Non-clairvoyant Scheduling with Predictions

SUNGJIN IM, University of California, Merced, USA
RAVI KUMAR, Google Research, USA
MAHSHID MONTAZER QAEM, University of California, Merced, USA
MANISH PUROHIT, Google Research, USA

In the single-machine *non-clairvoyant* scheduling problem, the goal is to minimize the total completion time of jobs whose processing times are *unknown a priori*. We revisit this well-studied problem and consider the question of how to effectively use (possibly erroneous) predictions of the processing times. We study this question from ground zero by first asking what constitutes a good prediction; we then propose a new measure to gauge prediction quality and design scheduling algorithms with strong guarantees under this measure. Our approach to derive a prediction error measure based on natural desiderata could find applications for other online problems.

CCS Concepts: • **Theory of computation** → **Approximation algorithms analysis**; **Online algorithms;**

Additional Key Words and Phrases: Scheduling, non-clairvoyance, competitive ratio, prediction

ACM Reference format:
Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. 2023. Non-clairvoyant Scheduling with Predictions. *ACM Trans. Parallel Comput.* 10, 4, Article 19 (December 2023), 26 pages.
https://doi.org/10.1145/3593969

1 INTRODUCTION

Non-clairvoyance, where the scheduler is not aware of the exact processing times of a job *a priori*, is a highly desired property in the design of scheduling algorithms. Due to its myriad practical applications, non-clairvoyant scheduling has been extensively studied in various settings in the scheduling literature [15, 18, 31]. With no access to the processing times (i.e., job sizes), non-clairvoyant algorithms inherently suffer from worse performance guarantees than the corresponding clairvoyant algorithms. For example, in the most basic version of scheduling, we have a set of jobs that need to be scheduled on a single machine with the goal of minimizing the total completion time of all jobs. In the non-clairvoyant setting, the job sizes are unknown to the algorithm and only become known *after* the job has completed; here, the *Round-Robin* algorithm that divides the machine equally among all incomplete jobs is 2-competitive [29] and this is known to be the best possible. In contrast, in the clairvoyant setting where job sizes are known *a priori*, the ***Shortest***

Sungjin Im and Mahshid Montazer Qaem were supported in part by NSF grants CCF-1617653, CCF-1844939, and CCF-2121745.
Authors' addresses: S. Im and M. Montazer Qaem, University of California, 5200 North Lake Rd, Merced, CA 95343, USA; emails: sim3@ucmerced.edu, mmontazerqaem@ucmerced.edu; R. Kumar and M. Purohit, Google Research, 1600 Amphitheatre Pkwy, Mountain View, CA 94043, USA; emails: ravi.k53@gmail.com, mpurohit@google.com.

2329-4949/2023/12-ART19
https://doi.org/10.1145/3593969

Job First **(SJF)** algorithm that schedules jobs in non-decreasing order of their sizes is known to be optimal.

Practitioners often face scheduling problems that lie somewhere in between clairvoyant and non-clairvoyant settings. While it might be impossible to know the exact job sizes, rather than assuming non-clairvoyance, it is possible to *estimate* job sizes based on their features using a predictor [2, 26, 30]. However, such an estimation can be error-prone. Can one use the possibly erroneous predicted job sizes to improve the performance of scheduling algorithms?

Augmenting traditional algorithms with machine-learned predictions is a fascinating and newly emerging line of work. In particular, this paradigm is applicable to online algorithms, which typically focus on obtaining worst-case guarantees against uncertain future inputs and thus settle for pessimistic bounds. Recent works have shown that, using predictions (that may be incorrect), one can provably improve the guarantees of traditional online algorithms for caching [17, 25, 32], ski-rental [3, 13, 20], scheduling [8, 20, 28], load balancing [21, 23], secretary problem [5], metrical task systems [4], set cover [9], flow and matching [22], knapsack [16, 34], and many others.

In this article, we continue the study of learning-augmented algorithms for single-machine non-clairvoyant scheduling. This problem, where an algorithm has access to predictions of each job size, was first investigated in Reference [20]. Without making any assumptions on the prediction quality, they design a non-clairvoyant algorithm that satisfies two important properties, namely, consistency and robustness. *Consistency* means that the guarantees of the algorithm improve with good predictions; in particular, the algorithm obtains a competitive ratio better than 2 if the predictions are good. *Robustness* ensures that the algorithm gracefully handles bad predictions, i.e., even if the predictions are adversarially bad, the competitive ratio stays bounded. For any $\lambda \in (0, 1)$, they design an algorithm that guarantees robustness of $\frac{2}{1-\lambda}$ and consistency of $\frac{1}{\lambda}$.[1]

1.1 The Need for a New Error Measure

Although Reference [20] demonstrates an appealing tradeoff between consistency and robustness for non-clairvoyant scheduling, a closer look reveals some brittleness of the result. Here, we discuss the issue at a high level and delve in more detail in the next section when we formally define the problem and the old and new error notions.

The main issue stems from the total completion time objective. Since this objective measures the total waiting time of all jobs, a shorter job could delay more jobs. In fact, different jobs can have different effects on how much they delay other jobs. The objective is thus *neither linear nor quadratic* in the job sizes.[2]

In Reference [20], it is assumed that the algorithm has a prediction $\hat{p}_j$ of each job size p_j. The quality of the prediction is the sum of the prediction errors of individual jobs, i.e., $\ell_1(p, \hat{p}) = \sum_j |\hat{p}_j - p_j|$. Intuitively, such a linear error measure is incompatible with the completion time objective and may not distinguish good predictions vs. poor predictions; in fact, small perturbations in the predictions can result in large changes to the optimal solution. Consequently, the results in Reference [20] are forced to be pessimistic and have a high dependence on the error term. In particular, they show that scheduling the jobs in non-decreasing order of their predicted sizes (SPJF) leads to a cost of at most $\textsc{opt} + (n-1) \cdot \ell_1(p, \hat{p})$ and this is tight, where $\textsc{opt}$ is the cost of the optimum solution and n the number of jobs.

We examine the $\ell_1(\cdot, \cdot)$ error measure and show that it violates a natural and desirable Lipschitz-like property for the total completion time objective. This prompts the search for a different error

[1]Here, *α-robustness* and *β-consistency* mean that the algorithm's cost is at most α times the optimum for all inputs but improves to at most β factor when the prediction coincides with the actual input. See Definition 2.

[2]For a concrete example, consider n jobs that have unit sizes with sufficiently small perturbations. The derivative of the objective is n with respect to the length of the smallest job; yet it is 1 with respect to that of the largest job.

measure based on two desiderata (see Section 2.2). Our new error measure better captures the sensitive nature of the objective and allows us to obtain an algorithm with total cost at most $(1+\epsilon)\text{OPT} + O_\epsilon(1) \cdot \nu(p, \hat{p})$ where $\nu(\cdot, \cdot)$ is the measure we propose.

In practice, job sizes are predicted using black-box machine learned models that utilize various features of the jobs (e.g., history) and may be expensive to train. While it is impossible to precisely define the goodness of a prediction, intuitively, an effective error measure should neither tag bad predictions as good nor ignore predictions that could improve the objective.

1.2 Our Contributions

Under the new notion of error (denoted $\nu(\cdot, \cdot)$), we give the following results, stated informally below. For a job j, let p_j be its actual size and let $\hat{p}_j$ be its predicted size. We assume all jobs are available for scheduling from time 0. Let OPT be the optimum solution.

(1) We obtain a non-clairvoyant algorithm that is $O(1)$-robust (with no dependency on ϵ) and $(1+\epsilon)$-consistent for any sufficiently small $\epsilon > 0$ w.h.p., if no subset of $O(\frac{1}{\epsilon^3} \log n)$ jobs dominates the objective. (Theorem 3 and Corollary 1)

(2) We obtain a non-clairvoyant algorithm that is $O(\frac{1}{\epsilon})$-robust and $(1+\epsilon)$-consistent in expectation for any sufficiently small $\epsilon > 0$. More precisely, the cost of the algorithm is at most $(1+\epsilon)\text{OPT} + O(\frac{1}{\epsilon^3} \log \frac{1}{\epsilon})\nu(p, \hat{p})$. (Theorem 4)

In contrast, Reference [20] obtains an algorithm that is $O(\frac{1}{\epsilon})$-robust and whose cost is at most $(1+\epsilon)\text{OPT} + (1+\epsilon)(n-1) \cdot \ell_1(p, \hat{p})$. Since our error measure satisfies $\ell_1(p, \hat{p}) \le \nu(p, \hat{p}) \le n \cdot \ell_1(p, \hat{p})$, our algorithm never has an asymptotically worse dependence on the prediction quality for any fixed $\epsilon > 0$ and is often sharper.

(3) We show that for any sufficiently small $\epsilon, \gamma > 0$, no algorithm can have a smaller objective than $(1+\epsilon)\text{OPT} + O(1/\epsilon^{1-\gamma})\nu(p, \hat{p})$. (Theorem 6)

We now discuss the high-level ideas. The main challenge is how to determine if a prediction is reliable or not *before* completing all jobs. If the predictions are somewhat reliable, then we can more or less follow them; otherwise, we will essentially have to rely on non-clairvoyant algorithms such as Round-Robin. Therefore, we repeatedly take a small sample of jobs over the course of the algorithm and partially process them. Informally, we estimate the median remaining size of jobs and estimate the prediction error considering job sizes up to the estimated median. Unfortunately, this estimation is not free, since we have to partially process the sampled jobs and it can delay all the existing jobs. Therefore, we are forced to stop sampling once there are very few jobs left. Depending on how long we sample, we obtain the first and second results.

Due to the dynamic nature of our algorithm, the analysis turns out to be considerably non-trivial. In a nutshell, we never see the true error until we finish a job. Nevertheless, we still have to decide whether to follow the predictions. The mismatch between partial errors we perceive and the actual errors makes it challenging to charge our algorithm's cost to the optimum and the error; special care is needed throughout the analysis to avoid overcharging. We note that unlike our algorithm, Reference [20] uses a static algorithm that linearly combines following the predictions and Round-Robin.

To summarize, our work demonstrates that it is possible to find quality solutions for a bigger class of predictions by using a more refined measure and it could lead to discovering new algorithmic ideas.

1.3 Other Related Work

Designing learning-augmented algorithms falls into the new beyond-worst-case algorithm design paradigm [33]. Starting with the work of Kraska et al. [19] on using ML predictions to speed up indexing, there have been many efforts to leverage ML predictions to better handle common instances that are found in practice. In addition to the aforementioned works, there also exist works on frequency counting [1, 11, 14], membership testing [27, 35], and online learning [10]. A recent work [12] shows how to speed up bipartite matching algorithms using ML predictions.

For single-machine scheduling in the clairvoyant setting, **Shortest Remaining Processing Time (SRPT)** is known to be optimal for minimizing the total completion time; it is in fact also optimal for minimizing the total flow/response time.[3] If all jobs arrive at time 0, then SJF coincides with SRPT. In the non-clairvoyant setting, when jobs have different arrival times, no algorithm is $O(1)$-competitive for minimizing the total flow time, but Round-Robin is known to be $O(1)$-competitive when compared to the optimum schedule running on a machine with speed less than $1/2 - \epsilon$, for any $\epsilon > 0$. For a survey on online scheduling algorithms, see Reference [31]. Very recently, References [6, 7] obtained algorithms with $O(1)$-competitive ratio for total flow time if every job's size is within a constant factor of its prediction.

1.4 Roadmap

In Section 2, we formally define our non-clairvoyant scheduling problem. In the same section, we continue to discuss what desiderata constitute a good measure of prediction error and propose a new measure meeting the desiderata. We also discuss other—both existing and candidate—measures and show that they fail to satisfy the desiderata. We present our algorithm in Section 3 and its analysis in Section 4. The lower bounds are presented in Section 5.

2 FORMULATION AND BASIC PROPERTIES

2.1 Non-clairvoyant Scheduling

Let J denote a set of n jobs. In the classical single-machine *non-clairvoyant scheduling* setting, each job $j \in J$ has an unknown *size* or *processing time* p_j. The processing time is known only after the job is complete. A job j *completes* when it has received p_j time units of processing, and we denote j's *completion time* as C_j. A job may be preempted at any time and resumed at a later time without any cost. Our goal is to find a schedule that completes all jobs and minimizes the *total completion time* of all jobs, i.e., $\sum_{j \in J} C_j$. In the *clairvoyant* case, an algorithm knows the p_j's in advance.

Definition 1 (Competitive Ratio). Let $\mathcal{I}$ denote the set of all instances of the non-clairvoyant scheduling problem. Let $\textsc{cost}_{\mathcal{A}}(I)$ be the total completion time of the schedule obtained by a non-clairvoyant algorithm $\mathcal{A}$ and $\textsc{opt}(I)$ be the cost of the optimum (clairvoyant) algorithm on instance $I \in \mathcal{I}$. Algorithm $\mathcal{A}$ is said to be *c-competitive* if

$$\max_{I \in \mathcal{I}} \frac{\textsc{cost}_{\mathcal{A}}(I)}{\textsc{opt}(I)} \leq c.$$

In the clairvoyant case, it is well-known that the ***Shortest Remaining Processing Time First (SRPT)*** algorithm minimizes the total completion time and becomes identical to the ***Shortest Job First*** **(SJF)** algorithm when all jobs arrive at time 0, which is the setting we consider in this article. In the non-clairvoyant case, the *Round-Robin* algorithm achieves a competitive ratio of 2, which is known to be optimal [29].

For any subset $Z \subseteq J$ of jobs, we let $\textsc{opt}(\{x_j\}_{j \in Z})$ denote the minimum objective to complete all jobs in Z when each job $j \in Z$ has size x_j and is known to the algorithm, i.e., it is the completion time of SJF when x_j is the size of job j. Here, we can think of $\textsc{opt}$ as a function that takes as input a multiset of non-negative job sizes and returns the minimum objective to complete all jobs with

[3]In the setting where job j has a release time r_j, the *flow time* of a job is defined as $C_j - r_j$ where C_j is the *completion time* of job j in the schedule. If all jobs are available at time 0, then the flow time coincides with the completion time.

the job sizes in the set. Note that this is well-defined, as SJF is oblivious to job identities. If x_j is j's true size, i.e., p_j, for notational convenience, then we use $\textsc{opt}(Z) := \textsc{opt}(\{p_j\}_{j\in Z})$; in particular, $\textsc{opt} := \textsc{opt}(J)$.

We consider the *learning-augmented scheduling* problem where the algorithm has access to *predictions* for each job size; let $\hat{p}_j$ denote the predicted size of job j. We emphasize that we make no assumptions regarding the validity of the predictions and they may even be adversarial. As in the usual non-clairvoyant scheduling setup, the true processing size p_j of job j is revealed only *after* the job has received p_j amount of processing time. In the learning-augmented setting, the competitive ratio of an algorithm $\mathcal{A}$ is a function of the prediction error. Our goal is to design an algorithm that satisfies the dual notions of robustness and consistency.

Definition 2 (Robustness and Consistency). Let $\mathcal{I}$ be the set of all instances of the learning-augmented non-clairvoyant scheduling problem.[4] The *robustness* of an algorithm $\mathcal{A}$ is the worst-case ratio of the algorithm's cost to the cost of the optimal solution independent of the quality of the predictions. The *consistency* of an algorithm $\mathcal{A}$ is the worst-case ratio when restricted to instances where the predictions are all correct, i.e., $\hat{p}_j = p_j, \forall j \in J$.

$$\text{Robustness}(\mathcal{A}) = \max_{I\in\mathcal{I}} \frac{\textsc{cost}_{\mathcal{A}}(I)}{\textsc{opt}(I)},$$

$$\text{Consistency}(\mathcal{A}) = \max_{\substack{I\in\mathcal{I}\\ \hat{p}_j=p_j,\forall j\in J}} \frac{\textsc{cost}_{\mathcal{A}}(I)}{\textsc{opt}(I)}.$$

2.1.1 Properties of $\textsc{opt}$. The following fact is well-known and follows from the definition of $\textsc{opt}$, i.e., SJF:

Proposition 1 ([29]). $\textit{OPT}(\{x_j\}_{j\in J}) = \sum_{j\in J} x_j + \sum_{i<j\in J}\min\{x_i,x_j\} \le \sum_{(i,j)\in J\times J}\min\{x_i,x_j\}$.

The following properties are simple consequences of SJF:

Proposition 2. *Let J denote an arbitrary set of jobs and $\{x_j\}_{j\in J}$ and $\{y_j\}_{j\in J}$ be two sets of non-negative job sizes. Then,*

(i) *If $x_j \ge y_j$ for all $j\in J$, then $\textit{OPT}(\{x_j\}_{j\in J}) \ge \textit{OPT}(\{y_j\}_{j\in J})$.*
(ii) *For any subset $Z\subseteq J$, $\textit{OPT}(\{x_j\}_{j\in J}) \ge \textit{OPT}(\{x_j\}_{j\in Z})$.*
(iii) $\textit{OPT}(\{x_j+y_j\}_{j\in J}) \ge \textit{OPT}(\{x_j\}_{j\in J}) + \textit{OPT}(\{y_j\}_{j\in J})$.
(iv) *Let $X_1,\dots,X_L$ be a partition of J, i.e., $J=\bigcup_{\ell\in[L]} X_\ell$ and $X_\ell\cap X_{\ell'}=\emptyset$ for $\ell\ne\ell'$, then we have*

$$\sum_{\ell\in[L]} \textit{OPT}(\{x_j\}_{j\in X_l}) \le \textit{OPT}(\{x_j\}_{j\in J}) \le L\cdot\sum_{\ell\in[L]} \textit{OPT}(\{x_j\}_{j\in X_l}).$$

Proof. Properties (i)–(iii) follow directly from Proposition 1. We now show Property (iv). For brevity, we show the claim only when $L=2$; extending the proof to arbitrary values of L is straightforward. Let $X=X_1$ and $Y=X_2$ denote the two disjoint subsets of J.

To prove the first inequality in the claim, consider the job set $J=X\cup Y$ and the sets of job sizes $\{x'_j\}_{j\in J}$ and $\{y'_j\}_{j\in J}$ given by $x'_j=x_j$ and $y'_j=0, \forall j\in X$ and $x'_j=0$ and $y'_j=x_j, \forall j\in Y$. The inequality now follows directly from Proposition 1.

To prove the second inequality, we consider the optimal schedule of jobs in set X and the optimal schedule of jobs in set Y. For any job $j\in X\cup Y$, let C'_j denote the completion time of job j in the corresponding optimal schedule. We can construct a schedule for all jobs in $X\cup Y$ by scheduling jobs in non-decreasing order of their completion times C'_j. We observe that in the newly constructed

[4]An instance here is specified by both the predicted job sizes and the true job sizes.

schedule, the relative ordering of all jobs in X and Y is maintained. Let C_j denote the completion time of job j in the new schedule. Consider any job $j \in X$; by definition, we have $C_j = C'_j + C'_k$ where $k \in Y$ is the last job from set Y that is scheduled before j. But by definition of the constructed schedule, we have $C'_k \le C'_j$. By a similar argument for any job in Y, we obtain $C_j \le 2C'_j$ for any job $j \in X \cup Y$. Since the optimal solution for jobs in $X \cup Y$ can only yield lower total cost, the second inequality follows. □

2.2 Prediction Error

A key question in the design of algorithms with predictions is how to define the prediction error, i.e., how to quantify the quality of predictions. While this definition can be problem-dependent, it must be *algorithm*-independent. For the non-clairvoyant scheduling problem, before we dive into a definition, we identify two desirable properties that we want of any such definition. Let $\textsc{err}(\{p_j\}_{j\in J}, \{\hat{p}_j\}_{j\in J})$ denote the prediction error for an instance with true sizes $\{p_j\}$ and predicted job sizes $\{\hat{p}_j\}$; note that an algorithm knows the $\hat{p}_j$'s but not the p_j's.

The first property is monotonicity, i.e., if more job sizes predictions are correct, then the error must decrease. Monotonicity is natural as better predictions are expected to decrease the error.

Property 1 (Monotonicity). *For any $I \subseteq J$, $ERR(\{p_j\}_{j\in J}, \{\hat{p}_j\}_{j\in J\setminus I} \cup \{p_i\}_{i\in I}) \le ERR(\{p_j\}_{j\in J}, \{\hat{p}_j\}_{j\in J})$.*

The second property is a Lipschitz-like condition that states that a prediction $\{\hat{p}_j\}_{j\in J}$ is said to be good (as measured by $\textsc{err}(\cdot,\cdot)$) only if the optimal solution of the predicted instance is close to the true optimal solution. Indeed, if the optimal solution of a predicted instance differs significantly from true optimal solution, i.e., $|\textsc{opt}(\{\hat{p}_j\}_{j\in J}) - \textsc{opt}(\{p_j\}_{j\in J})|$ is large, then the property requires that a good error measure assigns a large error to such predictions. Intuitively, this property allows us to effectively distinguish between good and bad predictions.

Property 2 (Lipschitzness). $|OPT(\{\hat{p}_j\}_{j\in J}) - OPT(\{p_j\}_{j\in J})| \le ERR(\{p_j\}_{j\in J}, \{\hat{p}_j\}_{j\in J})$.

A natural way to define the prediction error is to define it as the ℓ_1-norm between the predicted and the true job sizes, i.e., $\ell_1(p,\hat{p}) = \textsc{err}(\{p_j\}_{j\in J}, \{\hat{p}_j\}_{j\in J}) = \sum_{j\in J} |p_j - \hat{p}_j|$, as was done in Reference [20]. While this error definition satisfies monotonicity, it is not Lipschitz. Indeed, consider the following simple problem instance. Let $\epsilon > 0$ be a constant. The true job sizes are given by $p_1 = 1+\epsilon$ and $p_j = 1, \forall j \in J\setminus\{1\}$. The predicted job sizes are given by $\hat{p}_1 = 1+3\epsilon$ and $\hat{p}_j = 1, \forall j \in J\setminus\{1\}$. Let $\hat{q}$ be another set of predicted job sizes given by $\hat{q}_1 = 1-\epsilon$ and $\hat{q}_j = 1, \forall j \in J\setminus\{1\}$. By construction, $\ell_1(p,\hat{p}) = 2\epsilon = \ell_1(p,\hat{q})$. However, by the nature of the total completion time objective, there is a significant difference in the quality of the predictions in these two instances. Formally, $\textsc{opt}(\{\hat{p}_j\}_{j\in J}) - \textsc{opt}(\{p_j\}_{j\in J}) = 2\epsilon$, whereas $\textsc{opt}(\{p_j\}_{j\in J}) - \textsc{opt}(\{\hat{q}_j\}_{j\in J}) = (n-1)\cdot\epsilon \gg \ell_1(p,\hat{q})$. Intuitively, the lack of Lipschitzness causes the $\ell_1(\cdot,\cdot)$ error metric to not be able to distinguish between $\{\hat{p}\}$ and $\{\hat{q}\}$ predictions, although $\{\hat{p}\}$ is arguably a much better prediction for this instance.

However, to satisfy the Lipschitz property, one can consider simply defining the prediction error as $\textsc{err}(\{p_j\}_{j\in J}, \{\hat{p}_j\}_{j\in J}) = |\textsc{opt}(\{\hat{p}_j\}_{j\in J}) - \textsc{opt}(\{p_j\}_{j\in J})|$. Unfortunately, this may not be monotone. Indeed, consider a simple instance where the predictions are a reassignment of the true job sizes to the jobs, i.e., the job sizes are predicted correctly but the job identities are permuted. In this case, we have $|\textsc{opt}(\{\hat{p}_j\}_{j\in J}) - \textsc{opt}(\{p_j\}_{j\in J})| = 0$. However, an improvement to any of the predictions will only result in a different optimum, and hence a non-zero error. In other words, this definition does not satisfy monotonicity.

These examples motivate a different notion of prediction error.

Definition 3 (Prediction Error). For any instance of the non-clairvoyant scheduling problem with predictions where each job $j \in J$ has a true size p_j and a predicted size $\hat{p}_j$, the *prediction error* is defined as:

$$\begin{aligned}\nu(J;\{p_j\},\{\hat{p}_j\}) &:= \textsc{err}(\{p_j\}_{j\in J},\{\hat{p}_j\}_{j\in J}) \\ &= \textsc{opt}\Big(\{\hat{p}_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}\Big)-\textsc{opt}\Big(\{p_j\}_{j\in J_o}\cup\{\hat{p}_j\}_{j\in J_u}\Big),\end{aligned}$$

where $J_o = \{j \in J \mid \hat{p}_j > p_j\}$, $J_u = \{j \in J \mid \hat{p}_j \le p_j\}$ denote the set of jobs whose sizes are overestimated and underestimated, respectively.

Intuitively, the above definition aims to ensure

$$\nu \ge \textsc{opt}\Big(\{\hat{p}_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}\Big)-\textsc{opt}\Big(\{p_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}\Big),$$

which comes from pretending that all underestimated job sizes were predicted correctly. Similarly, we also want

$$\nu \ge \textsc{opt}\Big(\{p_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}\Big)-\textsc{opt}\Big(\{p_j\}_{j\in J_o}\cup\{\hat{p}_j\}_{j\in J_u}\Big).$$

Our error measure follows by adding the RHS of these two inequalities.

It is easy to see that this definition, besides being symmetric and non-negative, also satisfies both monotonicity and the Lipschitz property. While this may not be the unique such definition, it is simple. Further, we are *not* aware of any other error measures, including those used in the previous work [9, 20], that satisfy the two desired properties. For more details, see Section 2.2.2.

Proposition 3. *The error measure given in Definition 3 satisfies both monotonicity and Lipschitzness.*

Proof. We first show monotonicity. As before, we use the subscript o to refer to the subset of overestimated jobs in a given set. Likewise, we use u analogously to refer to the underestimated jobs. For any $I \subseteq J$,

$$\begin{aligned}&\nu(J;\{p_j\}_{j\in J},\{\hat{p}_j\}_{j\in J\setminus I}\cup\{p_i\}_{i\in I}) \\ &= \textsc{opt}\Big(\{\hat{p}_j\}_{j\in J_o\setminus I}\cup\{p_j\}_{j\in J_u\setminus I}\cup\{p_j\}_{j\in I}\Big)-\textsc{opt}\Big(\{p_j\}_{j\in J_o\setminus I}\cup\{\hat{p}_j\}_{j\in J_u\setminus I}\cup\{p_j\}_{j\in I}\Big) \\ &= \textsc{opt}\Big(\{\hat{p}_j\}_{j\in J_o\setminus I}\cup\{p_j\}_{j\in J_u\cup I}\Big)-\textsc{opt}\Big(\{p_j\}_{j\in J_o\cup I}\cup\{\hat{p}_j\}_{j\in J_u\setminus I}\Big) \\ &\le \textsc{opt}\Big(\{\hat{p}_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}\Big)-\textsc{opt}\Big(\{p_j\}_{j\in J_o}\cup\{\hat{p}_j\}_{j\in J_u}\Big) = \nu(J;\{p_j\},\{\hat{p}_j\}). \quad \text{(Proposition 2(ii))}\end{aligned}$$

Next, we show Lipschitzness. Due to Proposition 2(i), we have

$$\textsc{opt}(\{\hat{p}_j\}_{j\in J}) \le \textsc{opt}(\{\hat{p}_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}) \text{ and } \textsc{opt}(\{p_j\}_{j\in J}) \ge \textsc{opt}(\{p_j\}_{j\in J_o}\cup\{\hat{p}_j\}_{j\in J_u}).$$

Thus, we conclude

$$\begin{aligned}|\textsc{opt}(\{\hat{p}_j\}_{j\in J})-\textsc{opt}(\{p_j\}_{j\in J})| &\le |\textsc{opt}(\{\hat{p}_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u})-\textsc{opt}(\{p_j\}_{j\in J_o}\cup\{\hat{p}_j\}_{j\in J_u})| \\ &= \nu(J;\{p_j\},\{\hat{p}_j\}).\end{aligned}$$ □

When the scheduling instance is clear from context, we drop the arguments and let $\nu = \nu(J;\{p_j\},\{\hat{p}_j\})$. Note that in case all the predicted job sizes are overestimates (or underestimates) of the true sizes, then we have $\nu(J;\{p_j\},\{\hat{p}_j\}) = |\textsc{opt}(\{\hat{p}_j\}_{j\in J})-\textsc{opt}(\{p_j\}_{j\in J})|$.

2.2.1 Surrogate Error. For the sake of analysis, we define a surrogate (prediction) error where we measure the error for overestimated and underestimated jobs separately. The surrogate error is a lower bound on the prediction error in Definition 3. While it does not satisfy Lipschitzness, nevertheless, it will turn out to be a useful tool for analysis.

Definition 4 (Surrogate Error). For any set $Z \subseteq J$ of jobs, where each job $j \in Z$ has a true size p_j and a predicted size $\hat{p}_j$, the *surrogate error* is defined as:

$$\eta(Z; \{p_j\}, \{\hat{p}_j\}) := \Big(\textsc{opt}(\{\hat{p}_j\}_{j \in Z_o}) - \textsc{opt}(\{p_j\}_{j \in Z_o})\Big) + \Big(\textsc{opt}(\{p_j\}_{j \in Z_u}) - \textsc{opt}(\{\hat{p}_j\}_{j \in Z_u})\Big),$$

where $Z_o = \{j \in Z \mid \hat{p}_j > p_j\}$, $Z_u = \{j \in Z \mid \hat{p}_j \le p_j\}$ denote the set of jobs whose sizes are overestimated and underestimated, respectively.

Again, when the scheduling instance is clear from context, we drop the arguments and let $\eta = \eta(J; \{p_j\}, \{\hat{p}_j\})$. We first show that the surrogate error can be used to lower bound the prediction error.

Proposition 4. *For any set $Z \subseteq J$ of jobs where each job $j \in Z$ has true size p_j and predicted size $\hat{p}_j$, $\nu(Z; \{p_j\}, \{\hat{p}_j\}) \ge \eta(Z; \{p_j\}, \{\hat{p}_j\})$.*

Proof. By Definition 3, we have

$$\begin{aligned}\nu(Z; \{p_j\}, \{\hat{p}_j\}) &= \textsc{opt}\Big(\{\hat{p}_j\}_{j \in Z_o} \cup \{p_j\}_{j \in Z_u}\Big) - \textsc{opt}\Big(\{p_j\}_{j \in Z_o} \cup \{\hat{p}_j\}_{j \in Z_u}\Big) \\ &= \Big(\textsc{opt}\Big(\{\hat{p}_j\}_{j \in Z_o} \cup \{p_j\}_{j \in Z_u}\Big) - \textsc{opt}\left(\{p_j\}_{j \in Z}\right)\Big) \\ &\quad + \Big(\textsc{opt}\left(\{p_j\}_{j \in Z}\right) - \textsc{opt}\Big(\{p_j\}_{j \in Z_o} \cup \{\hat{p}_j\}_{j \in Z_u}\Big)\Big).\end{aligned}$$

We compare the first term above to the first term in the definition of η. By definition of the optimum solution (Proposition 1), we have the following:

$$\begin{aligned}\textsc{opt}\Big(\{\hat{p}_j\}_{j \in Z_o} \cup \{p_j\}_{j \in Z_u}\Big) - \textsc{opt}\left(\{p_j\}_{j \in Z}\right) &= \sum_{j \in Z_o} (\hat{p}_j - p_j) + \sum_{i < j \in Z_o} (\min\{\hat{p}_i, \hat{p}_j\} - \min\{p_i, p_j\}) \\ &\quad + \sum_{i \in Z_o, j \in Z_u} (\min\{\hat{p}_i, p_j\} - \min\{p_i, p_j\}),\end{aligned}$$

since $\hat{p}_i > p_i$ for all $i \in Z_o$, we have

$$\begin{aligned}&\ge \sum_{j \in Z_o} (\hat{p}_j - p_j) + \sum_{i < j \in Z_o} (\min\{\hat{p}_i, \hat{p}_j\} - \min\{p_i, p_j\}) \\ &= \textsc{opt}(\{\hat{p}_j\}_{j \in Z_o}) - \textsc{opt}(\{p_j\}_{j \in Z_o}).\end{aligned}$$

Similarly, using analogous arguments, we have

$$\textsc{opt}\left(\{p_j\}_{j \in Z}\right) - \textsc{opt}\Big(\{p_j\}_{j \in Z_o} \cup \{\hat{p}_j\}_{j \in Z_u}\Big) \ge \textsc{opt}(\{p_j\}_{j \in Z_u}) - \textsc{opt}(\{\hat{p}_j\}_{j \in Z_u}).$$

The proposition now follows from Definition 4. □

A key advantage of the surrogate error η is that it is easier to decompose as opposed to ν. As our analysis carefully charges our algorithm's cost in each round to the error and the optimum, decomposability will be very useful to avoid overcharging.

Proposition 5 (Superadditivity of Surrogate Error). *For any set $Z \subseteq J$ of jobs, any set of true and predicted job sizes $\{(p_j, \hat{p}_j)\}_{j \in Z}$ and any partition of Z into two disjoint subsets Z_1 and Z_2, we have $\eta(Z; \{p_j\}, \{\hat{p}_j\}) \ge \eta(Z_1; \{p_j\}, \{\hat{p}_j\}) + \eta(Z_2; \{p_j\}, \{\hat{p}_j\})$.*

Proof. Let Z_u be the underestimated jobs in Z. Similarly, let Z_{1u} and Z_{2u} be the underestimated jobs in Z_1 and Z_2, respectively. Likewise, we define Z_o, Z_{1o}, and Z_{2o} for overestimated jobs. By definition of η, it suffices to show

$$\eta(Z_o; \{p_j\}, \{\hat{p}_j\}) \geq \eta(Z_{1o}; \{p_j\}, \{\hat{p}_j\}) + \eta(Z_{2o}; \{p_j\}, \{\hat{p}_j\}),$$
$$\eta(Z_u; \{p_j\}, \{\hat{p}_j\}) \geq \eta(Z_{1u}; \{p_j\}, \{\hat{p}_j\}) + \eta(Z_{2u}; \{p_j\}, \{\hat{p}_j\}).$$

We show the first inequality above; the second one follows similarly. By definition of η and Proposition 1, we have

$$\begin{aligned}\eta(Z_o; \{p_j\}, \{\hat{p}_j\}) &= \text{OPT}(\{\hat{p}_j\}_{j\in Z_o}) - \text{OPT}(\{p_j\}_{j\in Z_o})\\ &= \sum_{j\in Z_o}(\hat{p}_j - p_j) + \sum_{i<j\in Z_o}\Big(\min\{\hat{p}_i,\hat{p}_j\} - \min\{p_i,p_j\}\Big)\\ &\geq \sum_{j\in Z_o}(\hat{p}_j - p_j) + \sum_{i<j\in Z_{1o}}\Big(\min\{\hat{p}_i,\hat{p}_j\} - \min\{p_i,p_j\}\Big)\\ &\quad + \sum_{i<j\in Z_{2o}}\Big(\min\{\hat{p}_i,\hat{p}_j\} - \min\{p_i,p_j\}\Big)\\ &= \eta(Z_{1o}; \{p_j\}, \{\hat{p}_j\}) + \eta(Z_{2o}; \{p_j\}, \{\hat{p}_j\}),\end{aligned}$$

where the inequality follows, since we have $\hat{p}_j \geq p_j$ for any $j \in Z_o$. □

2.2.2 Comparisons with Other Error Measures. We compare our new error measure with others, including those in References [9, 20]. First, we observe that our error measure is always lower bounded by the $\ell_1(p,\hat{p})$ error utilized by Reference [20] but is at most a factor of n larger.

Proposition 6. *For any set $\{p_j\}_{j\in J}$ and $\{\hat{p}_j\}_{j\in J}$ of true and predicted job sizes, we have*

$$\ell_1(p,\hat{p}) \leq \nu(J; \{p_j\}, \{\hat{p}_j\}) \leq n \cdot \ell_1(p,\hat{p}).$$

Proof. From Definition 3, we have

$$\begin{aligned}\nu(J; \{p_j\}, \{\hat{p}_j\}) &= \text{OPT}\Big(\{\hat{p}_j\}_{j\in J_o} \cup \{p_j\}_{j\in J_u}\Big) - \text{OPT}\Big(\{p_j\}_{j\in J_o} \cup \{\hat{p}_j\}_{j\in J_u}\Big)\\ &= \sum_{j\in J}|p_j - \hat{p}_j| + \sum_{i<j\in J}\min\{\max\{p_i,\hat{p}_i\},\max\{p_j,\hat{p}_j\}\}\\ &\quad - \min\{\min\{p_i,\hat{p}_i\},\min\{p_j,\hat{p}_j\}\}.\end{aligned}$$

Since every term in the second summation is non-negative, the first inequality in the proposition follows. To see why the second inequality holds, we observe that

$$\min\Big\{\max\{p_i,\hat{p}_i\},\max\{p_j,\hat{p}_j\}\Big\} - \min\Big\{\min\{p_i,\hat{p}_i\},\min\{p_j,\hat{p}_j\}\Big\} \leq \max\{|p_i - \hat{p}_i|, |p_j - \hat{p}_j|\}.$$

Substituting in the above expression, we get

$$\begin{aligned}\nu(J; \{p_j\}, \{\hat{p}_j\}) &\leq \ell_1(p,\hat{p}) + \sum_{i<j\in J}\max\{|p_i - \hat{p}_i|, |p_j - \hat{p}_j|\} \leq \ell_1(p,\hat{p})\\ &\quad + \sum_{i<j\in J}|p_i - \hat{p}_i| + |p_j - \hat{p}_j| = \ell_1(p,\hat{p}) + (n-1)\sum_{j\in J}|p_j - \hat{p}_j| = n\ell_1(p,\hat{p}).\end{aligned}$$ □

Thus, our error measure lends itself to asymptotically stronger algorithmic guarantees than the $\ell_1(p,\hat{p})$ measure. In Reference [20], the cost of their algorithm is shown to be bounded by $(1+\epsilon)\text{OPT} + (1+\epsilon)\cdot(n-1)\ell_1(p,\hat{p})$. In the following sections, we obtain an algorithm whose cost is bounded by $(1+\epsilon)\text{OPT} + O_\epsilon(1) \cdot \nu$. By Proposition 6, our bound is asymptotically never worse than that in Reference [20] and can often be sharper.

Next, we discuss the error measure used by Bamas et al. [9] in their primal-dual framework. Their measure, which we call η_{BMS}, can be defined as the cost of SPJF[5] minus the optimum. It is easy to see that η_{BMS} is neither monotone nor Lipschitz. This is because SPJF yields an optimal schedule as long as jobs have the same order both in their true sizes and estimated sizes, i.e., $p_j \leq p_i$ if and only if $\hat{p}_j \leq \hat{p}_i$. Further, it is hard to compare our error measure to η_{BMS}, as the latter does not directly factor in estimated job sizes but measures the cost for running an algorithm (that is based on the prediction) on the actual input. However, the following example shows that η_{BMS} can be excessively large even if a single job size is mispredicted: The true job sizes are given by $p_j = 1\ \forall j \in J \setminus \{n\}$ and $p_n = n^2$. All job sizes are predicted correctly, except job n, where $\hat{p}_n = 0$. Then, $\nu = 1 + \cdots + n - 1 + (n + n^2) - (1 + \cdots + n - 1) = n + n^2$, while $\eta_{BMS} \geq n^2 \cdot n - (1 + \cdots + n - 1 + n + n^2) = \Omega(n^3)$. Here, $n^2 \cdot n$ comes from the fact that job n completes first under SPJF.

Finally, we note that the error measure cannot be oblivious to job identities. For example, consider the Earth Mover's distance between the true job sizes and the estimated job sizes, i.e., find a mincost matching between two multi-sets $\{p_j\}_{j \in J}$ and $\{\hat{p}_i\}_{i \in J}$ where matching p_i to $\hat{p}_j$ incurs cost $|p_i - \hat{p}_j|$. It is easy see that such measures have zero error when the two multi-sets are identical yet the predictions are incorrect for individual jobs.

Subsequent to the publication of the conference version of this article, recent work [24] proposes a novel model that considers predictions of the relative order of job sizes instead of predictions of the actual job sizes.

3 ALGORITHM

In this section, we present our algorithm for scheduling with predictions. Our algorithm runs in rounds. To formalize, we need to set up some notation. We let J_k be the set of unfinished (alive) jobs at the beginning of round k, where $k \geq 1$. Let $n_k := |J_k|$. Let $q_{k,j}$ be the amount of processing done on job j in round k. We define

- $p_{k,j} = p_j - \sum_{w=1}^{k-1} q_{w,j}$: the *true* remaining size of j at the beginning of round k.
- $\hat{p}_{k,j} = \max\{0, \hat{p}_j - \sum_{w=1}^{k-1} q_{w,j}\}$: the *predicted* remaining size of j at the beginning of round k.

Note that if a job j has been processed by more than its predicted size $\hat{p}_j$ before the kth round, then we have $\hat{p}_{k,j} = 0$.

Our algorithm employs two subprocedures in each round to estimate the median m_k of the true remaining size of jobs in J_k and the magnitude of the error in the round. We first present the subprocedures and then present our main algorithm. We assume $n \geq 2$ throughout, since otherwise all of our results follow immediately.

3.1 Median Estimation

To streamline the analysis, we will assume without loss of generality that all remaining sizes are distinct.[6] Let $\tilde{m}_k$ denote our estimate of the true median m_k. Recall that Round-Robin processes all alive jobs equally at each time.

[5]Shortest Predicted Job First (SPJF) is the algorithm that blindly follows the predictions.

[6]For instance, this can be achieved almost surely by adding small random perturbations to the initial job sizes. So, if a tiny random number $\iota_j > 0$ is added to p_j, then we pretend that j has a remaining size of ι_j as soon as it actually completes. This perturbation has negligible effects on the objective.

ALGORITHM 1: Median-Estimator(J_k, δ, n)

```
1: Let $S$ be a uniform random sample, with replacement, of size $\lceil \frac{\ln 2n}{\delta^2} \rceil$ from $J_k$.
2: Run Round-Robin on $S$ until half of the jobs in $S$ complete; let $j_k$ be the job that completed the
   last.
3: Return $\tilde{m}_k = p_{k,j_k}$.
```

Algorithm 1 takes a sample S of the remaining jobs and returns as $\tilde{m}_k$ the median of the jobs in S in terms of their remaining size. Note that this can be done by completing half of the jobs in S using Round-Robin. The sampling with replacement is done as follows: When we take job j as a sample, we pretend to create a new job in S with size equal to $p_{k,j}$. Thus, S could contain multiple "copies" originating from the same job in J_k; however, we will pretend that they are distinct jobs and they get exactly the same processor share in Round-Robin.

When the condition in the following lemma holds, we will say that $\tilde{m}_k$ is a $(1+\delta)$ *order-approximation* of m_k, or $(1+\delta)$*-approximation* for brevity.

Lemma 1. *The order of $\tilde{m}_k$ among $\{p_{k,j} \mid j \in J_k\}$ is in $((\frac{1}{2} - \delta)n_k, (\frac{1}{2} + \delta)n_k]$, with probability at least $1 - \frac{1}{n^2}$.*

Proof. Let r_i be the order of x_i in the set $\{p_{k,j} \mid j \in J_k\}$. Let Y_i be an indicator random variable that has value 1 if and only if $r_i \le (1/2 - \delta)n_k$. To show the order of $\tilde{m}_k$ is smaller than or equal $(1/2 - \delta)$ with probability at most $\frac{1}{2n^2}$, it suffices to show:

$$\Pr\left[\sum_{i \in [|S|]} \frac{Y_i}{|S|} \ge \frac{1}{2}\right] \le \frac{1}{2n^2}.$$

This is equivalent to showing

$$\Pr\left[\sum_{i \in [S]} \frac{Y_i}{|S|} - \mathbb{E}\left[\sum_{i \in [|S|]} \frac{Y_i}{|S|}\right] \ge \delta\right] \le \frac{1}{2n^2},$$

as $\mathbb{E}\sum_{i \in [|S|]} Y_i/|S| = 1/2 - \delta$. The above inequality is a direct consequence of the Hoeffding bound (Theorem 7) with $\ell := |S| \ge \frac{\ln(4n^2)}{2\delta^2}$. The other inequality can be shown symmetrically, and the proof follows from a union bound. □

3.2 Error Estimation

Next, we would like to see if the prediction for the remaining jobs in round k is accurate enough to follow closely. However, measuring the error of the predictions even by running all jobs in a small sample to completion could take too much time. Thus, we estimate the error of the remaining jobs by capping all remaining sizes and predicted sizes at $(1 + 2\epsilon)\tilde{m}_k$. The error we seek to estimate is below.

Definition 5 (Error in Round k). $\eta_k := \textsc{opt}(\{d_{k,j}\}_{j \in J_k})$, where $d_{k,j} := |\min\{(1+2\epsilon)\tilde{m}_k, p_{k,j}\} - \min\{(1+2\epsilon)\tilde{m}_k, \hat{p}_{k,j}\}|$.

Recall that by Proposition 1, the error η_k can be rewritten as $\eta_k = \sum_{i \le j \in J_k} \min\{d_{k,i}, d_{k,j}\}$. Hence, to estimate η_k, we sample pairs of jobs from J_k and for each sampled job j calculate $d_{k,j}$. Since we only need to run job j for at most $(1+2\epsilon)\tilde{m}_k$ to compute $d_{k,j}$, this step does not incur too much additional cost. Algorithm 2 describes it formally.

ALGORITHM 2: Error-Estimator($J_k, \epsilon, n, \tilde{m}_k$)

```
1: Let $P$ be a uniform random sample, with replacement, of size $\lceil \frac{32^2}{\delta^4 \epsilon^2} \log n \rceil$ from a family $Q := \{(j,j) \mid j \in J_k\} \cup \{(i,j) \mid i < j \text{ and } i,j \in J_k\}$ of unordered pairs.
2: For every sampled job $j$, calculate $d_{k,j}$ by running $j$ up to $(1+2\epsilon)\tilde{m}_k$ units.
3: Return the estimate $\tilde{\eta}_k := |Q| \frac{1}{|P|} \sum_{(i,j)\in P} \min\{d_{k,i}, d_{k,j}\}$.
```

For any $k \in [K]$, we say for brevity that $\tilde{\eta}_k$ is a $(1+\epsilon)$-*approximation* of η_k if it satisfies

$$\eta_k - \epsilon \tilde{m}_k n_k^2 \leq \tilde{\eta}_k \leq \eta_k + \epsilon \tilde{m}_k n_k^2.$$

Lemma 2. *If $n_k \geq 9$ and $\epsilon \leq 1/10$, then $\tilde{\eta}_k$ is a $(1 + \frac{\delta^2}{32}\epsilon)$-approximation of η_k with probability at least $1 - \frac{1}{n^2}$.*

Proof. Note that by Proposition 1, we have

$$\eta_k = |Q| \left(\frac{1}{|Q|} \sum_{(i,j)\in Q} \min\{d_{k,i}, d_{k,j}\} \right).$$

For each unordered pair $(i,j) \in Q$, let $\Delta(i,j) := \min\{d_{k,i}, d_{k,j}\}/((1+2\epsilon)\tilde{m}_k)$. Note that $\Delta(i,j) \in [0,1]$ and $|Q| = n_k(n_k+1)/2$ and that $\mathbb{E}[\tilde{\eta}_k] = \eta_k$. We now have the following:

$$|\tilde{\eta}_k - \eta_k| = |Q|(1+2\epsilon)\tilde{m}_k \cdot \left| \frac{1}{|P|} \sum_{(i,j)\in P} \Delta(i,j) - \mathbb{E}\left[\frac{1}{|P|} \sum_{(i,j)\in P} \Delta(i,j) \right] \right|$$

$$\Pr\left[|\tilde{\eta}_k - \eta_k| \geq \left(\frac{\delta^2}{32}\right) \epsilon \tilde{m}_k n_k^2 \right] = \Pr\left[\left| \frac{1}{|P|} \sum_{(i,j)\in P} \Delta(i,j) - \mathbb{E}\left[\frac{1}{|P|} \sum_{(i,j)\in P} \Delta(i,j) \right] \right| \geq \frac{(\delta^2/32)\epsilon n_k^2}{|Q|(1+2\epsilon)} \right].$$

For $\epsilon \leq 1/10$ and $n_k \geq 9$ (Algorithm 2 will be used later only when $n_k \geq \frac{32^2}{\delta^4}(\log n)/\epsilon^3 \geq 9$), we have $\frac{(\delta^2/32)\epsilon n_k^2}{|Q|(1+2\epsilon)} \geq 1.5(\delta^2/32)\epsilon := \xi$ and hence applying the Hoeffding bound (Theorem 7) with $\ell := |P|$, we have:

$$\Pr\left[|\tilde{\eta}_k - \eta_k| \geq \frac{\delta^2}{32} \epsilon \tilde{m}_k n_k^2 \right] \leq \Pr[|\tilde{\eta}_k - \eta_k| \geq \xi] \leq 2\exp(-2\ell\xi^2).$$

The lemma now follows by observing that $2\exp(-2\ell\xi^2) \leq 1/n^2$ for all $n \geq 2$. □

3.3 Main Algorithm

Given the methods to estimate the median size of all jobs in J_k and the remaining jobs' error in round k, we now describe our algorithm running in rounds $k \geq 1$. We note that for a fixed $\epsilon < 1/10$, the following algorithm yields $(1 + O(\epsilon))$-consistency. To obtain the desired $(1+\epsilon)$-consistency, we can later scale ϵ by the necessary constant factor to obtain Theorem 3.

If there are enough jobs alive for accurate sampling, then we use our estimators to estimate the median and the error. If the estimated error is big, then we say that the current round is an *RR round* and run Round-Robin to process all jobs equally up to $2\tilde{m}_k$ units[7]; this is intuitive, as our estimator indicates that the prediction is unreliable. If not, then we closely follow the prediction.

[7]In fact, the jobs can be processed in an arbitrary order as long as they are processed up to $2\tilde{m}_k$ units.

ALGORITHM 3: Scheduling with Predictions

```
1: $k \leftarrow 1$ and $\delta \leftarrow 1/50$.
2: while $n_k \geq \frac{32^2}{\delta^4 \epsilon^3} \log n$ do
3:     $\tilde{m}_k \leftarrow$ Median-Estimator($J_k, \delta, n$).
4:     $\tilde{\eta}_k \leftarrow$ Error-Estimator($J_k, \epsilon, n, \tilde{m}_k$).
5:     if $\tilde{\eta}_k \geq \epsilon \delta^2 \tilde{m}_k n_k^2 / 16$                ▷ RR (big error) round
6:         Process each job in $J_k$ up to $2\tilde{m}_k$ units with Round-Robin.
7:     else                                                 ▷ Non-RR (small error) round
8:         Process jobs $j$ with $\hat{p}_{k,j} \leq (1+\epsilon)\tilde{m}_k$ up to $\hat{p}_{k,j} + 3\epsilon\tilde{m}_k$ units in increasing order of $\hat{p}_{k,j}$.
9:     $k \leftarrow k + 1$.
10: Complete the remaining jobs with Round-Robin.                 ▷ Round $K+1$
```

We only consider jobs that are predicted to be small and process them in increasing order of their (remaining) predicted size. To allow for a small prediction error, we allow a job to get processed $3\epsilon\tilde{m}_k$ more units than its remaining predicted size. In this case, we say that the current round is a *non-RR round*. Finally, when there are few jobs left, we run Round-Robin to complete all remaining jobs; this is the final round, indexed by $K + 1$.

The following easy observations will be useful for our analysis later:

Observation 1. (1) *Every overestimated job remains overestimated in every round, i.e., if $\hat{p}_j \geq p_j$, then $\hat{p}_{k,j} \geq p_{k,j}$ for all k. A similar statement holds for every underestimated job.*
(2) *If a job j is processed in a non-RR round k, then its remaining predicted size is 0 for all the subsequent rounds, i.e., $\hat{p}_{k',j} = 0$ for all $k' > k$.*
(3) *For each job, there is at most one round where the job's remaining predicted size becomes 0.*

4 ANALYSIS

To streamline the presentation of our analysis, we will first make the following simplifying assumptions; we will remove these assumptions later in Section 4.4.

Assumption 1. *(i) $\tilde{m}_k$ is a $(1+\delta)$-approximation of m_k for $\delta = 1/50$, (ii) $\tilde{\eta}_k$ is a $(1 + \frac{\delta^2}{32}\epsilon)$-approximation of η_k, (iii) the estimation procedures are instantaneous and do not incur any additional delay, and (iv) the sampled jobs themselves have not been processed in the estimation processes.*

Note that Assumption 1(iv) can only hurt the algorithm. For the analysis, we extend the definition of η_k.

Definition 6 (Error in Round k on a Subset). $\eta_k(X) := \textsc{opt}(\{d_{k,j}\}_{j \in X})$ for all $X \subseteq J_k$, where $d_{k,j} := |\min\{(1+2\epsilon)\tilde{m}_k, p_{k,j}\} - \min\{(1+2\epsilon)\tilde{m}_k, \hat{p}_{k,j}\}|$.

Note that $\eta_k = \eta_k(J_k)$.

4.1 Robustness

In this section, we show that our algorithm always yields a constant approximation assuming that our median and error estimation subroutines succeed. This guarantee holds in all cases even if the predicted job sizes are arbitrarily bad or even adversarially chosen.

Theorem 1. *Algorithm 3 is an $O(1)$-approximation, under Assumption 1.*

Key to the analysis is to show that a constant fraction of jobs complete in each round.

Lemma 3. *For all $k \in [K]$, we have $n_{k+1} \leq (1/2 + 2\delta) n_k$.*

Proof. Suppose round k is an RR round. By Assumption 1(i), there are at least $(1/2-\delta)n_k$ jobs with $p_{k,j} \le \tilde{m}_k$. Since all jobs are processed up to $2\tilde{m}_k$ units in an RR round, clearly, we complete at least $(1/2-\delta)n_k$ jobs in the round, and hence $n_{k+1} \le (1/2+\delta)n_k$.

Now suppose round k is a non-RR round. We first show that there are many jobs considered by the algorithm. Let $X := \{j \in J_k \mid \hat{p}_{k,j} \le (1+\epsilon)\tilde{m}_k\}$ and $Y := \{j \in J_k \mid p_{k,j} \le \tilde{m}_k\}$. We claim,

$$|X| \ge (1/2 - 1.5\delta)n_k.$$

Suppose not. Then, since by Assumption 1, $|Y| \ge (1/2-\delta)n_k$, we have $|Y \setminus X| \ge 0.5\delta n_k$. Note that for all $j \in Y \setminus X$, $d_{k,j} \ge \epsilon\tilde{m}_k$. We have a contradiction, as we have $\eta_k \ge \eta_k(Y\setminus X) \ge \frac{1}{2}\epsilon\tilde{m}_k(0.5\delta n_k)^2$.

Note that all jobs in X are processed by the algorithm. Now, we show most of jobs in X complete in the round. Indeed, let $Z = \{j \in X \mid p_{k,j} > \hat{p}_{k,j} + 3\epsilon\tilde{m}_k\}$ denote those jobs in X that do not complete in the round k. For every job $j \in Z$, we have $d_{k,j} \ge \epsilon\tilde{m}_k$. Thus, if we have $|Z| \ge 0.5\delta n_k$, as before, then we will have $\eta_k \ge \eta_k(Z) \ge \frac{1}{2}(\epsilon\tilde{m}_k)(0.5\delta n_k)^2$, another contradiction.

Thus, we have $|X \setminus Z| \ge (1/2-2\delta)n_k$, meaning the algorithm completes at least $(1/2-2\delta)n_k$ jobs in round k. □

Intuitively, from Lemma 3, we know that a large number of jobs must complete in each round and further, since we assume that $\tilde{m}_k$ approximates the true median well, many of those jobs have remaining sizes at least $\tilde{m}_k$. We next show that $\Omega(n_k)$ jobs must have remaining size at least $\tilde{m}_k$ and hence the optimal solution must incur a total cost of at least $\Omega(\sum_k \tilde{m}_k n_k^2)$.

Lemma 4. $\sum_{k=1}^{K} \tilde{m}_k n_k^2 \le 266 \cdot \textit{opt}(J \setminus J_{K+1})$.

Proof. By Proposition 2(iv), we know $\text{opt}(J \setminus J_{K+1})$ is lower bounded by $\sum_{\text{odd } k\in[K-1]} \text{opt}(J_k \setminus J_{k+2})$, and also by $\sum_{\text{even } k\in[K-1]} \text{opt}(J_k \setminus J_{k+2})$. Thus, we have

$$2\text{opt}(J \setminus J_{K+1}) \ge \sum_{k=1}^{K-1} \text{opt}(J_k \setminus J_{k+2}).$$

By Lemma 3, we know that at least $(1-(1/2+2\delta)^2)n_k$ jobs in J_k complete in round k or $k+1$. Further, as $\tilde{m}_k$ is a $(1+\delta)$-approximation, less than $(1/2+\delta)n_k$ jobs in J_k have $p_{k,j} \le \tilde{m}_k$. Thus, we conclude that there are at least $(1-((1/2+2\delta)^2)-(1/2+\delta))n_k \ge (1/8)n_k$ jobs in J_k with $p_{k,j} \ge \tilde{m}_k$ that complete in round k or $k+1$; here, $\delta \le 1/50$. Let F_k denote the set of those jobs. Note that $\text{opt}(J_k \setminus J_{k+2}) \ge \text{opt}(F_k) \ge (1/2)\tilde{m}_k((1/8)n_k)^2 = \tilde{m}_k n_k^2/128$.

Therefore, we have

$$2\text{opt}(J \setminus J_{K+1}) \ge \sum_{k=1}^{K-1} \frac{1}{128}\tilde{m}_k n_k^2.$$

Further, we have

$$\text{opt}(J \setminus J_{K+1}) \ge \text{opt}(J_K) \ge (1/2)\tilde{m}_K(0.45 n_K)^2 \ge 0.1\tilde{m}_K n_K^2,$$

as there are at least $(1/2-\delta)n_K \ge 0.45 n_K$ jobs of sizes $\ge \tilde{m}_K$. □

Next, we upper bound our algorithm's cost. Let A_k be the total delay incurred by our algorithm in round k. When our algorithm processes a job j in round k, all the other alive jobs are forced to wait; A_k counts this total waiting time. Formally, we define $A_k := \sum_{i\ne j \in J_k} D_k(i,j)$ where $D_k(i,j)$ is the total amount of processing received by job i in round k while job j is still alive. We observe the following simple upper bound:

Observation 2. *For any k, we have $A_k \leq \sum_{j \in J_k} q_{k,j} \cdot n_{k,j}$, where $n_{k,j}$ is the maximum number of other jobs that are still alive when job j is processed in round k.*[8]

Lemma 5. *For any $k \in [K]$, $A_k \leq 2\tilde{m}_k n_k^2$.*

Proof. We note that in any round $k \in [K]$, any job j gets processed by at most $2\tilde{m}_k$ units. Since a job j can delay at most $(n_k - 1)$ jobs in round k, we have $A_k \leq \sum_{j \in J_k} q_{k,j}(n_k - 1) \leq \sum_{j \in J_k} 2\tilde{m}_k n_k \leq 2\tilde{m}_k n_k^2$. □

Observe that we use Round-Robin in the final round $K+1$, which is known to be 2-competitive. Hence, to complete all the remaining jobs, by considering each job's contribution to the objective, our algorithm's cost can be upper bounded as follows:

Lemma 6. *The algorithm's cost is at most $\sum_{k=1}^{K} 2\tilde{m}_k n_k^2 + 2\textsc{opt}(J_{K+1}) + \sum_{j \in J} p_j$.*

By Lemmas 4 and 6, the algorithm's cost is at most $2 \cdot 266\ \textsc{opt}(J \setminus J_{K+1}) + 2\textsc{opt}(J_{K+1}) + \textsc{opt}(J) \leq 535\ \textsc{opt}(J)$,[9] where the last inequality follows from Proposition 2(iv). This completes the proof of Theorem 1.

4.2 Consistency

In this section, we show that Algorithm 3 also utilizes good predictions to obtain improved guarantees. We analyze the delay incurred by our algorithm in RR rounds and non-RR rounds separately.

4.2.1 RR Round. Intuitively, using the fact that the error is huge in each RR round, we can upper bound our algorithm's total delay in all RR rounds by the error. Recall $A_k \leq \sum_{j \in J_k} q_{k,j} \cdot n_{k,j}$. As δ is set to an absolute constant ($\delta = 1/50$), we will hide it in asymptotic notation. (Recall the surrogate error η from Definition 4.) This section is devoted to showing the following lemma:

Lemma 7. *$\sum_{k \in RR} A_k \leq O\left(\frac{1}{\epsilon}\right) \eta \leq O\left(\frac{1}{\epsilon}\right) \nu$, under Assumption 1.*

From Lemma 5, the total delay A_k incurred in round $k \in [K]$ in our algorithm is at most $2\tilde{m}_k n_k^2$. Thus, our goal is to carefully identify a part of the surrogate error of magnitude $\Omega(\epsilon \tilde{m}_k n_k^2)$ to charge A_k to, in each RR round k.

In the following lemma, we consider three types of jobs and show that the error is big enough for at least one job type. The job types are: (i) those completing in round k, (ii) whose remaining predicted sizes become 0 in the round, and (iii) whose remaining predicted sizes are 0 and that do not complete in this round. We need to be careful when extracting some error for type (iii) jobs as they may reappear as type (iii) jobs in subsequent RR rounds. This is why we measure the error by pretending their remaining sizes are $2\tilde{m}_k$, exactly the amount by which the jobs each are processed in the round.

Lemma 8. *In any RR round k, at least one of the following is $\Omega(\epsilon)\tilde{m}_k(n_k)^2$:*

(i) $\eta(J_k \setminus J_{k+1})$.
(ii) $\eta(\hat{F}_k)$ *where* $\hat{F}_k = \{j \in J_k \mid \hat{p}_{k,j} > 0$ *and* $\hat{p}_{k+1,j} = 0\}$.
(iii) $\eta(\hat{Z}_k; \{2\tilde{m}_k\}, \{0\}) = \textsc{opt}(\{2\tilde{m}_k\}_{j \in \hat{Z}_k})$, *where* $\hat{Z}_k := \{j \in J_k \mid \hat{p}_{k,j} = 0$ *and* $p_{k,j} > 2\tilde{m}_k\}$.

Proof. Let $S_k := \{j \in J_k \mid \hat{p}_{k,j} \leq (1+2\epsilon)\tilde{m}_k$ or $p_{k,j} \leq (1+2\epsilon)\tilde{m}_k\}$. Since $d_{k,j} = 0$ for all jobs $j \in J_k \setminus S_k$, by the definition of η_k, we have $\eta_k = \eta_k(S_k)$. For notational convenience, let

[8] Since all alive jobs are equally processed at each time in a RR round, the number of alive jobs can change while a job is being processed, which is not the case in non-RR rounds.
[9] Note that $\sum_{j \in J} p_j$ is already factored in the upper bound of A_k stated in Lemma 5. Thus, we can show that the algorithm's cost is at most $534\textsc{opt}(J)$.

$X := S_k \cap (J_k \setminus J_{k+1})$, $Y := S_k \cap \hat{F}_k$, and $Z := S_k \cap \hat{Z}_k$. As each job in S_k is of at least one of the above three types, we have $S_k = X \cup Y \cup Z$.

Because of k being an RR round, we have $\tilde{\eta}_k \geq \epsilon\delta^2\tilde{m}_k n_k^2/16$ and, therefore, we have $\eta_k \geq \tilde{\eta}_k - \frac{\delta^2}{32}\epsilon\tilde{m}_k n_k^2 \geq \frac{\delta^2}{32}\epsilon\tilde{m}_k n_k^2 = \Omega(\epsilon\tilde{m}_k n_k^2)$ due to Assumption 1. Because of the monotonicity of $\eta_k = \eta_k(S_k)$ (it can only become larger when more jobs are considered) and the fact that $S_k = X \cup Y \cup Z$, by Proposition 2, we know at least one of $\eta_k(X)$, $\eta_k(Y)$, $\eta_k(Z)$ must be no smaller than $(1/9)\eta_k$. We consider each case in the following:

Case (i). $\eta_k(X) \geq (1/9)\eta_k$. Let X_o, X_u denote the jobs in X that are overestimated and underestimated, respectively. By definition of η_k and Proposition 2(iv), we have $\eta_k(X_o) + \eta_k(X_u) \geq (1/2)\eta_k(X)$, and

$$\begin{aligned}\eta_k(X_u) &= \textsc{opt}(\{\min\{(1+2\epsilon)\tilde{m}_k, p_{k,j}\} - \min\{(1+2\epsilon)\tilde{m}_k, \hat{p}_{k,j}\}\}_{j\in X_u}) \ \leq\ \textsc{opt}(\{p_{k,j} - \hat{p}_{k,j}\}_{j\in X_u}) \\ &\leq \textsc{opt}(\{p_{k,j}\}_{j\in X_u}) - \textsc{opt}(\{\hat{p}_{k,j}\}_{j\in X_u}) = \eta(X_u),\end{aligned}$$

where the last inequality follows from Proposition 2(iii).

Similarly, we can show $\eta_k(X_o) \leq \eta(X_o)$. Thus, we have

$$\eta(J_k \setminus J_{k+1}) \geq \eta(X) \geq \eta(X_u) + \eta(X_o) \geq \eta_k(X_u) + \eta_k(X_o) \geq (1/2)\eta_k(X) \geq (1/18)\eta_k,$$

where the first two inequalities follow from Proposition 5.

Case (ii). $\eta_k(Y) \geq (1/9)\eta_k$. This case can be similarly handled as the first case to obtain $\eta(\hat{F}_k) \geq \eta(Y) \geq (1/18)\eta_k$.

Case (iii). $\eta_k(Z) \geq (1/9)\eta_k$. Note that all jobs j in Z are underestimated because of $p_{k,j} > 2\tilde{m}_k$, $\hat{p}_{k,j} = 0$, and Observation 1. Also, by definition of S_k, Z, $\hat{Z}_k$, we have $Z = \hat{Z}_k$. Therefore,

$$\begin{aligned}\eta_k(Z) &= \textsc{opt}(\{\min\{(1+2\epsilon)m_k, p_{k,j}\} - \min\{(1+2\epsilon)m_k, \hat{p}_{k,j}\}\}_{j\in Z}) \\ &= \textsc{opt}(\{(1+2\epsilon)\tilde{m}_k\}_{j\in Z}) \ \leq \textsc{opt}(\{2\tilde{m}_k\}_{j\in\hat{Z}_k}) = \eta(\hat{Z}_k; \{2\tilde{m}_k\}, \{0\}).\end{aligned}$$

Thus, we have $\eta(\hat{Z}_k; \{2\tilde{m}_k\}, \{0\}) \geq (1/9)\eta_k$. □

We next show that the above errors add up to $O(\eta)$.

Lemma 9. $\sum_{k\in RR}\left(\eta(J_k \setminus J_{k+1}) + \eta(\hat{F}_k) + \eta(\hat{Z}_k; \{2\tilde{m}_k\}, \{0\})\right) \leq 3\eta$.

Proof. We start by considering the first quantity. Since the sets in $\{J_k \setminus J_{k+1}\}_{k\geq 1}$ are disjoint, from Proposition 5, we know that $\sum_{k\in RR}\eta(J_k \setminus J_{k+1}) \leq \eta$. Similarly, we can show $\sum_{k\in RR}\eta(\hat{F}_k) \leq \eta$.

It now remains to show

$$\sum_{k\in RR}\eta(\hat{Z}_k; \{2m_k\}, \{0\}) \leq \eta. \tag{1}$$

By definition of $\hat{Z}_k$ and Observation 1, we know that if $j \in \hat{Z}_k$, then j must be underestimated, i.e., $\hat{p}_j \leq p_j$. Let J_u denote the set of all underestimated jobs. By definition of η, we know that $\eta \geq \textsc{opt}(\{p_j\}_{j\in J_u}) - \textsc{opt}(\{\hat{p}_j\}_{j\in J_u})$. Further, by Proposition 2(iii),

$$\eta \geq \textsc{opt}(\{p_j - \hat{p}_j\}_{j\in J_u}).$$

The key idea is to show the decrease of $\textsc{opt}(\{p_{k,j} - \hat{p}_{k,j}\}_{j\in J_u})$ in each RR round k is as big as $\eta(\hat{Z}_k; \{2\tilde{m}_k\}, \{0\})$. By the definition of $\hat{p}_{1,j}$ and $p_{1,j}$, we have

$$\textsc{opt}(\{p_{1,j} - \hat{p}_{1,j}\}_{j\in J_u}) = \textsc{opt}(\{p_j - \hat{p}_j\}_{j\in J_u}).$$

Further, by Observation 1, we know $\hat{Z}_k \subseteq J_u$. Note that for every job $j \in \hat{Z}_k$, $((p_{k,j} - \hat{p}_{k,j}) - (p_{k+1,j} - \hat{p}_{k+1,j})) = p_{k,j} - p_{k+1,j} = 2\tilde{m}_k$. Thus, we have

$$\textsc{opt}(\{p_{k,j} - \hat{p}_{k,j}\}_{j\in J_u}) - \textsc{opt}(\{p_{k+1,j} - \hat{p}_{k+1,j}\}_{j\in J_u})$$

$$
\begin{aligned}
&\geq \textsc{opt}(\{(p_{k,j}-\hat{p}_{k,j})-(p_{k+1,j}-\hat{p}_{k+1,j})\}_{j\in J_u}) && \text{(Proposition 2 and } (p_{k,j}-\hat{p}_{k,j}) \text{ is decreasing in } k, \forall j \in J_u)\\
&\geq \textsc{opt}(\{(p_{k,j}-\hat{p}_{k,j})-(p_{k+1,j}-\hat{p}_{k+1,j})\}_{j\in \hat{Z}_k}) && \text{(Monotonicity of } \textsc{opt})\\
&= \textsc{opt}(\{2\tilde{m}_k\}_{j\in \hat{Z}_k}) = \eta(\hat{Z}_k;\{2\tilde{m}_k\},\{0\}).
\end{aligned}
$$

Thus, we have shown that η restricted to underestimated jobs decreases by as much as $\eta(\hat{Z}_k;\{2m_k\},\{0\})$, which implies Equation (1), as desired. □

We are now ready to prove Lemma 7

$$
\begin{aligned}
\sum_{k\in RR} A_k &\leq \sum_{k\in RR} 2\tilde{m}_k n_k^2 && \text{(Lemma 5)}\\
&\leq \sum_{k\in RR} O\left(\frac{1}{\epsilon}\right)\Big(\eta(J_k\setminus J_{k+1})+\eta(\hat{F}_k)+\eta(\hat{Z}_k;\{2\tilde{m}_k\},\{0\})\Big) && \text{(Lemma 8)}\\
&= O\left(\frac{1}{\epsilon}\right)\eta && \text{(Lemma 9)}\\
&\leq O\left(\frac{1}{\epsilon}\right)\nu. && \text{(Proposition 4)}
\end{aligned}
$$

4.2.2 Non-RR Round. This section is devoted to proving the following lemma that bounds our algorithm's total delay in **non-RR rounds (NRR)**. As we do not have sufficiently large errors in non-RR rounds, we will have to bound it by both $\textsc{opt}$ and ν. Note that $\textsc{opt} - (\sum_{i\in J} p_i)$ is the total delay cost of the optimal schedule.

Lemma 10. *Under Assumption 1, we have* $\sum_{k\in \text{NRR}} A_k \leq (1+O(\epsilon))OPT - (\sum_{i\in J} p_i) + O(1/\epsilon^2)\nu$.

We begin our analysis of consistency for non-RR rounds by proving the following lemma, which shows how much error we can use for each pair of jobs.

Lemma 11. $\nu(J,\{p_j\},\{\hat{p}_j\}) \geq \sum_{i\neq j\in J}\nu(i,j)$, *where* $\nu(i,j) = |\min\{p_i,p_j\}-\min\{\hat{p}_i,\hat{p}_j\}|$.

Proof. By Proposition 1, we can decompose ν as follows:

$$
\begin{aligned}
&\nu(J,\{p_j\},\{\hat{p}_j\})\\
&= \textsc{opt}\Big(\{\hat{p}_j\}_{j\in J_o}\cup\{p_j\}_{j\in J_u}\Big) - \textsc{opt}\Big(\{p_j\}_{j\in J_o}\cup\{\hat{p}_j\}_{j\in J_u}\Big)\\
&\geq \sum_{i,j\in J_o}(\min\{\hat{p}_i,\hat{p}_j\}-\min\{p_i,p_j\}) + \sum_{i,j\in J_u}(\min\{p_i,p_j\}-\min\{\hat{p}_i,\hat{p}_j\})\\
&\quad + \sum_{i\in J_o, j\in J_u}(\min\{\hat{p}_i,p_j\}-\min\{p_i,\hat{p}_j\}).
\end{aligned}
$$

Since every term in the summation is non-negative, to prove the lemma it suffices to show

$$\min\{\hat{p}_i,p_j\}-\min\{p_i,\hat{p}_j\} \geq |\min\{p_i,p_j\}-\min\{\hat{p}_i,\hat{p}_j\}|, \quad \forall i\in J_o \text{ and } j\in J_u.$$

By definition, we have $\hat{p}_i \geq p_i$ for $i\in J_o$, and hence $\min\{\hat{p}_i,p_j\}\geq\min\{p_i,p_j\}$ and $\min\{p_i,\hat{p}_j\}\leq\min\{\hat{p}_i,\hat{p}_j\}$. Hence,

$$\min\{\hat{p}_i,p_j\}-\min\{p_i,\hat{p}_j\}\geq\min\{p_i,p_j\}-\min\{\hat{p}_i,\hat{p}_j\}.$$

Similarly, since $\hat{p}_j \leq p_j$ for $j\in J_u$, we have $\min\{\hat{p}_i,p_j\}\geq\min\{\hat{p}_i,\hat{p}_j\}$ and $\min\{p_i,\hat{p}_j\}\leq\min\{p_i,p_j\}$. Hence,

$$\min\{\hat{p}_i,p_j\}-\min\{p_i,\hat{p}_j\}\geq\min\{\hat{p}_i,\hat{p}_j\}-\min\{p_i,p_j\}.$$ □

Knowing how much error we can use for each pair of jobs, we are now ready to give an overview of the analysis. Recall that we let $D_k(i,j)$ denote the delay i causes to j in round k. Note that in a non-RR round, $D_k(i,j) = q_{k,i}$ if j is still alive while i gets processed; otherwise, $D_k(i,j) = 0$.

1. Total delay involving jobs with zero remaining predicted sizes. We show that the delay involving the following jobs across all non-RR rounds is at most $O(\epsilon) \cdot \textsc{opt}$:

$$\hat{Z}_k := \{i \in J_k \mid \hat{p}_{k,i} = 0\},$$

which overrides the definition of the same notation given in Lemma 8. Fix a job $i \in \hat{Z}_k$. Such a job i gets processed by at most $3\epsilon\tilde{m}_k$, since it is processed up to $\hat{p}_{k,i} + 3\epsilon\tilde{m}_k$. Further, if job j gets processed before i, then it implies $\hat{p}_{k,j} = 0$, where j can delay i by at most $3\epsilon\tilde{m}_k$ in the round. Similarly, job i can delay another job by at most $3\epsilon\tilde{m}_k$ in the round. It is an easy exercise to see that total delay involving a job i with $\hat{p}_{k,i} = 0$ is at most $3\epsilon\tilde{m}_k n_k$. As there are at most n_k jobs remaining in this round,

$$\sum_{k \in \text{NRR}} \sum_{i \in \hat{Z}_k} \sum_{j \in J_k : j \neq i} (D_k(i,j) + D_k(j,i)) \leq \sum_{k \in \text{NRR}} 3\epsilon\tilde{m}_k n_k^2 \leq O(\epsilon)\textsc{opt}(J \setminus J_{K+1}) \leq O(\epsilon)\textsc{opt}, \qquad (2)$$

where the second inequality follows from Lemma 4.

2. Total delay involving jobs that execute but do not complete. We show the total delay across all non-RR rounds is at most $O(\epsilon)\textsc{opt} + O(1/\epsilon^2)\nu$. To precisely articulate what we aim to prove, let

$$U_k := \{i \in J_k \mid 0 < \hat{p}_{k,i} \leq (1+\epsilon)\tilde{m}_k \text{ and } p_{k,i} > \hat{p}_{k,i} + 3\epsilon\tilde{m}_k\},$$

which, roughly speaking, are the jobs with relatively small non-zero remaining predicted sizes that execute but do not complete in round k. If $i \in U_k$, then $\hat{p}_{k,i} > 0$ and $\hat{p}_{k+1,i} = 0$ and therefore the family $\{U_k\}_{k \in [K]}$ is disjoint.

The following bounds the total delay incurred due to jobs in U_k.

Lemma 12. *For each $i \in U_k$, let $D_{k,i} := \sum_{j \in J_k : j \neq i}(D_k(i,j) + D_k(j,i)) = (\sum_{j \in J_k : j \neq i} q_{k,j} + \sum_{j \in J_k : j \neq i, C_j > L_{k,i}} (3\epsilon\tilde{m}_k + \hat{p}_{k,i}))$ be the total delay involving job i in a non-RR round k, where $L_{k,i}$ denotes the last time when i gets processed in round k and C_j is j's completion time. Then, we have*

$$\sum_{i \in U_k} D_{k,i} \leq O(\epsilon)\tilde{m}_k n_k^2 + O\left(\frac{1}{\epsilon^2}\right) \sum_{i \in U_k} \sum_{j \in J_k : j \neq i} \nu(i,j).$$

Note that in $D_{k,i}$, the first term is how much other jobs delay i and the second is how much job i delays other jobs: job i delays job j in the round by exactly $\hat{p}_{k,i} + 3\epsilon\tilde{m}_k$ if j is still alive when the algorithm stops processing i in the round. The proof is a bit subtle and is deferred to Section 4.3 but the intuition is the following: Suppose we made a bad mistake by working on job $i \in U_k$ in round k—we thought the job was small based on its prediction but it turned out to be big. This means that job i's processing delays many jobs in J_k, which we could have avoided had we known that i was in fact big. Thus, to charge the delay, we show that the considerable underprediction of job i creates a huge error as it makes a large difference in how much i delays other big jobs.

Assuming Lemma 12, we have

$$\begin{aligned}\sum_{k \in \text{NRR}} \sum_{i \in U_k} D_{k,i} &\leq \sum_{k \in \text{NRR}} O(\epsilon)\tilde{m}_k n_k^2 + O\left(\frac{1}{\epsilon^2}\right) \sum_{i \in U_k} \sum_{j \in J_k : j \neq i} \nu(i,j) \\ &\leq O(\epsilon)\textsc{opt} + O\left(\frac{1}{\epsilon^2}\right) \sum_{k \in \text{NRR}} \sum_{i \in U_k, j \in J : j \neq i} \nu(i,j) \qquad \text{(Lemma 4 and } J_k \subseteq J)\end{aligned}$$

$$\le O(\epsilon)\textsc{opt} + O\left(\frac{1}{\epsilon^2}\right) \sum_{i \ne j \in J} \nu(i,j) \qquad (U_1, \ldots, U_K \text{ are disjoint})$$
$$\le O(\epsilon)\textsc{opt} + O\left(\frac{1}{\epsilon^2}\right) \cdot \nu. \qquad \text{(Lemma 11)} \tag{3}$$

3. Total delay due to the other jobs. Finally, we consider delay not considered by the two cases. Let us see for which pairs of jobs we did not consider their pairwise delay. For job i to delay job j in a non-RR round k, i must be processed, meaning that $\hat{p}_{k,i} \le (1+\epsilon)\tilde{m}_k$. Since Case 1 already considered $\hat{p}_{k,i} = 0$, $i \in \hat{Z}_k$, we assume $\hat{p}_{k,i} > 0$. Further, if i does not complete in round k, then we already covered the delay in Case 2 as $i \in U_k$. Thus, we only need to consider the case when $i \in V_k$, where V_k is defined as follows:

$$V_k := \{i \in J_k \mid 0 < \hat{p}_{k,i} \le (1+\epsilon)\tilde{m}_k, p_{k,i} \le \hat{p}_{k,i} + 3\epsilon\tilde{m}_k\}.$$

Note that every $i \in V_k$ completes in round k. The following upper bounds the total delay we did not consider in the previous cases:

Lemma 13. *For any $i \in V_k$, $j \in J_k \setminus (\hat{Z}_k \cup U_k)$, the delay i causes to j in non-RR round k, $D_k(i,j)$, is at most* $\min\{p_i, p_j\} + \nu(i,j) + 3\epsilon\tilde{m}_k$.

Note that $\{V_k\}_k$ is a family of disjoint job sets because every job $i \in V_k$ completes in round k. Therefore, we have

$$\sum_{k \in \mathrm{NRR}} \sum_{i \in V_k, j \in J_k \setminus (\hat{Z}_k \cup U_k) : \hat{p}_{k,j} > \hat{p}_{k,i}} D_k(i,j) \tag{4}$$
$$= \sum_{k \in \mathrm{NRR}} \sum_{i \in V_k, j \in J_k \setminus (\hat{Z}_k \cup U_k) : \hat{p}_{k,j} > \hat{p}_{k,i}} (\min\{p_i, p_j\} + \nu(i,j) + 3\epsilon\tilde{m}_k) \qquad \text{(Lemma 13)}$$
$$\le \sum_{\{i,j\} \subseteq J : i \ne j} \Big(\min\{p_i, p_j\} + \nu(i,j)\Big) + \sum_{k \in [K]} 3\epsilon\tilde{m}_k n_k^2 \qquad (V_1, \ldots, V_K \text{ are disjoint})$$
$$\le \textsc{opt} - \left(\sum_{i \in J} p_i\right) + \nu + O(\epsilon)\textsc{opt}. \qquad \text{(Proposition 1, Lemma 4, and Lemma 11)}$$

Putting the pieces together. Note that the delay incurred between every pair of jobs i and j in every non-RR round k falls into at least one of the above three categories. Thus, from Equations (2), (3), and (4), the total pairwise delay in non-RR rounds is at most

$$O(\epsilon)\textsc{opt} + O\left(\frac{1}{\epsilon^2}\right)\nu + \textsc{opt} - \left(\sum_{i \in J} p_i\right) + \nu + O(\epsilon)\textsc{opt}. \tag{5}$$

We are now ready to give the final upper bound on the objective of our algorithm, which is obtained by combining the upper bound in Lemma 7 and Equation (5) and by factoring in the total job size, $\sum_{i \in J} p_j$. Note that $2\textsc{opt}(J_{K+1})$ comes from the last round $K+1$ where the remaining jobs J_{K+1} are processed by Round-Robin, which is known to be 2-competitive.

Theorem 2. *Under Assumption 1, Algorithm 3's objective is at most* $(1 + O(\epsilon))\textsc{opt} + 2\textsc{opt}(J_{K+1}) + O(\frac{1}{\epsilon^2})\nu$.

4.3 Proof of Lemma 12 and Lemma 13

For the sake of analysis, similar to $\nu(i,j)$, let

$$\nu_k(i,j) := |\min\{p_{k,i}, p_{k,j}\} - \min\{\hat{p}_{k,i}, \hat{p}_{k,j}\}|.$$

Recall that $\nu(i,j) := |\min\{p_i, p_j\} - \min\{\hat{p}_i, \hat{p}_j\}|$. The following relates $\nu(i,j)$ to $\nu_k(i,j)$:

Lemma 14. *Any two jobs $i, j \in J_k$ with $\hat{p}_{k,i}, \hat{p}_{k,j} > 0$ have been processed by an equal amount in each previous round and hence, $\nu(i,j) = \nu_k(i,j)$.*

Proof. Note that both jobs have non-zero remaining predicted sizes. This implies that neither job was processed in a previous non-RR round if such a round exists (Observation 1). Thus, the two jobs have been processed by an equal amount in all the previous rounds. Hence, $\nu(i,j) = |\min\{p_i, p_j\} - \min\{\hat{p}_i, \hat{p}_j\}| = |\min\{p_{k,i}, p_{k,j}\} - \min\{\hat{p}_{k,i}, \hat{p}_{k,j}\}| = \nu_k(i,j)$. □

We are now ready to prove Lemma 12 and Lemma 13.

Proof of Lemma 12. Let $U'_k := \{i \in U_k \mid \hat{p}_{k,i} \le (1-\epsilon)\tilde{m}_k\}$. To prove the lemma, we will bound the inequality for jobs in U'_k and $U_k \setminus U'_k$ separately:

$$\sum_{i \in U'_k} D_{k,i} \le O\left(\frac{1}{\epsilon}\right) \sum_{i \in U'_k} \sum_{j \in J : j \ne i} \nu(i,j), \tag{6}$$

and

$$\sum_{i \in U_k \setminus U'_k} D_{k,i} \le 3\epsilon \tilde{m}_k n_k^2 + O\left(\frac{1}{\epsilon^2}\right) \sum_{i \in U_k \setminus U'_k} \sum_{j \in J : j \ne i} \nu(i,j). \tag{7}$$

We begin by showing Equation (6). Let $B := \{j \in J_k \mid p_{k,j} \ge \tilde{m}_k\}$. We will show

$$\nu(i,j) \ge \epsilon \tilde{m}_k \text{ for any } i \in U'_k \text{ and } j \in B.$$

As $\hat{p}_{k,i} > 0$ for any $i \in U'_k \subseteq U_k$, we know that i has not been processed in any previous non-RR round (Observation 1). Thus, $j \in B$ must have been processed as much as i in the previous rounds, i.e., $p_j - p_{k,j} \ge p_i - p_{k,i}$. Let $\Delta := p_i - p_{k,i}$. As $\hat{p}_{k,i} > 0$, i's predicted size must have decreased by Δ prior to round k and thus $\hat{p}_i - \hat{p}_{k,i} = \Delta$. Then, we obtain

$$\begin{aligned}\nu(i,j) &\ge \min\{p_i, p_j\} - \min\{\hat{p}_i, \hat{p}_j\} \ \ge\ \min\{p_i, p_j\} - \hat{p}_i \ \ge\ \min\{p_{k,i} + \Delta, p_{k,j} + \Delta\} - (\hat{p}_{k,i} + \Delta) \\ &= \min\{p_{k,i} - \hat{p}_{k,i}, p_{k,j} - \hat{p}_{k,i}\} \ \ge\ \min\{3\epsilon\tilde{m}_k, \epsilon\tilde{m}_k\} \ =\ \epsilon\tilde{m}_k,\end{aligned}$$

as desired. Note that

$$D_{k,i} \le 2(1+4\epsilon)\tilde{m}_k n_k,$$

as each job gets processed up to $(1+4\epsilon)\tilde{m}_k$ units, and i can delay at most $n_k - 1$ jobs and can get delayed by at most $n_k - 1$ jobs. Further, under the assumption that $\tilde{m}_k$ is $(1+\delta)$-order approximation of m_k, we know that $|B| \ge (1/2 - \delta)n_k$. Therefore, we have Equation (6) as follows:

$$\sum_{i \in U'_k} D_{k,i} \le |U'_k| \cdot 2(1+4\epsilon)\tilde{m}_k n_k \le O\left(\frac{1}{\epsilon}\right) \sum_{i \in U'_k} \sum_{j \in B : j \ne i} \nu(i,j) \le O\left(\frac{1}{\epsilon}\right) \sum_{i \in U'_k} \sum_{j \in J : j \ne i} \nu(i,j).$$

We now shift to showing Equation (7). We consider two cases. The first case is when $|U_k \setminus U'_k| \le \epsilon n_k$. We clearly have

$$\sum_{i \in U_k \setminus U'_k} D_{k,i} \le 2(1+4\epsilon)\tilde{m}_k n_k \cdot \epsilon \tilde{n}_k \le 3\epsilon \tilde{m}_k n_k^2. \tag{8}$$

In this other case, we will show

$$\sum_{i \in U_k \setminus U'_k} D_{k,i} \le O\left(\frac{1}{\epsilon^2}\right) \sum_{i \in U_k \setminus U'_k} \sum_{j \in J : j \ne i} \nu(i,j). \tag{9}$$

Then, Equations (8) and (9) together will imply Equation (7).

It now remains to show Equation (9). Observe $\nu(i,j) \geq 3\epsilon\tilde{m}_k$ for any $i \neq j \in U_k \setminus U'_k$. To see this, say $p_i \leq p_j$, as the other case is symmetric. Then, we have $\nu(i,j) \geq \min\{p_i, p_j\} - \min\{\hat{p}_i, \hat{p}_j\} \geq p_i - \hat{p}_i \geq 3\epsilon\tilde{m}_k$. Therefore, we have

$$\sum_{i<j \in U_k \setminus U'_k} \nu(i,j) \geq 3\epsilon\tilde{m}_k \binom{|U_k \setminus U'_k|}{2} \geq \epsilon\tilde{m}_k \cdot |U_k \setminus U'_k|^2 \geq \epsilon^2 \tilde{m}_k n_k \cdot |U_k \setminus U'_k|,$$

as $|U_k \setminus U'_k| \geq \epsilon n_k$ and $n_k \geq \frac{1}{\epsilon^3}\log n \geq 10$ for any $k \in [K]$. Thus, we obtain the desired bound

$$\sum_{i \in U_k \setminus U'_k} D_{k,i} \leq |U_k \setminus U'_k| \cdot 2(1+4\epsilon)\tilde{m}_k n_k \leq O\left(\frac{1}{\epsilon^2}\right) \sum_{i<j \in U_k \setminus U'_k} \nu(i,j) \leq O\left(\frac{1}{\epsilon^2}\right) \sum_{i \in U_k \setminus U'_k} \sum_{j \in J} \nu(i,j). \qquad \square$$

Proof of Lemma 13. Consider any $i \in V_k$ and $j \in J_k \setminus (\hat{Z}_k \cup U_k)$. Since i gets processed and completes in round k, job i can delay job j only when $\hat{p}_{k,i} \leq \hat{p}_{k,j}$. Also, it is worth noting that if i delays j in round k, then j does not delay i. As job i completes in the round k, it gets processed by $p_{k,i} \leq \hat{p}_{k,i} + 3\epsilon\tilde{m}_k$. Assume $p_j < p_i$, since otherwise the lemma follows immediately. As $\hat{p}_{k,i}, \hat{p}_{k,j} > 0$, by Lemma 14, we have $p_{k,j} < p_{k,i}$. Again, by Lemma 14, we have $\nu(i,j) = \nu_k(i,j) = |\min\{p_{k,i}, p_{k,j}\} - \min\{\hat{p}_{k,i}, \hat{p}_{k,j}\}| = |p_{k,j} - \hat{p}_{k,i}|$. Then, the delay can be upper bounded by $\hat{p}_{k,i} + 3\epsilon\tilde{m}_k \leq p_{k,j} + \nu(i,j) + 3\epsilon\tilde{m}_k \leq p_j + \nu(i,j) + 3\epsilon\tilde{m}_k$. $\square$

4.4 Removing Simplifying Assumptions

Our goal here is to extend Theorem 2 by removing Assumption 1. First, we can remove assumption (iv) by pretending that jobs have not been processed during the estimation processes. Thus, we might have to waste processing a job after completing it for the purpose of simulation.

Removing other assumptions need more care. We say that a *bad event* $\mathscr{B}_k$ occurs in round k if $\tilde{m}_k$ fails to be $(1+\delta)$-approximate or $\tilde{\eta}_k$ fails to be $(1+\frac{\delta^2\epsilon}{32})$-approximate. By Lemma 1 and Lemma 2, $\mathscr{B}_k$ occurs with probability at most $2/n^2$. If $\mathscr{B}_k$ does not occur, then we know that a constant fraction of jobs complete in round k, thanks to Lemma 3. Thus, if no bad events occur, then we have $K = O(\log n)$. Hence, by a union bound, the probability that at least one bad event occurs is at most $O((\log n)/n^2)$.

To remove Assumption 1(i) and (ii), we note that any non-idle algorithm, including ours, is n-approximate. Thus, in expectation, the above bad events can only increase the objective by $\lceil(\log n/n^2) \cdot n \cdot \textsc{opt}\rceil$, which is negligible.

We now remove Assumption 1(iii) by factoring in the extra delays due to estimating m_k and η_k, assuming no bad events occur. In the median estimation, we took a sample S of size $\lceil\frac{\log 2n}{\delta^2}\rceil$ and processed every job in S by exactly $\tilde{m}_k$. So, the maximum delay due to the processing is at most $(\tilde{m}_k) \cdot |S| \cdot |J_k| = O((\log n)\tilde{m}_k n_k)$. Similarly, in estimating η_k, we took a sample P of size $O(\frac{1}{\epsilon^2}\log n)$ and processed both jobs in each pair in P up to $(1+2\epsilon)\tilde{m}_k$ units. Thus, this processing causes total extra delay at most $2(1+2\epsilon)\tilde{m}_k \cdot |P| \cdot |J_k| = O(\frac{1}{\epsilon^2}(\log n)\tilde{m}_k n_k)$.

Hence, the extra delay cost due to the estimation is bounded by

$$\begin{aligned} O\left(\frac{1}{\epsilon^2}\right)(\log n)\sum_{k \in [K]} \tilde{m}_k n_k &\leq O(\epsilon)\sum_{k \in [K]} \tilde{m}_k n_k^2 && (n_k = |J_k| \geq \tfrac{1}{\epsilon^3}\log n \text{ for all } k \in [K]) \\ &\leq O(\epsilon)\textsc{opt}(J \setminus J_{K+1}), && (\text{Lemma } 4) \end{aligned}$$

with probability $1 - O((\log n)/n^2)$. Thus, the extra delay is negligible with high probability. The above discussion, Theorem 1, and Theorem 2, with ϵ scaled appropriately by a constant factor, yield:

Theorem 3. *There exists an algorithm such that for any sufficiently small constant $\epsilon > 0$, it yields a schedule with objective at most* $\min\{O(1)\textsc{opt}, (1+\epsilon)\textsc{opt}+2\textsc{opt}(J_{K+1})+O(\frac{1}{\epsilon^2})\nu\}$ *with high probability, where* $|J_{K+1}| \leq \frac{1}{\epsilon^3}\log n$. *Furthermore, the same bound holds in expectation.*

Corollary 1. *Suppose for any* $Z \subseteq J$ *with* $|Z| \leq \frac{1}{\epsilon^3}\log n$, $\textsc{opt}(Z) \leq \epsilon \cdot \textsc{opt}$. *Then, there exists an algorithm whose objective is at most* $\min\{O(1)\textsc{opt}, (1+\epsilon)\textsc{opt} + O(\frac{1}{\epsilon^2})\nu\}$ *with high probability. Furthermore, the same bound holds in expectation.*

4.5 Guarantees in Expectation

Previously, we showed high probability guarantees on our algorithm's objective. However, high probability guarantees inherently require $\Omega(\log n)$ samples and, therefore, we are forced to stop sampling once the number of jobs alive becomes $o(\log n)$. Here, we show that we can further continue to sample, if we only need guarantees in expectation, until we have $O(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon})$ jobs unfinished.

Towards this end, we slightly change the algorithm.

(1) Reduce the sample sizes: For estimating m_k take a sample of size $\lceil\frac{1}{\delta^2}\log 2n_k\rceil$ in Algorithm 1 and for estimating η_k take a sample of size $\lceil\frac{32^2}{\delta^4\epsilon^2}\log n_k\rceil$ in Algorithm 2. Note that the sample size depends on n_k.

(2) Run Round-Robin concurrently: We divide each instantaneous time to run Round-Robin for ϵ fraction and to run our algorithm for $(1-\epsilon)$ fraction. More precisely, we pretend that we only run each of the algorithms without changing the jobs' processing times. Let $A_j(t)$ and $B_j(t)$ denote how much Round-Robin and our algorithm have processed job j at time t. Then, job j will complete at the first time when $A_j(t) + B_j(t) = p_j$. But, we pretend that j is unfinished in our algorithm's execution until it has processed job j by its actual processing time—we do the same for Round-Robin. This way, we can analyze each of the two algorithms as if only one of them is being run. Since this simulation of the hybrid algorithm can only slow down the execution of our algorithm by a factor of $(1-\epsilon)$, intuitively, the bound in Theorem 3 only increases by a factor of $1/(1-\epsilon)$, which has no effect on our asymptotic bounds. But by running the 2-competitive Round-Robin concurrently, our final schedule will always be $2/\epsilon$-competitive.

(3) Stop sampling when $n_k = O(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon})$ (Line 2, Algorithm 3): This is doable, as we can tolerate higher probabilities of bad events, thanks to the concurrent execution of Round-Robin.

(4) In the final round $K+1$, process all jobs in increasing order of their predicted size: As we only have $|J_{K+1}| = O\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)$ jobs left, following the prediction blindly will not hurt much!

Theorem 4. *Let $\epsilon > 0$ be a sufficiently small fixed constant. Then there exists an algorithm whose expected objective value is at most*

$$\min\left\{O\left(\frac{1}{\epsilon}\right)\textsc{opt}, (1+\epsilon)\textsc{opt} + O\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu\right\}.$$

To show this, we first need the claim below. As before, we use $\mathscr{B}_k$ to denote the bad event in round k. Our first goal is to upper bound the probability that any bad event occurs.

Claim 1. $\Pr[\vee_{k\in[K]}\mathscr{B}_k] \leq O(\epsilon^6/\log^2(1/\epsilon))$.

Proof. As we changed the sample size in estimating the median and error by replacing n with n_k, it is straightforward to see $\Pr[\mathscr{B}_k] \leq 1/n_k^2$ from Lemma 1 and Lemma 2—previously the bound was $1/n^2$. We also have $\Pr[\mathscr{B}_k \mid \wedge_{k'\in[K-1]}\neg\mathscr{B}_{k'}] \leq 1/n_k^2$, as we use fresh random bits for sampling in each round. Further, we have $n_{k+1} \leq (3/4)n_k$ by Lemma 3 with $\delta \leq 1/50$ if $\neg\mathscr{B}_k$. Then, we have

$$\Pr[\wedge_{k\in[K]}\neg\mathscr{B}_k] \geq \prod_{k\in[K]}\left(1-\frac{1}{n_k^2}\right) \geq 1 - \sum_{k\in[K]}\frac{1}{n_k^2},$$

where $n_{k+1} \leq (3/4)n_k$ for all $k \in [K-1]$, and $n_K = \Omega(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon})$. Thus, we have

$$\Pr[\vee_{k\in[K]}\mathscr{B}_k] \leq O\Big(1/\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)^2\Big)\Big(1+(3/4)^2+(3/4)^4+\cdots\Big) = O\left(\frac{\epsilon^6}{\log^2(1/\epsilon)}\right). \qquad \square$$

We are now ready to prove Theorem 4.

Proof. As discussed, our algorithm augmented with Round-Robin is $(2/\epsilon)$-competitive. Thus, the expected cost of our algorithm when any bad event occurs is $O(\epsilon^6/\log^2(1/\epsilon)) \cdot (2/\epsilon)\text{OPT} = O(\epsilon)\text{OPT}$, which does not affect the asymptotic bound.

Next, we bound the extra delay due to the median and error estimation assuming no bad events occur. Since $n_k \geq \Omega(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon})$, we have $\frac{n_k}{\log n_k} = \Omega(1/\epsilon^3)$. From the observation we made in Section 4.4, the extra delay we should upper bound in round $k \in [K]$ is $O(\frac{1}{\epsilon^2})(\log n_k)\tilde{m}_k n_k \leq O(\epsilon)\tilde{m}_k n_k^2$. As shown before, this extra delay can be charged to $O(\epsilon)\text{OPT}$.

Thus, we still have the bounds in Theorem 1 and Theorem 2 and hence, our algorithm's cost until round K is at most,

$$(1+\epsilon)\text{OPT}(J \setminus J_{K+1}) + O\left(\frac{1}{\epsilon^2}\right)\nu.$$

Finally, we need to bound how much the jobs alive in the final round $K+1$ contribute to the objective. As we already bounded how much they were delayed in the previous rounds, we can pretend w.l.o.g. that we only have jobs in J_{K+1}, process them from time 0 in increasing order of their predicted size, and upper bound the resulting objective. Then, by Reference [20, Lemma 3.2], the objective is at most

$$\text{OPT}(J_{K+1}) + (n_{K+1}-1)\sum_{j\in J_{K+1}}|p_j - \hat{p}_j| \leq \text{OPT}(J_{K+1}) + n_{K+1}\nu = \text{OPT}(J_{K+1}) + O\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu,$$

where the first step is by Proposition 6. The final objective, using Proposition 2(iv), is thus at most

$$(1+\epsilon)\text{OPT}(J\setminus J_{K+1}) + O\left(\frac{1}{\epsilon^2}\right)\nu + \text{OPT}(J_{K+1}) + O\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu \leq (1+\epsilon)\text{OPT} + O\left(\frac{1}{\epsilon^3}\log\frac{1}{\epsilon}\right)\nu.$$

As we simulate our main algorithm at a rate of $(1-\epsilon)$ at each time, the consistency bound will increase by $\frac{1}{1-\epsilon}$ factor. The $(2/\epsilon)$-competitiveness follows from the concurrent execution of Round-Robin. By scaling ϵ appropriately, we obtain Theorem 4. $\square$

5 LOWER BOUNDS

We first show our analysis of Algorithm 3 is tight. Recall Theorem 2, which states that the algorithm's objective is at most $(1+\epsilon)\text{OPT} + \frac{1}{\epsilon^2}\nu$ if for any subset $Z \subseteq J$ of jobs whose size is polylogarithmic in n, $\text{OPT}(Z) = o(\text{OPT})$.

Theorem 5. *For any $\gamma > 0$, there exists a sufficiently small $\epsilon > 0$, such that there is an instance that shows our algorithm's objective is greater than $(1+\epsilon)\text{OPT} + \Omega(1/\epsilon^{2-\gamma})\nu$.*

Proof. Let $\beta := \epsilon^{1-\gamma}$. There are two groups of jobs, X and Y. Group X consists of βn jobs that each have true size $p_j = 1+\beta$ and predicted size $\hat{p}_j = 1$. Group Y consists of the remaining $(1-\beta)n$ jobs with unit true and predicted sizes, i.e., $p_j = \hat{p}_j = 1$ for all $j \in Y$.

Let A denote the total completion time of the schedule found by our algorithm, and let OPT denote that of the optimal solution for the true job sizes. In this case, it is easy to verify that $\text{OPT} = \Theta(n^2)$ and the total error $\nu = \Theta(\beta^3 n^2)$. For brevity, assume that the algorithm's median and error estimation is exact. Then, $\tilde{m}_1 = 1$ and $\tilde{\eta}_1 = \Theta(\epsilon\beta^2 n^2)$. Thus, the algorithm's first round is non-RR. All jobs have the same predicted size and therefore are indistinguishable by our algorithm. Say it first considers jobs in X. Unfortunately, it finishes no jobs in X in the first round as $\beta = \omega(\epsilon)$.

Thus, the algorithm has at least $|X| \cdot |Y|$ more units of delay than the optimum solution that first completes all jobs in Y and then those in X. Thus, we have $A - \textsc{opt} \geq |X||Y| \geq \beta(1-\beta)n^2 = \Theta(\beta n^2)$. But, since $\epsilon = o(\beta)$ and $\textsc{opt} = \Theta(n^2)$, we have $A - (1+\epsilon)\textsc{opt} \geq \Omega(\beta n^2) = \Omega(1/\beta^2)\nu$. □

The following theorem concerns the tradeoff between $\textsc{opt}$ and ν in the consistency bound:

Theorem 6. *For any sufficiently small $\epsilon > 0$ and $\gamma > 0$, no algorithm's objective is at most $(1+\epsilon)\textsc{opt} + O(1/\epsilon^{1-\gamma})\nu$.*

Proof. Consider the following lower bound instance: There are $n := 1/\epsilon^{1-\gamma/4}$ jobs. All jobs have predicted sizes 1. Suppose all jobs have true sizes exactly 1, except one job having size 2, which we call *big*. Note that $\textsc{opt} = n(n+1)/2 + 1$ and $\nu = 1$. Thus, we have

$$\epsilon\textsc{opt} + O(1/\epsilon^{1-\gamma})\nu = O(1/\epsilon^{1-\gamma/2} + 1/\epsilon^{1-\gamma}) = O(1/\epsilon^{1-\gamma/2}). \tag{10}$$

We use Yao's Min-Max theorem. Assume the adversary permutes the job identities uniformly at random to hide the big job. Let us consider the time when a fixed deterministic algorithm has processed the big job by one unit. Then, the expected number of small jobs that are still alive is $(n-1)/2$. Once the algorithm knows the big job, we can assume that it uses SRPT. Since all small jobs that are still alive are delayed by one unit of time due to processing the big job, we have

$$\mathbb{E}[A] - \textsc{opt} \geq (n-1)/2 = \Theta(1/\epsilon^{1-\gamma/4}),$$

where A denotes the algorithm's objective. This, together with Equation (10), completes the proof. □

6 CONCLUSIONS AND FUTURE DIRECTIONS

In this article, we defined a new prediction error measure based on natural desiderata. We believe that the new measure could be useful for other optimization problems with ML predictions where the ℓ_1-norm measure is not suitable. We remark that our contributions are primarily theoretical and focus on asymptotic analysis; obtaining practical algorithms that achieve similar guarantees while incurring a small overhead (say, over Round-Robin) is an interesting question. Other future research directions include finding a deterministic algorithm with similar guarantees, obtaining a better dependence on ν, and extending the error notion to the setting where jobs have different arrival times.

ACKNOWLEDGMENTS

Part of this work was done while Sungjin Im was visiting Google Research. We thank Joan Boyar and Martin Kristian Lorenzen for bringing to our attention some issues in an earlier version of the article.

A APPENDIX

A.1 Concentration Bounds

Let $X_1, \ldots, X_\ell$ be independent random variables such that $0 \leq X_i \leq 1$ for all $i \in [\ell]$. We define the empirical mean of these variables by $\bar{X} = \frac{1}{\ell}(X_1 + \cdots + X_\ell)$. Then,

Theorem 7 (Hoeffding's Inequality). *$P(|\bar{X} - E[\bar{X}]| \geq \xi) \leq 2e^{-2\ell\xi^2}$, where $\xi \geq 0$.*

REFERENCES

[1] Anders Aamand, Piotr Indyk, and Ali Vakilian. 2019. (Learned) frequency estimation algorithms under Zipfian distribution. (2019). arXiv:1908.05198

[2] Maryam Amiri and Leyli Mohammad-Khanli. 2017. Survey on prediction models of applications for resources provisioning in cloud. *J. Netw. Comput. Applic.* 82 (2017), 93–113.

[3] Keerti Anand, Rong Ge, and Debmalya Panigrahi. 2020. Customizing ML predictions for online algorithms. In *ICML*. PMLR, 303–313.

[4] Antonios Antoniadis, Christian Coester, Marek Elias, Adam Polak, and Bertrand Simon. 2020. Online metric algorithms with untrusted predictions. In *ICML*. 345–355.

[5] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. 2020. Secretary and online matching problems with machine learned advice. In *NeurIPS*. 7933–7944.

[6] Yossi Azar, Stefano Leonardi, and Noam Touitou. 2021. Flow time scheduling with uncertain processing time. In *STOC*. 1070–1080.

[7] Yossi Azar, Stefano Leonardi, and Noam Touitou. 2022. Distortion-oblivious algorithms for minimizing flow time. In *SODA*. 252–274.

[8] Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. 2020. Learning augmented energy minimization via speed scaling. In *NeurIPS*. 15350–15359.

[9] Étienne Bamas, Andreas Maggiori, and Ola Svensson. 2020. The primal-dual method for learning augmented algorithms. In *NeurIPS*.

[10] Aditya Bhaskara, Ashok Cutkosky, Ravi Kumar, and Manish Purohit. 2020. Online learning with imperfect hints. In *ICML*. 822–831.

[11] Edith Cohen, Ofir Geri, and Rasmus Pagh. 2020. Composable sketches for functions of frequencies: Beyond the worst case. In *ICML*.

[12] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. 2021. Faster matchings via learned duals. In *NeurIPS*. 10393–10406.

[13] Sreenivas Gollapudi and Debmalya Panigrahi. 2019. Online algorithms for rent-or-buy with expert advice. In *ICML*. 2319–2327.

[14] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. 2019. Learning-based frequency estimation algorithms. In *ICLR*.

[15] Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. 2017. Competitive algorithms from competitive equilibria: Non-clairvoyant scheduling under polyhedral constraints. *J. ACM* 65, 1 (2017), 1–33.

[16] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. 2021. Online knapsack with frequency predictions. In *NeurIPS*. 2733–2743.

[17] Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. 2020. Online algorithms for weighted paging with predictions. In *ICALP*. 69:1–69:18.

[18] Bala Kalyanasundaram and Kirk Pruhs. 2000. Speed is as powerful as clairvoyance. *J. ACM* 47, 4 (2000), 617–643.

[19] Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. 2018. The case for learned index structures. In *SIGMOD*. 489–504.

[20] Ravi Kumar, Manish Purohit, and Zoya Svitkina. 2018. Improving online algorithms using ML predictions. In *NeurIPS*. 9661–9670.

[21] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. 2020. Online scheduling via learned weights. In *SODA*. 1859–1877.

[22] Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. 2021. Learnable and instance-robust predictions for online matching, flows and load balancing. In *ESA*. 59:1–59:17.

[23] Shi Li and Jiayi Xian. 2021. Online unrelated machine load balancing with predictions revisited. In *ICML*. 6523–6532.

[24] Alexander Lindermayr and Nicole Megow. 2022. Permutation predictions for non-clairvoyant scheduling. In *SPAA*. 357–368.

[25] Thodoris Lykouris and Sergei Vassilvtiskii. 2018. Competitive caching with machine learned advice. In *ICML*. 3296–3305.

[26] Andréa Matsunaga and José A. B. Fortes. 2010. On the use of machine learning to predict the time and resources consumed by applications. In *CCGRID*. 495–504.

[27] Michael Mitzenmacher. 2018. A model for learned Bloom filters and optimizing by sandwiching. In *NeurIPS*. 464–473.

[28] Michael Mitzenmacher. 2020. Scheduling with predictions and the price of misprediction. In *ITCS*. 14:1–14:18.

[29] Rajeev Motwani, Steven Phillips, and Eric Torng. 1994. Nonclairvoyant scheduling. *Theoret. Comput. Sci.* 130, 1 (1994), 17–47.

[30] Ilia Pietri, Gideon Juve, Ewa Deelman, and Rizos Sakellariou. 2014. A performance model to estimate execution time of scientific workflows on the Cloud. In *Workshop on Workflows in Support of Large-Scale Science*. 11–19.

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

[32] Dhruv Rohatgi. 2020. Near-optimal bounds for online caching with machine learned advice. In *SODA*. 1834–1845.

[33] Tim Roughgarden. 2020. *Beyond the Worst-case Analysis of Algorithms*. Cambridge University Press.

[34] Bo Sun, Ali Zeynali, Tongxin Li, Mohammad Hassan Hajiesmaili, Adam Wierman, and Danny H. K. Tsang. 2020. Competitive algorithms for the online multiple knapsack problem with application to electric vehicle charging. *Proc. ACM Meas. Anal. Comput. Syst.* 4, 3 (2020), 51:1–51:32.

[35] Kapil Vaidya, Eric Knorr, Tim Kraska, and Michael Mitzenmacher. 2021. Partitioned learned Bloom filter. In *ICLR*.

Received 18 January 2022; accepted 20 April 2023