j)^2}$ [MSU99, SU05, JS18, Sch08, SSU16, GMUX20]. Since this squared coefficient of variation could be very large in general (even for Bernoulli jobs), we want to obtain approximations which are *distribution-independent*, and in particular, do not depend on the coefficient of variation.

There are significant roadblocks to obtaining such distribution-independent guarantees: the known algorithmic toolkit for deterministic jobs relies on greedy policies and linear program-based algorithms [HSSW97, PST04, Pin08]. For the former, the natural *Shortest Expected Processing Time* (SEPT) policy has an approximation ratio no better than $\Omega(n^{1/4})$ [IMP15]. Moreover, even for instances consisting of only two types of jobs—identical unit-sized deterministic jobs and identical Bernoulli jobs $X_j \sim s \cdot \mathrm{Ber}(p)$, no *index policy* (which assigns each job an "index" depending only on its size distribution, and then schedules the jobs in order of their indices) can have bounded approximation ratio [EFMM19]. Approaches based on linear programming also do not seem to extend to the stochastic setting: the most expressive time-indexed linear program commonly used for such settings has an integrality gap of $\Omega(\Delta)$ [SSU16]. Finally, we know that *adaptivity gap*—the gap between the optimal adaptive and fixed-assignment policies[1]—is $\Omega(\Delta)$ [SSU16]. Taken together, these lower bounds rule out most of the tools that work for the setting of deterministic jobs.

The only distribution-independent approximations for stochastic completion time minimization have approximation ratios at least linear in m [IMP15, EFMM19]. In fact, nothing better than an $O(m)$ approximation is known even for instances consisting of only two types of jobs: identical unit-sized deterministic jobs and identically distributed Bernoulli jobs $X_j \sim s \cdot \mathrm{Ber}(p)$ [EFMM19]. For general job distributions, the best known approximation is $O(m \operatorname{poly}\log n)$ [IMP15].

Again, there are barriers to obtaining approximations that are sublinear in m: these previous works use "volume" lower bounds, which rely on the fact that the processing capacity of m machines is m times larger than that of a single machine. Indeed, the objective is extremely sensitive to the number of machines m: decreasing the number of machines from m to $\frac{m}{2}$ can change the optimal adaptive policy's objective value by an exponential in m factor. (This is in stark contrast to the deterministic setting, where the optimal solution's objectives for m and $\frac{m}{2}$ machines differ by at most a *constant factor*.) See Appendix A for proofs of these two claims. This gives a sense for why lower bounds on the optimal objective value based on the number of machines m do not generalize well to the stochastic setting, except with a loss of a factor of m.

In summary, the main question we ask is:

> *Can we break through both the Δ- and m-barriers for the basic problem of completion time minimization for stochastic jobs?*

Despite the difficulty in obtaining improved approximations for this problem, it is possible that the problem has a constant-factor approximation!

1.1 Our Results In this paper, we consider the case of (non-identical) *Bernoulli jobs*, i.e., with independent processing times $X_j \sim s_j \cdot \mathrm{Ber}(p_j)$ for size $s_j \geq 0$ and probability $p_j \in [0, 1]$. Our main result is the first algorithm that is both distribution-independent and has an approximation ratio sublinear in m. We use the notation $\widetilde{O}(\cdot)$ to hide $\operatorname{poly}\log n$-factors.

THEOREM 1.1. (MAIN THEOREM) *There exists an efficient deterministic algorithm for completion time minimization for Bernoulli jobs that computes a list schedule that $\widetilde{O}(\sqrt{m})$-approximates the optimal adaptive policy.*

By list schedule, we mean our algorithm produces a list (i.e., an ordering) of all the jobs, and whenever a machine is free, it schedules the next job according to this ordering. Bernoulli jobs already are a significant generalization of the setting of [EFMM19], and our result improves (up to $\operatorname{poly}\log n$-factors) the $O(m)$-approximation of [EFMM19] and the $\widetilde{O}(m)$-approximation of [IMP15] for this special case of Bernoulli jobs. A corollary of our result is an upper-bound of $\widetilde{O}(\sqrt{m})$ on the adaptivity gap between the optimal adaptive policy and list schedules for the special case of Bernoulli jobs.

[1] Such a policy non-adaptively assigns jobs to machines and runs each machines' jobs in a fixed order.

We view Bernoulli jobs as an important testbed for new techniques for this central problem in stochastic scheduling. Progress on this problem has stalled since the the $\tilde{O}(m)$-approximation of [IMP15], and our work gives develops new analysis techniques towards a $o(m)$-approximation for general distributions. In particular, we refine the weighted free time objective (which was implicitly considered in [IMP15]) by incorporating the batch constraint in our algorithm and analysis. We remark that a consequence of our techniques is a simple $\tilde{O}(m)$-approximation for Bernoulli jobs, matching (up to poly $\log n$-factors) the result of [IMP15] in this special case (see Section 4.2 for details.)

Considering Bernoulli jobs has been an important stepping-stone in other stochastic problems (e.g., for stochastic makespan minimization [KRT00]), where algorithms for Bernoulli jobs could be extended to general distributions. While we do not see how to get the extension yet, we hope that our technical framework will soon lead to such extensions via our new proxy objective – weighted free time (which is valid for general distributions) – and the techniques we develop to optimize it.

1.2 Technical Overview Our algorithm design will be informed by a proxy objective function, which we call the *weighted free time*. We first observe that to bound the completion time of a job, it suffices to bound its starting time, S_j. This is because on identical machines, we have $C_j = S_j + X_j$, where $\sum_j X_j$ is a lower-bound on the optimal total completion time. The key idea of our proxy objective is to relate the per-job starting times to a more global quantity, which we call the *free time.*

Definition 1.1. (Free Time) *Consider any fixed schedule. The ith free time of the schedule, which we denote by $F(i)$ is the first time when i jobs have been started and at least one machine is free to start the $(i+1)$st job.*

For schedules that do not idle machines, the ith free time is the load of the least-loaded machine after starting i jobs. By definition of free time, there are $\Theta(n/2^k)$ jobs with starting times in $[F(n-n/2^{k-1}), F(n-n/2^k)]$ for all $k = 1, \ldots, \log n$ (all logarithms are base 2 in this paper.) Thus we have:

$$\sum_j S_j = \sum_{k=1}^{\log n} \Theta(n/2^k) F(n - n/2^k). \tag{1.1}$$

We call this final expression, $\sum_{k=1}^{\log n} n/2^k \cdot F(n-n/2^k)$, the *weighted free time* of the schedule. We can view this objective as defining $\log n$ work checkpoints for our algorithm. These checkpoints are the time that we have $n/2^1$ jobs left to start (i.e. $F(n-n/2^1)$), $n/2^2$ left to start (i.e. $F(n-n/2^2)$), and so on. Roughly, the goal of our algorithm is to ensure that at each work checkpoint, our free time is comparable with the optimal schedule's free time at the same checkpoint.

We can now illustrate the reason for considering free times rather than the completion time directly. Indeed, let $C(i)$ be the time that we complete i jobs (and note the difference with C_j, which is the time at which we finish a specific job j). We analogously have $\sum_j C_j = \Theta(\sum_k n/2^k \cdot C(n-n/2^k))$. However, one difficulty of stochastic jobs is we cannot easily control what are the first $n - n/2^k$ jobs *to complete*. On the other hand, for free times, we have complete control over what $n - n/2^k$ jobs we decide *to start* first, which then contribute to $F(n-n/2^k)$. This suggests two natural informal subproblems for our algorithm design:

- **Subset Selection:** Compute nested sets of jobs $J_1 \subset J_2 \subset \ldots$ such that for all k, J_k is comparable to the first $n - n/2^k$ jobs of the optimal adaptive policy (i.e. the jobs contributing to $F(n-n/2^k)$ for opt.)
- **Batch Free Time Minimization:** Given nested sets of jobs $J_1 \subset J_2 \subset \ldots$, schedule the J_k's such that our free time after scheduling J_k is comparable to $F(n-n/2^k)$ for opt. Our schedule must satisfy the *batch constraint* that we schedule J_k (i.e. start every job in J_k) before $J_{k+1} \setminus J_k$ for all k.

The main technical challenge in both subproblems is the interaction between the free time and the batch constraint. Since our final algorithm will be a list schedule, the J_k-sets are chosen non-adaptively. However, the optimal policy chooses its first $n-n/2^k$ jobs *adaptively*, so it is not clear that there even exist good sets J_k. Our first contribution is that we can indeed efficiently find good J_k sets non-adaptively by delaying slightly more jobs than opt.

Theorem 1.2. (Subset Selection, Informal) *Given Bernoulli jobs, we can efficiently find nested sets of jobs $J_1 \subset \cdots \subset J_{\log n}$ such that $|J_k| = n - \widetilde{O}(n/2^k)$ and J_k is a subset of the first $n-n/2^k$ jobs of the optimal adaptive policy for all realizations.*

Our subset selection algorithm is a simple greedy algorithm: to construct J_k, for each possibly Bernoulli size parameter (which we may assume there are $O(\log n)$ of by standard discretization techniques) we remove the $n/2^k$ jobs with largest probability parameters. Thus, for each size, we keep the "smallest possible" jobs. The analysis relies on the following structural characterization of the optimal adaptive policy for Bernoulli jobs.

LEMMA 1.1. *Consider a collection of Bernoulli jobs. Then for each possible size parameter, the optimal adaptive completion time schedule for these jobs starts the jobs with this size parameter in increasing order of their probabilities for all realizations of the job sizes.*

In light of Theorem 1.2, it remains to compute good list schedule of the J_k's subject to the batch constraint. Our free time minimization algorithm is also greedy: For each batch $J_k \setminus J_{k-1}$, we list-schedule them in increasing order of size parameter.

THEOREM 1.3. (BATCH FREE TIME MINIMIZATION, INFORMAL) *Given the nested sets of Bernoulli jobs $J_1 \subset \cdots \subset J_{\log n}$ guaranteed by Theorem 1.2, list scheduling them in increasing order of size parameter (subject to the batch constraint) is $\widetilde{O}(\sqrt{m})$-approximate for weighted free time.*

Combining Theorems 1.2 and 1.3 gives our desired $\widetilde{O}(\sqrt{m})$-approximation for completion time minimization for Bernoulli jobs. We now give an overview of our analysis.

For a moment, suppose that we could list-schedule the batches output by Theorem 1.2, $J_1 \subset \ldots J_{\log n}$, *optimally* subject to the batch constraint (i.e. for each $J_k \setminus J_{k-1}$, we compute an ordering of the jobs and start the jobs in this order for all $k = 1, \ldots, \log n$) to minimize the weighted free time. At first glance, it might seem like we are done, because J_k is always a subset of opt's first $n - n/2^k$ jobs, so we are only scheduling fewer jobs than opt at every work checkpoint. However, this reasoning fails because of the batch constraint. Indeed, let us contrast the classic makespan minimization problem (schedule *deterministic* jobs to minimize the load of the most-loaded machine) with its free time analogue (schedule n deterministic jobs to minimize the nth free time). While an arbitrary list schedule is $O(1)$-approximate for makespan, it is $\Omega(m)$-approximate for nth free time:

LEMMA 1.2. *For all $m > 1$, there exists a set of n jobs J and a list-schedule of J whose nth free time $\Omega(m)$-approximates the optimal nth free time.*

Proof. Consider m "small" jobs of size 1 and $m - 1$ "big" jobs of size m. The optimal free time schedule is to first schedule one small job on each machine, and then one big job on $m - 1$ machines. Thus the optimal nth free time is 1. Now consider the list-schedule of all big jobs before small jobs. Then each big job is scheduled on a separate machine, and all m small jobs are scheduled on the remaining machine. This gives nth free time m. ∎

The instances from the above lemma suggest that we should schedule small jobs before bigger ones so that the big jobs do not clog up the machines and delay the starting times of the small jobs. Implicitly, this is why previous work loses a m-factor: While opt has m machines to schedule small jobs before the machines are clogged by big ones, it can be the case that alg first clogs all but one machine with big jobs and then schedules all small jobs on a single machine. For our algorithm, because of the batch constraint, it could be the case that opt does some small jobs in $J_k \setminus J_{k-1}$ much earlier than our algorithm when fewer machines are clogged by big jobs. This is the main technical challenge that we overcome to obtain our improvement.

Concretely, consider scheduling J_k subject to the batch constraint. For this subproblem, we say a job is big if its realized size is larger than opt's (random) $n - n/2^k$th free time. Scheduling such jobs effectively turns off a machine for the remainder of the schedule. We call such machines *clogged*. One should imagine that afterwards we are averaging the volume of the remaining small jobs over one less machine. Our first insight is a stronger lower bound on the rate that opt clogs machines using Lemma 1.1. We let J_k^* be the adaptively-chosen set of the first $n - n/2^k$ jobs started by opt. Because $J_{k'} \subset J_{k'}^*$ for every batch k' *for every realization of job sizes*, the number of big jobs in J_k' is at most the number of big jobs in $J_{k'}^*$. This implies that our algorithm clogs machines at a slower rate than opt for every realization.

Similarly, Lemma 1.1 also implies that the total size of small jobs in $J_{k'}$ is at most the total size of small jobs in $J_{k'}^*$ for every $k' \leq k$. However, this does *not* guarantee that the total size of small jobs in $J_{k'} \setminus J_{k'-1}$ is at most that in $J_{k'}^* \setminus J_{k'-1}^*$, so although our algorithm has more unclogged machines, we may also be trying to average more small jobs over these machines. Our second insight is a delicate charging argument that characterizes how

the free times of previous batches affect the current one. Roughly, we argue that for our particular batches, while moving a small job to an earlier batch allows us to average it over more unclogged machines, this also delays the free time of all later batches. This ensures that there is not much benefit to moving small jobs to earlier batches. To achieve this, we initiate a systematic study of the free time.

1.3 Comparison to prior work Prior $O(\Delta)$-approximations rely on bounding with respect to an LP solution, e.g. [MSU99, SSU16]. These have an integrality gap of $\Omega(\Delta)$. On the other hand, our algorithm is combinatorial and avoids this gap by comparing directly to the optimal adaptive policy.

The distribution-independent approximation of [IMP15] also partitions jobs into batches (as in our subset selection problem). Roughly, they guarantee that their batches are "better" than the optimal solution's jobs "in expectation." However, we will show that our algorithm's batches are better than the optimal solution's jobs for *every* realization (using our structural characterization of the optimal policy Lemma 1.1.)

Further, their algorithm schedules jobs within batches in arbitrary order (i.e. they give a trivial solution to the subproblem we call free time minimization). It seems likely that a loss of $\Omega(m)$ is necessary if one considers an arbitrary list schedule because of the lower bound in Lemma 1.2. To overcome this, we choose a particular list schedule (i.e. in increasing order of size parameter), which we show is $\widetilde{O}(\sqrt{m})$-approximate. To summarize, using a deeper technical understanding, we give more refined guarantees for both subset selection and free time minimization than [IMP15].

The only other work is [EFMM19], which considers even more restricted instances: those with only two types of jobs, identical deterministic and identical Bernoulli. Their algorithm is to schedule either all deterministic jobs first or all Bernoullis first (depending on the relative number of each type of jobs.) Our subset selection algorithm vastly generalizes this idea to arbitrary Bernoulli jobs with varying size and probability parameters. Further, while our algorithm in Theorem 1.3 runs an index policy within each $J_k \setminus J_{k-1}$-batch, we overcome the lower bound on index policies due to [EFMM19] because our subset selection algorithm constructs the batches by taking into account the relative number of different types of jobs—not only the distributions of individual jobs.

1.4 Related Work Many stochastic combinatorial optimization problems have been studied from an approximation perspective; the previous results closest to this work are packing problems including those on stochastic versions of knapsack [DGV04, GKMR11, BGK11, LY13], orienteering [GKNR12], multi-armed bandits [GMS10, Ma14], generalized assignment [AHL13], and packing integer programs [DGV05]. Some stochastic versions of covering problems include k-TSP [ENS17, JLLS20] and submodular cover [AAK19, GGN21]. Another important class of stochastic problems is probing/selection problems [GGM10, GNS16, GNS17, FLX18].

For stochastic scheduling problems, approximations are known for load balancing [GI99, KRT00, GKNS18, DKLN20] and completion time minimization with precedence constraints [SU01], preemption [MV14], release dates and online arrivals [MUV06]; however the latter works have approximations that depend on the variance of job sizes.

2 Subset Selection

The goal of this section is to solve the *subset selection subproblem* for Bernoulli jobs: we want to find nested sets of jobs $J_1 \subset \cdots \subset J_{\log n}$ such that J_k is comparable to the first $n - n/2^k$ jobs of the optimal adaptive completion time schedule. Formally, let J_k^* be the random set consisting of the first $n - n/2^k$ jobs scheduled by the optimal completion time schedule. Our main theorem here is the following:

THEOREM 2.1. (SUBSET SELECTION) *Let L be the number of distinct Bernoulli size parameters. There is an algorithm* CHOOSEJOBS *that outputs sets J_k satisfying:*

(i) $J_1 \subset \cdots \subset J_{\log n} \subset J$
(ii) $|J_k| \in [n - L \cdot n/2^k, n - n/2^k]$
(iii) $J_k \subset J_k^*$ *for all k and all realizations.*

We show later how to use standard rescaling and discretization techniques to assume $L = O(\log n)$ while losing only an extra constant factor in our final approximation ratio.

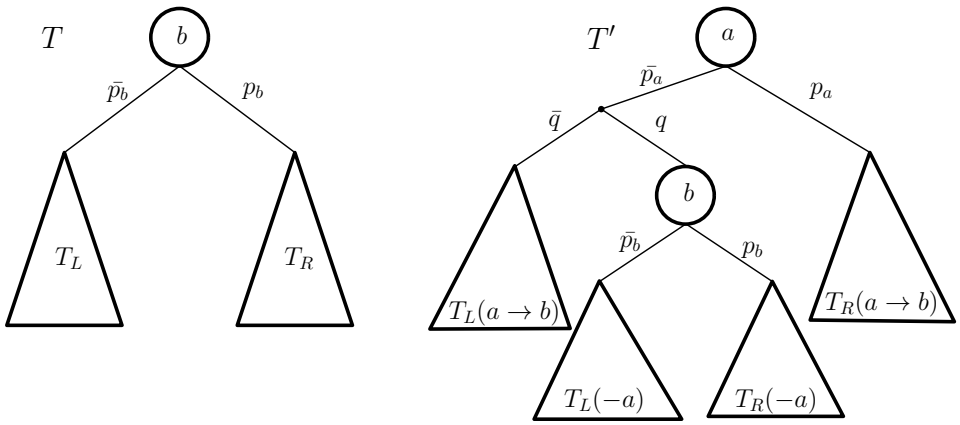

Figure 1: Original decision tree T and modified decision tree T'

It is convenient to think of the $n/2^k$ jobs that the optimal schedule *excludes* from J_k^* rather than the jobs it chooses to start. Similarly, we specify our algorithm's set of jobs also by the exclusions. The next lemma gives our structural characterization for the optimal adaptive completion time schedule for Bernoulli jobs, which allows us to characterize which jobs the optimal schedule chooses to exclude.

Because jobs are Bernoullis, upon scheduling a job j, the scheduler immediately learns the realized size of X_j because it is either 0 or s_j. Thus, the optimal schedule can be represented by a decision tree, where each node is labeled by a job j, corresponding to the decision to schedule j on the currently least loaded machine, and has a left- and right child, corresponding to the realized size of j being 0 or s_j, respectively. Every root-leaf path on this tree gives an ordering to schedule the jobs for a particular realization of job sizes.

LEMMA 1.1. *Consider a collection of Bernoulli jobs. Then for each possible size parameter, the optimal adaptive completion time schedule for these jobs starts the jobs with this size parameter in increasing order of their probabilities for all realizations of the job sizes.*

Proof Sketch. Our proof is an exchange argument. Consider the optimal decision tree (as described above), and suppose there exists a root-leaf path that schedules job $X_b \sim s \cdot \mathrm{Ber}(p_b)$ before $X_a \sim s \cdot \mathrm{Ber}(p_a)$ with $p_a \le p_b$. Then there exists a subtree T rooted at b such that a is scheduled on every root-leaf path in this subtree.

We now show how to modify T to start a before b while not increasing the expected completion time. Let T_L and T_R be the left- and right subtrees (corresponding to the root job b coming up size 0 or s) of T, respectively. We define $T_L(a \to b)$ to be T_L with the job a replaced by job b and $T_L(-a)$ to be T_L, but at a's node, we do not schedule anything and instead go to a's left child. The subtrees $T_R(a \to b)$ and $T_R(-a)$ are defined analogously. See Figure 1 for the modified tree T'.

We choose the parameter q so that the probability that T' enters $T_L(a \to b)$ or $T_L(-a)$ is exactly $\bar{p}_b$. This is our replacement for the event that T enters T_L. A calculation now shows that the expected completion time weakly decreases from T to T'. (See the proof in Appendix B for details.) ∎

By definition opt can exclude only $n/2^k$ jobs from J_k^*. On the other hand, our algorithm will exclude $n/2^k$ jobs of each Bernoulli size *simultaneously*. Lemma 1.1 suggests that we might as well exclude the jobs with largest p_j's. In particular, our algorithm to choose sets of jobs that are comparable to the J_k^*'s is the following:

CHOOSEJOBS: For each $k = 1, \ldots, \log n$, let J_k be the set of jobs constructed as follows:

i. Initialize $J_k = J$.
ii. For each Bernoulli size s, remove from J_k the $n/2^k$ jobs of size s with largest p_j's.
iii. Output J_k.

Proof of Theorem 2.1. It is immediate that the sets $J_1, \ldots, J_{\log n}$ are nested and have the desired size. It remains to show that $J_k \subset J_k^*$. Note that opt excludes at most $n/2^k$ jobs of the largest probabilities of each size by Lemma 1.1. This holds for all realizations. On the other hand, we exclude $n/2^k$ jobs of largest probability of all sizes simultaneously. ∎

Morally, Theorem 2.1 states that we know the jobs that opt starts to achieve $F^*(n - n/2^k)$ for all k (up to a L-factor.) This suggests that we should first schedule J_1 to get free time comparable to $F^*(n - n/2^1)$, and then $J_2 \setminus J_1$ to get free time comparable to $F^*(n - n/2^2)$, and so on.

The goal of the next section is to show how to schedule the J_k's subject to this batch constraint (we must schedule all jobs in J_{k-1} before any in $J_k \setminus J_{k-1}$ for all k) such that our weighted free time is comparable to that of opt. Note that in general, even though $J_k \subset J_k^*$ for all k, the *optimal* completion time schedule may not satisfy the batch constraint—this is precisely is the main technical challenge that we have to overcome in the next section.

3 Batch Free Time Minimization

We now turn to the *batch free time minimization* problem. Our starting point is the nested sets of jobs $J_1 \subset \cdots \subset J_{\log n} \subset J$ output by ChooseJobs. Recall that J_k^* is the random set of the first $n - n/2^k$ jobs scheduled by the optimal completion time policy opt.

3.1 Free Time Basics

To motivate our final algorithm, we explore some basic properties of the free time. We first recall the lower bound instance from Lemma 1.2: there are m small jobs of size 1 and $m-1$ big jobs of size m. The optimal nth free time is 1 by first scheduling all small jobs and then all big jobs. Observe that the final $m-1$ big jobs do not contribute to the optimal nth free time. This gave the intuition that we should schedule small jobs before big ones clog up the machines. This intuition turns out to be correct, which we formalize in the next lemma.

Lemma 3.1. *List-scheduling deterministic jobs in increasing order of size is a 4-approximation for nth free time.*

Proof. Let J be the set of input jobs and opt the optimal nth free time. We partition J into *small* and *big* jobs: a job is big if its size is strictly greater than opt, and is small otherwise. This definition means that opt can schedule at most one big job per machine. Moreover, there are strictly less than m big jobs, otherwise opt would need to schedule at least one big job per machine, which contradicts the optimal nth free time.

On the other hand, our algorithm starts all small jobs before all big jobs. We claim that the nth free time of our algorithm, which we denote by $F(J)$, is at most the makespan of our algorithm after only scheduling the small jobs, which we denote by $M(S)$. To see this, consider the time right before we start the first big job. All machines have loads within $[M(S) - \mathsf{opt}, M(S)]$ (the lower bound follows by noting that we list-scheduled only jobs of size at most opt up until this point.) Thus, we schedule the at most $m-1$ remaining big jobs each on separate machines. All of the big jobs are started by time $M(S)$, and there exists a machine that schedules no big job, which is free by $M(S)$ as well. It follows, $F(J) \le M(S)$.

Now, because every list-schedule is 2-approximate for makespan, we have $M(S) \le 2M^{\mathsf{opt}}(S)$, where $M^{\mathsf{opt}}(S)$ is the makespan of the small jobs under opt (i.e. when opt finishes its final small job.) To complete the proof, we relate $M^{\mathsf{opt}}(S)$ with opt. It suffices to show $M^{\mathsf{opt}}(S) \le 2\mathsf{opt}$. To see this, consider time opt in the optimal schedule. At this time, all machines are either free or working on their final job. In particular, any machine working on a small job completes by time $\mathsf{opt} + \mathsf{opt}$. ∎

While we do not apply Lemma 3.1 directly for our algorithm, the ideas in the analysis will be crucial. In particular, a key concept in our analysis is to differentiate between small and big jobs. Roughly, when we consider J_k, a job is small if its size is at most $F^*(n - n/2^k)$ and big otherwise. We are concerned about the volume (total size) of the small jobs and number of big jobs. However, because of the batch constraint, we cannot ensure that all small jobs are scheduled before all big jobs in general.

As we schedule batches $J_1, J_2 \setminus J_1, \ldots, J_k \setminus J_{k-1}$, more and more machines are getting clogged by big jobs. For the purposes of scheduling J_k, these machines are effectively turned off. Thus, as we proceed through the batches, we are averaging the volume of small jobs over fewer and fewer machines. The goal of our algorithm will be to ensure that we do small jobs as early as possible (subject to the batch constraint) so that we have the most unclogged machines available.

3.2 Final Algorithm With the goal of §3.1 in mind, we are ready to describe our final algorithm, which is the $\widetilde{O}(\sqrt{m})$-approximation guaranteed by Theorem 1.1. Although we cannot ensure that within a batch, jobs are scheduled in increasing order of *realized* size as in the analysis of Lemma 3.1, because our jobs are Bernoullis, we can ensure that all jobs that come up heads (have realized size s_j) are scheduled in increasing order of realized size. Here, we crucially use the fact that our jobs are Bernoullis, so if the come up tails (have size 0), they do not affect the free time. For the rest of the description, we make the following assumption, which we justify in Appendix C.

ASSUMPTION 3.1. *We assume that there are $L = O(\log n)$ distinct Bernoulli size parameters s_j, each at most n^8.*

By losing a constant factor in the final approximation ratio, we may assume Assumption 3.1 for the rest of the analysis.

LEMMA 3.2. *Let $m \geq 2$. Suppose there exists an algorithm for completion time minimization for Bernoulli jobs on m machines satisfying Assumption 3.1 that outputs a list schedule with expected completion time at most $\alpha(\mathbb{E}\mathsf{opt} + O(1))$. Then there exists a $O(\alpha)$-approximate algorithm for the same problem without the assumption. Further, the resulting algorithm is also a list schedule, and it preserves efficiency and determinism.*

The proof uses standard rescaling and discretization ideas, but it is more involved because of the stochastic jobs; we justify it in Appendix C. Our final algorithm is now the following:

STOCHFREE: Given input collection J of Bernoulli jobs:

i. Run CHOOSEJOBS to obtain nested sets of jobs $J_1 \subset \cdots \subset J_{\log n} \subset J$

ii. List-schedule each batch $J_k \setminus J_{k-1}$ in increasing order of Bernoulli size parameter s_j for all batches $k = 1, \ldots, \log n$.

iii. List-schedule all remaining jobs $J \setminus J_{\log n}$ in arbitrary order.

It is clear that STOCHFREE outputs a list schedule in polynomial time and is deterministic. Our main approximation guarantee for STOCHFREE is the following.

THEOREM 3.2. (BATCH FREE TIME MINIMIZATION) *Given Bernoulli jobs, if $m \geq 2$ and Assumption 3.1 holds, then* STOCHFREE *outputs a list schedule with expected completion time at most $\widetilde{O}(\sqrt{m}) \cdot (\mathbb{E}[\mathsf{opt}] + O(1))$, where opt is the optimal adaptive policy.*

Note that composing Theorem 3.2 with Lemma 3.2 gives the desired $\widetilde{O}(\sqrt{m})$ without the assumption for all $m \geq 2$. For the remaining case of $m = 1$, scheduling the jobs in increasing order of their expected processing times is an optimal policy [Rot66]. This gives the desired $\widetilde{O}(\sqrt{m})$-approximation for all m, and completes the proof of Theorem 1.1. In the remainder of the paper, we analyze STOCHFREE (Theorem 3.2.)

4 Analysis of the StochFree Algorithm

The goal of this section is to prove Theorem 3.2, given Assumption 3.1. Our proof has four conceptual steps.

i. Bound the weighted free time of alg by averaging the volume of small jobs within each batch over the unclogged machines—those that have not yet scheduled a big job. (§4.1)

ii. Show that STOCHFREE is $\widetilde{O}(m)$-approximate for all $m \geq 2$. This serves as a warm-up to the improved $\widetilde{O}(\sqrt{m})$-approximation, and it allows us to focus on the remaining case where $m = \Omega(1)$ is sufficiently large. (§4.2)

iii. Control the rate that machines become clogged by a large job. We show that the rate that machines become clogged for alg is slow enough so that opt cannot benefit much by putting small volume in earlier batches than alg. (§4.3)

iv. Finally, bound the contribution of the volume of small jobs to the free time. Here we handle the main challenge, which is that opt may schedule small volume "in the past" compared to alg. (§4.4–4.6)

4.1 Weighted free time First, we pass from total completion time to our proxy objective of weighted free time. We let opt denote the optimal adaptive completion time policy as well as its completion time. Recall that J_k^* is the first $n - n/2^k$ jobs scheduled by this policy achieving free time $F^*(n - n/2^k)$. Analogously, we let alg denote the completion time of our algorithm, and $F(J_k)$ the free time of our algorithm after scheduling J_k.

LEMMA 4.1. *We have* $\mathsf{alg} = O(\log n)(\sum_k n/2^k \cdot F(J_k) + \mathsf{opt})$ *and* $\mathsf{opt} = \Omega(\sum_k n/2^k \cdot F^*(n - n/2^k))$.

Proof. We rewrite $\mathsf{alg} = \sum_j C_j = \sum_j S_j + \sum_j X_j$. First, note that $\sum_j X_j \le \mathsf{opt}$. It remains to bound $\sum_j S_j$. Recall that in StochFree, first we list-schedule $J_{\log n}$ subject to the batch constraint and then $J \setminus J_{\log n}$. We first handle the starting times of $J_{\log n}$. For all k, note that $J_k \setminus J_{k-1}$ consists of at most $O(L) \cdot n/2^k$ jobs with starting times in $[F(J_{k-1}), F(J_k)]$ by Theorem 2.1. Thus we have $\sum_{j \in J_{\log n}} S_j = O(L) \cdot (\sum_k n/2^k \cdot F(J_k)) = O(\log n) \cdot (\sum_k n/2^k \cdot F(J_k))$ using Assumption 3.1.

For the jobs in $J \setminus J_{\log n}$, by Theorem 2.1, there are at most $O(L) = O(\log n)$ such jobs. Each of these jobs completes by the makespan of alg's schedule. Further, alg is a list schedule, which is 2-approximate for makespan, so the makespan of alg is at most twice the makespan of opt. The makespan of opt is a lower bound on opt (because some job must complete at this time.) We conclude, $\sum_{j \in J \setminus J_{\log n}} S_j = O(\log n)\mathsf{opt}$. Combining our bounds for $J_{\log n}$ and $J \setminus J_{\log n}$ gives the desired result for alg.

The bound on opt follows from Equation (1.1). ∎

We refer to $\sum_k n/2^k \cdot F(J_k)$ as alg's weighted free time and $\sum_k n/2^k \cdot F^*(n - n/2^k)$ as opt's. The remainder of the analysis will focus on bounding alg's weighted free time with respect to opt's. Our main result is the following:

THEOREM 4.1. *If* $m = \Omega(1)$ *is sufficiently large, then the weighted free time of* alg *satisfies:*

$$\mathbb{E}\left[\sum_k n/2^k \cdot F(J_k)\right] = \widetilde{O}(\sqrt{m}) \cdot \left(\mathbb{E}\left[\sum_k n/2^k \cdot F^*(n - n/2^k)\right] + \mathbb{E}[\mathsf{opt}]\right) + O(1).$$

Note that Theorem 4.1 along with Lemma 4.1 implies the desired guarantee in Theorem 3.2 for the case $m = \Omega(1)$ sufficiently large.

We now introduce some notations. For all k, we call $I_k = J_k \setminus J_{k-1}$ the kth *batch* of jobs. Recall that the J_k's are nested, so the batch constraint says we schedule in order $I_1, \ldots, I_{\log n}$. We define $I_k^* = J_k^* \setminus J_{k-1}^*$ analogously. For any set of jobs, J' and $\tau \ge 0$, we define $J'(= \tau)$ to be the random subset consisting of all jobs in J' with *realized* size exactly τ. We define $J'(> \tau)$ and $J'(\le \tau)$ analogously. Further, for a set of jobs J', we let $\mathrm{Vol}(J') = \sum_{j \in J'} X_j$ be the volume of J'. Finally, we say job j is τ-big for $\tau \ge 0$ if $X_j > \tau$. Otherwise j is τ-small.

As in the analysis for minimizing the free time for a single batch of deterministic jobs (Lemma 3.1), the key concept is to differentiate between small and big jobs. To this end, for all k we define the random threshold $\tau_k = 2 \cdot \max(\mathbb{E}F^*(n - n/2^k), F^*(n - n/2^k))$. Morally, one should imagine that τ_k is $F^*(n - n/2^k)$, but there is an edge case where $F^*(n - n/2^k) < \mathbb{E}F^*(n - n/2^k)$ and a multiplicative factor for concentration. When bounding $F(J_k)$, we will take τ_k to be our threshold between small- and big jobs. This threshold has the following crucial property that alg always has at least as many unclogged machines as opt. In particular, alg always has at least one unclogged machine.

PROPOSITION 4.2. *For all* k*, the following holds per-realization:* $|J_k(> \tau_k)| \le |J_k^*(> \tau_k)| < m$.

Proof. The first inequality follows from Theorem 2.1, because $J_k(> \tau_k) \subset J_k^*(> \tau_k)$ per-realization. For the second inequality, note that $\tau_k \ge F^*(n - n/2^k)$, so by definition of free time, opt schedules strictly less than m jobs bigger than τ_k to achieve $F^*(n - n/2^k)$. ∎

Using this threshold, we re-write $F(J_k)$ by averaging the volume of small jobs over the unclogged machines (the ones with no big job.)

LEMMA 4.2. *For all* k*, the following holds per-realization:*

$$F(J_k) \le F(J_{k-1}) + \frac{\mathrm{Vol}(I_k(\le \tau_k))}{m - |J_{k-1}(> \tau_k)|} + 2\tau_k.$$

Proof. First, we note by Proposition 4.2 that the denominator $m-|J_{k-1}(>\tau_k)| \geq 1$. Then consider time $F(J_{k-1})$. There are at least $m-|J_{k-1}(>\tau_k)|$ machines that have not scheduled a τ_k-big job in J_{k-1}. At this time, each machine is either free or working on its final job in J_{k-1}. In particular, each machine that has not scheduled a τ_k-big job in J_{k-1} is free to start working on I_k by time $F(J_{k-1})+\tau_k$, and there are at least $m-|J_{k-1}(>\tau_k)|$ such machines.

We need the following monotonicity property of list schedules.

LEMMA 4.3. *Consider a set of deterministic jobs and a fixed list schedule of those jobs. Then increasing the initial load or decreasing the number of machines weakly increase the free time of the schedule.*

Proof. Let J be the set of jobs. Consider initial load vectors $\ell, \ell' \in \mathbb{R}^m$, where the ith entry of each vector denotes the initial load on machine i. Now suppose $\ell \leq \ell'$, entry-wise. It suffices to show that $F(J,\ell) \leq F(J,\ell')$, where $F(J,\ell)$ is the free time achieved by our list-schedule with initial load ℓ. This suffices, because we can decrease the number of machines by making the initial loads of some machines arbitrarily large so that they will never be used.

We prove $F(J,\ell) \leq F(J,\ell')$ by induction on the number of jobs, $|J|$. In the base case, $|J|=0$, so the claim is trivial because $\ell \leq \ell'$. For $|J|>0$, let j be the first job in the list, which is scheduled, without loss of generality, on the first machine for both initial loads ℓ and ℓ'. Then:

$$F(J,\ell) = F(J\setminus\{j\}, \ell + s_j e_1) \leq F(J\setminus\{j\}, \ell' + s_j e_1) = F(J,\ell'),$$

where e_1 is the first standard basis vector, so we have $\ell + s_j e_1 \leq \ell' + s_j e_i 1$ entry-wise. Then we assumed inductively that $F(J\setminus\{j\}, \ell + s_j e_1) \leq F(J\setminus\{j\}, \ell' + s_j e_1)$. ∎

By Lemma 4.3, we can upper-bound $F(J_k)$ by list-scheduling I_k with initial load $F(J_{k-1})+\tau_k$ on $m-|J_{k-1}(>\tau_k)|$ machines that have not scheduled a τ_k-big job in J_{k-1}. Recall that alg list-schedules I_k in increasing order of size parameter, so - ignoring jobs that come up tails with realized size 0 - we schedule all τ_k-small jobs in I_k before any τ_k-big one. Further, $|I_k(>\tau_k)| < m-|J_{k-1}(>\tau_k)|$ by Proposition 4.2, so there exists some machine that schedules only τ_k-small jobs in I_k. This machine is free by time $F(J_k) \leq F(J_{k-1}) + \tau_k + \frac{\mathrm{Vol}(I_k(\leq\tau_k))}{m-|J_{k-1}(>\tau_k)|} + \tau_k$. ∎

Using Lemma 4.2 and the exponentially decreasing weights, we can re-write alg's weighted free time as:

$$\sum_k n/2^k \cdot F(J_k) = O\Big(\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\leq\tau_k))}{m-|J_{k-1}(>\tau_k)|} + \sum_k n/2^k \cdot \tau_k\Big) \tag{4.2}$$

By definition of τ_k, the second sum is $O(\mathbb{E}\sum_k n/2^k \cdot F^*(n-n/2^k))$ in expectation, which is exactly opt's weighted free time. It remains to bound the first sum.

4.2 Warm up: $\widetilde{O}(m)$-approximation Before proceeding with the proof of Theorem 4.1, we observe that Equation (4.2) along with our basic weighted free time properties is enough to give a $\widetilde{O}(m)$-approximation. Interestingly, this gives a simple proof that nearly matches the previously best-known guarantees for Bernoulli jobs.

LEMMA 4.4. *Given Bernoulli jobs, if $m \geq 2$ and Assumption 3.1 holds, then* StochFree *outputs a list schedule whose expected completion time $\widetilde{O}(m)$-approximates the optimal adaptive policy.*

Proof. Starting from Equation (4.2):

$$\sum_k n/2^k \cdot F(J_k) = O\Big(\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\leq\tau_k))}{m-|J_{k-1}(>\tau_k)|} + \sum_k n/2^k \cdot \tau_k\Big),$$

we note that $I_k \subset J_k^*$ by Theorem 2.1 and $m-|J_{k-1}(>\tau_k)| \geq 1$ by Proposition 4.2. Thus, we can bound:

$$\frac{\mathrm{Vol}(I_k(\leq\tau_k))}{m-|J_{k-1}(>\tau_k)|} \leq \mathrm{Vol}(J_k^*(\leq\tau_k)).$$

We claim that $\text{Vol}(J_k^*(\leq \tau_k)) = O(m) \cdot \tau_k$. To see this, observe that by averaging the volume of $J_k^*(\leq \tau_k)$ over the m machines, after scheduling J_k^*, each machine in opt has load at least $\frac{\text{Vol}(J_k^*(\leq \tau_k))}{m} - \tau_k$. This gives $F^*(n - n/2^k) \geq \frac{\text{Vol}(J_k^*(\leq \tau_k))}{m} - \tau_k$. Noting that $\tau_k \geq F^*(n - n/2^k)$ completes the proof that $\text{Vol}(J_k^*(\leq \tau_k)) = O(m) \cdot \tau_k$.

Applying this to our above expression gives $\frac{\text{Vol}(I_k(\leq \tau_k))}{m - |J_{k-1}(> \tau_k)|} = O(m) \cdot \tau_k$, so alg's weighted free time satisfies:

$$\sum_k n/2^k \cdot F(J_k) = O(m) \cdot (\sum_k n/2^k \cdot \tau_k).$$

Taking expectations and applying Lemma 4.1 completes the proof. ∎

The loss of m in the above proof was because alg averages the small volume over at least 1 unclogged machine, but opt may average the same volume over at most m machines. Intuitively, this reasoning is why previous work loses a m-factor as well.

Further, this is the main technical challenge that we will overcome to get our improvement. Indeed, even though $J_k \subset J_k^*$ for all k, it is not true that $I_k \subset I_{k^*}$. This means that while we are averaging $\text{Vol}(I_k(\leq \tau_k))$ over $m - |J_{k-1}(> \tau_k)|$ machines (which is at least as many machines as opt has for batch k), it can be the case that opt actually did jobs in I_k in much earlier batches. In the remainder of our analysis, we do a more fine-grained analysis of the rate that alg and opt clog machines, and when they choose to do the same small volume. This allows us to break through the linear dependence in m.

4.3 Bounding the unclogged machines In this section, we are interested in controlling the quantity $m - |J_{k-1}(> \tau_k)|$, which is the number of machines we have left to schedule I_k (the unclogged machines.) Note that there are two sources of randomness: the realizations of jobs in J_{k-1} and the threshold τ_k.

Our strategy is to control $m - \mathbb{E}|J_{k-1}(> \tau)|$ for a fixed threshold τ. Then because $|J_{k-1}(> \tau)|$ is a sum of independent $\{0, 1\}$-valued random variables, a Chernoff-union bound argument allows us to control $m - |J_{k-1}(> \tau)|$ as well.

However, we will see that concentration alone is not enough; this is because there is an unbounded difference between $|J_{k-1}(> \tau)| = m$ and $|J_{k-1}(> \tau)| < m - 1$. In the former case, all machines are clogged by big jobs, whose size we cannot upper bound. Thus, we cannot make any progress towards reaching time $F(J_k)$ (by starting more jobs.) In the latter, we have at least one machine, so we can still make some progress towards $F(J_k)$. The situation to keep in mind is when $\mathbb{E}|J_{k-1}(> \tau)|$ is close to m, so concentration around the mean will fail to preserve this hard constraint that we need at least one unclogged machine. To remedy this, we will combine concentration arguments with the per-realization properties of StochFree.

We begin with the concentration arguments, so we wish to understand $m - \mathbb{E}|J_{k-1}(> \tau)|$. We first use the properties of ChooseJobs to bound $\mathbb{E}|J_{k-1}(> \tau)|$:

Proposition 4.3. *For all fixed thresholds τ and batches k, we have $\mathbb{E}|I_k(> \tau)| \geq \frac{1}{2}\mathbb{E}\tau|I_{k-1}(> \tau)|$.*

Proof. By summing over the relevant sizes, it suffices to prove $\mathbb{E}|I_k(= s)| \geq \frac{1}{2}\mathbb{E}|I_{k-1}(= s)|$ for any Bernoulli size parameter s. We may assume $\mathbb{E}|I_{k-1}(= s)| > 0$ or else the proposition is trivial.

Then when ChooseJobs constructs J_{k-1}, it includes at least one job with size parameter s. It follows, there exist $n/2^{k-1}$ remaining jobs in $J \setminus J_{k-1}$ with size parameter s. When constructing J_k, ChooseJobs will include $n/2^k$ of these remaining jobs. In conclusion, I_{k-1} has at most $n/2^{k-1}$ jobs with size parameter s, while I_k has at least $n/2^k$. The result follows because ChooseJobs includes jobs in increasing order of p_j. ∎

Proposition 4.3 allows us to relate the expected number of machines left (with respect to fixed threshold τ) at batch k with the number of machines left at $k' \leq k$:

Lemma 4.5. *For all fixed thresholds τ and batches $k' \leq k$, we have $m' - \mathbb{E}|J_{k-1}(> \tau)| \geq 2^{-(k-k'+1)} \cdot (m' - \mathbb{E}|J_{k'-1}(> \tau)|)$, where $m' \geq \mathbb{E}|J_k(> \tau)|$.*

Proof. We may assume $\mathbb{E}|I_k(> \tau)| > 0$ or else the lemma is trivial, because by definition of ChooseJobs, if $\mathbb{E}|I_k(> \tau)| = 0$, then $\mathbb{E}|J_{k-1}(> \tau)| = 0$ and $\mathbb{E}|J_{k'-1}(> \tau)| = 0$.

In particular, we may assume $m' - \mathbb{E}|J_{k-1}(> \tau)| \geq \mathbb{E}|I_k(> \tau)| > 0$. Then we compute:

$$\frac{m' - \mathbb{E}|J_{k'-1}(> \tau)|}{m' - \mathbb{E}|J_{k-1}(> \tau)|} = 1 + \frac{\mathbb{E}|I_{k'}(> \tau)| + \cdots + \mathbb{E}|I_{k-1}(> \tau)|}{m' - \mathbb{E}|J_{k-1}(> \tau)|} \leq 1 + \frac{\mathbb{E}|I_{k'}(> \tau)| + \cdots + \mathbb{E}|I_{k-1}(> \tau)|}{\mathbb{E}|I_k(> \tau)|}.$$

Repeatedly applying Proposition 4.3 to the numerator gives:

$$1 + \frac{\mathbb{E}|I_{k'}(> \tau)| + \cdots + \mathbb{E}|I_{k-1}(> \tau)|}{\mathbb{E}|I_k(> \tau)|} \leq 1 + (2^{k-k'} + \cdots + 2^1) \leq 2^{k-k'+1}.$$

□

To see the utility of Lemma 4.5, suppose $\mathbb{E}|J_k(> \tau)| = m$. Then roughly the lemma says in expectation, we lose at most half of our remaining machines between each batch. However, in the weighted free time the coefficient $n/2^k$ (corresponding to the number of jobs delayed by the current batch) also halves between each batch. Thus, although we are losing half of our machines, only half as many jobs are affected by this loss.

First, we bound the expectation of $|J_k(> \tau)|$ when τ is sufficiently large (i.e. for all possible realizations of τ_k.) The proof uses a Chernoff bound along with the definition of big jobs; see Appendix D for proof.

LEMMA 4.6. *Let $m = \Omega(1)$ be sufficiently large. Then there exists a constant $c \geq 0$ such that for all batches k and thresholds $\tau > 2\mathbb{E}F^*(n - n/2^k)$, we have $\mathbb{E}|J_k(> \tau)| \leq m + c\sqrt{m}$.*

Now, because $\mathbb{E}|J_k(> \tau)| = O(m)$, we can bound the deviation of $|J_k(> \tau)|$ by $\widetilde{O}(\sqrt{m})$ with high probability.

We define the notation $|J_k(> \tau)| \overset{\pm\Delta}{\approx} \mathbb{E}|J_k(> \tau)|$ to denote the event

$$|\,|J_k(> \tau)| - \mathbb{E}|J_k(> \tau)|\,| \leq \Delta.$$

The proof of the next lemma is a Chernoff-union argument; see Appendix D for proof.

LEMMA 4.7. *Let $\Delta = O(\sqrt{m}\log n)$ and $m = \Omega(1)$ be sufficiently large. Then with probability at least $1 - \frac{1}{\text{poly}(n)}$, the following events hold:*

$$\{|J_k(> \tau)| \overset{\pm\Delta}{\approx} \mathbb{E}|J_k(> \tau)| \quad \forall \text{ batches } k \text{ and thresholds } \tau > 2\mathbb{E}F^*(n - n/2^k)\}. \tag{4.3}$$

Combining Lemma 4.5 and Lemma 4.7, we can show the number of remaining machines is concentrated as well. Here we also need to bring in the per-realization properties of StochFree to handle the case where concentration is not enough to ensure that we have at least one remaining machine. This is the main result of this section. Recall that we defined $\tau_k = 2\max(\mathbb{E}F^*(n - n/2^k), F^*(n - n/2^k))$, so in particular $\tau_k \geq 2\mathbb{E}F^*(n - n/2^k)$.

LEMMA 4.8. *Suppose Event (4.3) holds. Then for all pairs of batches $k' \leq k$, we have $m - |J_{k-1}(> \tau_k)| \geq (3\Delta)^{-1}2^{-(k-k'+1)}(m - |J_{k'-1}(> \tau_k)|)$, where $\Delta = O(\sqrt{m}\log n)$.*

Proof. Consider fixed batches $k' \leq k$, and let $\mu_k = \mathbb{E}|J_k(> \tau_k)|$ and $\mu_{k'} = \mathbb{E}|J'_k(> \tau_k)|$. Note that $\tau_k \geq 2\mathbb{E}F^*(n - n/2^k) \geq 2\mathbb{E}F^*(n - n/2^{k'})$, so Event (4.3) gives $|J_k(> \tau_k)| \overset{\pm\Delta}{\approx} \mu_k$ and $|J_{k'}(> \tau_k)| \overset{\pm\Delta}{\approx} \mu_{k'}$. Further, we may choose $\Delta = O(\sqrt{m}\log n)$ large enough so that $\mu_k \leq m + \Delta$. Using these approximations with Lemma 4.5 gives:

$$\begin{aligned} m - |J_k(> \tau_k)| &= m + \Delta - |J_k(> \tau_k)| - \Delta \\ &\geq m + \Delta - \mu_k - 2\Delta \\ &\geq 2^{-(k-k'+1)}(m + \Delta - \mu_{k'}) - 2\Delta \\ &\geq 2^{-(k-k'+1)}(m - |J_k(> \tau_k)|) - 2\Delta. \end{aligned}$$

Finally, by Proposition 4.2, $m - |J_k(> \tau_k)| \geq 1$, so rearranging gives:

$$3\Delta(m - |J_k(> \tau_k)|) \geq m - |J_k(> \tau_k)| + 2\Delta \geq 2^{-(k-k'+1)}(m + |J_k(> \tau_k)|).$$

□

To summarize, we showed that up to a multiplicative $\widetilde{O}(\sqrt{m})$-factor, the number of unclogged machines with respect to threshold τ_k at worst halves in each batch up to k.

4.4 Bounding small-in-the-past jobs Recall that our goal is to bound $\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\le \tau_k))}{m-|J_{k-1}(>\tau_k)|}$. To this end, consider fixed k. Because $I_k \subset J_k^* = \cup_{k' \le k} I_k^*$ (by Theorem 2.1), we can write:

$$\frac{\mathrm{Vol}(I_k(\le \tau_k))}{m-|J_{k-1}(>\tau_k)|} = \sum_{k' \le k} \frac{\mathrm{Vol}(I_k \cap I_{k'}^*(\le \tau_{k'}))}{m-|J_{k-1}(>\tau_k)|} + \sum_{k' \le k} \frac{\mathrm{Vol}(I_k \cap I_{k'}^*(>\tau_{k'}, \le \tau_k))}{m-|J_{k-1}(>\tau_k)|}.$$

Thus, we split I_k depending on which batch opt decided to schedule that job in. Further, we split $I_k \cap I_{k'}^*$ (i.e. the jobs our algorithm does in batch k that opt did in the past batch $k' \le k$) into the jobs that are small-in-the-past (size at most $\tau_{k'}$) and big-in-the-past (size greater than $\tau_{k'}$ and at most τ_k.)

The goal of this section is to bound the small-in-the-past jobs. This formalizes the idea that the rate at which we lose machines, guaranteed by Lemma 4.8, is offset by the number of jobs opt is delaying, captured by the exponentially decreasing weights $n/2^k$. More precisely, if opt decides to do a small job from I_k in an earlier batch, say $I_{k'}^*$, then opt is averaging this small volume over at most a $\widetilde{O}(\sqrt{m}) \cdot 2^{k-k'}$-factor more unclogged machines. However, the weight of this term in opt's weighted free time increased by a $2^{k-k'}$-factor as well, corresponding to the number of jobs delayed by batch k'. Thus, up to a $\widetilde{O}(\sqrt{m})$-factor, there is no benefit to doing the small-in-the-past jobs any earlier. We show the following.

LEMMA 4.9. *Suppose Event (4.3) holds. Then the small-in-the-past jobs satisfy:*

$$\sum_k n/2^k \cdot \sum_{k' \le k} \frac{\mathrm{Vol}(I_k \cap I_{k'}^*(\le \tau_{k'}))}{m-|J_{k-1}(>\tau_k)|} = \widetilde{O}(\sqrt{m}) \cdot \sum_k n/2^k \cdot \tau_k.$$

Proof. Because there are $O(\log n)$ batches, it suffices to show for fixed k and $k' \le k$ that we have

$$\frac{\mathrm{Vol}(I_k \cap I_{k'}^*(\le \tau_{k'}))}{m-|J_{k-1}(>\tau_k)|} = O(\Delta) \cdot 2^{k-k'} \tau_{k'},$$

where $\Delta = O(\sqrt{m} \log n)$. Summing over all k and $k' \le k$ would give the desired result.

We upper bound the numerator using $I_k \cap I_{k'}^* \subset I_{k'}^*$ and apply Lemma 4.8 to the denominator. This gives:

$$\frac{\mathrm{Vol}(I_k \cap I_{k'}^*(\le \tau_{k'}))}{m-|J_{k-1}(>\tau_k)|} = O(\Delta) \cdot 2^{k-k'} \frac{\mathrm{Vol}(I_{k'}^*(\le \tau_{k'}))}{m-|J_{k'-1}(>\tau_k)|} = O(\Delta) \cdot 2^{k-k'} \frac{\mathrm{Vol}(I_{k'}^*(\le \tau_{k'}))}{m-|J_{k'-1}^*(>\tau_{k'})|}.$$

In the final step, we used $J_{k'-1} \subset J_{k'-1}^*$ (by Theorem 2.1) and $\tau_k \ge \tau_{k'}$.

Finally, we show

$$\frac{\mathrm{Vol}(I_{k'}^*(\le \tau_{k'}))}{m-|J_{k'-1}^*(>\tau_{k'})|} = O(\tau_{k'}).$$

Recall that $\tau_{k'} > F^*(n - n/2^{k'})$, so opt schedules at most one $\tau_{k'}$-big job per machine in $J_{k'}^*$. Further, opt schedules $I_{k'}^*(\le \tau_{k'})$ only on the $m - |J_{k'-1}^*(>\tau_{k'})|$ machines that have not yet scheduled a $\tau_{k'}$-big job yet. By averaging, after scheduling $I_{k'}^*(\le \tau_{k'})$, every such machine in opt has load at least

$$\frac{\mathrm{Vol}(I_{k'}^*(\le \tau_{k'}))}{m-|J_{k'-1}^*(>\tau_{k'})|} - \tau_{k'}.$$

One of these machines must achieve the free time $F^*(n - n/2^{k'})$, because every other machine has already scheduled a $\tau_{k'}$-big job. This implies

$$F^*(n - n/2^{k'}) \ge \frac{\mathrm{Vol}(I_{k'}^*(\le \tau_{k'}))}{m-|J_{k'-1}^*(>\tau_{k'})|} - \tau_{k'}.$$

Rearranging and using $\tau_{k'} > F^*(n - n/2^{k'})$ give the desired result. ∎

Thus, the contribution of the small-in-the-past jobs to alg's weighted free time is comparable to opt's weighted free time, up to a $\widetilde{O}(\sqrt{m})$-factor.

4.5 Bounding big-in-the-past jobs The goal of this section is to bound the big-in-the-past jobs, that is:

$$\sum_k n/2^k \cdot \sum_{k' \le k} \frac{\mathrm{Vol}(I_k \cap I^*_{k'}(> \tau_{k'}, \le \tau_k))}{m - |J_{k-1}(> \tau_k)|}.$$

For convenience, we define $I_{kk'} = I_k \cap I^*_{k'}(> \tau_{k'}, \le \tau_k)$. Note that we cannot apply volume arguments as in §4.4, because the big-in-the-past jobs are $\tau_{k'}$-big. Instead, we will use the fact that opt schedules at most one $I_{kk'}$-job per machine.

There are two types of jobs in $j \in I_{kk'}$: We say j is *blocked* if opt later schedules a τ_k-big job in J_{k-1} on the same machine as j (recall that $J_{k-1} \subset J^*_{k-1}$ by Theorem 2.1.) Otherwise, j is *unblocked*. Further, a machine is blocked/unblocked if the $I_{kk'}$-job scheduled on that machine is blocked/unblocked. Thus we can partition $I_{kk'} = B_{kk'} \cup U_{kk'}$ into blocked and unblocked jobs, respectively.

By splitting the volume of jobs into unblocked and blocked, we can rewrite:

$$\sum_k n/2^k \cdot \sum_{k' \le k} \frac{\mathrm{Vol}(I_k \cap I^*_{k'}(> \tau_{k'}, \le \tau_k))}{m - |J_{k-1}(> \tau_k)|} = \sum_k n/2^k \cdot \sum_{k' \le k} \frac{\mathrm{Vol}(U_{kk'})}{m - |J_{k-1}(> \tau_k)|} + \sum_k n/2^k \cdot \frac{\mathrm{Vol}(B_{kk'})}{m - |J_{k-1}(> \tau_k)|}.$$

Intuitively, the unblocked jobs are not problematic because there can be at most $m - |J_{k-1}(> \tau_k)|$ such jobs.

LEMMA 4.10. *The unblocked jobs satisfy* $\sum_k n/2^k \cdot \sum_{k' \le k} \frac{\mathrm{Vol}(U_{kk'})}{m - |J_{k-1}(> \tau_k)|} \le O(\log n) \cdot \sum_k n/2^k \cdot \tau_k$.

Proof. Because there are $O(\log n)$ batches, it suffices to show for fixed k and $k' \le k$ that

$$\frac{\mathrm{Vol}(U_{kk'})}{m - |J_{k-1}(> \tau_k)|} \le \tau_k.$$

We recall that every job in $U_{kk'}$ is τ_k-small, so:

$$\frac{\mathrm{Vol}(U_{kk'})}{m - |J_{k-1}(> \tau_k)|} \le \tau_k \cdot \frac{|U_{kk'}|}{m - |J_{k-1}(> \tau_k)|}.$$

We note that every job in $U_{kk'}$ is $\tau_{k'}$-big, and opt schedules these jobs in batch $I^*_{k'}$. Thus, there is at most one $U_{kk'}$-job per unblocked machine. Further, there are at most $m - |J_{k-1}(> \tau_k)|$ unblocked machines, because each τ_k-big job in J_{k-1} must be scheduled on a separate machine of opt (because $J_{k-1} \subset J^*_k$ by Theorem 2.1.) We conclude, $\frac{|U_{kk'}|}{m - |J^*_{k-1}(> \tau_k)|} \le 1$, as required. ∎

It remains to handle the blocked jobs. Again, the central issue is that opt does blocked jobs in an earlier batch before some machines get clogged. On the other hand, ChooseJobs puts these jobs in a later batch when we have fewer machines.

Unlike our previous arguments, for the blocked jobs we will charge the volume of these jobs to the completion time of opt directly. Because these jobs are blocked, opt must schedule a τ_k-big job later on the same machine. In particular, opt must have kept scheduling Bernoulli jobs with size parameter at least τ_k until one comes up heads. We will charge $B_{kk'}$ to the completion time of all of these coin flips.

As before, we consider a fixed threshold τ and later union bound over all relevant thresholds. In this section, for any batch k and threshold τ, we define $p_{k\tau} \in [0, 1]$ to be the largest probability parameter across all jobs j in J_{k-1} with $s_j > \tau$ (if no such job exists, then we follow the convention $p_{k\tau} = 0$.) Note that $p_{k\tau}$ is deterministic for fixed τ. We first relate the number of remaining machines with the expected number of heads in the kth batch.

PROPOSITION 4.4. *Consider any batches k, $k' \le k$ and threshold $\tau \ge 2\mathbb{E}F^*(n - n/2^k)$. Suppose Event (4.3) holds. Then we have $m - |J_{k-1}(> \tau)| \ge p_{k\tau} \cdot n/2^k - O(\Delta)$, where $\Delta = O(\sqrt{m} \log n)$.*

Proof. First, if $p_{k\tau} = 0$, then $|J_{k-1}(> \tau)| = 0$, so the proposition is trivial. Thus, we may assume $p_{k\tau} > 0$. In particular, ChooseJobs included at least one job $j \in J_{k-1}$ with $s_j > \tau$ and $p_j = p_{k\tau}$. It follows, ChooseJobs will include $n/2^k$ further jobs in I_k with size parameter larger than τ_k and probability parameter at least $p_{k\tau}$. Thus, we have $\mathbb{E}|I_k(> \tau)| \ge p_{k\tau} \cdot n/2^k$. Rewriting $\mathbb{E}|I_k(> \tau)| = \mathbb{E}|J_k(> \tau)| - \mathbb{E}|J_{k-1}(> \tau)|$ and applying Lemma 4.6 and Event (4.3) to the first and second expectations, respectively gives:

$$p_{k\tau} \cdot n/2^k \le \mathbb{E}|I_k(> \tau)| = \mathbb{E}|J_k(> \tau)| - \mathbb{E}|J_{k-1}(> \tau)| \le (m + O(\sqrt{m})) - (|J_{k-1}(> \tau)| - O(\Delta)).$$

Rearranging gives the desired result. ∎

To see the utility of Proposition 4.4, we assume for a moment that τ_k is deterministic and ignore the additive $O(\Delta)$ term in the proposition. Then we could rewrite $n/2^k \cdot \frac{\mathrm{Vol}(B_{kk'})}{m-|J_{k-1}(>\tau_k)|} \lesssim \frac{1}{p_{k\tau}} \cdot \mathrm{Vol}(B_{kk'})$.

To relate this expression with opt, we note that opt schedules a τ_k-big job on top of each job in $B_{kk'}$. In particular, opt must schedule enough Bernoulli jobs j with $s_j > \tau_k$ until at one comes up heads on each such machine. Each such job also satisfies $p_j \le p_{k\tau_k}$, so - roughly - in opt we expect each blocked job to delay at least $\frac{1}{p_{k\tau}}$ jobs in order for that machine to become blocked. This would give $\frac{1}{p_{k\tau_k}} \cdot \mathrm{Vol}(B_{kk'}) \lesssim \mathsf{opt}$, as required.

4.6 Coin Game It remains to formalize this idea using a martingale argument. We begin by defining an (artificial) game, which will model the process of a machine becoming blocked.

DEFINITION 4.1. (COIN GAME) *The game is played with n coins and m machines by a single player. The coins are independent such that coin j comes up heads with probability p_j. Initially, all machines are* available. *At each turn, the player can either choose to flip a previously unflipped coin on an available machine or to end the game. In the former case, if the coin comes up heads, then the machine becomes unavailable. The game ends when the player chooses to, or if we run out of unflipped coins or available machines.*

Now we are ready to interpret opt as implicitly playing a coin game to block machines.

DEFINITION 4.2. (INDUCED COIN GAME) *Consider pairs of batches $k' \le k$ and thresholds $\tau' \le \tau$. Then the (k', k, τ', τ)-induced coin game (with respect to policy opt) is a distribution over coin games defined as follows:*

- *The machines are the ones of opt whose final job in $J^*_{k'}$ has size exactly τ'.*
- *For every job in $j \in J_{k-1} \setminus J^*_{k'}$ with $s_j > \tau$, we have a coin with the same probability parameter.*

The player of the coin game simulates opt as follows. Starting from after opt schedules $J^*_{k'}$, if opt subsequently schedules a job on a machine that is still available (in the coin game), then the player flips the corresponding coin (if such a coin exists) on the same machine. The player decides to stop when it runs of out coins or all machines are unavailable.

One should imagine that the machines in the induced coin game are exactly those that can become blocked. Thus, a machine becoming unavailable in the coin game corresponds to it becoming blocked in opt, and the total number of flipped coins records how many jobs were delayed by τ'.

Using a martingale argument, we relate the number of machines that become unavailable with the number of flipped coins. The next lemma formalizes the idea that to block a machine, we expect opt to flip $\frac{1}{p_{k\tau_k}}$ coins per blocked machine. Recall that for any batch k and threshold τ, we define $p_{k\tau}$ to be the largest probability parameter across all jobs j in J_{k-1} with $s_j > \tau$.

LEMMA 4.11. *With probability $1 - \frac{1}{\mathrm{poly}(n)}$, the following event holds:*

$$\{\#(\textit{unavailable machines}) \le p_{k\tau} \cdot \#(\textit{flipped coins}) + \Delta \quad \forall\ (k', k, \tau', \tau)\textit{- induced coin games}\}, \tag{4.4}$$

where $\Delta = O(\sqrt{m} \log n)$.

Proof. Because there are $O(\log n)$ batches and $L = O(\log n)$ relevant thresholds, by union-bounding over all pairs of batches and thresholds, it suffices to show that a fixed (k', k, τ', τ)-induced coin game satisfies:

$$\mathbb{P}(\#(\text{unavailable machines}) \le p_{k\tau} \cdot \#(\text{flipped coins}) + \Delta) = 1 - \frac{1}{\mathrm{poly}(n)}.$$

We will define a martingale to count the number of unavailable machines. For all $t \ge 0$, let A_t be the (adaptively chosen) set of the first t coins flipped by the player. If the player stops before flipping t coins, then we define $A_t = A_{t-1}$. Now consider the sequence of random variables $M_t = \sum_{j \in A_t} C_j - \sum_{j \in A_t} p_j$ for all $t \ge 0$, where $C_j \sim Ber(p_j)$ is the distribution of coin j. Note that $\sum_{j \in A_t} C_j$ is exactly the number of heads in the first t coin flips, which is the number of unavaiable machines.

We claim that M_t is a martingale. Consider any $t \ge 0$. There are two cases. If $A_t = A_{t-1}$, then $M_t = M_{t-1}$, so trivially $\mathbb{E}[M_t \mid M_{t-1}, \dots, M_0] = M_{t-1}$. Otherwise, $A_t = A_{t-1} \cup \{j\}$ for some adaptively chosen coin j. It suffices to show the martingale property conditioned on the next coin being j for any fixed coin j:

$$\mathbb{E}[M_t \mid M_{t-1}, \dots, M_0, A_t = A_{t-1} \cup \{j\}] = \mathbb{E}[M_{t-1} + C_j - p_j \mid M_{t-1}, \dots, M_0, A_t = A_{t-1} \cup \{j\}]$$

$$= M_{t-1} + p_j - p_j = M_{t-1},$$

as required.

To bound the deviation of M_t, we apply Freedman's inequality [Fre75] to the martingale difference sequence of M_t.

PROPOSITION 4.5. (FREEDMAN'S INEQUALITY) *Consider a real-valued martingale sequence $\{X_t\}_{t\geq 0}$ such that $X_0 = 0$ and $|X_t| \leq M$ almost surely for all t. Let $Y_t = \sum_{s=0}^{t} \mathbb{E}[X_s^2 \mid X_{s-1}, \ldots, X_0]$ denote the quadratic variation process of $\{X_t\}_t$. Then for any $\ell \geq 0, \sigma^2 > 0$ and stopping time τ, we have:*

$$\mathbb{P}(|\sum_{t=0}^{\tau} X_t| \geq \ell \text{ and } Y_\tau \leq \sigma^2) \leq 2 \cdot \exp\left(-\frac{\ell^2/2}{\sigma^2 + M\ell/3}\right).$$

We let X_t denote the martingale difference sequence of M_t, which is defined as $X_0 = 0$ and $X_t = M_t - M_{t-1}$ for all $t > 0$. Because M_t is a martingale, X_t is as well. Furthermore, we have $|X_t| \leq 1$ almost surely for all t. For any $t \geq 0$, we let j_t be the (adaptively chosen) tth coin flip. Then we can bound the quadratic variation process by:

$$\begin{aligned} Y_t = \sum_{s=0}^{t} \mathbb{E}[X_s^2 \mid X_{s-1}, \ldots, X_0] &= \sum_{s=0}^{t} \mathbb{E}[(C_{j_s} - p_{j_s})^2 \mid X_{s-1}, \ldots, X_0] \\ &\leq \sum_{s=0}^{t} \mathbb{E}[C_{j_s}^2 \mid X_{s-1}, \ldots, X_0] \\ &= \sum_{s=0}^{t} \mathbb{E}[C_{j_s} \mid X_{s-1}, \ldots, X_0]. \end{aligned}$$

Note that the $C_{j_1} + \ldots C_{j_t} \leq m$ almost surely, because the induced coin game has at most m machines, and any adaptive policy can flip at most one heads per machine. Thus, we have $Y_t \leq m$ for all t.

Now let T be the stopping time when the induced coin game ends, so T is exactly the number of flipped coins. Then Freedman's inequality gives:

$$\mathbb{P}(|\sum_{t=0}^{T} X_t| \geq \Delta) = \mathbb{P}(|\sum_{t=0}^{T} X_t| \geq \Delta \text{ and } Y_T \leq m) \leq 2 \cdot \exp(-\frac{\Delta^2/2}{m + \Delta/3}).$$

Taking $\Delta = O(\sqrt{m} \log n)$ gives $\mathbb{P}(|\sum_{t=0}^{T} X_t| \geq \Delta) \leq \frac{1}{\text{poly}(n)}$.

Finally, we observe that #(unavailable machines) $= \sum_{j \in A_T} C_j$. Further, we have $p_{k\tau} \cdot$ #(flipped coins) $\geq \sum_{j \in A_T} p_j$, because every coin j corresponds to a job in J_{k-1} with $s_j > \tau$, so $p_j \leq p_{k\tau}$ for all coins. Thus, we conclude:

$$\mathbb{P}(\#(\text{unavailable machines}) > p_{k\tau} \cdot \#(\text{flipped coins}) + \Delta) \leq \mathbb{P}(|\sum_{t=0}^{T} X_t| > \Delta) \leq \frac{1}{\text{poly}(n)}.$$

□

Combining Proposition 4.4 and Lemma 4.11, we can bound the blocked jobs:

LEMMA 4.12. *Suppose Events (4.3) and (4.4) hold. Then the blocked jobs satisfy:*

$$\sum_k n/2^k \cdot \sum_{k' \leq k} \frac{\text{Vol}(B_{kk'})}{m - |J_{k-1}(> \tau_k)|} = \widetilde{O}(\sqrt{m})(\sum_k n/2^k \cdot \tau_k + \mathit{opt}).$$

Proof. Because there are $O(\log n)$ batches k, it suffices to show for fixed $k, k' \leq k$ that $n/2^k \cdot \frac{\text{Vol}(B_{kk'})}{m - |J_{k-1}(>\tau_k)|} = O(\Delta \log n)(n/2^k \cdot \tau_k + \mathsf{opt})$ for $\Delta = O(\sqrt{m} \log n)$. We consider two cases.

First, on the event that $p_{k\tau_k} = 0$, we have $m - |J_{k-1}(> \tau_k)| = m$. Recall that every job in $B_{kk'}$ is $\tau_{k'}$-big and in $I^*_{k'}$, so there is at most one such job per machine in opt. Then we can bound:

$$n/2^k \cdot \frac{\mathrm{Vol}(B_{kk'})}{m - |J_{k-1}(> \tau_k)|} \le n/2^k \cdot \tau_k \frac{m}{m} = n/2^k \cdot \tau_k.$$

Otherwise, we have $p_{k\tau_k} > 0$. Here we related the blocked jobs to the induced coin games:

$$\begin{aligned}
\mathrm{Vol}(B_{kk'}) &= \sum_{\tau' \le \tau_k} \tau' \cdot |\{j \in B_{kk'} \mid X_j = \tau'\}| \\
&= \sum_{\tau' \le \tau_k} \tau' \cdot \#(\text{unavailable machines in } (k', k, \tau', \tau_k)\text{-induced coin game}) \\
&\le \sum_{\tau' \le \tau_k} \tau' \cdot (p_{k\tau_k} \cdot \#(\text{flipped coins in } (k', k, \tau', \tau_k)\text{-induced coin game}) + \Delta) \\
&\le O(\log n) \cdot p_{k\tau_k} \cdot \mathsf{opt} + O(\Delta \log n) \cdot \tau_k.
\end{aligned}$$

where the first inequality follows from Event (4.4). The second follows because there are $O(\log n)$ relevant thresholds $\tau' \le \tau_k$, and every flipped coin in the (k', k, τ', τ_k)-induced coin game corresponds to opt scheduling a job on a machine that already scheduled some job with size τ', so every such job has completion time at least τ'. It follows:

$$n/2^k \cdot \frac{\mathrm{Vol}(B_{kk'})}{m - |J_{k-1}(> \tau_k)|} \le n/2^k \cdot O(\log n) \frac{p_{k\tau_k}}{m - |J_{k-1}(> \tau_k)|} \cdot \mathsf{opt} + n/2^k \cdot O(\Delta \log n) \frac{\tau_k}{m - |J_{k-1}|}.$$

By Proposition 4.4, we can bound the first term by:

$$\begin{aligned}
n/2^k \cdot O(\log n) \frac{p_{k\tau_k}}{m - |J_{k-1}(> \tau_k)|} \cdot \mathsf{opt} &= O(\log n) \frac{m - |J_{k-1}(> \tau_k)| + O(\Delta)}{m - |J_{k-1}(> \tau_k)|} \cdot \mathsf{opt} \\
&= O(\Delta \log n) \cdot \mathsf{opt}.
\end{aligned}$$

We can bound the second term by:

$$n/2^k \cdot O(\Delta \log n) \frac{\tau_k}{m - |J_{k-1}(> \tau_k)|} = O(\Delta \log n) \cdot n/2^k \cdot \tau_k.$$

Combining both bounds completes the proof. ∎

We summarize our bounds for the unblocked and blocked jobs by the next lemma, which follows immediately from Lemma 4.10 and Lemma 4.12.

LEMMA 4.13. *Suppose Events (4.3) and (4.4) hold. Then the big-in-the-past jobs satisfy:*

$$\sum_k n/2^k \cdot \sum_{k' \le k} \frac{\mathrm{Vol}(I_k \cap I^*_{k'}(> \tau_{k'}, \le \tau_k))}{m - |J_{k-1}(> \tau_k)|} = \widetilde{O}(\sqrt{m}) \cdot \left(\sum_k n/2^k \cdot \tau_k + \mathsf{opt} \right).$$

4.7 Putting it all together We are ready to prove Theorem 3.2 and Theorem 4.1, which we restate here for convenience.

THEOREM 3.2. (BATCH FREE TIME MINIMIZATION) *Given Bernoulli jobs, if $m \ge 2$ and Assumption 3.1 holds, then* STOCHFREE *outputs a list schedule with expected completion time at most $\widetilde{O}(\sqrt{m}) \cdot \big(\mathbb{E}[\mathsf{opt}] + O(1)\big)$, where opt is the optimal adaptive policy.*

THEOREM 4.1. *If $m = \Omega(1)$ is sufficiently large, then the weighted free time of alg satisfies:*

$$\mathbb{E}\left[\sum_k n/2^k \cdot F(J_k) \right] = \widetilde{O}(\sqrt{m}) \cdot \left(\mathbb{E}\left[\sum_k n/2^k \cdot F^*(n - n/2^k) \right] + \mathbb{E}[\mathsf{opt}] \right) + O(1).$$

Theorem 4.1 follows from partitioning alg's weighted free time into the contribution due to small-in-the-past and big-in-the-past jobs (which we further partitioned into unblocked and blocked jobs.)

Proof of Theorem 4.1. We assume $m = \Omega(1)$ is sufficiently large. Then we complete the proof of Theorem 4.1 by combining our bounds for the small-in-the-past- and big-in-the-past jobs. We bound alg's weighted free time by Lemma 4.2:

$$\sum_k n/2^k \cdot F(J_k) = O(\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\leq \tau_k))}{m - |J_{k-1}(> \tau_k)|} + \sum_k n/2^k \cdot \tau_k).$$

We recall $\tau_k = 2 \cdot \max(\mathbb{E}F^*(n - n/2^k), F^*(n - n/2^k))$, so $\mathbb{E}\tau_k = O(\mathbb{E}F^*(n - n/2^k))$. Thus, in expectation, the second sum is at most:

$$\mathbb{E}\sum_k n/2^k \cdot \tau_k = O(\mathbb{E}\sum_k n/2^k \cdot F^*(n - n/2^k)).$$

It remains to bound the first sum, which we split into the contribution due to small-in-the-past and big-in-the-past jobs:

$$\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\leq \tau_k))}{m - |J_{k-1}(> \tau_k)|} = \sum_k n/2^k \cdot \sum_{k' \leq k} \frac{\mathrm{Vol}(I_k \cap I^*_{k'}(\leq \tau_{k'}))}{m - |J_{k-1}(> \tau_k)|} + \sum_k n/2^k \cdot \sum_{k' \leq k} \frac{\mathrm{Vol}(I_k \cap I^*_{k'}(> \tau_{k'}, \leq \tau_k))}{m - |J_{k-1}(> \tau_k)|}.$$

On Events (4.3) and (4.4), we can apply Lemma 4.9 to the first term and Lemma 4.13 to the second to obtain:

$$\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\leq \tau_k))}{m - |J_{k-1}(> \tau_k)|} = \widetilde{O}(\sqrt{m})(\sum_k n/2^k \cdot \tau_k + \mathsf{opt}).$$

Again, in expectation, this contributes $\widetilde{O}(\sqrt{m})(\mathbb{E}\sum_k n/2^k \cdot F^*(n - n/2^k) + \mathbb{E}\mathsf{opt})$ to alg's expected weighted free time.

Finally, we consider the event that Event (4.3) or Event (4.4) does not hold. Recall that by Lemma 4.7 and Lemma 4.11, this happens with probability at most $\frac{1}{\mathrm{poly}(n)}$ because $m = \Omega(1)$ is sufficiently large. Further, on this event, we can trivially upper bound $\sum_k n/2^k \cdot \frac{\mathrm{Vol}(I_k(\leq \tau_k))}{m - |J_{k-1}(> \tau_k)|} = \mathrm{poly}(n)$, because there are n jobs each with size at most $\mathrm{poly}(n)$ almost surely. Thus, the contribution of this event to the overall expectation is $O(1)$. We conclude, alg's expected weighted free time is at most $\widetilde{O}(\sqrt{m})(\mathbb{E}\sum_k n/2^k \cdot F^*(n - n/2^k) + \mathbb{E}\mathsf{opt}) + O(1)$. ∎

To complete the proof of Theorem 4.1, we relate the weighted free time to the completion time. We also use our warm-up $\tilde{O}(m)$-approximation when m is too small to apply Theorem 3.2.

Proof of Theorem 3.2. First, suppose $m = \Omega(1)$ is sufficiently large. Then by Theorem 4.1, alg's weighted free time satisfies:

$$\mathbb{E}\left[\sum_k n/2^k \cdot F(J_k)\right] = \widetilde{O}(\sqrt{m}) \cdot \left(\mathbb{E}\left[\sum_k n/2^k \cdot F^*(n - n/2^k)\right] + \mathbb{E}[\mathsf{opt}]\right) + O(1).$$

Applying Lemma 4.1 to relate weighted free time to completion time gives:

$$\mathbb{E}[\mathsf{alg}] = \widetilde{O}(1) \cdot \mathbb{E}\left[\sum_k n/2^k \cdot F(J_k)\right] + \widetilde{O}(\mathbb{E}[\mathsf{opt}]) = \widetilde{O}(\sqrt{m}) \cdot \Big(\mathbb{E}[\mathsf{opt}] + O(1)\Big).$$

This gives the desired guarantee if $m = \Omega(1)$.

Otherwise, if $m = O(1)$, Lemma 4.4 immediately gives that StochFree is $\widetilde{O}(m) = \widetilde{O}(1)$-approximate, so $\mathbb{E}[\mathsf{alg}] = \widetilde{O}(\mathbb{E}[\mathsf{opt}])$. ∎

This completes the analysis of StochFree. Since the proof had several conceptual parts, let us a give a quick summary.

Summary. Recall that our analysis began by passing from completion time to our new proxy objective: weighted free time in §4.1. As we mentioned earlier, a key benefit of working with free times rather than completion times was that we could completely control what jobs we *started* to achieve the ith free time, whereas we have far less control over the first i jobs to *finish*. This allowed us to make the contribution of each job to the weighted free time more modular: either the job contributed to the small volume in a batch, or it contributed to the clogged machines—see Equation (4.2). We then controlled the rate at which alg and opt clog up machines in §4.3. Then in §4.4–4.6 we compared the times at which alg and opt chose to do the same volume of small jobs. Since these were the only two ways in which a job affected the weighted free time, we could combine these two ideas in §4.7 to complete our analysis.

5 Conclusion

We gave an improved approximation for stochastic completion times, which does not depend on the job size variances, and has a sublinear dependence on the number of machines m. Observe that the weighted free time is a valid proxy objective for *any* job size distributions, not just Bernoulli jobs, so extending our result to general stochastic jobs requires us to solve subset selection and batch free time minimization for these settings.

Many interesting open problems remain: can we improve our approximation ratio even further? We also do not have a good grasp on the complexity of this problem: is the stochastic problem provably hard to solve/approximate? Can we use the idea of passing from completion times to free times more broadly? Can we extend the results to other scheduling objectives, such as flow/response times? In general, stochastic scheduling problems (apart from the makespan objective) are not well understood from a distribution-independent approximation perspective, and we hope that our work will lead to further interesting developments.

References

[AAK19] Arpit Agarwal, Sepehr Assadi, and Sanjeev Khanna. Stochastic submodular cover with limited adaptivity. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 323–342. SIAM, 2019.

[AHL13] Saeed Alaei, MohammadTaghi Hajiaghayi, and Vahid Liaghat. The online stochastic generalized assignment problem. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, volume 8096 of *Lecture Notes in Computer Science*, pages 11–25. Springer, 2013.

[BGK11] Anand Bhalgat, Ashish Goel, and Sanjeev Khanna. Improved approximation results for stochastic knapsack problems. In Dana Randall, editor, *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*, pages 1647–1665. SIAM, 2011.

[BJS74] John L. Bruno, Edward G. Coffman Jr., and Ravi Sethi. Scheduling independent tasks to reduce mean finishing time. *Commun. ACM*, 17(7):382–387, 1974.

[DGV04] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Approximating the stochastic knapsack problem: The benefit of adaptivity. In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 208–217. IEEE Computer Society, 2004.

[DGV05] Brian C. Dean, Michel X. Goemans, and Jan Vondrák. Adaptivity and approximation for stochastic packing problems. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 395–404. SIAM, 2005.

[DKLN20] Anindya De, Sanjeev Khanna, Huan Li, and Hesam Nikpey. An efficient PTAS for stochastic load balancing with poisson jobs. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming, ICALP 2020, July 8-11, 2020, Saarbrücken, Germany (Virtual Conference)*, volume 168 of *LIPIcs*, pages 37:1–37:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[EFMM19] Franziska Eberle, Felix A. Fischer, Jannik Matuschke, and Nicole Megow. On index policies for stochastic minsum scheduling. *Oper. Res. Lett.*, 47(3):213–218, 2019.

[ENS17] Alina Ene, Viswanath Nagarajan, and Rishi Saket. Approximation algorithms for stochastic k-tsp. In Satya V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 27:27–27:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017.

[FLX18] Hao Fu, Jian Li, and Pan Xu. A PTAS for a Class of Stochastic Dynamic Programs. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, volume 107 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 56:1–56:14, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Fre75] David A. Freedman. On tail probabilities for Martingales. *The Annals of Probability*, 3(1):100–118, 1975.
[GGM10] Ashish Goel, Sudipto Guha, and Kamesh Munagala. How to probe for an extreme value. *ACM Trans. Algorithms*, 7(1):12:1–12:20, 2010.
[GGN21] Rohan Ghuge, Anupam Gupta, and Viswanath Nagarajan. The power of adaptivity for stochastic submodular cover. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 3702–3712. PMLR, 18–24 Jul 2021.
[GI99] Ashish Goel and Piotr Indyk. Stochastic load balancing and related problems. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 579–586. IEEE Computer Society, 1999.
[GKMR11] Anupam Gupta, Ravishankar Krishnaswamy, Marco Molinaro, and R. Ravi. Approximation algorithms for correlated knapsacks and non-martingale bandits. In Rafail Ostrovsky, editor, *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 827–836. IEEE Computer Society, 2011.
[GKNR12] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for stochastic orienteering. In Yuval Rabani, editor, *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1522–1538. SIAM, 2012.
[GKNS18] Anupam Gupta, Amit Kumar, Viswanath Nagarajan, and Xiangkun Shen. Stochastic load balancing on unrelated machines. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 1274–1285. SIAM, 2018.
[GMS10] Sudipto Guha, Kamesh Munagala, and Peng Shi. Approximation algorithms for restless bandit problems. *J. ACM*, 58(1):3:1–3:50, 2010.
[GMUX20] Varun Gupta, Benjamin Moseley, Marc Uetz, and Qiaomin Xie. Greed works - online algorithms for unrelated machine stochastic scheduling. *Math. Oper. Res.*, 45(2):497–516, 2020.
[GNS16] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Algorithms and adaptivity gaps for stochastic probing. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1731–1747. SIAM, 2016.
[GNS17] Anupam Gupta, Viswanath Nagarajan, and Sahil Singla. Adaptivity gaps for stochastic probing: Submodular and XOS functions. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1688–1702. SIAM, 2017.
[HSSW97] Leslie A. Hall, Andreas S. Schulz, David B. Shmoys, and Joel Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Math. Oper. Res.*, 22(3):513–544, 1997.
[IMP15] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. Stochastic scheduling of heavy-tailed jobs. In Ernst W. Mayr and Nicolas Ollinger, editors, *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, volume 30 of *LIPIcs*, pages 474–486. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2015.
[JLLS20] Haotian Jiang, Jian Li, Daogao Liu, and Sahil Singla. Algorithms and adaptivity gaps for stochastic k-tsp. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 45:1–45:25. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.
[JS18] Sven Jäger and Martin Skutella. Generalizing the Kawaguchi-Kyan bound to stochastic parallel machine scheduling. In Rolf Niedermeier and Brigitte Vallée, editors, *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, volume 96 of *LIPIcs*, pages 43:1–43:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018.
[KRT00] Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217, 2000.
[LY13] Jian Li and Wen Yuan. Stochastic combinatorial optimization via poisson approximation. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 971–980. ACM, 2013.
[Ma14] Will Ma. Improvements and generalizations of stochastic knapsack and multi-armed bandit approximation algorithms: Extended abstract. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1154–1163. SIAM, 2014.
[MSU99] Rolf H. Möhring, Andreas S. Schulz, and Marc Uetz. Approximation in stochastic scheduling: the power of LP-based priority policies. *J. ACM*, 46(6):924–942, 1999.
[MUV06] Nicole Megow, Marc Uetz, and Tjark Vredeveld. Models and algorithms for stochastic online scheduling. *Math. Oper. Res.*, 31(3):513–525, 2006.
[MV14] Nicole Megow and Tjark Vredeveld. A tight 2-approximation for preemptive stochastic scheduling. *Math. Oper. Res.*, 39(4):1297–1310, 2014.
[Pin08] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition, 2008.

[PST04] Kirk Pruhs, Jirí Sgall, and Eric Torng. Online scheduling. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.

[Rot66] Michael H. Rothkopf. Scheduling with random service times. *Management Science*, 12(9):707–713, 1966.

[Sch08] Andreas S. Schulz. Stochastic online scheduling revisited. In Boting Yang, Ding-Zhu Du, and Cao An Wang, editors, *Combinatorial Optimization and Applications, Second International Conference, COCOA 2008, St. John's, NL, Canada, August 21-24, 2008. Proceedings*, volume 5165 of *Lecture Notes in Computer Science*, pages 448–457. Springer, 2008.

[SSU16] Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Math. Oper. Res.*, 41(3):851–864, 2016.

[SU01] Martin Skutella and Marc Uetz. Scheduling precedence-constrained jobs with stochastic processing times on parallel machines. In S. Rao Kosaraju, editor, *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 589–590. ACM/SIAM, 2001.

[SU05] Martin Skutella and Marc Uetz. Stochastic machine scheduling with precedence constraints. *SIAM J. Comput.*, 34(4):788–802, 2005.

[WP80] Gideon Weiss and Michael Pinedo. Scheduling tasks with exponential service times on non-identical processors to minimize various cost functions. *Journal of Applied Probability*, 17(1):187–202, 1980.

[WVW86] R. R. Weber, P. Varaiya, and J. Walrand. Scheduling jobs with stochastically ordered processing times on parallel machines to minimize expected flowtime. *Journal of Applied Probability*, 23(3):841–847, 1986.

A Sensitivity of number of machines

For a fixed collection of jobs, and any number of machines m, we let $\mathsf{opt}(m)$ be the optimal completion time for these jobs on m machines.

LEMMA A.1. *For any number of machines m sufficiently large, there exists a collection of* identical *Bernoulli jobs with $\mathbb{E}\mathsf{opt}(\frac{m}{2}) = e^{\Omega(m)} \cdot \mathbb{E}\mathsf{opt}(m)$.*

Proof. We fix a number of machines m. Define $L = e^{cm}$ for a constant $c > 0$. Then consider the collection of $\frac{7}{8}mL$ Bernoulli jobs distributed as $\mathrm{Ber}(\frac{1}{L})$. Note that because jobs are identically distributed, we may assume opt list schedules jobs in arbitrary order.

We first claim that $\mathbb{E}\mathsf{opt}(m) = O(m)$. To see this, let $H \sim \mathrm{Binom}(\frac{7}{8}mL, \frac{1}{L})$ be the number of jobs that come up heads. On the event $H \le m$, each machine schedules some number of jobs with realized size zero and then at most one job with realized size 1. Thus, on this event we have $\mathsf{opt}(m) \le H$. Further, by Chernoff (Proposition D.1), we have:

$$\mathbb{P}(H > m) \le \mathbb{P}(H \ge \mathbb{E}[H] + \frac{m}{8}) = e^{-\Theta(m)}.$$

We conclude, the contribution of the event $H \le m$ to $\mathbb{E}\mathsf{opt}(m)$ is at most $\mathbb{E}H = \frac{7}{8}m$, and the contribution of the event $H < m$ is at most $\mathrm{poly}(mL) \cdot \mathbb{P}(H > m) = \mathrm{poly}(me^{cm}) \cdot e^{-\Theta(m)} = O(1)$ for c sufficiently small. This gives $\mathbb{E}\mathsf{opt}(m) \le \frac{7}{8}m + O(1) = O(m)$.

On the other hand, we have $\mathbb{E}\mathsf{opt}(\frac{m}{2}) = \Omega(mL)$. Let $H' \sim \mathrm{Binom}(\frac{3}{4}mL, \frac{1}{L})$ be the number of heads among the first $\frac{3}{4}mL$ jobs. Analogously by Chernoff we have $\mathbb{P}(H' < \frac{m}{2}) \le e^{-\Theta(m)}$. Thus, on the event $H' \ge \frac{m}{2}$ (which happens with probability $(1 - o(1))$, after scheduling the first $\frac{3}{4}mL$ jobs, each of the $\frac{m}{2}$ machines has a job of realized size 1. It follows, the remaining $\Omega(mL)$ unscheduled jobs all have completion times at least 1. Thus, we can lower bound $\mathbb{E}\mathsf{opt}(\frac{m}{2}) = \Omega(mL) \cdot (1 - o(1))$. Taking m sufficiently large gives the desired gap. ∎

LEMMA A.2. *For any collection of deterministic jobs and any $m \ge 2$, we have $\mathsf{opt}(\frac{m}{2}) \le 3 \cdot \mathsf{opt}(m)$.*

Proof. Consider the schedule achieving $\mathsf{opt}(m)$, and let C_j^m be the completion time of job j in this schedule. We construct a schedule on $\frac{m}{2}$ machines with completion time at most $3 \cdot \mathsf{opt}(m)$. Our algorithm is to list schedule the jobs on $\frac{m}{2}$ machines in increasing order of C_j^m.

Let C_j be the completion time of job j in this schedule. We claim that $C_j \le 3C_j^m$ for all jobs j, which gives the desired result. Assume for contradiction that this is not the case, so let j be the first job with $C_j > 3C_j^m$. It must be the case that up until time $2C_j^m$, all $\frac{m}{2}$ machines are busy running jobs j' with $C_{j'}^m \le C_j^m$. The total size of such j' jobs is strictly larger than $\frac{m}{2} \cdot 2C_j^m = m \cdot C_j^m$. However, $\mathsf{opt}(m)$ must complete all such j' jobs by time C_j^m. This is a contradiction. ∎

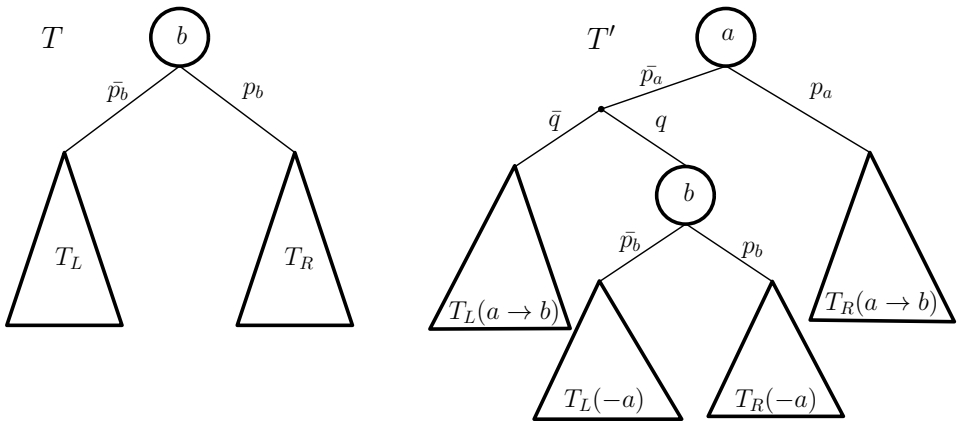

Figure 2: Original and modified decision trees

B Exchange Argument

LEMMA 1.1. *Consider a collection of Bernoulli jobs. Then for each possible size parameter, the optimal adaptive completion time schedule for these jobs starts the jobs with this size parameter in increasing order of their probabilities for all realizations of the job sizes.*

Proof. We suppose there exist jobs a, b such that $X_a \sim s \cdot Ber(p_a)$ and $X_b \sim s \cdot Ber(p_b)$ with $p_a \leq p_b$ such that the optimal completion time policy schedules b before a in some realization. Consider the decision tree corresponding to this policy (described in §2.) Thus, we assume this tree schedules b before a is some realization (i.e. some root-leaf path.) It follows, there exists a subtree rooted at b such that a is scheduled on each root-leaf path of this subtree. We denote this subtree by T. Entering this subtree, the machines have some fixed initial loads and T schedules a fixed set of jobs J.

We will modify the subtree T so that we start a before b on each root-leaf path. Further, this will not increase the expected completion time of the overall schedule. We construct the modified subtree T' as follows. Let the left- and right subtrees (corresponding to the root job b coming up size 0 or s) of T be T_L and T_R, respectively. T' is rooted at job a. In T', the right subtree of a is T_R, but with job a replaced by job b. We denote this modifed subtree by $T_R(a \to b)$. On the left subtree of a, first, independently of all jobs, we flip a coin that comes up heads with probability q. We will choose q later. If the coin is tails, then we schedule subtree T_L with job a replaced by job b, so $T_L(a \to b)$. Otherwise, if the coin is heads, then we schedule b. The left- and right subtrees of b are T_L and T_R, except the job a is replaced by a dummy job that is always zero. In particular, this job does not contribute to the completion time, but upon reaching this node we will always follows the left subtree. We denote these subtrees by $T_L(-a), T_R(-a)$, respectively. This completes the description of T'. See Figure 2 for the modified tree T'.

Note that T' schedules the same jobs as T and always starts a before b. It remains to choose q such that the expected completion time of T' is the same as T. We choose q such that the probability of entering $T_R(a \to b)$ or $T_R(-a)$ is exactly p_b. We conflate the name of a subtree (e.g. $T_R(a \to b)$) with the event that we enter the subtree. Thus, we want $\mathbb{P}(T_R(a \to b) \vee T_R(-a)) = p_b$. The former probability is exactly $p_a + \bar{p}_a q p_b$, where we define $\bar{p} = 1 - p$ for a probability p. This gives $q = \frac{p_b - p_a}{\bar{p}_a p_b}$. Thus, we have chosen q such that $\mathbb{P}(T_R(a \to b) \vee T_R(-a)) = p_b$ and $\mathbb{P}(T_L(a \to b) \vee T_L(-a)) = \bar{p}_b$. One should imagine that these two events are our replacements for the original tree T entering T_R and T_L.

In both subtrees $T_L(a \to b)$ and $T_L(-a)$, we replace the original job a from T_L with b and a zero job, respectively. Let $\tilde{X}_a$ denote the size of the the replacement job, which is supported on $\{0, s\}$. We compute the distribution of $\tilde{X}_a$:

$$\mathbb{P}(\tilde{X}_a = s \mid T_L(a \to b) \vee T_L(-a)) = \frac{\mathbb{P}(T_L(a \to b))}{\mathbb{P}(T_L(a \to b) \vee T_L(-a))} p_b = \frac{\bar{p}_a \bar{q}}{\bar{p}_b} p_b = p_a.$$

It follows, conditioned on $T_L(a \to b) \vee T_L(-a)$, our replacement job for a has the same distribution as a. An

analogous computation for the right subtree gives:

$$\mathbb{P}(\tilde{X}_a = s \mid T_R(a \to b) \vee T_R(-a)) = \frac{\mathbb{P}(T_R(a \to b))}{\mathbb{P}(T_R(a \to b) \vee T_L(-a))} p_b = \frac{p_a}{p_b} p_b = p_a,$$

so the distribution of our replacement job conditioned on $T_R(a \to b) \vee T_R(-a)$ has the same distribution as a as well.

To summarize, we have constructed a tree T' that starts a before b. T' enters $T_L(a \to b)$ or $T_L(-a)$ with probability $\bar{p}_b$: exactly the same as the probability that T enters T_L. Further, T' enters $T_L(a \to b)$ or $T_L(-a)$ with the same initial loads as T entering T_L, because both correspond to all previous jobs in the subtree having size 0. Finally, upon entering $T_L(a \to b)$ or $T_L(-a)$, the job we replace a with has the same distribution as a. The analogous properties hold for the right subtree as well. We conclude, for any job $j \in J \setminus \{a, b\}$, the expected completion time of j in T' is the same as in T (subject to the same initial loads.)

It remains to show that the expected completion time of a and b weakly decreases from T to T'. We define $\ell' \sim T_L$ to be the load of the least-loaded machine upon reaching the node a in subtree T_L (we define $\ell' \sim T_R$ analogously.) This is well-defined, because a is scheduled on every root-leaf path in T_L. Note that ℓ' does not depend on the job scheduled at node a. It follows, the expected completion time of a and b in T' are:

$$\mathbb{E}_{T'} C_a = \ell + sp_a$$

$$\mathbb{E}_{T'} C_b = \mathbb{P}(T_L(a \to b))(\mathbb{E}_{\ell' \sim T_L} \ell' + sp_b) + \mathbb{P}(T_L(-a))\ell + \mathbb{P}(T_R(-a))(\ell + s) + \mathbb{P}(T_R(a \to b))(\mathbb{E}_{\ell' \sim T_R} \ell' + sp_b).$$

Now we simplify the completion time of b. First, we consider the terms corresponding to the left subtree. We have $\mathbb{P}(T_L(a \to b)) = \bar{p}_b \frac{p_a}{p_b}$, $\mathbb{P}(T_L(a \to b)) + \mathbb{P}(T_L(-a)) = \bar{p}_b$, and $\ell' \geq \ell$ for $\ell' \sim T_L$. Combining these three observations:

$$\begin{aligned}\mathbb{P}(T_L(a \to b))(\mathbb{E}_{\ell' \sim T_L} \ell' + sp_b) + \mathbb{P}(T_L(-a))\ell &= \mathbb{P}(T_L(a \to b))\mathbb{E}_{\ell' \sim T_L} \ell' + \bar{p}_b s p_a + \mathbb{P}(T_L(-a))\ell \\ &\leq \bar{p}_b(\mathbb{E}_{\ell' \sim T_L} \ell' + sp_a).\end{aligned}$$

Now we consider the right subtree. Analogously, we have $\mathbb{P}(T_R(a \to b)) = p_a$, $\mathbb{P}(T_R(a \to b)) + \mathbb{P}(T_R(-a)) = p_b$, and $\ell' \geq \ell$ for $\ell' \sim T_R$. We compute:

$$\begin{aligned}\mathbb{P}(T_R(-a))(\ell + s) + \mathbb{P}(T_R(a \to b))(\mathbb{E}_{\ell' \sim T_R} \ell' + sp_b) &= (p_b - p_a)(\ell + s) + p_a \mathbb{E}_{\ell' \sim T_R} \ell' + sp_b + p_a s p_b \\ &\leq p_b(\mathbb{E}_{\ell' \sim T_R} \ell' + p_a s) + (p_b - p_a)s.\end{aligned}$$

Combining our expressions for the left- and right-subtrees gives our final bound on the completion time of a and b:

$$\mathbb{E}_{T'} C_a + \mathbb{E}_{T'} C_b \leq \ell + sp_b + \bar{p}_b(\mathbb{E}_{\ell' \sim T_L} \ell' + sp_a) + p_b(\mathbb{E}_{\ell' \sim T_R} \ell' + p_a s) = \mathbb{E}_T C_b + \mathbb{E}_T C_a.$$

□

C Justification for Assumption 3.1

LEMMA 3.2. *Let $m \geq 2$. Suppose there exists an algorithm for completion time minimization for Bernoulli jobs on m machines satisfying Assumption 3.1 that outputs a list schedule with expected completion time at most $\alpha(\mathbb{E}\mathsf{opt} + O(1))$. Then there exists a $O(\alpha)$-approximate algorithm for the same problem without the assumption. Further, the resulting algorithm is also a list schedule, and it preserves efficiency and determinism.*

Proof. Let $\mathcal{A}$ be the algorithm assumed by the lemma. We will run $\mathcal{A}$ on a subinstance of jobs satisfying Assumption 3.1. Suppose we have a collection J of Bernoulli jobs of the form $X_j \sim s_j \cdot Ber(p_j)$ for arbitrary size parameters s_j.

First, we round up all size parameters to the nearest power of 2. This at most doubles opt. Then, we rescale all s_j's uniformly so that $\sum_j \mathbb{E}X_j = 1$. Note that now we have $\mathbb{E}\mathsf{opt} \geq \sum_j \mathbb{E}X_j = \Omega(1)$. Finally, we partition $J = S \cup M \cup L$ into small, medium, and large jobs, respectively such that S consists of the jobs j with $s_j < \frac{1}{n^2}$, M the jobs j with $\frac{1}{n^2} \leq s_j < n^8$, and L the jobs j with $s_j \geq n^8$. Thus, M is a collection of Bernoulli jobs satisfying Assumption 3.1.

Our algorithm to schedule J is the following:

i. List-schedule all large jobs L in arbitrary order.
ii. List-schedule all small jobs S in arbitrary order.
iii. Run $\mathcal{A}$ to schedule the medium jobs M.

It is clear that this algorithm is efficient, deterministic, and outputs a list schedule as long as $\mathcal{A}$ does as well. It remains to bound the total completion time of this schedule, which we denote by alg. We let B be the event that some large job comes up heads (i.e. has realized size at least n^8.)

On the event $\bar{B}$, every large job comes up tails, so they contribute 0 to alg. Then we list-schedule the small jobs with initial load 0 on every machine. The total completion time of all jobs in S can be crudely upper-bounded by the max load after S times the number of jobs, which is at most $\frac{1}{n} \cdot n = O(\mathbb{E}\mathsf{opt})$.

After this, we schedule the medium jobs using $\mathcal{A}$. After scheduling S, all machines are free by time $\frac{1}{n}$. Let $\mathcal{A}$ be the total completion time of running $\mathcal{A}$ on jobs M starting at time 0. We need the following monotonicity property of list schedules, which is analogous to Lemma 4.3.

LEMMA C.1. *Consider a set of deterministic jobs and a fixed list schedule of those jobs. Then increasing the initial load or decreasing the number of machines weakly increase the total completion time of the schedule.*

Proof. Let J be the set of jobs. Consider initial load vectors $\ell, \ell' \in \mathbb{R}^m$, where the ith entry of each vector denotes the initial load on machine i. Now suppose $\ell \le \ell'$, entry-wise. It suffices to show that $C(J, \ell) \le C(J, \ell')$, where $C(J, \ell)$ is the total completion time achieved by our list-schedule with initial load ℓ. This suffices, because we can decrease the number of machines by making the initial loads of some machines arbitrarily large so that they will never be used.

We prove $C(J, \ell) \le C(J, \ell')$ by induction on the number of jobs, $|J|$. In the base case, $|J| = 0$, so the claim is trivial because $C(J, \ell) = 0$ and $C(J, \ell') = 0$. For $|J| > 0$, let j be the first job in the list, which is scheduled, without loss of generality, on the first machine for both initial loads ℓ and ℓ'. Then:

$$C(J, \ell) = (\ell_1 + s_j) + C(J \setminus \{j\}, \ell + s_j e_1) \le (\ell'_1 + s_j) + C(J \setminus \{j\}, \ell' + s_j e_1) = C(J, \ell'),$$

where e_1 is the first standard basis vector, so we have $\ell + s_j e_1 \le \ell' + s_j e_i 1$ entry-wise. Then we assumed inductively that $C(J \setminus \{j\}, \ell + s_j e_1) \le C(J \setminus \{j\}, \ell' + s_j e_1)$. ∎

By the above lemma, we can upper-bound the total completion time of $\mathcal{A}$ on jobs M by starting once all machines are free after scheduling S, so at time $\frac{1}{n}$. This increases the completion time of each job by $\frac{1}{n}$. To summarize, on the event that every large job comes up tails, we have:

$$\mathbb{E}\mathsf{alg} \cdot 1_{\bar{B}} \le \frac{1}{n} \cdot n + \frac{1}{n} \cdot n + \mathbb{E}\mathcal{A} = O(\mathbb{E}\mathsf{opt}) + \alpha(\mathbb{E}\mathsf{opt} + O(1)) = O(\alpha) \cdot \mathbb{E}\mathsf{opt},$$

where we used the guarantee of $\mathcal{A}$ and $\mathbb{E}\mathsf{opt} = \Omega(1)$.

It remains to consider the event where some large job comes up heads. In this case, we will not use the guarantee of $\mathcal{A}$. Instead, we will upper bound alg by the cost of an arbitrary list schedule. We define $S_1 = \max_{j \in J} X_j$ and S_2 to be the size of the second-largest job in J. On the event $B \cap \{S_2 \le \frac{1}{n^2} S_1\}$, we note that no job is scheduled after the largest job with size S_1 on the same machine (using $m \ge 2$.) Noting all other jobs have size at most $\frac{1}{n^2} S_1$, we can upper bound alg by:

$$\mathsf{alg} \le S_1 + n \cdot \frac{1}{n} S_1 \le 2S_1 \le 2\mathsf{opt},$$

so we have $\mathbb{E}\mathsf{alg} \cdot 1_{B, S_2 \le \frac{1}{n^2} S_1} = O(\mathbb{E}\mathsf{opt})$.

Finally, we bound $\mathbb{E}\mathsf{alg} \cdot 1_{B, S_2 > \frac{1}{n^2} S_1}$. We partition $B = \cup_{k=0}^{\infty} B_k$, where $B_k = \{\max_{j \in J} X_j \in [2^k n^8, 2^{k+1} n^8)\}$. On the event $B_k \cap \{S_2 > \frac{1}{n^2} S_1\}$, there are at least two jobs of size at least $2^k n^6$. Recall that $\sum_{j \in J} \mathbb{E} X_j = 1$, so in particular $\mathbb{E} X_j \le 1$ for all $j \in J$. Thus by Markov's inequality, $\mathbb{P}(X_j \ge 2^k n^6) \le 2^{-k} n^{-6}$ for all $j \in J$. By union-bounding over all pairs of jobs in J:

$$\mathbb{P}(B_k,\ S_2 > \frac{1}{n^2} S_1) \le \mathbb{P}(\exists \text{ two jobs in } J \text{ with size at least } 2^k n^6) \le O(n^2)(2^{-k} n^{-6})^2.$$

Further, on the event $B_k \cap \{1_{B,S_2>\frac{1}{n^2}S_1}\}$, every job has size at most $2^{k+1}n^8$, so we have $\mathsf{alg} \le n \cdot n 2^{k+1} n^8 = 2^{k+1}n^{10}$. Thus, for each k, we have:

$$\begin{aligned}\mathbb{E}\mathsf{alg} \cdot 1_{B_k, S_2 > \frac{1}{n^2}S_1} &\le 2^{k+1}n^{10} \cdot \mathbb{P}(B_k,\, S_2 > \frac{1}{n^2}S_1) \\ &= 2^{k+1}n^{10} \cdot O(n^2)(2^{-k}n^{-6})^2 = O(2^{-k}).\end{aligned}$$

To complete the proof, we partition $B = \cup_{k=0}^{\infty} B_k$ to bound $\mathbb{E}\mathsf{alg} \cdot 1_{B,S_2>\frac{1}{n^2}S_1}$:

$$\mathbb{E}\mathsf{alg} \cdot 1_{B,S_2>\frac{1}{n^2}S_1} = \sum_{k=0}^{\infty} \mathbb{E}\mathsf{alg} \cdot 1_{B_k,S_2>\frac{1}{n^2}S_1} = O(\sum_{k=0}^{\infty} 2^{-k}) = O(\mathbb{E}\mathsf{opt}).$$

∎

D Concentration arguments

We need the following standard Chernoff bound.

PROPOSITION D.1. (CHERNOFF BOUND) *Let $X = X_1 + \ldots X_n$ be a sum of independent, $\{0,1\}$-valued random variables and $\mu = \mathbb{E}X$. Then we have:*

- $\mathbb{P}(X \le (1-\delta)\mu) \le \exp\big(-\frac{\delta^2\mu}{2}\big)$ *for all* $0 \le \delta \le 1$.
- $\mathbb{P}(X \ge (1+\delta)\mu) \le \exp\big(-\frac{\delta^2\mu}{2+\delta}\big)$ *for all* $0 \le \delta$.

LEMMA 4.6. *Let $m = \Omega(1)$ be sufficiently large. Then there exists a constant $c \ge 0$ such that for all batches k and thresholds $\tau > 2\mathbb{E}F^*(n - n/2^k)$, we have $\mathbb{E}|J_k(>\tau)| \le m + c\sqrt{m}$.*

Proof. Fix $c \ge 0$ which we will choose sufficiently large later. Then assume for contradiction that there exists a batch k and threshold $\tau > 2\mathbb{E}F^*(n - n/2^k)$ such that $\mathbb{E}|J_k(>\tau)| > m + c\sqrt{m}$.

To reach a contradiction, it suffices to show that $\mathbb{P}(|J_k(>\tau)| \le m) < \frac{1}{2}$. This is because on the complement event $|J_k(>\tau)| > m$ (which we assume happens with probability strictly larger than $\frac{1}{2}$), we also have $|J_k^*(>\tau)| > m$ by Theorem 2.1. This implies $F^*(n - n/2^k) > \tau \ge 2\mathbb{E}F^*(n - n/2^k)$. This would contradict the definition of $\mathbb{E}F^*(n - n/2^k)$.

For convenience, let $\mu = \mathbb{E}|J_k(>\tau)|$. By Chernoff, we have:

$$\mathbb{P}(|J_k(>\tau)| \le m) = \mathbb{P}(|J_k(>\tau)| \le \mu(1 - \frac{\mu - m}{\mu})) \le \exp\big(-\frac{(\mu-m)^2}{2\mu}\big).$$

There are two cases to consider. Recall that by assumption, we have $\mu > m + c\sqrt{m}$. If $\mu \ge 2m$, then $\mathbb{P}(|J_k(>\tau)| \le m) \le \exp(-\frac{\mu}{8}) \le \exp(-\frac{m}{4}) < \frac{1}{2}$ for $m = \Omega(1)$ sufficiently large. Otherwise, $m + c\sqrt{m} < \mu < 2m$. Then $\mathbb{P}(|J_k(>\tau)| \le m) \le \exp(-\frac{c^2 m}{2m}) < \frac{1}{2}$ for $c = O(1)$ sufficiently large. ∎

LEMMA 4.7. *Let $\Delta = O(\sqrt{m}\log n)$ and $m = \Omega(1)$ be sufficiently large. Then with probability at least $1 - \frac{1}{\mathrm{poly}(n)}$, the following events hold:*

$$\{|J_k(>\tau)| \overset{\pm\Delta}{\approx} \mathbb{E}|J_k(>\tau)| \quad \forall \text{ batches } k \text{ and thresholds } \tau > 2\mathbb{E}F^*(n - n/2^k)\}. \tag{4.3}$$

Proof. Note that there are $O(\log n)$ choices for k and $L = O(\log n)$ relevant choices for τ. Thus, by a standard union bound argument it suffices to show that for fixed k and $\tau > 2\mathbb{E}F^*(n - n/2^k)$, we have:

$$\mathbb{P}(||J_k(>\tau)| - \mathbb{E}|J_k(>\tau)|| > \Delta) = \frac{1}{\mathrm{poly}(n)}.$$

Now we may assume m is large enough so that $\mathbb{E}|J_k(>\tau)| \le m + c\sqrt{m} \le (c+1)m$ for sufficiently large constant $c \ge 0$ (guaranteed by Lemma 4.6.) Then we can bound the deviation of $|J_k(>\tau)|$ again with a Chernoff bound. Let $\mu = \mathbb{E}|J_k(>\tau)|$. We take $\Delta = O(\sqrt{\mu}\log n) = O(\sqrt{m}\log n)$.

There are two cases to consider. If $\mu < \Delta$, then the lower tail is trivial:

$$\mathbb{P}(|J_k(> \tau)| \leq \mu - \Delta) \leq \mathbb{P}(|J_k(> \tau)| < 0) = 0.$$

For the upper tail we use Chernoff:

$$\mathbb{P}(|J_k(> \tau)| \geq \mu + \Delta) = \mathbb{P}(|J_k(> \tau)| \geq (1 + \frac{\Delta}{\mu})\mu) \leq \exp\left(-\frac{\Delta^2}{2\mu + \Delta}\right) \leq \exp\left(-\frac{\Delta^2}{3\Delta}\right) = \frac{1}{\text{poly}(n)}.$$

Otherwise, $\mu \geq \Delta$, so in particular $\frac{\Delta}{\mu} \leq 1$. Then we use Chernoff for both the lower- and upper tails:

$$\mathbb{P}(|J_k(> \tau)| \leq \mu + \Delta) = \mathbb{P}(|J_k(> \tau)| \leq (1 + \frac{\Delta}{\mu})\mu) \leq \exp\left(-\frac{\Delta^2}{2\mu}\right) = \frac{1}{\text{poly}(n)}.$$

$$\mathbb{P}(|J_k(> \tau)| \geq \mu + \Delta) = \mathbb{P}(|J_k(> \tau)| \geq (1 + \frac{\Delta}{\mu})\mu) \leq \exp\left(-\frac{\Delta^2}{2\mu + \Delta}\right) \leq \exp\left(-\frac{\Delta^2}{3\mu}\right) = \frac{1}{\text{poly}(n)}.$$

□