Lindermayr, Alexander; Megow, Nicole

Permutation Predictions for Non-Clairvoyant Scheduling

Conference paper as: peer-reviewed accepted version (Postprint)

DOI of this document*(secondary publication): https://doi.org/10.26092/elib/3188

Publication date of this document: 01/08/2024

* for better findability or for reliable citation

Recommended Citation (primary publication/Version of Record) incl. DOI:

Alexander Lindermayr and Nicole Megow. 2022. Permutation Predictions for Non-Clairvoyant Scheduling. In Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '22). Association for Computing Machinery, New York, NY, USA, 357–368. https://doi.org/10.1145/3490148.3538579.

Permutation Predictions for Non-Clairvoyant Scheduling

Alexander Lindermayr
University of Bremen
Bremen, Germany
linderal@uni-bremen.de

Nicole Megow
University of Bremen
Bremen, Germany
nicole.megow@uni-bremen.de

ABSTRACT

In non-clairvoyant scheduling, the task is to find an online strategy for scheduling jobs with a priori *unknown* processing requirements with the objective to minimize the total (weighted) completion time. We revisit this well-studied problem in a recently popular learning-augmented setting that integrates (untrusted) predictions in online algorithm design. While previous works used predictions on processing requirements, we propose a new prediction model, which provides a relative order of jobs which could be seen as predicting algorithmic actions rather than parts of the unknown input. We show that these predictions have desired properties, admit a natural error measure as well as algorithms with strong performance guarantees and that they are learnable in both, theory and practice. We generalize the algorithmic framework proposed in the seminal paper by Kumar et al. (NeurIPS'18) and present the first learning-augmented scheduling results for weighted jobs and unrelated machines. We demonstrate in empirical experiments the practicability and superior performance compared to the previously suggested single-machine algorithms.

CCS CONCEPTS

• **Theory of computation → Approximation algorithms analysis**; **Online algorithms**.

KEYWORDS

Scheduling, non-clairvoyant, unrelated machines, competitive ratio, predictions, learning-augmented algorithms

1 INTRODUCTION

Non-clairvoyant scheduling requires to schedule jobs without knowing their processing requirements a priori. This is a fundamental problem and has been studied extensively in many variations [15, 28, 29, 34, 46].

We consider non-clairvoyant scheduling with the objective of minimizing the sum of weighted completion times in different settings. Generally, we are given a set of jobs, each job with individual weight and unknown processing time, possibly arriving online at its release date. All jobs must be scheduled on a single or identical parallel machines; preemption is allowed. Using classical scheduling notation, we refer to the problems we consider as the non-clairvoyant versions of $1|pmtn|\sum w_jC_j$ and $P|r_j, pmtn|\sum w_jC_j$. We also investigate non-clairvoyant scheduling on unrelated machines, denoted by $R|r_j, pmtn|\sum w_jC_j$, where jobs may have very different processing times on each of the machines, but a machine-dependent processing rate is given. (Precise definitions follow later.)

The performance of online algorithms is typically assessed by *competitive analysis*. An online algorithm is *ρ-competitive* if, for all instances I, the algorithm has cost $\textsc{Alg}(I) \leq \rho \cdot \textsc{Opt}(I)$, where $\textsc{Opt}(I)$ is the objective value of an optimal solution for I.

Non-clairvoyant algorithms assign processing *rates* to jobs and assume time sharing, that is, parallel processing of jobs with rates that sum up to at most one per machine and per job. One could see this as processing each job by a certain amount in every infinitesimal time interval. The most prominent strategy is the Round-Robin (RR) algorithm, which assigns equal rates to all alive jobs and is 2-competitive for $1|pmtn|\sum C_j$, which is best possible [46]. The same guarantee is possible using a natural generalization of RR to weighted jobs [34] and/or to identical machines [15, 46]. Scheduling on unrelated machines is much harder and requires careful migration between machines [25]. Nevertheless, it is possible to compute rates proportional to job properties and machine constraints and obtain an $O(1)$-competitive algorithm ([28]; also implicitly in [29]).

The assumption of non-clairvoyance seems too strong for many applications. While the exact processing time might be unknown, often some estimate is available, e.g., extracted information from past data is commonly used to predict the future. The recently emerging line of research on *learning-augmented* algorithms proposes to design algorithms that have access to additional (possibly erroneous) input, called *prediction*, to achieve an improved performance if the prediction is accurate while performing not much worse than algorithms without access to predictions, if the predictions are completely wrong. Ideally, the performance of a learning-augmented algorithm is a function of the quality of the prediction for some well defined error measure. Here, defining an appropriate error measure is a key task. Given a definition for the prediction error $\eta \geq 0$ that quantifies the quality of the prediction, the goal is express the competitive ratio of the algorithm by a monotone function $f(\eta)$. A learning-augmented algorithm is called $f(0)$-consistent (in case of perfect prediction) and β-robust if $f(\eta) \leq \beta$ for all possible errors η.

Recent work on non-clairvoyant scheduling with predictions [30, 48, 56] studies the single-machine problem $1|pmtn|\sum C_j$ with predicted processing requirements $\{y_j\}_{j\in J}$, which we call *length predictions*. Commonly, we distinguish two categories of prediction models: either predict parts of the online input (*input-predictions*) [8–10, 40, 48] or algorithmic actions (*action-predictions*) [4, 11, 36]. Length predictions clearly fall in to the first category.

In their seminal paper [48], Kumar et al. propose an algorithm that is controlled by a parameter $\lambda \in (0, 1)$, which can be seen as an indicator of the algorithm's trust in the accuracy of the prediction. Measuring the quality of a prediction $\{y_j\}_{j\in J}$ w.r.t. the actual processing requirements $\{p_j\}_{j\in J}$ by the ℓ_1 metric ($\ell_1 = \sum_{j\in J}|p_j - y_j|$), they prove a competitive ratio of at most $(1/(1-\lambda))(1 + n\ell_1/\textsc{Opt})$ while also maintaining a robustness factor of $2/\lambda$. However, the ℓ_1-metric does not seem to distinguish well between "good" and "bad" predictions, as has been noted recently by Im et al. [30]. They argue that, intuitively, the linear error measure ℓ_1 is incompatible with the sum of weighted completion time objective and using $n \cdot \ell_1$ as upper bound may overestimate the "actual" error, substantially.

Im et al. [30] propose a different error measure ν that satisfies certain desired properties and is based on the optimal solution of artificial instances mixing y_j and p_j. It satisfies $\ell_1 \le \nu \le n\ell_1$. Using this error, they design a learning-augmented randomized algorithm with competitive ratio $\min\{1 + \lambda + O(1/\lambda^3 \log(1/\lambda)) \cdot \nu/\textsc{Opt}, 2/\lambda\}$, for sufficiently small $\lambda > 0$, in expectation. Their algorithm is quite sophisticated, requires large constants to diverge from RR (we give more details later), and it seems very challenging to generalize it to scheduling settings with release dates, weights or even heterogeneous machines. Further, the error measure ν is still sensitive to changes in the predicted job lengths which would not affect an optimal schedule at all which seems an undesired property.

Our contribution. In this work, we contribute to non-clairvoyant scheduling with predictions in two ways: (i) we propose a new prediction model with a new error definition, as an alternative to length predictions studied so far, and (ii) we revisit the classical idea of time sharing and develop a general framework for designing learning-augmented scheduling algorithms for more general settings, beyond the simple single-machine setting.

We propose a novel prediction model for scheduling problems, which we call *permutation prediction model*. Intuitively, it provides a permutation of jobs suggesting a priority order for scheduling. In a way, this is an action-prediction in contrast to previously studied input-predictions. The idea is that, instead of predicting job lengths, we take structural properties of an input instance into account that an optimal algorithm may exploit. Notice that for minimizing the sum of weighted completion time, the *Weighted Shortest Remaining Processing Time (WSPT)* order, i.e., jobs in order of weight over processing time ratios, has proven to be useful in various settings. Indeed, for the non-clairvoyant version of $1|pmtn|\sum w_jC_j$, knowing the WSPT order of jobs would be sufficient to determine an optimal schedule [53]. While this knowledge is not sufficient for optimally scheduling with release dates and/or on multiple machines, it still admits strategies with good approximations on an optimal solution [3, 26, 42]. For unrelated machines, we also include a job-to-machine assignment in the prediction model.

Clearly, a WSPT-based permutation prediction could be derived from a length prediction. The advantage of our model is that it is much more compact, captures a crucial structural property of an optimal solution and makes error measures less vulnerable to small noise in the prediction compared to the length prediction model.

As a key contribution, we define a new, meaningful error measure that quantifies the impact of an error in the prediction to an algorithm's cost explicitly in terms of the objective function. It has several desirable properties such as (i) monotonicity and (ii) Lipschitzness (both highly advertised recently by Im el al. [30]), (iii) theoretical learnability of our prediction model with respect to the error definition, which we show by proving that our predictions are efficiently PAC-learnable in the agnostic sense, as well as (iv) practical learnability, which we demonstrate in empirical experiments, showing that our implemented learning algorithm quickly improves the performance of our scheduling algorithms and appears superior to previously presented algorithms.

Further, we revisit the algorithmic technique of time sharing introduced by Kumar et al. [48] in their seminal work on non-clairvoyant scheduling with predictions. We extend this technique to a general framework for designing learning-augmented scheduling algorithms allowing for release dates, job weights and unrelated machines. As a main contribution, we give the first algorithm for non-clairvoyant scheduling with predictions on unrelated machines and prove strong performance bounds, smoothly degrading with prediction quality. More precisely, we show for the permutation prediction model and two appropriate error definitions η^S and η^R that there exists for every $\lambda \in (0, 1)$ a learning-augmented non-clairvoyant online algorithm for minimizing the total weighted completion time on

(i) a single machine, $1|pmtn|\sum w_jC_j$, with a competitive ratio of at most
$$\min\left\{\frac{1}{1-\lambda}\left(1 + \frac{\eta^S}{\textsc{Opt}}\right), \frac{2}{\lambda}\right\},$$

(ii) m identical machines with release dates, $\mathrm{P}|r_j, pmtn|\sum w_jC_j$, with a competitive ratio of at most
$$\min\left\{\frac{1}{1-\lambda}\left(2 + \frac{\eta^S}{m \cdot \textsc{Opt}}\right), \frac{3}{\lambda}\right\}, \text{ and}$$

(iii) unrelated machines with release dates, $\mathrm{R}|r_j, pmtn|\sum w_jC_j$, with a competitive ratio of at most
$$\min\left\{\frac{1}{1-\lambda}\left(5.8284 + \frac{\eta^R}{\textsc{Opt}}\right), \frac{128}{\lambda}\right\}.$$

Our framework requires a clairvoyant and a non-clairvoyant algorithm for a given scheduling problem, both of them must satisfy a certain monotonicity property. Then, we design a learning-augmented variation of the clairvoyant algorithm that admits a competitive ratio as a function of the error. Intuitively, the errors η^R and η^S measure how much an erroneous prediction influences the objective value compared to an accurate prediction. For a single and identical machines, we require even less predicted information (no machine assignment) and the simpler measure η^S suffices.

While we use non-clairvoyant algorithms as a black box from the literature, the new contribution lies in proving error-dependent

competitive ratios for monotone clairvoyant algorithms that use predictions as input. This may require designing new algorithms. In particular, we show a competitive ratio of $3+2\sqrt{2} \approx 5.8284$ for a natural Greedy algorithm for the clairvoyant problem $R|r_j, pmtn| \sum w_j C_j$. This does not match the recent and best known deterministic bound of 3 [16], but our algorithm satisfies the desired properties of being error-sensitive and monotone.

Further related work. There has been significant interest in the recent framework of learning-augmented online algorithms. Many problems have been considered, e.g., caching [4, 49, 55], further scheduling [7, 8, 10, 36, 44, 45, 50, 56], rent-or-buy problems [2, 5, 12, 24, 48, 54, 56], paging [13, 23, 33], graph problems [9, 22, 39, 57], secretary problems [6, 20], matching [4, 37, 38] and many more.

Non-clairvoyant and clairvoyant online scheduling models have been studied extensively; see the surveys [47, 51]. Most relevant for our work are WSPT-based algorithms such as [3, 26, 42].

Paper Organization. In Section 2 we give precise definitions for the problem, prediction model and error measure. Then, we introduce our algorithmic framework and apply it to concrete scheduling problems in Section 3. We prove efficient PAC learnability of our predictions in Section 4 and discuss empirical results in Section 5.

2 PROBLEM AND PREDICTION MODEL

2.1 Problem definition

We consider the problem of scheduling n jobs $J = \{1, \dots, n\} =: [n]$ preemptively on m unrelated machines. Every job $j \in J$ has an associated weight w_j and processing requirement p_j. Further, for every machine $i \in [m]$ there is given a rate ℓ_{ij} which is the amount of processing that job j receives if it is processed one time unit on machine i, resulting in a total processing time $p_{ij} = \ell_{ij} \cdot p_j$ if job j is scheduled on machine i. Jobs arrive online at their individual release dates $\{r_j\}_{j \in J}$. A non-clairvoyant online algorithm has to schedule jobs J on the given machines, but is oblivious to unreleased jobs and has no information on processing requirements. The objective is to minimize the weighted sum of completion times $\sum_{j \in J} w_j C_j$, where the completion time C_j of a job j is the first point in time when it has been processed for p_j units. In the standard three-field notation, this problem is denoted as non-clairvoyant version of $R|r_j, pmtn| \sum w_j C_j$. Note that a non-clairvoyant algorithm is oblivious to the processing requirement p_j but needs access to the machine rates ℓ_{ij} to admit a constant competitive ratio [29]. The setting where $\ell_{ij} = 1$ for all jobs j and machines i is called identical machine setting, $P|r_j, pmtn| \sum w_j C_j$, and the single machine setting without release dates is $1|pmtn| \sum w_j C_j$.

2.2 Permutation prediction model

We propose a prediction model that is heavily inspired by the relevance of the WSPT order (jobs ordered by non-increasing densities $\mu_{ij} = w_j/p_{ij}$) for scheduling to minimize the total weighted completion time.

For a scheduling instance with job set $[n]$ and a single or multiple identical machines, our prediction is a permutation $\hat{\sigma} : [n] \to [n]$ of all jobs. Given the aforementioned power of the WSPT order, we call the associated permutation of jobs, σ, *perfect prediction.*

On unrelated machines, the job-to-machine assignment crucially matters. Therefore, we add such an assignment to our prediction. In this most general model, our prediction is defined as $\hat{\sigma} = \{\hat{\sigma}_i\}_{i \in [m]}$, where $\hat{\sigma}_i$ is the permutation of jobs assigned to machine i, and every job is assigned to exactly one machine. We denote the machine to which job j is assigned in $\hat{\sigma}$ by $m(\hat{\sigma}, j)$. Given a scheduling instance without release dates and the optimal job-to-machine allocation, it would be optimal to schedule jobs in WSPT order on each machine individually. Therefore, we speak of perfect prediction $\sigma = \{\sigma_i\}_{i \in [m]}$, if σ_i involves exactly those jobs that are scheduled in an optimal solution on machine i and orders them in WSPT order, for each $i \in [m]$.

In the permutation prediction model, jobs still arrive online and, at any time, an algorithm has access only to predictions on jobs that have been released already. At any release date, the permutation is updated consistently with the previous permutation. That is, the prediction model is not allowed to change the relative order of previously known jobs.

2.3 Prediction error

The prediction error defines a measure for the quality of a prediction. It is a crucial element in the design of learning-augmented algorithms. Intuitively, the error measure shall quantify the impact that an erroneous prediction has on an (optimal) scheduling algorithm. It is not unnatural to express the error as $|\textsc{Opt}(\hat{\sigma}) - \textsc{Opt}(\sigma)|$, as has been done in [11, 22, 39], but for more complex scheduling environments the optimal solution is hard to compute and, more importantly, this error could be even negligible whereas the impact of running an optimal algorithm with the wrong prediction could be significant. The latter is what we want to quantify.

In more detail, our error measure shall capture the change in the cost that an optimal schedule must face when two jobs j and j' are inverted in a prediction $\hat{\sigma}$ with respect to σ. For example, on a single machine without release dates, if j and its successor j' in $\hat{\sigma}$ are swapped in σ, the schedule that follows $\hat{\sigma}$ pays an additional cost of $w_{j'} p_j$ but saves $w_j p_{j'}$ compared to the schedule that follows σ. However, in presence of release dates and on multiple machines, just knowing the orders may not allow us to express the change in the exact optimal cost. Therefore, we rely on an approximation as a surrogate for the optimal cost, namely, the *change in the sum of weighted completion times when preemptively scheduling jobs in the given priority order,* $\hat{\sigma}$ resp. σ.

We define two different error measures. Firstly, we define our simple error η^S for predictions that consist of a single permutation on all jobs. Then, our general measure η^R describes the quality of permutation predictions with predicted job assignments, $\sigma = \{\sigma_i\}_{i \in [m]}$. We show that η^S is special case of η^R.

Definition 2.1. For an instance of non-clairvoyant scheduling with permutation prediction $\hat{\sigma}$ consisting of a single permutation, and WSPT order σ, let $\mathcal{I}(J, \hat{\sigma}) = \{(j', j) \in J^2 \mid \sigma(j') < \sigma(j) \wedge \hat{\sigma}(j') > \hat{\sigma}(j)\}$ be the set of inverted job pairs. The prediction error of $\hat{\sigma}$ is defined as

$$\eta^S(J, \hat{\sigma}) = \sum_{(j', j) \in \mathcal{I}(J, \hat{\sigma})} (w_{j'} p_j - w_j p_{j'}).$$

This error measures the *exact* change in the objective value, in the absence of release dates. The single permutation prediction and this error will be sufficient for designing algorithms with appealing error-dependency for a single and parallel identical machines.

For scheduling on unrelated machines and predictions including a job assignment, we need a more elaborate error definition which, nevertheless, follows the same idea. Given an instance with job set J and prediction $\hat{\sigma} = \{\hat{\sigma}_i\}_{i\in[m]}$, we define for every job $j \in J$ a partial error η_j, which measures how much the different positions of j in $\hat{\sigma}$ resp. σ increases the objective value assuming preemptive scheduling according to a given permutation.

To this end, we consider for an arbitrary assignment and permutation $\pi = \{\pi_i\}_{i\in[m]}$ the schedule that processes at every point in time t on every machine i the available job j' with $m(\pi, j') = i$ that has the highest priority in π_i. Let $A(j)$ denote the set of jobs that are released but unfinished by time r_j and that are assigned to machine $m(\pi, j)$. Note that $j \in A(j)$. For two jobs j and j' with $r_j = r_{j'}$ and $m(\pi, j) = m(\pi, j')$, we assume that they are assigned to the machine in order of their indices. By denoting the remaining processing requirement of job j at time t by $p_j(t)$, and $p_{ij}(t) = \ell_{ij}p_j(t)$, the increase in the schedule's objective value for adding job j to machine $i = m(\pi, j)$ is equal to

$$W_j(J, \pi) = p_{ij} \sum_{\substack{j'\in A(j)\\ \pi_i(j')>\pi_i(j)}} w_{j'} + w_j\left(r_j + \sum_{\substack{j'\in A(j)\\ \pi_i(j')\leq\pi_i(j)}} p_{ij'}(r_j)\right).$$

Definition 2.2. For an instance of non-clairvoyant scheduling with permutation prediction $\hat{\sigma} = \{\hat{\sigma}_i\}_{i\in[m]}$ and perfect prediction $\sigma = \{\sigma_i\}_{i\in[m]}$, the prediction error for job $j \in J$ is defined as

$$\eta_j(J, \hat{\sigma}) = W_j(J, \hat{\sigma}) - W_j(J, \sigma).$$

The prediction error of $\hat{\sigma}$ is given by $\eta^R(J, \hat{\sigma}) = \sum_{j\in J} \eta_j(J, \hat{\sigma})$.

It is not difficult to see that η^R reduces to the compact error measure η^S for predictions that consist of a single permutation (without machine assignment) and without release dates.

Proposition 2.3. *For a job set J and a permutation prediction $\hat{\sigma}$, if $\hat{\sigma}$ is a single permutation and $r_j = 0$, for all $j \in J$, then*

$$\eta^R(J, \hat{\sigma}) = \eta^S(J, \hat{\sigma}).$$

Proof. Let $j \in J$. Observe that $\eta_j(J, \hat{\sigma}) = W_j(J, \hat{\sigma}) - W_j(J, \sigma)$ equals under the stated assumptions

$$\sum_{\substack{j'\in J\\ \hat{\sigma}(j)<\hat{\sigma}(j')}} w_{j'}p_j + \sum_{\substack{j'\in J\\ \hat{\sigma}(j')<\hat{\sigma}(j)}} w_j p_{j'} - \sum_{\substack{j'\in J\\ \sigma(j)<\sigma(j')}} w_{j'}p_j - \sum_{\substack{j'\in J\\ \sigma(j')<\sigma(j)}} w_j p_{j'}.$$

Combining the first with the third sum and the second with the fourth gives

$$\sum_{\substack{j'\in J\\ \sigma(j)>\sigma(j')\\ \hat{\sigma}(j)<\hat{\sigma}(j')}} w_{j'}p_j - \sum_{\substack{j'\in J\\ \sigma(j')<\sigma(j)\\ \hat{\sigma}(j')>\hat{\sigma}(j)}} w_j p_{j'} = \sum_{\substack{j'\in J\\ \sigma(j)>\sigma(j')\\ \hat{\sigma}(j)<\hat{\sigma}(j')}} (w_{j'}p_j - w_j p_{j'}).$$

Summing over all jobs and inversion pairs $\mathcal{I}$ yields

$$\sum_{(j',j)\in\mathcal{I}(J,\hat{\sigma})} (w_{j'}p_j - w_j p_{j'}) = \eta^S(J, \hat{\sigma}). \qquad \square$$

2.4 Properties of the error measure

Our new error measure satisfies several desired properties such as (*i*) monotonicity, (*ii*) Lipschitzness, (*iii*) theoretical learnability, and (*iv*) practical learnability.

Im et al. [30] advocate particularly the first two properties. *Monotonicity* requires, in the length prediction model, that the error grows as more length predictions become incorrect. In our setting, we have $\eta(\hat{\sigma}) = 0$ if $\hat{\sigma} = \sigma$, and for any inversion added to $\hat{\sigma}$, the error grows. This is because an inversion $(j', j) \in \mathcal{I}$ increases the error by $w_{j'}p_j - w_j p_{j'}$, since $\sigma(j') < \sigma(j)$ implies $w_{j'}/p_{j'} \geq w_j/p_j$. Thus, our definition satisfies monotonicity.

Lipschitzness requires the error to bound the absolute difference of the optimal objective values for the actual and predicted instance from above. Our error definition *precisely* measures the cost between a solution that follows $\hat{\sigma}$ and one that follows σ, when scheduling the actual instance preemptively according to the given order. Hence, our error measures immediately satisfy Lipschitzness for our prediction setup.

Our prediction model is *theoretically learnable* in the framework of PAC-learnability [52]. We show that permutations are efficiently PAC-learnable in the agnostic sense w.r.t. our error definition (Section 4). While this theoretic result gives a rather large bound on the required number of samples to get a low prediction error, we further demonstrate that our predictions are *learnable and useful in practice*. We implement a learning algorithm and show that even a small number of seen samples results in a drastic performance improvement of our algorithm in practical instances (Section 5).

In general, it is difficult to compare different prediction and error models. However, we can convert a given length prediction into a permutation prediction by simply computing the WSPT order based on the predicted processing requirements. For the case of unrelated machines, we further require predicted machine assignments. This conversion allows us to compare our error to the previously proposed measures ν and ℓ_1 for the case of $1|pmtn|\sum C_j$.

Firstly, we note that our error η^S is less vulnerable than ν and ℓ_1 to changes in the predicted instance which do not affect the *structure* of an optimal solution. Indeed, the optimal solution of an instance with $p_j = j$ for all $j \in [n]$ has the same structure as the optimal solution of a predicted instance with $y_j = j - 1$ for all $j \in [n]$. One would expect a small error, and indeed $\eta^S = 0$. In contrast, previously defined errors are large: $\nu = \textsc{Opt}(\{\max\{p_j, y_j\}\}) - \textsc{Opt}(\{\min\{p_j, y_j\}\}) = n(n+1)/2 - (n-1)n/2 = n$ and $\ell_1 = \sum_{j\in J}|p_j - y_j| = n$. This shows that our prediction and error seem to capture well the relevant characteristics of an input-prediction in terms of derived actions, while ν and ℓ_1 also track insignificant numerical differences between the actual and predicted instances.

In contrast to this example, there are other instances where ν and ℓ_1 underestimate the actual difficulty that is caused by the inaccuracy of the prediction given to an (optimal) algorithm. Im et al. [30] give such an example with $p_1 = y_1 = \ldots = p_{n-1} = y_{n-1} = 1$ and $p_n = n^2$ but $y_n = 0$. While the structural difference of the

optimal solutions for predicted and true values is large ($\eta^S = \Omega(n^3)$) the other error definitions only measure $\nu = n^2 + n$ and $\ell_1 = n^2$.

It is not difficult to see that our prediction error never exceeds $n\ell_1$.

Proposition 2.4. *For any instance of* $1|pmtn|\sum C_j$ *and length prediction,* $\eta^S \leq n \cdot \ell_1$.

Proof. Consider an instance with job set J and length prediction $\{y_j\}_{j\in[n]}$. Let $\hat{\sigma}$ be the corresponding predicted permutation. Since $(j', j) \in \mathcal{I}(J, \hat{\sigma})$ implies $\hat{\sigma}(j') > \hat{\sigma}(j)$, which must be due to $y_j \leq y_{j'}$, we conclude

$$\begin{aligned}\eta^S(J,\hat{\sigma}) &= \sum_{(j',j)\in\mathcal{I}(J,\hat{\sigma})} p_j - y_j + y_j - y_{j'} + y_{j'} - p_{j'} \\ &\leq \sum_{(j',j)\in\mathcal{I}(J,\hat{\sigma})} |p_j - y_j| + |p_{j'} - y_{j'}| \leq n\ell_1. \qquad \square\end{aligned}$$

Our results for non-uniform job weights on a single and identical machines translate to the length prediction model, as one can similarly show that η^S is bounded by the natural weighted generalization of $n \cdot \ell_1$, that is $\sum_{j'\in J} w_{j'} \sum_{j\in J} |p_j - y_j|$.

3 PREFERENTIAL TIME SHARING

We describe a framework for designing algorithms for non-clairvoyant scheduling with untrusted predictions, which we apply to several concrete scheduling settings in the following subsections.

In their seminal paper, Kumar et al. [48] proposed a single-machine time sharing algorithm for executing two algorithms 'in parallel', a clairvoyant (assuming predicted processing times to be correct) and a non-clairvoyant algorithm. The rate, at which each of these algorithms is executed, is determined by the confidence parameter $\lambda \in (0, 1)$. We extend this idea to a general framework for scheduling jobs with non-uniform weights and arbitrary release dates on unrelated machines.

This technique requires that both algorithms are *monotone* [48].

Definition 3.1. A scheduling algorithm is *monotone*, if for two instances with identical inputs but actual job processing requirements $\{p_1, \ldots, p_n\}$ and $\{p'_1, \ldots, p'_n\}$ such that $p_j \leq p'_j$ for all $j \in [n]$, the objective value of the algorithm for the first instance is at most its objective value for the second one.

Given two monotone algorithms $\mathcal{A}$ and $\mathcal{B}$ and a confidence parameter $\lambda \in (0, 1)$, we define a new preemptive algorithm: we run on all machines and for every infinitesimal time interval, algorithm $\mathcal{A}$ in the first $(1-\lambda)$-fraction of the interval and algorithm $\mathcal{B}$ in the remaining λ-fraction of the interval. The new algorithm hides arrived jobs until they are released in the simulated, i.e., slowed down, schedule of $\mathcal{A}$ resp. $\mathcal{B}$. The following result generalizes a single-machine version without weights and release dates [48].

Lemma 3.2. *Given a parameter* $\lambda \in (0, 1)$ *and two monotonic algorithms with competitive ratios* $\rho_{\mathcal{A}}$ *and* $\rho_{\mathcal{B}}$ *for the online problem* $\mathrm{R}|r_j, pmtn|\sum_j w_j C_j$*, there exists an algorithm for the same problem with a competitive ratio* $\min\left\{\frac{\rho_{\mathcal{A}}}{1-\lambda}, \frac{\rho_{\mathcal{B}}}{\lambda}\right\}$.

Proof. Assume that the competitive ratios of $\mathcal{A}$ and $\mathcal{B}$ are at most $\rho_{\mathcal{A}}$ and $\rho_{\mathcal{B}}$. By monotonicity of both algorithms, whenever one algorithm processes a job, the other one will not have a higher objective value due to shorter processing requirements. Since we execute $\mathcal{A}$ for a $(1-\lambda)$-fraction of time and $\mathcal{B}$ for a λ-fraction of time, the weighted completion time of a job increases by a factor of at most $1/(1-\lambda)$ resp. $1/\lambda$ compared to the schedules of $\mathcal{A}$ resp. $\mathcal{B}$, which implies the competitive ratio of the new algorithm. $\square$

Our *Preferential Time Sharing* framework crucially builds on Lemma 3.2 and takes as input two monotone algorithms, a clairvoyant algorithm $\mathcal{A}^C$ with a competitive ratio of at most ρ_C and a non-clairvoyant algorithm $\mathcal{A}^N$ with a competitive ratio of at most ρ_N. Intuitively, the non-clairvoyant algorithm will ensure robustness, while the clairvoyant algorithm, being executed based on the given predictions, gives a good consistency. As $\mathcal{A}^C$ will have access to predictions while being oblivious of true processing requirements, we call it *prediction-clairvoyant*. Our framework then gives, using Lemma 3.2 with $\mathcal{A} = \mathcal{A}^C$ and $\mathcal{B} = \mathcal{A}^N$, a time sharing algorithm with consistency $\rho_C/(1-\lambda)$ and robustness ρ_N/λ.

When aiming for error-sensitive guarantees, we require an error-dependent performance guarantee for $\mathcal{A}^C$.

Definition 3.3. A prediction-clairvoyant algorithm is *η-error-dependent* for an error measure η if its objective value is bounded by $\rho_C \cdot \mathrm{Opt}(J) + \eta(J, \hat{\sigma})$ for any instance J and prediction $\hat{\sigma}$.

We note that these definitions are independent of the used prediction model. A straightforward consequence is as follows.

Corollary 3.4. *Preferential Time Sharing with a monotone,* η*-error-dependent algorithm* $\mathcal{A}^C$ *with competitive ratio at most* ρ_C *and a monotone, non-clairvoyant algorithm* $\mathcal{A}^N$ *with competitive ratio at most* ρ_N *has, for every* $\lambda \in (0, 1)$*, a competitive ratio of at most*

$$\min\left\{\frac{1}{1-\lambda}\left(\rho_C + \frac{\eta}{\mathrm{Opt}}\right), \frac{\rho_N}{\lambda}\right\}$$

for non-clairvoyant scheduling with predictions $\mathrm{R}|r_j, pmtn|\sum w_j C_j$.

In the following subsections, we apply the Preferential Time Sharing framework to different concrete scheduling problems and prove our main algorithmic results. This requires:

(i) develop a monotone prediction-clairvoyant algorithm $\mathcal{A}^C$ with error-dependent competitive ratio; and
(ii) select an applicable non-clairvoyant monotone algorithm.

By Corollary 3.4, both algorithms combined give the desired performance bounds for preemptive scheduling with predictions. While non-clairvoyant algorithms for our problems are available in the literature, our main contribution lies in designing prediction-clairvoyant algorithms with provable low error-dependency.

3.1 Single machine

Consider non-clairvoyant scheduling of weighted jobs on a single machine, $1|pmtn|\sum w_j C_j$.

Prediction-clairvoyant algorithm. It is well-known that scheduling non-preemptively in the order given by $\hat{\sigma}$ gives the optimal schedule [53] if $\hat{\sigma}$ coincides with the WSPT order. We refer to this algorithm as prediction-clairvoyant WSPT. It is monotone since, for a fixed prediction, shrinking a job does not affect $\hat{\sigma}$ and only results in a lower completion time for this job and all its successors in $\hat{\sigma}$. We now show that it is η^S-error-dependent.

Lemma 3.5. *The prediction-clairvoyant WSPT algorithm is η^S-error-dependent.*

Proof. Consider an instance J with jobs being indexed by σ, a prediction $\hat{\sigma}$, and the schedule obtained by the prediction-clairvoyant WSPT algorithm. In this schedule, let $d(j', j)$ denote the amount of job j' that has been processed before job j completed. Thus, $d(j', j) = p_{j'}$ if and only if $\hat{\sigma}(j') < \hat{\sigma}(j)$. This implies

$$\begin{aligned}
\textsc{Alg}(J, \hat{\sigma}) &= \sum_{j=1}^{n} w_j p_j + \sum_{j=1}^{n} \sum_{j'=1}^{j-1} \left(w_j \cdot d(j', j) + w_{j'} \cdot d(j, j') \right) \\
&= \sum_{j=1}^{n} w_j p_j + \sum_{j=1}^{n} \sum_{\substack{j'=1 \\ \hat{\sigma}(j') < \hat{\sigma}(j)}}^{j-1} w_j p_{j'} + \sum_{j=1}^{n} \sum_{\substack{j'=1 \\ \hat{\sigma}(j') > \hat{\sigma}(j)}}^{j-1} w_{j'} p_j \\
&= \sum_{j=1}^{n} w_j \sum_{j'=1}^{j} p_{j'} + \sum_{j=1}^{n} \sum_{\substack{j'=1 \\ \hat{\sigma}(j') > \hat{\sigma}(j)}}^{j-1} (w_{j'} p_j - w_j p_{j'}) \\
&= \textsc{Opt}(J) + \eta^S(J, \hat{\sigma}).
\end{aligned}$$

The last equation holds since the first sum equals the objective value of the true WSPT schedule, i.e., a schedule according to σ, which is optimal and the second sum equals $\eta^S(J, \hat{\sigma})$ by Definition 2.1, since we assumed the jobs to be indexed according to σ. □

Non-clairvoyant algorithm. The *Weighted Round Robin (WRR)* algorithm distributes processing rates across all alive jobs proportional to their weights. Motwani et al. [46] showed that the algorithm has a competitive ratio of 2 for jobs with uniform weights, and Kim and Chwa [34] proved the same competitive ratio for arbitrary weights. In both cases, this ratio is best possible. It is not difficult to see that WRR is monotone, since shrinking a job's processing requirement only decreases its completion time and thus gives all other jobs more rate earlier, also reducing their completion time.

By Corollary 3.4 we conclude with the following result.

Theorem 3.6. *Preferential Time Sharing with the prediction-clairvoyant WSPT algorithm and the non-clairvoyant WRR algorithm has, for every $\lambda \in (0, 1)$, a competitive ratio of at most*

$$\min\left\{ \frac{1}{1-\lambda}\left(1 + \frac{\eta^S}{\textsc{Opt}}\right), \frac{2}{\lambda} \right\}$$

for non-clairvoyant scheduling with predictions $1|pmtn|\sum w_j C_j$.

3.2 Identical parallel machines

Consider non-clairvoyant scheduling of weighted jobs with release dates on m identical parallel machines, $P|r_j, pmtn|\sum w_j C_j$. As prediction we assume a single permutation $\hat{\sigma}$ over all jobs, i.e., we do not require a machine assignment.

Prediction-clairvoyant algorithm. Consider the *preemptive WSPT (P-WSPT)* algorithm that schedules, at any moment in time, the m available jobs with the highest priority in the predicted order $\hat{\sigma}$. Assuming $\hat{\sigma}$ is a perfect prediction and gives the true WSPT order, P-WSPT is known to be 2-competitive [42]. Further notice that, for a fixed permutation prediction, smaller processing requirements will not increase the objective value of this algorithm. Thus, it is monotone. We show the following error-dependence.

Lemma 3.7. *The prediction-clairvoyant P-WSPT algorithm is (η^S/m)-error-dependent.*

Proof. Consider an instance J with jobs being indexed by σ, a prediction $\hat{\sigma}$, and the schedule obtained by the prediction-clairvoyant P-WSPT. After job j has been released, it is either being processed on a machine or it is delayed by another job. Let $d(j', j)$ denote the total amount of job j' that delays the completion of j. Note that $d(j', j) \leq p_{j'}$. Such a delay can only occur if there are at least m alive jobs before j in $\hat{\sigma}$, and these jobs will be distributed over all m machines. Since j has received p_j units of processing by its completion time, we conclude

$$\begin{aligned}
&\textsc{Alg}(J, \hat{\sigma}) \\
&\leq \sum_{j=1}^{n} w_j (r_j + p_j) + \frac{1}{m} \sum_{j=1}^{n} \sum_{j'=1}^{j-1} \left(w_j \cdot d(j', j) + w_{j'} \cdot d(j, j') \right) \\
&\leq \textsc{Opt}(J) + \frac{1}{m} \sum_{j=1}^{n} \sum_{\substack{j'=1 \\ \hat{\sigma}(j') < \hat{\sigma}(j)}}^{j-1} w_j p_{j'} + \frac{1}{m} \sum_{j=1}^{n} \sum_{\substack{j'=1 \\ \hat{\sigma}(j') > \hat{\sigma}(j)}}^{j-1} w_{j'} p_j \\
&= \textsc{Opt}(J) + \frac{1}{m} \sum_{j=1}^{n} w_j \sum_{j'=1}^{j-1} p_{j'} + \frac{1}{m} \sum_{j=1}^{n} \sum_{\substack{j'=1 \\ \hat{\sigma}(j') > \hat{\sigma}(j)}}^{j-1} (w_{j'} p_j - w_j p_{j'}) \\
&\leq 2 \cdot \textsc{Opt}(J) + \frac{1}{m} \cdot \eta^S(J, \hat{\sigma}).
\end{aligned}$$

The second and third inequality hold due to two classical lower bounds on an optimal solution: Every job has to be processed by at least its p_j after its release in any solution. And $\frac{1}{m} \sum_{j=1}^{n} w_j \sum_{j'=1}^{j-1} p_{j'}$ equals the objective value of the WSPT schedule on a single machine with speed m without release dates, which is a known relaxation of our problem and therefore also a lower bound on $\textsc{Opt}(J)$. Since we assumed that the jobs are indexed according to σ, the sum of inversions is equal to $\eta^S(J, \hat{\sigma})$ by Definition 2.1. □

Non-clairvoyant algorithm. Beaumont et el. [15] analyzed a natural extension of the WRR algorithm [34] to identical parallel machines and prove the same competitive ratio of 2 for non-clairvoyant $P|pmtn|\sum w_j C_j$. Like WRR, their algorithm *Weighted Dynamic EQuipartition* (WDEQ) assigns processing rates to jobs proportional to their weights making sure that no job receives a higher rate than executable on one machine simultaneously.

When release dates are present, it is not hard to prove that WDEQ has a competitive ratio of at most 3. This result might be folkloric. To see it, consider the schedules S and S' of WDEQ with and without release dates for the same job set. Let C_j resp. C'_j be the completion time of job j in S resp. S'. Notice that the total sum of rates job j receives in the interval $[r_j, C_j]$ in S is not more than in the interval $[0, C'_j]$ in S'. This is because the total weight of other jobs running during $[r_j, C_j]$ in S cannot be higher compared to the case when all jobs are released at the same time, which is the case in S'. Thus, $[r_j, C_j]$ is not longer than $[0, C'_j]$, giving $C_j \leq r_j + C'_j$. The facts that $\sum_j w_j r_j$ is a lower bound on the optimal objective value with release dates and $\sum_j w_j C'_j$ is at most twice the optimal objective value without release dates [15] imply that WDEQ has a competitive ratio of at most 3 for $P|r_j, pmtn|\sum w_j C_J$.

Note that WDEQ is monotone as shrinking a job only decreases its completion time and thus gives other jobs more rate earlier, which also decreases their completion times.

Lemma 3.8. *WDEQ is a monotone* 3*-competitive algorithm for the non-clairvoyant version of* $\mathrm{P}|r_j, pmtn|\sum w_jC_j$.

By Corollary 3.4 we conclude with the following result.

Theorem 3.9. *Preferential Time Sharing with the prediction-clairvoyant P-WSPT algorithm and the non-clairvoyant WDEQ algorithm has, for every* $\lambda \in (0,1)$*, a competitive ratio of*

$$\min\left\{\frac{1}{1-\lambda}\left(2+\frac{\eta^S}{m\cdot \textsc{Opt}}\right), \frac{3}{\lambda}\right\}$$

for non-clairvoyant scheduling with predictions on m identical parallel machines, $\mathrm{P}|r_j, pmtn|\sum w_jC_j$.

3.3 Unrelated machines

We consider our most general non-clairvoyant scheduling problem, preemptive scheduling of weighted jobs on unrelated machines $\mathrm{R}|r_j, pmtn|\sum w_jC_j$, with predictions. Recall that we are given a predicted permutation $\hat{\sigma}_i$ for each machine $i \in [m]$ including a predicted machine allocation $m(\hat{\sigma}, j)$ for each job j.

Prediction-clairvoyant algorithm. The best known algorithm for clairvoyant scheduling $\mathrm{R}|r_j, pmtn|\sum w_jC_j$ by Bienkowski et al. [16] has a competitive ratio of 3. It uses a guess-and-double framework and processing times; it is unclear how to run it based on permutation predictions and how to track its error-dependence.

Other clairvoyant algorithms where proposed (for different problems) [3, 26, 32, 43] that use a greedy strategy for assigning jobs to machines in the following way. Assuming a fixed single-machine rule Π, they assign a newly arriving job to the machine where it causes the (approximately) minimum increase in the objective value, assuming that jobs on each machine are scheduled according to Π. We refer to such algorithm as *MinIncrease* Π.

While these algorithms are similar in flavor, none of the existing results proven in the literature seems to directly match our purpose w.r.t. the precise scheduling model and the possibility for proving an error-sensitivity.

Most promising seems a result for minimizing the total weighted flow time on unrelated machines, where the flow time of a job j is defined as $C_j - r_j$. Anand et al. [3] use the *Weighted Shortest Remaining Processing Time first (WSRPT)* rule as single-machine algorithm Π, which schedules, at any time t, an available job with largest residual density $w_j/p_j(t)$. For the (simpler) objective of minimizing the weighted completion time, WSRPT is known to be 2-competitive on a single machine with release dates [41]. A straightforward adaption of the analysis in [3] shows that MinIncrease WSRPT is 8-competitive for our clairvoyant problem. A more careful analysis even proves a competitive ratio of at most 4 [32]. However, it is unclear how to turn this algorithm into a prediction-clairvoyant algorithm in our setting. While the machine assignment is given, we do not have information about (remaining) processing times to apply WSRPT. Further, it is unclear how to obtain an error-dependency for our permutation prediction model, as the order of the jobs given by WSRPT changes when jobs are processed.

Nevertheless, we take inspiration from the analysis, replace WSRPT by preemptive WSPT and adopt the MinIncrease P-WSPT algorithm for our framework. We first prove that the clairvoyant MinIncrease P-WSPT algorithm is at most 5.8284-competitive using a dual-fitting analysis borrowing different ideas from [3, 26, 32]. Without release dates, our algorithm is essentially the same as the algorithms in [3, 26, 32] and a lower bound of 4 is known [17, 26]. We also prove an error-dependent competitive ratio for its prediction-clairvoyant version with release dates.

Theorem 3.10. *The MinIncrease P-WSPT algorithm has a competitive ratio of at most* $3+2\sqrt{2} \approx 5.8284$ *for clairvoyant scheduling on unrelated machines,* $\mathrm{R}|r_j, pmtn|\sum w_jC_j$.

In the following we denote the MinIncrease P-WSPT algorithm by $\mathcal{A}$. Fix an instance J and let $s > 1$ be a real number that we will fix later. We assume w.l.o.g. by scaling the instance that all processing requirements and release dates in J are integer multiples of s.

Let $M_i(j)$ be the set of available jobs that are assigned to machine i at time r_j, excluding job j. As this definition is ambiguous if there are two jobs j and j' with $r_j = r_{j'}$ being assigned to i, we assume that we assign them in the order of their index. By defining $\mu_{ij} = w_j/p_{ij}$, the increase of the objective value of $\mathcal{A}$ due to assigning job j to machine i at time r_j equals

$$Q_{ij} = w_j\left(r_j + p_{ij} + \sum_{\substack{j'\in M_i(j)\\ \mu_{ij'}\geq\mu_{ij}}} p_{ij'}(r_j)\right) + p_{ij}\sum_{\substack{j'\in M_i(j)\\ \mu_{ij'}<\mu_{ij}}} w_{j'}.$$

Then, algorithm $\mathcal{A}$ assigns job j to machine $g(j) = \arg\min_i Q_{ij}$.

The following linear program is a relaxation of our scheduling problem [3, 26, 32]. The variable x_{ijt} denotes the fractional assignment of job j to machine i at time t.

$$\begin{aligned}
\min\quad & \sum_{i,j,t} w_j\cdot\left(\frac{x_{ijt}}{2}+\frac{x_{ijt}}{p_{ij}}\cdot\left(t+\frac{1}{2}\right)\right) && && \text{(LP)}\\
& \sum_{i,t\geq r_j}\frac{x_{ijt}}{p_{ij}} \geq 1 && \forall j\\
& \sum_j x_{ijt} \leq 1 && \forall i,t\\
& x_{ijt} \geq 0 && \forall i,j,t\\
& x_{ijt} = 0 && \forall i,j,t<r_j
\end{aligned}$$

The dual of (LP) is equal to the following linear program with variables a_j and b_{it}.

$$\begin{aligned}
\max\quad & \sum_j a_j - \sum_{i,t} b_{it} && && \text{(DLP)}\\
& \frac{a_j}{p_{ij}} \leq b_{it} + w_j\cdot\left(\frac{t+1/2}{p_{ij}}+\frac{1}{2}\right) && \forall i,j,t\geq r_j && (1)\\
& a_j, b_{it} \geq 0 && \forall i,j,t
\end{aligned}$$

We define a solution of (DLP) for instance J which depends on the schedule produced by algorithm $\mathcal{A}$ for J. Let $U_i(t) = \{j \in J \mid g(j) = i \wedge t < C_j\}$, where C_j denotes the completion time of job j in the schedule of $\mathcal{A}$ for instance J. Note that $U_i(t)$ includes unreleased jobs. Consider the following assignment:

- $\hat{a}_j = Q_{g(j)j}$ for every job j and

- $\hat{b}_{it} = \sum_{j \in U_i(s \cdot t)} w_j$ for every machine i and time t.

We first show that the objective value of (DLP) for the solution $(\hat{a}_j, \hat{b}_{it})$ is close to the objective value of $\mathcal{A}$ w.r.t. s.

Lemma 3.11. $\sum_j \hat{a}_j - \sum_{i,t} \hat{b}_{it} = \left(1 - \frac{1}{s}\right) \cdot \mathcal{A}(J)$.

Proof. The definition of $Q_{g(j)j}$ implies $\sum_j \hat{a}_j = \sum_j Q_{g(j)j} = \mathcal{A}(J)$. Since we assumed that all release dates and processing times in J are integer multiples of s, all preemptions occur at integer multiples of s and therefore also all job completions. Thus, $\sum_t \sum_{j \in U_i(s \cdot t)} w_j = \frac{1}{s} \sum_t \sum_{j \in U_i(t)} w_j$ for every machine i, and

$$\sum_{i,t} \hat{b}_{it} = \sum_{i,t} \sum_{j \in U_i(s \cdot t)} w_j = \frac{1}{s} \sum_{i,t} \sum_{j \in U_i(t)} w_j = \frac{1}{s} \cdot \mathcal{A}(J),$$

which implies the desired equality. □

Second, we show that scaling the defined variables makes them feasible for (DLP).

Lemma 3.12. *Assigning $a_j = \hat{a}_j/(s+1)$ and $b_{it} = \hat{b}_{it}/(s+1)$ gives a feasible solution for* (DLP).

Proof. Since our defined variables are non-negative by definition, it suffices to show that this assignment satisfies (1). Fix a job j, a machine i and a time $t \geq r_j$. We assume that no new job arrives after j, since such a job may only increase $\hat{b}_{it}$ while $\hat{a}_j$ stays unchanged. Let $j_1, \ldots, j_z$ be the jobs of $M_i(j)$ indexed in WSPT order by densities $\mu_{ij} = w_j/p_{ij}$. Defining

- $H = \{j' \in M_i(j) : \mu_{ij'} \geq \mu_{ij}\} = \{j_1, \ldots, j_r\}$ and
- $L = \{j' \in M_i(j) : \mu_{ij'} < \mu_{ij}\} = \{j_{r+1}, \ldots, j_z\}$,

and using $\hat{a}_j = Q_{g(j)j} \leq Q_{ij}$ and $s + 1 > 2$ yields

$$\frac{a_j}{p_{ij}} = \frac{\hat{a}_j}{(s+1)p_{ij}} \leq \frac{\mu_{ij}}{s+1}\left(r_j + \sum_{j' \in H} p_{ij'}(r_j)\right) + \frac{w_j}{2} + \sum_{j' \in L} \frac{w_{j'}}{s+1}.$$

Thus, asserting (1) reduces to proving

$$\frac{\mu_{ij}}{s+1}\left(r_j + \sum_{j' \in H} p_{ij'}(r_j)\right) + \sum_{j' \in L} \frac{w_{j'}}{s+1} \leq \mu_{ij} t + b_{it}. \tag{2}$$

Observe that the total processing time of all jobs in $M_i(j)$ that are completed before time $s \cdot t$ is at most $s \cdot t$. Further, $r_j + s \cdot t \leq (s+1)t$. Now consider the case that machine i processes a job j_k at time $s \cdot t$. If $j_k \in H$, using $\mu_{ij} \leq \frac{w_{j_\ell}}{p_{ij_\ell}} \leq \frac{w_{j_\ell}}{p_{ij_\ell}(r_j)}$ for all $j_\ell \in H$ gives

$$\begin{aligned}
&\frac{\mu_{ij}}{s+1}\left(r_j + \sum_{\ell=1}^{k-1} p_{ij_\ell}(r_j)\right) + \frac{\mu_{ij}}{s+1} \sum_{\ell=k}^{r} p_{ij_\ell}(r_j) + \sum_{j' \in L} \frac{w_{j'}}{s+1} \\
&\leq \mu_{ij} t + \frac{1}{s+1} \sum_{\ell=k}^{r} w_{j_\ell} + \sum_{j' \in L} \frac{w_{j'}}{s+1} \leq \mu_{ij} t + \frac{\hat{b}_{it}}{s+1} = \mu_{ij} t + b_{it}.
\end{aligned}$$

The last inequality holds since all jobs in $M_i(j)$ that are processed after job j_{k-1} are unfinished at time $s \cdot t$ and assigned to i in $\mathcal{A}'s$ schedule, hence part of $U_i(s \cdot t)$. If $j_k \in L$, using $w_{j_\ell} < \mu_{ij} \cdot p_{ij_\ell}$ for all $j_\ell \in L$ implies

$$\begin{aligned}
&\frac{\mu_{ij}}{s+1}\left(r_j + \sum_{\ell=1}^{r} p_{ij_\ell}(r_j)\right) + \frac{1}{s+1} \sum_{\ell=r+1}^{k-1} w_{j_\ell} + \frac{1}{s+1} \sum_{\ell=k}^{z} w_{j_\ell} \\
&\leq \frac{\mu_{ij}}{s+1}\left(r_j + \sum_{\ell=1}^{r} p_{ij_\ell}(r_j) + \sum_{\ell=r+1}^{k-1} p_{ij_\ell}\right) + \frac{1}{s+1} \sum_{\ell=k}^{z} w_{j_\ell} \\
&\leq \mu_{ij} t + \frac{1}{s+1} \sum_{\ell=k}^{z} w_{j_\ell} \leq \mu_{ij} t + \frac{\hat{b}_{it}}{s+1} = \mu_{ij} t + b_{it}.
\end{aligned}$$

If no job is running at time $s \cdot t$, we conclude that all jobs in $M_i(j)$ must already be completed, because algorithm $\mathcal{A}$ does not idle unnecessarily, and we assumed that no job is released after j. By using $w_{j_\ell} < \mu_{ij} \cdot p_{ij_\ell}$ for all $j_\ell \in L$ we assert (2) for this final case

$$\begin{aligned}
&\frac{\mu_{ij}}{s+1}\left(r_j + \sum_{\ell=1}^{r} p_{ij_\ell}(r_j)\right) + \sum_{\ell=r+1}^{z} w_{j_\ell} \\
&\leq \frac{\mu_{ij}}{s+1}\left(r_j + \sum_{\ell=1}^{r} p_{ij_\ell}(r_j) + \sum_{\ell=r+1}^{z} p_{ij_\ell}\right) \leq \mu_{ij} t.
\end{aligned}$$

□

Proof of Theorem 3.10. Weak duality and Lemma 3.12 imply that the objective value of (DLP) for the assigned variables is a lower bound on the optimal objective value. Lemma 3.11 gives

$$\begin{aligned}
\textsc{Opt}(J) &\geq \sum_{j \in J} a_j - \sum_{i,t} b_{it} = \sum_{j \in J} \frac{\hat{a}_j}{s+1} - \sum_{i,t} \frac{\hat{b}_{it}}{s+1} \\
&= \frac{1}{s+1}\left(\sum_{j \in J} \hat{a}_j - \sum_{i,t} \hat{b}_{it}\right) = \left(\frac{1 - 1/s}{s+1}\right) \cdot \mathcal{A}(J).
\end{aligned}$$

We conclude that algorithm $\mathcal{A}$ has a competitive ratio of at most $3 + 2\sqrt{2} \approx 5.8284$ for the optimal choice $s = 1 + \sqrt{2}$. □

We now consider the prediction-clairvoyant version of the MinIncrease P-WSPT algorithm. It assigns the jobs to machines according to the predicted assignment $\{\hat{\sigma}_i\}_{i \in [m]}$. At any time and for every machine i it schedules the available job with highest priority according to $\hat{\sigma}_i$. This algorithm is monotone, because shrinking a job's processing requirements does not affect $\{\hat{\sigma}_i\}_{i \in [m]}$ and thus may only decrease the completion time of jobs that are scheduled after this job on the assigned machine.

Lemma 3.13. *The prediction-clairvoyant MinIncrease P-WSPT algorithm is η^R-error-dependent.*

Proof. Consider job set J. Scheduling a job j according to a prediction $\hat{\sigma}$ contributes a value equal to $W_j(J, \hat{\sigma})$ to the objective of our algorithm $\textsc{Alg}(J, \hat{\sigma})$. Thus, $\textsc{Alg}(J, \hat{\sigma}) = \sum_j W_j(J, \hat{\sigma})$. Since the machine assignment of the clairvoyant MinIncrease P-WSPT algorithm $\mathcal{A}$ can be encoded into a prediction that orders the jobs by WSPT on every machine, the cost of following σ is a lower bound on the objective value of $\mathcal{A}$, i.e. $\sum_j W_j(J, \sigma) \leq \mathcal{A}(J)$, or $-\mathcal{A}(J) \leq -\sum_j W_j(J, \sigma)$. We conclude using Theorem 3.10 that our

algorithm is η^R-error-dependent, since

$$\begin{aligned}\textsc{Alg}(J,\hat{\sigma}) &= \mathcal{A}(J) - \mathcal{A}(J) + \sum_{j\in J} W_j(J,\hat{\sigma}) \\ &\leq \mathcal{A}(J) + \sum_{j} W_j(J,\hat{\sigma}) - \sum_{j\in J} W_j(J,\sigma) \\ &= \mathcal{A}(J) + \eta^R(J,\hat{\sigma}) \leq 5.8284 \cdot \textsc{Opt}(J) + \eta^R(J,\hat{\sigma}). \quad \square\end{aligned}$$

Non-clairvoyant algorithm. Im, Kulkarni and Munagala [28] show that the Proportional Fairness (PF) algorithm is 128-competitive. They actually state a smaller competitive ratio of 64 in [28, Theorem 1.2, Page 16] but it seems to miss a factor of 2 as we argue briefly in Appendix A. A similar argumentation as for WRR and WDEQ shows that this algorithm is monotone.

Lemma 3.14 ([28], Appendix A). *The Proportional Fairness algorithm is* 128*-competitive for non-clairvoyant scheduling on unrelated machines,* $\mathrm{R}|r_j, pmtn|\sum_j w_j C_j$.

By Corollary 3.4 we conclude with the following result.

Theorem 3.15. *Preferential Time Sharing with the prediction-clairvoyant MinIncrease P-WSPT algorithm and the non-clairvoyant Proportional Fairness algorithm has, for every* $\lambda \in (0,1)$*, a competitive ratio of at most*

$$\min\left\{\frac{1}{1-\lambda}\left(5.8284 + \frac{\eta^R}{\textsc{Opt}}\right), \frac{128}{\lambda}\right\}$$

for non-clairvoyant scheduling with predictions on unrelated machines, $\mathrm{R}|r_j, pmtn|\sum_j w_j C_j$.

4 LEARNABILITY OF PERMUTATIONS

We show that permutation predictions for identical machines are PAC-learnable in the agnostic sense w.r.t. η^S.

Theorem 4.1. *For any* $\epsilon, \delta \in (0,1)$ *and any distribution* $\mathcal{D}$ *over the instances of length n, there exists a learning algorithm which, given an i.i.d. sample of* $\mathcal{D}$ *of size* $z \in O\left(\frac{1}{\epsilon^2}\cdot(n\log n - \log\delta)n^2\right)$*, returns in polynomial time depending on n and z a prediction* $\hat{\sigma}_p \in \mathcal{H}$ *from the set of all possible permutations of the set* $\{1,\dots,n\}$*, such that with probability of at least* $(1-\delta)$ *it holds* $\mathbb{E}_{J\sim\mathcal{D}}[\eta^S(J,\hat{\sigma}_p)] \leq \mathbb{E}_{J\sim\mathcal{D}}[\eta^S(J,\sigma)] + \epsilon$*, where* $\eta^S(J,\hat{\sigma})$ *denotes the error of* $\hat{\sigma}$ *for instance J, and* $\sigma = \arg\min_{\hat{\sigma}\in\mathcal{H}} \mathbb{E}_{J\sim\mathcal{D}}[\eta^S(J,\hat{\sigma})]$.

Proof. Let $\epsilon, \delta \in (0,1)$. We prove that we can use the classic Empirical Risk Minimization (ERM) learning method to find such a prediction. Let $\mathcal{S} = \{J_1,\dots,J_z\}$ be a set of i.i.d. samples from $\mathcal{D}$. The ERM method then determines the prediction that minimizes the empirical error $\eta^S_{\mathcal{S}}(\hat{\sigma}) = \frac{1}{z}\sum_{s=1}^{z}\eta^S(J_s,\hat{\sigma})$. Since there are $n!$ possible permutations of the set $\{1,\dots,n\}$, we conclude that $\mathcal{H}$ is finite, and we can assume by scaling processing requirements and weights to $[0,1]$ that our error function is bounded by n. Classic results, see e.g. [52], imply for this case that $\mathcal{H}$ is agnostically PAC learnable using the ERM method with sample complexity

$$z \leq \left\lceil \frac{2\log(2|\mathcal{H}|/\delta)n^2}{\epsilon^2}\right\rceil \in O\left(\frac{(n\log n - \log\delta)n^2}{\epsilon^2}\right),$$

which is polynomial in the number of jobs, n, as $\log n! \in O(n\log n)$.

It remains to prove that the ERM algorithm can be implemented efficiently in our setting, that is, given a sample set of size z, determine in time polynomial in n, a prediction that minimizes the empirical error. Rewriting the empirical error gives

$$\eta^S_{\mathcal{S}}(\hat{\sigma}) = \frac{1}{z}\sum_{s=1}^{z}\eta^S(J_s,\hat{\sigma}) = \frac{1}{z}\sum_{s=1}^{z}\sum_{j=1}^{n}\left(W_j(J_s,\hat{\sigma}) - W_j(J_s,\sigma)\right).$$

Since the values $W_j(J_s,\sigma)$ are independent of $\hat{\sigma}$, it suffices to find a prediction $\hat{\sigma}$ that minimizes $\frac{1}{z}\sum_{s=1}^{z}\sum_{j=1}^{n} W_j(J_s,\hat{\sigma})$. For the special error η^S, by denoting for a job $j \in J_s$ its weight by $w_j^{(s)}$ and its processing requirement by $p_j^{(s)}$, this is equal to

$$\begin{aligned}\frac{1}{z}\sum_{s=1}^{z}\sum_{j=1}^{n} W_j(J_s,\hat{\sigma}) &= \frac{1}{z}\sum_{s=1}^{z}\sum_{j=1}^{n} w^{(s)}_{\hat{\sigma}(j)}\sum_{\ell=1}^{j} p^{(s)}_{\hat{\sigma}(\ell)} \\ &= \sum_{j=1}^{n}\left(\frac{1}{z}\sum_{s=1}^{z} w^{(s)}_{\hat{\sigma}(j)}\right)\sum_{\ell=1}^{j}\left(\frac{1}{z}\sum_{s=1}^{z} p^{(s)}_{\hat{\sigma}(\ell)}\right).\end{aligned}$$

By defining the average weight $\bar{w}_{\hat{\sigma}(j)} = \frac{1}{z}\sum_{s=1}^{z} w^{(s)}_{\hat{\sigma}(j)}$ and average processing requirement $\bar{p}_{\hat{\sigma}(j)} = \frac{1}{z}\sum_{s=1}^{z} p^{(s)}_{\hat{\sigma}(j)}$ over $\mathcal{S}$ for all $j \in [n]$, this is equal to minimizing

$$\sum_{j=1}^{n}\bar{w}_{\hat{\sigma}(j)}\sum_{\ell=1}^{j}\bar{p}_{\hat{\sigma}(\ell)}.$$

Consider the *average* instance of $\mathcal{S}$, i.e. the scheduling instance of n jobs with weights $\{\bar{w}_j\}_{j\in[n]}$ and processing requirements $\{\bar{p}_j\}_{j\in[n]}$. Since the above expression is equal to the objective value of this instance when scheduling jobs in order $\hat{\sigma}(1),\dots,\hat{\sigma}(n)$, we can minimize it by ordering the jobs according to WSPT in polynomial time in z and n [53]. $\square$

The space of permutation predictions with predicted machine assignments $\{\hat{\sigma}_i\}_{i\in[m]}$ is also finite and we can use similar arguments to prove that they are agnostically PAC-learnable with respect to η^R with bounded sample complexity. This implies that ERM minimizes the empirical error. However, it is not clear how to achieve this with polynomial running time in n, m and the number of samples z. Yet one can approximately minimize the empirical error by computing an approximately perfect prediction using the MinIncrease P-WSPT algorithm on the average instance of $\mathcal{S}$.

5 EXPERIMENTS

In empirical experiments we demonstrate the practicability of our approach in comparison to the previously proposed learning-augmented algorithms by Im et al. [30] and Wei and Zhang [56]. These algorithms consider the *single-machine* problem without weights and release dates, $1|pmtn|\sum C_j$. Notice that in this setting the Preferential Time Sharing (PTS) algorithm and the Preferential-Round-Robin (PRR) algorithm of Kumar et al. [48] are equivalent. The only difference is the theoretically different prediction model. However, since all previous algorithms use the length prediction model, we compute permutation predictions based on predicted processing

times. The results for the single machine setting are given in Section 5.1. We further give experimental results for PTS for scheduling weighted jobs with release dates on parallel identical machines in Section 5.2. But first we describe the instance generation and experiment setups.

Dataset. We generate synthetic instances. Each instance is composed of 1000 jobs. We choose this size as a compromise between computational effort and giving the algorithms enough jobs to work properly. The processing requirements for the jobs are individually sampled from a Pareto distribution with scale 1 and shape 1.1. This distribution was used in the seminal work on learning-augmented scheduling [48] and is (similar to the related Zipf distribution) generally considered to model scheduling applications very well [1, 14, 18, 21, 27, 31, 35]. Intuitively, it gives many tiny jobs and few very large jobs. We also performed our experiments with processing requirements sampled from an exponential distribution with mean 1 as well as from a Weibull distribution with scale 2 and shape 0.5, which were used in [44, 45].

Types of experiments. We perform two types of experiments. In *sensitivity experiments*, we generate length predictions by adding Gaussian noise to the processing requirements with an increasing standard deviation ω for a fixed instance. This type of experiment was also performed by Kumar et al. [48] to evaluate PRR as well as in other works on learning-augmented algorithms [4, 5, 39, 40].

In *online learning experiments* we first fix a synthetic instance, called base instance. Then, we consider 10 subsequent rounds, where in every round t an instance J_t arrives, which is generated by adding independently sampled Gaussian noise to the base instance. To calculate this noise we use scaled standard deviations parameterized by a factor $\gamma \geq 0$. That is, we compute noise for true processing requirement p with a standard deviation equal to $\gamma \cdot \sqrt{p}$. We feel that this is more realistic for this type of experiment than only using a fixed standard deviation for all jobs, as small jobs may vary less than large jobs over time. We then compute a prediction for round t using the ERM algorithm on the set of previous instances $\{J_0, \ldots, J_{t-1}\}$, as these are in round t known to the algorithms. As length prediction for round 0 we use an independently sampled random instance. This type of experiment was also performed in [19] to demonstrate the speedup of predictions for the bipartite matching problem.

5.1 Experiments for a single machine

Algorithms. We present implementation details of the considered algorithms. As online benchmark (without predictions), we use the best-possible non-clairvoyant algorithm Round-Robin (RR) [46].

TwoStage (algorithm by Wei and Zhang [56]) executes RR until a certain time point depending on the predicted processing requirements and the confidence parameter $\lambda \in [0, 1]$. Then, it schedules the jobs in non-decreasing order of their predicted processing requirements. If at any time a job finishes before or after their predicted length, it finishes the remaining instance with RR. This algorithm achieves for instances with at most two jobs a consistency-robustness tradeoff that matches a lower bound shown in [56].

MultiStage (algorithm by Im et al. [30]), works in phases and decides whether to follow the prediction or to execute RR by tracking

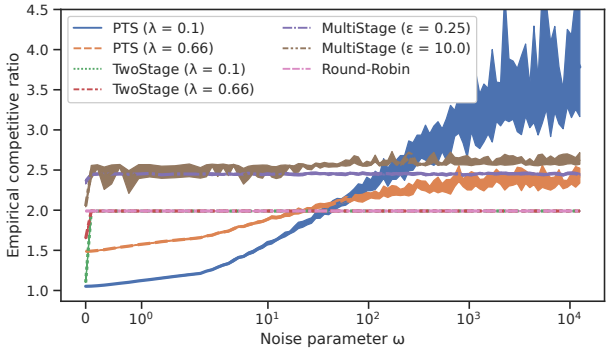

Figure 1: Sensitivity experiment.

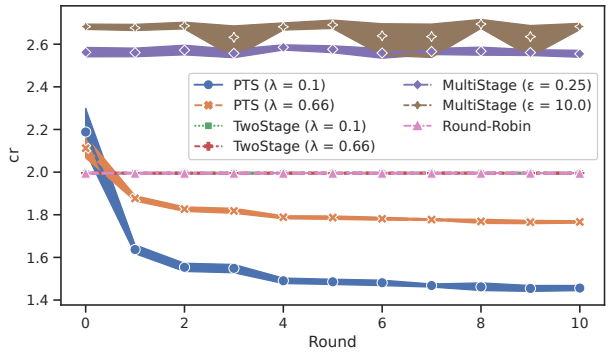

Figure 2: Online learning experiment, $\gamma = 10$. Note that the plots of TwoStage coincide with the plot of Round-Robin.

the quality of the prediction. This is done by processing and computing the error of small random samples, which is then projected to the whole set of remaining jobs. We implemented a basic variant of this algorithm, which is $O(1)$-robust and $(1+\epsilon)$-consistent for any $\epsilon > 0$ with high probability under some assumptions (Corollary 34 in [30]). Our implementation uses base two for unspecified logarithms. A consequence of this choice is that if $\epsilon < 0.215$, MultiStage executes solely RR on our instances. Therefore, we performed the experiments with rather large $\epsilon = 0.25$ and $\epsilon = 10.0$. The authors of [30] also give a modification of this algorithm which achieves bounds in expectation. We omitted the implementation of this variant as it requires a further parallel execution of RR which makes the calculation of precise completion times very difficult.

Results. For every parameter setting we perform 10 runs and measure the performance of the algorithms for this setting in terms of *empirical competitive ratio*. That is the average objective value of an algorithm over all runs divided by the optimal objective value for the instance. We further report error bars that denote the 95% confidence interval of the runs.

In the following we discuss results for Pareto-distributed processing requirements. For the other considered distributions, we observed very similar results, where in the online learning experiment we use varied noise parameters due to different job characteristics.

We first discuss results of the sensitivity experiment, which are visualized in Figure 1. For the consistency case ($\omega = 0$) the algorithms achieve their best performance, as expected. However, even for very small noise ($\omega = 0.1$), we observe that TwoStage and MultiStage experience drastic performance losses compared to having access to precise predictions. This behavior is explainable by the design of the algorithms, which switch their execution to the robust fallback procedure RR when detecting incorrect predictions. While TwoStage stays in this mode until the instance completes, MultiStage still estimates medians and errors, incurring an additional overhead. While the performance of PTS smoothly degrades for larger noise depending on λ, it still outperforms RR until $\omega \approx 20$. For very large noise, the performance of TwoStage and MultiStage stays unchanged, while PTS with $\lambda = 0.1$ still grows. For larger values of λ, e.g. $\lambda = 0.66$ as in the figure, PTS shows a constantly superior performance than MultiStage and, w.r.t. TwoStage, a smoother performance with substantially better consistency and only slightly larger robustness.

In the online learning experiment (Figure 2), TwoStage and MultiStage do not improve their performance over RR by using predictions. We suspect that this is again due to the fact that the prediction is still too erroneous over the first ten rounds to activate their trustful subroutines. We performed these experiments also with 100 rounds, but did not observe a significant difference. While in round 0 without any prediction PTS performs slightly worse than the other algorithms, it improves over RR already after seeing one sample. This shows that in our setup one sample is enough to approximately distinguish small jobs from large jobs, and this classification is enough to prevent large jobs from delaying the completion of many small jobs. This also demonstrates that permutation predictions capture the relevant information of practical instances.

5.2 Experiments for multiple machines

We generate 10 synthetic instances with 1000 jobs each. Processing requirements are again sampled from a Pareto-distribution with shape 1.1 and scale 1, weights and release dates from a Pareto-distribution with shape 2 and scale 1. We implement PTS according to Theorem 3.9 and compare it to the non-clairvoyant WDEQ algorithm [15]. To compute empirical competitive ratios and error bars, we use the objective value of the clairvoyant, 2-competitive P-WSPT algorithm [42] as baseline. The results of the sensitivity experiment for five machines (Figure 3) show that for small noise PTS outperforms WDEQ. For growing noise the performance of PTS slowly degrades, but still improves upon WDEQ until $\omega \approx 35$. For large values of ω, the empirical competitive ratio of PTS with $\lambda = 0.1$ continues growing, while for $\lambda = 0.5$ and $\lambda = 0.8$ the ratios quickly converge to their robustness bounds.

6 CONCLUSION

In this paper we proposed a new compact prediction model and error measure which fulfill desired properties in theory and practice. We revisited a learning-augmented time sharing framework, generalized it, and derived the first results for more complex scheduling problems with weights, release dates and multiple machines.

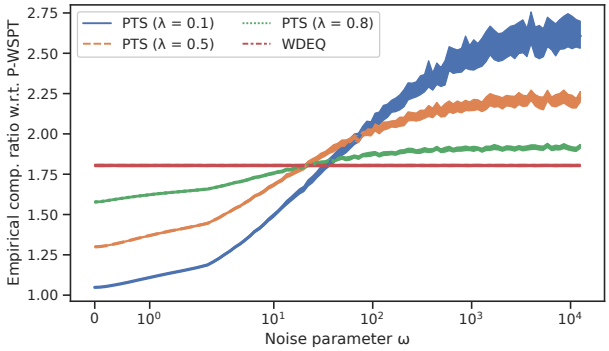

Figure 3: Sensitivity experiment for five identical parallel machines.

It would be interesting whether better guarantees are possible by exploiting the fact that processing at a slower rate makes jobs "earlier" available, or by exploiting communication between combined algorithms, or by more adaptive algorithms.

ACKNOWLEDGMENTS

Partially supported by the German Science Foundation (DFG) under contracts ME 3825/1 and 146371743 – TRR 89 Invasive Computing.

REFERENCES

[1] Lada A. Adamic and Bernardo A. Huberman. 2002. Zipf's law and the Internet. *Glottometrics* 3 (2002), 143–150.
[2] Keerti Anand, Rong Ge, and Debmalya Panigrahi. 2020. Customizing ML Predictions for Online Algorithms. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 303–313.
[3] S. Anand, Naveen Garg, and Amit Kumar. 2012. Resource augmentation for weighted flow-time explained by dual fitting. In *SODA*. SIAM, 1228–1241.
[4] Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. 2020. Online metric algorithms with untrusted predictions. In *ICML (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 345–355.
[5] Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. 2021. Learning-Augmented Dynamic Power Management with Multiple States via New Ski Rental Bounds. In *NeurIPS*. 16714–16726.
[6] Antonios Antoniadis, Themis Gouleakis, Pieter Kleer, and Pavel Kolev. 2020. Secretary and Online Matching Problems with Machine Learned Advice. In *NeurIPS*. 7933–7944.
[7] Yossi Azar, Stefano Leonardi, and Noam Touitou. 2021. Flow time scheduling with uncertain processing time. In *STOC*. ACM, 1070–1080.
[8] Yossi Azar, Stefano Leonardi, and Noam Touitou. 2022. Distortion-Oblivious Algorithms for Minimizing Flow Time. In *SODA*. SIAM, 252–274.
[9] Yossi Azar, Debmalya Panigrahi, and Noam Touitou. 2022. Online Graph Algorithms with Predictions. In *SODA*. SIAM, 35–66.
[10] Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. 2020. Learning Augmented Energy Minimization via Speed Scaling. In *NeurIPS*. 15350–15359.
[11] Étienne Bamas, Andreas Maggiori, and Ola Svensson. 2020. The Primal-Dual method for Learning Augmented Algorithms. In *NeurIPS*. 20083–20094.
[12] Soumya Banerjee. 2020. Improving Online Rent-or-Buy Algorithms with Sequential Decision Making and ML Predictions. In *NeurIPS*. 21072–21080.
[13] Nikhil Bansal, Christian Coester, Ravi Kumar, Manish Purohit, and Erik Vee. 2022. Learning-Augmented Weighted Paging. In *SODA*. SIAM, 67–89.
[14] Nikhil Bansal and Mor Harchol-Balter. 2001. Analysis of SRPT scheduling: investigating unfairness. In *SIGMETRICS/Performance*. ACM, 279–290.
[15] Olivier Beaumont, Nicolas Bonichon, Lionel Eyraud-Dubois, and Loris Marchal. 2012. Minimizing Weighted Mean Completion Time for Malleable Tasks Scheduling. In *IPDPS*. IEEE Computer Society, 273–284.
[16] Marcin Bienkowski, Artur Kraska, and Hsiang-Hsuan Liu. 2021. Traveling Repairperson, Unrelated Machines, and Other Stories About Average Completion Times. In *ICALP (LIPIcs, Vol. 198)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 28:1–28:20.

[17] José R. Correa and Maurice Queyranne. 2012. Efficiency of equilibria in restricted uniform machine scheduling with total weighted completion time as social cost. *Naval Research Logistics (NRL)* 59, 5 (2012), 384–395.

[18] Mark Crovella and Azer Bestavros. 1997. Self-similarity in World Wide Web traffic: evidence and possible causes. *IEEE/ACM Trans. Netw.* 5, 6 (1997), 835–846.

[19] Michael Dinitz, Sungjin Im, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. 2021. Faster Matchings via Learned Duals. In *NeurIPS*. 10393–10406.

[20] Paul Dütting, Silvio Lattanzi, Renato Paes Leme, and Sergei Vassilvitskii. 2021. Secretaries with Advice. In *EC*. ACM, 409–429.

[21] David A. Easley and Jon M. Kleinberg. 2010. *Networks, Crowds, and Markets - Reasoning About a Highly Connected World*. Cambridge University Press.

[22] Franziska Eberle, Alexander Lindermayr, Nicole Megow, Lukas Nölke, and Jens Schlöter. 2021. Robustification of Online Graph Exploration Methods. *CoRR* abs/2112.05422 (2021).

[23] Yuval Emek, Shay Kutten, and Yangguang Shi. 2021. Online Paging with a Vanishing Regret. In *ITCS (LIPIcs, Vol. 185)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 67:1–67:20.

[24] Sreenivas Gollapudi and Debmalya Panigrahi. 2019. Online Algorithms for Rent-Or-Buy with Expert Advice. In *ICML (Proceedings of Machine Learning Research, Vol. 97)*. PMLR, 2319–2327.

[25] Anupam Gupta, Sungjin Im, Ravishankar Krishnaswamy, Benjamin Moseley, and Kirk Pruhs. 2012. Scheduling heterogeneous processors isn't as easy as you think. In *SODA*. SIAM, 1242–1253.

[26] Varun Gupta, Benjamin Moseley, Marc Uetz, and Qiaomin Xie. 2020. Greed Works - Online Algorithms for Unrelated Machine Stochastic Scheduling. *Math. Oper. Res.* 45, 2 (2020), 497–516.

[27] Mor Harchol-Balter and Allen B. Downey. 1997. Exploiting Process Lifetime Distributions for Dynamic Load Balancing. *ACM Trans. Comput. Syst.* 15, 3 (1997), 253–285.

[28] Sungjin Im, Janardhan Kulkarni, and Kamesh Munagala. 2018. Competitive Algorithms from Competitive Equilibria: Non-Clairvoyant Scheduling under Polyhedral Constraints. *J. ACM* 65, 1 (2018), 3:1–3:33.

[29] Sungjin Im, Janardhan Kulkarni, Kamesh Munagala, and Kirk Pruhs. 2014. SelfishMigrate: A Scalable Algorithm for Non-clairvoyantly Scheduling Heterogeneous Processors. In *FOCS*. IEEE Computer Society, 531–540.

[30] Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. 2021. Non-Clairvoyant Scheduling with Predictions. In *SPAA*. ACM, 285–294.

[31] Sungjin Im, Benjamin Moseley, and Kirk Pruhs. 2015. Stochastic Scheduling of Heavy-tailed Jobs. In *STACS (LIPIcs, Vol. 30)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 474–486.

[32] Sven Joachim Jäger. 2021. *Approximation in deterministic and stochastic machine scheduling*. Ph. D. Dissertation. Technical University of Berlin, Germany.

[33] Zhihao Jiang, Debmalya Panigrahi, and Kevin Sun. 2020. Online Algorithms for Weighted Paging with Predictions. In *ICALP (LIPIcs, Vol. 168)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 69:1–69:18.

[34] Jae-Hoon Kim and Kyung-Yong Chwa. 2003. Non-clairvoyant scheduling for weighted flow time. *Inf. Process. Lett.* 87, 1 (2003), 31–37.

[35] Balachander Krishnamurthy and Jennifer Rexford. 2001. *Web Protocols and Practice - HTTP/1.1, Networking Protocols, Caching, and Traffic Measurement*. Addison-Wesley.

[36] Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. 2020. Online Scheduling via Learned Weights. In *SODA*. SIAM, 1859–1877.

[37] Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. 2021. Learnable and Instance-Robust Predictions for Online Matching, Flows and Load Balancing. In *ESA (LIPIcs, Vol. 204)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 59:1–59:17.

[38] Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. 2021. Using Predicted Weights for Ad Delivery. In *ACDA*. SIAM, 21–31.

[39] Alexander Lindermayr, Nicole Megow, and Bertrand Simon. 2022. Double Coverage with Machine-Learned Advice. In *ITCS (LIPIcs, Vol. 215)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 99:1–99:18.

[40] Thodoris Lykouris and Sergei Vassilvitskii. 2018. Competitive Caching with Machine Learned Advice. In *ICML (Proceedings of Machine Learning Research, Vol. 80)*. PMLR, 3302–3311.

[41] Nicole Megow. 2007. Coping with Incomplete Information in Scheduling – Stochastic and Online Models. Dissertation 2006, Technische Universität Berlin. Cuvillier Göttingen.

[42] Nicole Megow and Andreas S. Schulz. 2004. On-line scheduling to minimize average completion time revisited. *Oper. Res. Lett.* 32, 5 (2004), 485–490.

[43] Nicole Megow, Marc Uetz, and Tjark Vredeveld. 2006. Models and Algorithms for Stochastic Online Scheduling. *Math. Oper. Res.* 31, 3 (2006), 513–525.

[44] Michael Mitzenmacher. 2020. Scheduling with Predictions and the Price of Misprediction. In *ITCS (LIPIcs, Vol. 151)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 14:1–14:18.

[45] Michael Mitzenmacher. 2021. Queues with Small Advice. In *ACDA*. SIAM, 1–12.

[46] Rajeev Motwani, Steven J. Phillips, and Eric Torng. 1994. Non-Clairvoyant Scheduling. *Theor. Comput. Sci.* 130, 1 (1994), 17–47.

[47] K.R. Pruhs, J. Sgall, and E. Torng. 2004. Online Scheduling. In *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Joseph Y-T. Leung (Ed.). Chapman & Hall/CRC, Chapter 15.

[48] Manish Purohit, Zoya Svitkina, and Ravi Kumar. 2018. Improving Online Algorithms via ML Predictions. In *NeurIPS*. 9684–9693.

[49] Dhruv Rohatgi. 2020. Near-Optimal Bounds for Online Caching with Machine Learned Advice. In *SODA*. SIAM, 1834–1845.

[50] Ziv Scully, Isaac Grosof, and Michael Mitzenmacher. 2022. Uniform Bounds for Scheduling with Job Size Estimates. In *ITCS (LIPIcs, Vol. 215)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 114:1–114:30.

[51] J. Sgall. 1998. On-line scheduling – a survey. In *Online Algorithms: The State of the Art*, Amos Fiat and Gerhard J. Woeginger (Eds.). LNCS, Vol. 1442. Springer, Berlin, 196–231.

[52] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding Machine Learning - From Theory to Algorithms*. Cambridge University Press.

[53] Wayne E Smith et al. 1956. Various optimizers for single-stage production. *Naval Research Logistics Quarterly* 3, 1-2 (1956), 59–66.

[54] Shufan Wang, Jian Li, and Shiqiang Wang. 2020. Online Algorithms for Multi-shop Ski Rental with Machine Learned Advice. In *NeurIPS*. 8150–8160.

[55] Alexander Wei. 2020. Better and Simpler Learning-Augmented Online Caching. In *APPROX-RANDOM (LIPIcs, Vol. 176)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 60:1–60:17.

[56] Alexander Wei and Fred Zhang. 2020. Optimal Robustness-Consistency Tradeoffs for Learning-Augmented Online Algorithms. In *NeurIPS*. 8042–8053.

[57] Chenyang Xu and Benjamin Moseley. 2021. Learning-Augmented Algorithms for Online Steiner Tree. *CoRR* abs/2112.05353 (2021).

A COMPETITIVE RATIO OF PF IN [28]

For the sake of completeness, we shortly argue why we use a different competitive ratio for the non-clairvoyant proportional fairness (PF) algorithm by Im, Kulkarni, Munagala than stated in their paper [28, Theorem 1.2].

On [28, page 16] the authors state that $\sum_j \alpha_j \geq (1/2) \sum_j w_j C_j^{\mathcal{A}}$ [28, Lemma 3.2] and $\sum_{d,t} \beta_{dt} \leq \frac{8}{s} \sum_j w_j C_j^{\mathcal{A}}$ [28, Corollary 3.5] imply that the dual objective value $\sum_j \alpha_j - \sum_{d,t} \beta_{dt}$ is at least half of PF's objective value when $s = 32$. However, combining both results gives that the dual objective value is at least a quarter of the algorithms objective value, since with $s = 32$,

$$\mathrm{DUAL}_s = \sum_j \alpha_j - \sum_{d,t} \beta_{dt} \geq \left(\frac{1}{2} - \frac{8}{s}\right) \sum_j w_j C_j^{\mathcal{A}} = \frac{1}{4} \sum_j w_j C_j^{\mathcal{A}}.$$

Indeed, from [28, Proposition 3.1], we conclude for any s that the algorithm has a competitive ratio of at most

$$\frac{s}{\frac{1}{2} - \frac{8}{s}},$$

which has a minimum value of $32 \cdot 4 = 128$ for $s = 32$.