Non-Clairvoyant Makespan Minimization Scheduling with Predictions

Evripidis Bampis
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Alexander Kononov
Sobolev Institute of Mathematics, Novosibirsk, Russia
Novosibirsk State University, Russia

Giorgio Lucarelli
LCOMS, University of Lorraine, Metz, France

Fanny Pascual
Sorbonne Université, CNRS, LIP6, F-75005 Paris, France

Abstract

We revisit the classical non-clairvoyant problem of scheduling a set of n jobs on a set of m parallel identical machines where the processing time of a job is not known until the job finishes. Our objective is the minimization of the makespan, i.e., the date at which the last job terminates its execution. We adopt the framework of learning-augmented algorithms and we study the question of whether (possibly erroneous) predictions may help design algorithms with a competitive ratio which is good when the prediction is accurate (consistency), deteriorates gradually with respect to the prediction error (smoothness), and not too bad and bounded when the prediction is arbitrarily bad (robustness). We first consider the non-preemptive case and we devise lower bounds, as a function of the error of the prediction, for any deterministic learning-augmented algorithm. Then we analyze a variant of Longest Processing Time first (LPT) algorithm (with and without release dates) and we prove that it is consistent, smooth, and robust. Furthermore, we study the preemptive case and we provide lower bounds for any deterministic algorithm with predictions as a function of the prediction error. Finally, we introduce a variant of the classical Round Robin algorithm (RR), the Predicted Proportional Round Robin algorithm ($PPRR$), which we prove to be consistent, smooth and robust.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases scheduling, online, learning-augmented algorithm

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.9

Funding *Evripidis Bampis*: This work was partially supported by the French National Research Agency (Algoridam ANR-19-CE48-0016).
Alexander Kononov: This research was carried out within the framework of the state contract of the Sobolev Institute of Mathematics (project FWNF-2022-0019).

1 Introduction

We consider the problem of scheduling a set of n jobs on m identical machines so that the makespan, i.e., the time when the last job completes its execution, to be minimized. This is one of the most fundamental and well studied problems in scheduling [27, 34, 38]. We focus on the online paradigm of unknown running times where the processing requirement of a job is unknown until the end of its processing (see e.g. [38]), that is the non-clairvoyant setting.

The performance of an online algorithm in the competitive analysis framework is usually evaluated using the competitive ratio [10, 40]. An online algorithm for a minimization problem is ρ-competitive if for every instance of the problem, the value of the objective function of a solution produced by the algorithm is at most ρ times the value of the objective function of an

optimal offline solution. The non-clairvoyant non-preemptive makespan minimization problem on identical machines was the first scheduling problem, and perhaps the first optimization problem, that has been studied using competitive analysis. In 1966, Graham [17] proved that a simple deterministic greedy algorithm, the *List Scheduling algorithm* (LS), has a makespan within a factor of $(2 - 1/m)$ of the makespan of an optimal algorithm with no preemption allowed. The analysis of Graham works also in the non-preemptive case where each job has a release time as well as for the preemptive case, where the processing of any job may be interrupted and resumed at a later time [17, 18]. Given that LS does not use the jobs' processing times, it is also an online non-clairvoyant scheduling algorithm with competitive ratio $(2 - 1/m)$. Shmoys et al. [39] proved that the competitive ratio of any deterministic online algorithm for the non-preemptive non-clairvoyant makespan minimization problem is at least $(2-1/m)$. They also proved that the same tight bound on the competitive ratio holds in the preemptive case. These results showed that in the non-clairvoyant setting there is no difference with respect to the competitive ratio between the preemptive and non-preemptive variants of the makespan minimization problem.

Nevertheless, the assumption of the standard competitive analysis framework that no information is available about the input instance is quite pessimistic. Given the success of Machine Learning methods and Artificial Intelligence in the last years, predictions become available for many optimization problems [2, 33, 44]. However, no guarantees are available concerning the quality of the predictions and several works focus on the following question: "For a given optimization problem and an (unreliable) prediction of the input, is it possible to devise an algorithm with a performance guarantee that is good when the prediction is accurate (consistency), deteriorates gracefully with respect to the prediction error (smoothness), and not too bad and bounded when the prediction is arbitrarily bad (robustness)?" In this paper, we propose to revisit the classical non-clairvoyant makespan minimization scheduling problem in the context of the vibrant area of learning-augmented algorithms that has been formalized in [29] by Lykouris and Vassilvitskii (see [1] for a list of papers in this area) focusing on low complexity learning-augmented algorithms.

2 Further related works

Classical setting. The problem of minimizing the makespan of a schedule of a set of jobs is one of the most fundamental and well studied problems in scheduling theory. As mentioned earlier, Graham proved that LS is a $(2 - \frac{1}{m})$-approximate algorithm [17]. In [18], Graham showed that if the list is ordered in the decreasing order of the processing times of the jobs, then the *Longest Processing Time first* (LPT) algorithm is $(\frac{4}{3} - \frac{1}{3m})$-approximate. Later Coffman et al. [22] proposed a new algorithm, the $MULTIFIT$ algorithm, which leverages the bin packing problem. Improvements to the approximation ratio followed in the works of Friesen [13] and Langston [24, 14]. For a fixed number of machines, a fully polynomial time approximation scheme has been proposed in [37]. Hochbaum and Shmoys [20] proposed a polynomial time approximation scheme for an arbitrary number of processors.For the online (clairvoyant) case with release times, Chen and Vestjens [11] showed that LPT is $\frac{3}{2}$-competitive. When the preemption of jobs is allowed, a simple wrap-around algorithm has been proposed by McNaughton for the offline case [30] which first computes a lower bound of the optimal makespan and then determines a schedule matching this lower bound.

Learning-augmented setting. The framework of learning-augmented algorithms has been formalized by Lykouris and Vassilvitskii [29]. A series of learning-augmented algorithms has been proposed for various problems (see [31, 32]): caching [4, 29, 36, 42], ski-rental [3, 15, 35,

41], routing problems [8, 9, 12, 23], etc. In scheduling, learning-augmented algorithms have been proposed for different criteria, e.g. for the average completion time [7, 21, 28, 35, 43] or the energy consumption in the speed scaling setting [5, 6]. We focus here only on the related work for makespan.

In [25], Lattanzi et al. studied the makespan minimization problem for scheduling a set of jobs with restricted assignments where each job is characterized by a job-dependent subset of the m available machines on which the job can be executed, as well as its processing time. When job j arrives it must be immediately and irrevocably assigned to a machine. They associate a weight to each machine and based on these weights they propose a prediction model where the predicted quantity is the weight of each machine. By using a multiplicative error measure, they show how to obtain a near optimal robust solution for the fractional version based only on the weight-predictions, and use a randomized algorithm for rounding the fractional assignments online with a polylogarithmic loss in the competitive ratio. In [26], Lavastida et al. consider the fractional version of the restricted assignment problem and they prove that the predictions used in [25] can be learned. Moreover, they showed that the predictions are instance-robust. Zhao et al. [45] considered the preemptive-with-restarts makespan minimization problem on a set of uniform parallel machines and they studied it in a learning-augmented setting. They considered an input-prediction model, similar to the one in [35], where for each job a prediction of its processing time is given in advance and they proposed a learning-augmented algorithm with respect to the error prediction that is consistent and robust.

3 Problem Definition, Notations and Preliminaries

In the classical makespan minimization scheduling problem [17], we are given a set $\mathcal{J}$ of n jobs that have to be executed on a set of m parallel identical machines. The execution of each job $j \in \mathcal{J}$ takes a processing time of x_j time units, while it is available for execution only after its release date r_j. Let C_j be the completion time of a job j in a given schedule. The objective is to minimize the completion time of the last job, also known as *makespan* and denoted by $C_{\max} = \max_j\{C_j\}$. We consider both the non-preemptive (Section 4) and the preemptive case (Section 5).

In the non-clairvoyant setting, the *real processing time (or real length)* x_j of each job $j \in \mathcal{J}$ is not known in advance and it becomes known only when j completes its execution. Here, we consider an input-prediction model where for each job $j \in \mathcal{J}$ a prediction of its processing time is given, as it is the case in [21, 35, 45]. Let y_j be the *predicted processing time (or predicted length)* of a job $j \in \mathcal{J}$. We consider the error measure used also in [45].

▶ **Definition 1** (Prediction error, α)**.** *The error of the prediction for job j is defined as $\alpha_j = \max\{\frac{x_j}{y_j}, \frac{y_j}{x_j}\}$ and the error of the prediction is $\alpha = \max_j\{a_j\}$.*

The prediction is perfect if $\alpha = 1$, and in general $\alpha \geq 1$. For example, if $\alpha = 2$ the predicted processing time of a job cannot be more than twice its real processing time or less than half its real processing time. We use the competitive analysis framework to evaluate the performance of the algorithms and our aim is to express the competitive ratio as a function of the prediction error in order to find a trade-off between consistency and robustness.

Given an algorithm $\mathcal{A}$ and an instance I, we denote by $C_{\max}(\mathcal{A}, I)$ the makespan obtained by the algorithm $\mathcal{A}$ on the instance I. In a similar way, we denote by $OPT(I)$ the makespan obtained by the optimal solution on the instance I. Note that the same notation is used for both the classical problem and the problem with predictions, while the optimal solution does not depend on predictions in the latter one. In the case where the instance is clear by the context, we simplify the above notations to $C_{\max}(\mathcal{A})$ and OPT.

Let I be an instance of the classical scheduling problem consisting of n jobs with processing times set to be the real processing times (x_j). Similarly, let I_p be an instance of the problem consisting of the same n jobs with processing times set to be the predicted processing times (y_j). Since (i) the optimal makespan function is monotonic, (ii) the length of each job in I_p is larger than or equal to $\frac{1}{\alpha}$ its real length, and (iii) the length of each job in I_p is smaller than or equal to α times its true length, then the following proposition directly follows.

▶ **Proposition 2.** $\frac{1}{\alpha}OPT(I_p) \leq OPT(I) \leq \alpha OPT(I_p)$

Before continuing with the previously mentioned results, let us present the following example.

▶ **Observation 3.** Consider the following example: we are given $2m-1$ jobs with predicted lengths $y_j = 1$ and m identical machines. Let $x_1 = x_2 = \ldots = x_{m-1} = m-1$, $x_m = \ldots = x_{2m-2} = 1$, and $x_{2m-1} = m$ be the real processing times of the jobs. Hence, $\alpha = m$. For this instance, an optimal solution which knows the real processing times x_j, schedules the job $2m-1$ to a single machine and each of the $m-1$ couples of jobs with processing times 1 and $m-1$ to the remaining $m-1$ machines. Hence, $OPT = m$. However, any deterministic list scheduling algorithm $\mathcal{A}$ (it does not leave any idle time before the starting time of the last scheduled job) cannot take any scheduling decision since it does not know the real processing times, while the known predicted processing times are all identical. Therefore, $\mathcal{A}$ is obliged to create an arbitrary solution which in the worst case will be to schedule the jobs $1, 2, \ldots, m-1$ to the first $m-1$ machines, the jobs $m, m+1, \ldots, 2m-2$ to machine m, and finally the job $2m-1$ to any machine, leading to $C_{\max}(\mathcal{A}) = 2m-1$.

This example shows that any deterministic list scheduling algorithm is at least $(2-\frac{1}{m})$-competitive with predictions. Since any list scheduling algorithm is also at most $(2-\frac{1}{m})$-competitive, we cannot differentiate between different list scheduling algorithms without taking into account some other parameter, such as the value α for example. In what follows, we provide lower and upper bounds as a function of the value of α.

We now formally define the notions of consistency, smoothness and robustness as in [16].

▶ **Definition 4** (Consistency, Smoothness, Robustness)**.** *An algorithm $\mathcal{A}$ is:*

- *ρ_c-consistent, if it is ρ_c-competitive when the prediction is correct, i.e. $\alpha = 1$.*
- *ρ_s-smooth for a continuous function $\rho_s(\alpha)$, if it is $\rho_s(\alpha)$-competitive, where α is the prediction error.*
- *ρ_r-robust, if it is ρ_r-competitive regardless of the prediction error.*

3.1 Our contribution and articulation of the paper

We consider three variants of the problem of scheduling identical machines in the learning-augmented setting. In many works in this area and especially in scheduling (see e.g. [6, 7, 28, 35]), it is common to combine two algorithms, a clairvoyant (assuming predictions to be correct) for consistency and a non-clairvoyant algorithm for robustness. In this work, we propose a single algorithm for each variant (*one stone*) that achieves simultaneously consistency, smoothness and robustness (*for many birds*). In our work, we adapt and analyse two among the most popular algorithms in scheduling, namely LPT and Round Robin (RR), in the learning-augmented framework.

More precisely, for the non-preemptive variants (with and without release dates), we exploit the classical result of Graham [17, 18] which states that LS is a non-clairvoyant $(2-1/m)$-competitive algorithm. This allows us to devise learning-augmented algorithms

that use the predictions in order to create a priority list and then to apply LS. Note that by using such an approach robustness comes for free. In the preemptive case, we devise a new learning-augmented algorithm which is based on the predictions and whose smoothness analysis shows that the competitive ratio gracefully deteriorates with respect to the prediction error from 1 (consistency, when $\alpha = 1$) to $2 - 1/m$ (robustness, when $\alpha \to \infty$). Recall, that for the preemptive case the offline problem can be solved optimally [30] and that for the non-clairvoyant setting, no deterministic algorithm is possible with competitive ratio better than $2 - 1/m$ [39]. In addition, we provide lower bounds for both the non-preemptive and the preemptive variants. Tables 1 and 2 summarize our results with respect to consistency, robustness and smoothness.

The paper is articulated as follows. In Section 4.1, we prove lower bounds for any deterministic learning-augmented algorithm for the identical machines non-clairvoyant non-preemptive makespan minimization problem with predictions. Then, in Section 4.2, we analyze a variant of LPT and we prove that it is consistent, smooth and robust. We also study the non-preemptive problem with release dates and predictions, in Section 4.3, and we show that the generalization of LPT in this setting is also consistent, smooth and robust. Furthermore, we investigate the preemptive case with predictions and we provide lower bounds for any deterministic learning-augmented algorithm as a function of the error (Section 5.1), and then we introduce $PPRR$ that we prove to be consistent, smooth and robust (Section 5.2).

Table 1 Consistency, robustness guarantees.

	Without Predictions		**With Predictions**	
	Competitiveness		*Consistency*	*Robustness*
	Lower Bounds	Upper Bounds	$\alpha = 1$	
Non-preemptive	$2 - 1/m$ [39]	$2 - 1/m$ [17]	4/3	$2 - 1/m$
Non-preemptive with release dates	$2 - 1/m$ [39]	$2 - 1/m$ [17]	3/2	$2 - 1/m$
Preemptive	$2 - 1/m$ [39]	$2 - 1/m$ [17]	1	$2 - 1/m$

4 Non-preemptive Scheduling

In this section, we consider the case with no preemption allowed. We start with two generic lower bounds that hold for any deterministic algorithm, and then we propose and analyze a learning-augmented algorithm based on an adaptation of LPT.

4.1 Lower Bounds

▶ **Proposition 5.** *If $1 \leq \alpha < \sqrt{2}$, there is no deterministic non-clairvoyant algorithm with predictions for scheduling identical machines, with no preemption allowed, which has a competitive ratio smaller than $\frac{1}{2} + \frac{\alpha^2}{2}$.*

Proof. Consider the following instance: m machines, one job of real length α, and m jobs of real length $\frac{1}{\alpha}$. All jobs have predicted length 1. The minimal makespan of a schedule of this instance is $OPT = \max\{\alpha, \frac{2}{\alpha}\} = \frac{2}{\alpha}$ since $\alpha < \sqrt{2}$.

Table 2 Smoothness guarantees.

	Smoothness	
	Lower Bounds	Upper Bounds
Non-preemptive	If $1 \leq \alpha < \sqrt{2}$ $\frac{1}{2} + \frac{\alpha^2}{2}$	$\min\{\frac{2(\alpha^2+1)}{3}, 1 + \frac{\alpha^2}{2}(1 - \frac{1}{m}), 2 - \frac{1}{m}\}$
Non-preemptive with release dates	If $\alpha \geq \sqrt{2}$ $1 + \frac{1}{\lfloor \alpha^2 \rfloor} \left\lfloor \frac{\lfloor \alpha^2 \rfloor (m-1)}{m} \right\rfloor$	$1 + \min\{1, \frac{\alpha^2}{2}\}$
Preemptive	$\frac{m-1}{m} + 1 - \frac{1}{\alpha^2}$ If $\alpha < \sqrt{2}$ $\frac{m\alpha^2+m-1}{\alpha^2+2(m-1)}$ If $\alpha \geq \sqrt{2}$ $\frac{m-1}{m} + \frac{1}{m\lfloor \alpha^2 \rfloor} + 1 - \frac{1}{\lfloor \alpha^2 \rfloor}$	$2 - \frac{\alpha^2+m-2}{\alpha^2 m-1}$

Consider a deterministic algorithm $\mathcal{A}$, and let J be the job scheduled in the last position. Let us assume that J is the job with real processing time α. Therefore, job J starts at a time larger than or equal to $\frac{1}{\alpha}$, and the completion time of J is thus at least $\frac{1}{\alpha} + \alpha$. Therefore, the competitive ratio of $\mathcal{A}$ is at least $\frac{1/\alpha+\alpha}{2/\alpha} = \frac{1}{2} + \frac{\alpha^2}{2}$. ◀

▶ **Proposition 6.** *If $\alpha \geq \sqrt{2}$, there is no deterministic non-clairvoyant algorithm with predictions for scheduling identical machines, with no preemption allowed, which has a competitive ratio smaller than $1 + \frac{1}{\lfloor \alpha^2 \rfloor} \left\lfloor \frac{\lfloor \alpha^2 \rfloor (m-1)}{m} \right\rfloor$.*

Proof. Consider the following instance : m machines, one job of real length α, and $(m-1)\lfloor \alpha^2 \rfloor$ jobs of real length $\frac{\alpha}{\lfloor \alpha^2 \rfloor}$. The optimal makespan of such an instance is $OPT = \alpha$. Let us assume that the predicted length of all the jobs is 1.

Consider a deterministic algorithm $\mathcal{A}$, and let J be the last job to be started in the schedule returned by $\mathcal{A}$. Let us assume that J is the job of real length α. Since J is the last job to be scheduled, it starts at the earliest at time $\left\lfloor \frac{(m-1)\lfloor \alpha^2 \rfloor}{m} \right\rfloor \times \frac{\alpha}{\lfloor \alpha^2 \rfloor}$. Its completion time is thus at least $\left\lfloor \frac{(m-1)\lfloor \alpha^2 \rfloor}{m} \right\rfloor \times \frac{\alpha}{\lfloor \alpha^2 \rfloor} + \alpha$. Since $OPT = \alpha$, the competitive ratio of $\mathcal{A}$ is larger than or equal to $1 + \frac{1}{\lfloor \alpha^2 \rfloor} \left\lfloor \frac{\lfloor \alpha^2 \rfloor (m-1)}{m} \right\rfloor$. ◀

4.2 Common Release Dates

In the case where all jobs are released at time zero, our algorithm works as follows: consider the jobs in non-increasing order of their predicted processing times, i.e. $y_1 \geq y_2 \geq \ldots \geq y_n$. Then, whenever a machine becomes idle, assign to it and schedule non-preemptively the next job according to this order. We call this algorithm the *Longest Predicted Processing Time algorithm* ($LPPT$). Note that each job j finishes x_j units of time after its starting time, while the scheduling decisions are taken based only on the predicted processing times.

In what follows, we establish the following result.

▶ **Theorem 7** (Consistency, Smoothness and Robustness). *LPPT is a non-clairvoyant algorithm with predictions for scheduling identical machines, with no preemption allowed, that achieves a competitive ratio of*

$$\min\{\frac{2(\alpha^2+1)}{3}, 1+\frac{\alpha^2}{2}(1-\frac{1}{m}), 2-\frac{1}{m}\}.$$

Proof. We first give a simple analysis of $LPPT$ for any $\alpha \geq 1$.

▶ **Lemma 8.** *LPPT is a non-clairvoyant algorithm with predictions for scheduling identical machines, with no preemption allowed, that achieves a competitive ratio of*

$$1+\min\{1, \frac{\alpha^2}{2}\}(1-\frac{1}{m}).$$

Proof. Let us consider the schedule returned by $LPPT$ on a given instance I. We assume that the jobs are indexed with respect to the $LPPT$ order, that is in non-increasing order of their predicted processing times: $y_1 \geq y_2 \geq \cdots \geq y_n$. Let t be a job which is completed last (i.e. $C_t = C_{\max}(LPPT)$). We now consider two cases.

Case 1: $y_t > \frac{OPT(I_p)}{2}$. In this case $t \leq m$ and the job t is alone on its machine, and starts at time 0, since otherwise there will be a machine in $OPT(I_p)$ executing two jobs of processing times strictly greater than $\frac{OPT(I_p)}{2}$, which is a contradiction to the value of $OPT(I_p)$. Therefore, $C_{\max} = x_t$, and the schedule is optimal (indeed $OPT(I) \geq x_t$).

Case 2: $y_t \leq \frac{OPT(I_p)}{2}$. By the definition of α, we have that $x_t \leq \alpha y_t \leq \alpha\frac{OPT(I_p)}{2} \leq \alpha^2\frac{OPT(I)}{2}$, where the last inequality holds by Proposition 2. Let s_t be the time at which job t starts to be scheduled. Until s_t, all the machines are busy: this date is thus at most $\frac{\sum_{j\neq t} x_j}{m}$. Since $C_{\max}(LPPT) = s_t + x_t$, we have: $C_{\max}(LPPT) \leq \frac{\sum_{j\neq t} x_j}{m} + x_t = \frac{\sum_j x_j}{m} - \frac{x_t}{m} + x_t$. Since $OPT(I) \geq \frac{\sum_j x_j}{m}$ and $x_t \leq \min\{\frac{\alpha^2 OPT(I)}{2}, OPT(I)\}$, we get: $C_{\max}(LPPT) \leq (1+\min\{1, \frac{\alpha^2}{2}\}(1-\frac{1}{m}))OPT(I)$. ◀

Note, this bound is better than 2 when $\alpha < \sqrt{2}$. Moreover, when the predictions are correct i.e. when $\alpha = 1$, it is $\frac{3}{2} - \frac{1}{2m}$ (whereas LPT is $(\frac{4}{3} - \frac{1}{3m})$-approximate).

We give a better analysis of the $LPPT$ algorithm when $\alpha < \sqrt{2}$.

▶ **Lemma 9.** *When $\alpha < \sqrt{2}$, LPPT is a non-clairvoyant algorithm with predictions for scheduling identical machines, with no preemption allowed, that achieves a competitive ratio of $\frac{2(\alpha^2+1)}{3}$.*

Proof. Let us consider the schedule returned by $LPPT$ on a given instance I. We assume that the jobs are indexed with respect to the $LPPT$ order, that is in non-increasing order of their predicted processing times: $y_1 \geq y_2 \geq \cdots \geq y_n$. Let t be a job which is completed last (i.e. $C_t = C_{\max}(LPPT)$). We now consider two cases.

Case 1: $y_t > \frac{OPT(I_p)}{3}$. If $t \leq m$, then the job t is alone on its machine, and starts at time 0. Therefore, $C_{\max}(LPPT) = x_t$, and the schedule is optimal (indeed $OPT(I) \geq x_t$). Note also that $t \leq 2m$, since otherwise there will be a machine in $OPT(I_p)$ executing three jobs of processing times strictly greater than $\frac{OPT(I_p)}{3}$, which is a contradiction to the value of $OPT(I_p)$. Moreover, we can ignore the jobs $t+1, t+2, \ldots, n$, since the makespan of $LPPT$ is not affected by their removal, while the optimal can only decrease. In what follows in this case, we assume that the instance is reduced to contain only the jobs $1, 2, \ldots, t$.

We next transform the reduced instance I to a new instance I' as follows:

- For $j = 1, 2, \ldots, m$, we set $x'_j = \alpha x_j$.
- For $j = m+1, m+2, \ldots, t$, we set
  $x'_j = \frac{x_j}{\alpha}$, if $x_j > y_j$, and
  $x'_j = x_j$, otherwise.

Recall that we have $\frac{y_j}{\alpha} \leq x_j \leq \alpha y_j$. So, we get $x'_j = \alpha x_j \geq y_j$, for all $j = 1, 2, \ldots, m$, as well as, $x'_j \leq y_j$, for all $j = m+1, m+2, \ldots, t$. It follows that $x'_j \geq x'_{j'}$ for any pair j, j' such that $j \leq m$ and $j' \geq m+1$.

Next, we further modify the instance I' to obtain the instance $\bar{I}$.

- For $j = 1, 2, \ldots, m$, we set $\bar{x}_j = x'_j$.
- For the remaining jobs, we initialize $\mu = m$. In an iterative way and while $\mu < t$, we search for the job $k = \text{argmax}_{\mu+1 \leq j \leq t} x'_j$. Then, for $j = \mu+1, \mu+2, \ldots, k$, we set $\bar{x}_j = x'_k$. We reset $\mu = k$ and pass to the next iteration.

▶ **Property 1.** *Consider a reduced instance I and the corresponding transformed instance $\bar{I}$. Then, the following properties hold.*

(1) *For $j = m+1, m+2, \ldots, t$, we have that $\bar{x}_j \leq \alpha x_j$.*

(2) *$\bar{x}_{m+1} \geq \bar{x}_{m+2} \geq \cdots \geq \bar{x}_t$.*

(3) *For any pair j, j' such that $j \leq m$ and $j' \geq m+1$, we have that $\bar{x}_j \geq \bar{x}_{j'}$.*

Recall that, without loss of generality, we assumed that the instance I is reduced in the first t jobs in non-increasing order of predicted processing times and that $t \leq 2m$. Based on this, we define the algorithm $LPPT_2$ which works like $LPPT$ under the constraint that each machine can execute at most two jobs. It is clear that

$$C_{\max}(LPPT, I) \leq C_{\max}(LPPT_2, I) \tag{1}$$

Similarly, we define the algorithm LPT_2 which works like LPT under the constraint that each machine can execute at most two jobs.

▷ Claim 10. $C_{\max}(LPPT_2, I) \leq \frac{\alpha^2+1}{2\alpha} C_{\max}(LPT_2, \bar{I})$.

▷ Claim 11. $C_{\max}(LPT_2, \bar{I}) \leq \frac{4}{3} OPT(\bar{I})$.

▷ Claim 12. $OPT(\bar{I}) \leq \alpha OPT(I)$.

By combining Equation 1 and Claims 10, 11, 12, we get:

$$\begin{aligned} C_{\max}(LPPT, I) &\leq C_{\max}(LPPT_2, I) \leq \frac{\alpha^2+1}{2\alpha} C_{\max}(LPT_2, \bar{I}) \\ &\leq \frac{2(\alpha^2+1)}{3\alpha} OPT(\bar{I}) \leq \frac{2(\alpha^2+1)}{3} OPT(I). \end{aligned}$$

Case 2: $y_t \leq \frac{OPT(I_p)}{3}$. By the definition of α, we have that $x_t \leq \alpha y_t \leq \alpha \frac{OPT(I_p)}{3} \leq \alpha^2 \frac{OPT(I)}{3}$, where the last inequality holds by Proposition 2. Let s_t be the time at which job t starts to be scheduled. Until s_t, all the machines are busy: this date is thus at most $\frac{\sum_{j \neq t} x_j}{m}$. Since $C_{\max}(LPPT) = s_t + x_t$, we have: $C_{\max}(LPPT) \leq \frac{\sum_{j \neq t} x_j}{m} + x_t = \frac{\sum_j x_j}{m} - \frac{x_t}{m} + x_t$. Since $OPT(I) \geq \frac{\sum_j x_j}{m}$ we get:

$$C_{max}(LPPT) \leq 1 + \frac{\alpha^2}{3}(1 - \frac{1}{m}) OPT(I) \leq \frac{2(\alpha^2+1)}{3}$$ ◀

Lemmas 8 and 9 imply Theorem 7. ◀

▶ Remark 13. For example, if $m = 5$ we have the competitive ratio of $\frac{2(\alpha^2+1)}{3}$ for $\alpha \in [1, \sqrt{\frac{5}{4}}]$, $1 + \frac{\alpha^2}{2}(1 - \frac{1}{m})$ for $\alpha \in [\sqrt{\frac{5}{4}}, \sqrt{2}]$, and $\frac{9}{5}$ for $\alpha > \sqrt{2}$.

4.3 Arbitrary Release Dates

In the case where the jobs have arbitrary release dates, then the algorithm chooses the next *available* job to assign to an idle machine, that is a job which is already released but not yet scheduled. We call this algorithm the *Longest Predicted Processing Time with Release dates algorithm* ($LPPTR$). Here also, each job j finishes x_j units of time after its starting time, while the scheduling decisions are taken based only on the predicted processing times.

▶ **Theorem 14** (Consistency and Smoothness). *$LPPTR$ is a non-clairvoyant algorithm with predictions for scheduling identical machines, with release dates and no preemption allowed, that achieves a competitive ratio of* $1 + \min\{1, \frac{\alpha^2}{2}\}$.

Proof. Let l be a job that is completed last. Let r_l be the release date of job l. If $s_l = r_l$ we have an optimal schedule.

Case 1: $y_l > \frac{OPT(I_p)}{2}$. In the interval between r_l and s_l, no more than one other job is performed. Let $J(r_l)$ be a set of jobs that were performed immediately after the moment of time r_l. Let $Y = \min_{j \in J(r_l)} y_j$. At most m jobs have a processing time greater than $\frac{OPT(I_p)}{2}$. Thus, $Y \leq \frac{OPT(I_p)}{2}$. Hence, $s_l - r_l \leq Y \leq \frac{OPT(I_p)}{2} \leq \frac{\alpha^2 OPT}{2}$. So, we get $C_{\max} = s_l + x_l = s_l - r_l + r_l + x_l = OPT + \frac{\alpha^2 OPT}{2} = OPT(1 + \frac{\alpha^2}{2})$.

Case 2: $y_l \leq \frac{OPT(I_p)}{2}$. Since $OPT(I_p) \leq \alpha OPT$, we have $x_l \leq \alpha y_l \leq \alpha \frac{OPT(I_p)}{2} \leq \alpha^2 \frac{OPT}{2}$. Assume that we have an instance in which $C_{\max} > OPT(1 + \frac{\alpha^2}{2})$. Since $C_{\max} = s_l + x_l$, and since $OPT \geq r_l + x_l$, we have:

$$OPT(1 + \frac{\alpha^2}{2}) < s_l + x_l = s_l - r_l + r_l + x_l \leq s_l - r_l + OPT.$$

We get that $s_l - r_l > \frac{\alpha^2 OPT}{2}$. Let $[t_s, t_f]$ be the last non-empty interval of idle time before the job l begins processing. If such an interval does not exist, then all machines would be busy up to time s_l and $OPT > s_l$. Then, $C_{\max} = s_l + x_l < OPT + \frac{\alpha^2 OPT}{2}$ which is a contradiction, so there exists at least a non-empty interval of idle time before that job l begins.

▶ **Lemma 15.** *In the $LPPTR$ schedule, some jobs begin at or before t_s and complete at or after t_f.*

▶ **Lemma 16.** *Let t_f be the latest point before r_l that some machine is idle. Then $x_l \leq \frac{\alpha^2 OPT}{2} - \frac{t_f}{2}$.*

Additionally we have $OPT \geq s_l - \frac{t_f}{2}$ (see Formula (1.10) in Hochbaum's book [19]). The proof is based on the Lemma 15 and the properties of greedy schedules. Finally, we get $C_{\max} = s_l + x_l \leq s_l + \frac{\alpha^2 OPT}{2} - \frac{t_f}{2} \leq OPT(1 + \frac{\alpha^2}{2})$, contradicting the original assumption on $C_{\max}$. The first inequality follows from Lemma 16. ◀

▶ Remark 17. The fact that $LPPTR$ is $(2 - 1/m)$-robust comes for free from [17, 18] since it is a list scheduling algorithm.

5 Preemptive Scheduling

In this section, we consider the preemptive case and we assume that all jobs and the predictions of their processing times are available at time zero.

5.1 Lower bounds

▶ **Proposition 18.** *If $\alpha \geq \sqrt{2}$, there is no deterministic non-clairvoyant algorithm with predictions for scheduling identical machines, with preemption allowed, which has a competitive ratio smaller than $\frac{m-1}{m} + \frac{1}{m\lfloor\alpha^2\rfloor} + 1 - \frac{1}{\lfloor\alpha^2\rfloor}$.*

▶ **Proposition 19.** *There is no deterministic non-clairvoyant algorithm with predictions for scheduling identical machines, with preemption allowed, which has a competitive ratio smaller than $\frac{m-1}{m} + 1 - \frac{1}{\alpha^2}$.*

▶ **Proposition 20.** *If $\alpha < \sqrt{2}$, there is no deterministic non-clairvoyant algorithm with predictions for scheduling identical machines, with preemption allowed, which has a competitive ratio smaller than $\frac{m\alpha^2+m-1}{\alpha^2+2(m-1)}$.*

5.2 Competitive Algorithm

For this variant, we propose the *Predicted Proportional Round Robin (PPRR) algorithm* which, at each time instant, shares the processing power of the machines to the uncompleted jobs proportionally to their predicted processing times. More specifically, consider a time t. $PPRR$ considers the uncompleted jobs at t in non-increasing order of their predicted processing times, i.e. $y_1 \geq y_2 \geq \ldots \geq y_k$, where k is the number of uncompleted jobs at t. We say that a job i is *mandatory* at time t if $y_i(m-i) \geq \sum_{j=i+1}^{k} y_j$. Each mandatory job is executed alone in a separate machine. Let r be the number of mandatory jobs at time t. Then, a non-mandatory job j at t is executed with speed $\frac{m-r}{\sum_{\ell=r+1}^{k} y_\ell} y_j$. That is, the non-mandatory jobs are executed at a rate proportional to their predicted processing times.

Note that, if $k > m$, then the number of mandatory jobs at t does not exceed $m-1$, while if $k \leq m$, then all k jobs are mandatory. Moreover, we need to recompute the set of mandatory jobs and the speeds of non-mandatory jobs only at time instants corresponding either to the begin of the schedule or to a completion time of a job.

The following lemma shows intuitively that, at a given time t where the total predicted processing time is fixed, the presence of mandatory jobs speeds up the execution of non mandatory jobs.

▶ **Lemma 21.** *Let $\phi(i) = \frac{m-i}{\sum_{\ell=i+1}^{k} y_\ell}$. Consider a r such that $y_i(m-i) \geq \sum_{j=i+1}^{k} y_j$, for each $i = 1, 2, \ldots, r$. For each i, $1 \leq i \leq r$, it holds that $\phi(i-1) \leq \phi(i)$.*

Let us now present some interesting properties of the solution obtained by the $PPRR$ algorithm.

▶ **Property 2.**

(1) *The execution speed of each job can be only increased by the time.*

(2) *If a job becomes mandatory at a time t, then it remains mandatory until its completion.*

(3) *If a job i is mandatory at time t, then any job j such that $j < i$ ($y_j \geq y_i$) is also mandatory.*

(4) *If two jobs i and j do not become mandatory during their execution and $y_i/x_i > y_j/x_j$, then the job i is completed before the job j.*

(5) *If two jobs do not become mandatory during their execution and have the same prediction error α, then they are completed simultaneously.*

(6) *If the prediction is accurate or all jobs have the same prediction error then $PPRR$ is optimal.*

▶ **Theorem 22** (Consistency, Smoothness, Robustness). *$PPRR$ is a non-clairvoyant algorithm with predictions for scheduling identical machines, with preemption allowed, that achieves a competitive ratio of $2 - \frac{\alpha^2+m-2}{\alpha^2 m-1}$. (Hence, $PPRR$ is 1-consistent and $(2-\frac{1}{m})$-robust.)*

Proof. In a schedule constructed by an algorithm $\mathcal{A}$, we call the job i *critical* if $C_i = C_{\max}(\mathcal{A})$. We assume, without loss of generality, that in an optimal schedule, all jobs are completed simultaneously. Indeed, even if this is not the case, then a critical job, say i, is mandatory from time 0 to $OPT = x_i$. Let $X = \sum_j x_j$. We have $x_i > \frac{X}{m}$. We add jobs with a total processing time of $mx_i - X$. The value of the optimum will not change, and the solution of the algorithm with an incorrect prediction can only worsen. Henceforth, we assume that $OPT = X/m$ and in the optimal schedule, all jobs are completed simultaneously.

Let σ^* be an optimal schedule, and σ be the schedule obtained by the algorithm $PPRR$. Moreover, let $s_j(t)$ be the processing speed of a job j at time t in σ. We denote by $x_j(\tau)$ the total execution time of the job j during the interval $[0,\tau]$ in σ. In other words, $x_j(\tau) = \int_0^\tau s_j(t)dt$. Let $\tilde{s}_j(\tau))$ be the average speed of the job j in the interval $[0,\tau]$ in σ, i.e., $\tilde{s}_j(\tau) = x_j(\tau)/\tau$.

Let job c be the critical job in σ, i.e. $C_c = C_{\max}(PPRR)$. Assume that c becomes mandatory at time τ_c in σ. We have

$$C_{\max}(PPRR) = C_c = \tau_c + x_c - x_c(\tau_c) = \tau_c + x_c - \tilde{s}_c(\tau_c)\cdot\tau_c \tag{2}$$

Note that all machines work without any idle time during the interval $[0,\tau_c]$ in σ. It follows that $m\tau_c \le X - (x_c - \tilde{s}_c(\tau_c)\tau_c)$, where $X = \sum_j x_j$. Hence, $\tau_c \le \frac{X-x_c}{m-\tilde{s}_c(\tau_c)}$ and by substituting τ_c in (2), we get

$$C_{\max}(PPRR) \le \frac{(X-x_c)(1-\tilde{s}_c(\tau_c))}{m-\tilde{s}_c(\tau_c)} + x_c \tag{3}$$

Consider the right-hand side of the expression (3) as a function h of $s = \tilde{s}_c(\tau_c)$. Then, we have

$$h'(s) = (X-x_c)\cdot\frac{-(m-s)+(1-s)}{(m-s)^2} = \frac{(X-x_c)(1-m)}{(m-s)^2} < 0$$

Thus, $h(s)$ reaches a maximum when $s = \tilde{s}_c(\tau_c)$ is as small as possible.

In order to get a lower bound to $\tilde{s}_c(\tau_c)$, observe that $s_c(0) = \min\{1, \frac{m-r}{\sum_{\ell=r+1}^n y_\ell} y_c\}$, where r is the number of mandatory jobs at time 0. If $s_c(0) = 1$, then the job c is mandatory starting from time 0, and hence $C_{\max}(PPRR) = x_c$ and $PPRR$ is optimal. In what follows in this proof we consider that $s_c(0) = \frac{m-r}{\sum_{\ell=r+1}^n y_\ell} y_c$. By Lemma 21, we get

$$s_c(0) = \frac{m-r}{\sum_{\ell=r+1}^n y_\ell} y_c \ge \frac{m-0}{\sum_{\ell=0+1}^n y_\ell} y_c = \frac{m}{\sum_{\ell=1}^n y_\ell} y_c = \frac{m y_c}{Y}$$

where $Y = \sum_{\ell=1}^n y_\ell$. By the definition of α, we have that $y_c \ge \frac{x_c}{\alpha}$ and $\alpha X > Y$. So, it holds that $s_c(0) \ge \frac{m x_c}{\alpha^2 X}$. From Property 2(1) of the $PPRR$ algorithm, we have that $s_c(0) \le s_c(t)$ for all $t \in [0,\tau_c]$. Then, it follows that

$$\tilde{s}_c(\tau_c) \ge \frac{m x_c}{\alpha^2 X} \tag{4}$$

By using Equation (4) as a lower bound on $\tilde{s}_c(\tau_c)$ and replacing it in Equation (3) we get

$$C_{\max}(PPRR) \leq \frac{(X-x_c)(1-\frac{mx_c}{\alpha^2 X})}{m-\frac{mx_c}{\alpha^2 X}} + x_c = \frac{(X-x_c)(\alpha^2 X - mx_c)}{\alpha^2 mX - mx_c} + x_c$$

$$= \frac{\alpha^2X^2 - mXx_c - \alpha^2 Xx_c + mx_c^2 + \alpha^2 mXx_c - mx_c^2}{\alpha^2 mX - mx_c}$$

$$= \frac{\alpha^2X^2 - mXx_c - \alpha^2 Xx_c + \alpha^2 mXx_c}{\alpha^2 mX - mx_c} \tag{5}$$

Note that m and α are constants. Fix X and consider the right-hand side of the expression (5) as a function $f(x_c)$. We have

$$f'(x_c) = \frac{(\alpha^2 mX - \alpha^2 X - mX)(\alpha^2 mX - mx_c) + m(\alpha^2X^2 - mXx_c - \alpha^2 Xx_c + \alpha^2 mXx_c)}{(\alpha^2 mX - mx_c)^2}$$

$$= \frac{\alpha^2 mX^2(\alpha^2(m-1) - (m-1))}{(\alpha^2 mX - mx_c)^2} = \frac{\alpha^2 mX^2(\alpha^2-1)(m-1)}{(\alpha^2 mX - mx_c)^2} \geq 0$$

Thus, $f(x_c)$ reaches a maximum when x_c is as large as possible. By our initial observation, we have that $x_c \leq X/m$ and by replacing in Equation (5) we get

$$C_{\max}(PPRR) \leq \frac{\alpha^2X^2 - X^2 - \alpha^2X^2/m + \alpha^2X^2}{\alpha^2 mX - X}$$

Observe that in an optimal schedule the total execution load is equally partitioned to all machines, and hence $OPT \geq \frac{X}{m}$. Therefore, for the competitive ratio ρ of $PPRR$ we have

$$\rho \leq \frac{\alpha^2X^2 - X^2 - \alpha^2X^2/m + \alpha^2X^2}{\alpha^2X^2 - X^2/m} = \frac{2\alpha^2 m - m - \alpha^2}{\alpha^2 m - 1} = 2 - \frac{\alpha^2 + m - 2}{\alpha^2 m - 1}.$$

The consistency (resp. robustness) ratio is achieved by replacing $\alpha = 1$ (resp. taking the bound when $\alpha \to \infty$). ◀

Figure 1 shows the competitive ratio of algorithm PPRR as well as lower bounds on the ratio of any deterministic preemptive algorithm. As we can see, lower bounds and upper bounds are quite close, and get closer when m increases.

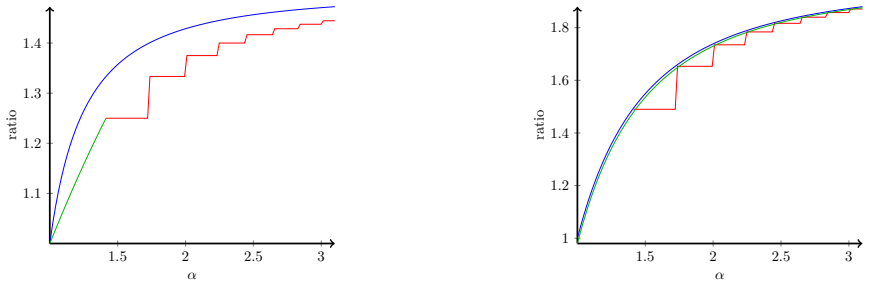

Figure 1 In blue: competitive ratio of algorithm PPRR as a function of α (x axis). In green and red: lower bounds on the competitive ratio of a deterministic algorithm with preemption. In red: lower bound given by Proposition 18. Left (resp. Right): ratio when $m = 2$ (resp. $m = 50$). On the left, in green: lower bound given by Proposition 20 (the bound of Proposition 19 is not drawn here since when $m = 2$, it is lower than the other lower bounds). On the right, in green: lower bound given by Proposition 19 (the bound of Proposition 20 is not drawn here since when $m = 50$, it is lower than the other lower bounds).

References

1. https://algorithms-with-predictions.github.io.
2. Maryam Amiri and Leili Mohammad Khanli. Survey on prediction models of applications for resources provisioning in cloud. *J. Netw. Comput. Appl.*, 82:93–113, 2017.
3. Spyros Angelopoulos, Christoph Dürr, Shendan Jin, Shahin Kamali, and Marc P. Renault. Online computation with untrusted advice. In *ITCS*, 2020.
4. Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. In *ICML*, 2020.
5. Antonios Antoniadis, Peyman Jabbarzade Ganje, and Golnoosh Shahkarami. A novel prediction setup for online speed-scaling. In Artur Czumaj and Qin Xin, editors, *18th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2022, June 27-29, 2022, Tórshavn, Faroe Islands*, volume 227 of *LIPIcs*, pages 9:1–9:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.
6. Étienne Bamas, Andreas Maggiori, Lars Rohwedder, and Ola Svensson. Learning augmented energy minimization via speed scaling. In *NeurIPS*, 2020.
7. Evripidis Bampis, Konstantinos Dogeas, Alexander V. Kononov, Giorgio Lucarelli, and Fanny Pascual. Scheduling with untrusted predictions. In Luc De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pages 4581–4587. ijcai.org, 2022.
8. Evripidis Bampis, Bruno Escoffier, Themis Gouleakis, Niklas Hahn, Kostas Lakis, Golnoosh Shahkarami, and Michalis Xefteris. Learning-augmented online TSP on rings, trees, flowers and (almost) everywhere else. In Inge Li Gørtz, Martin Farach-Colton, Simon J. Puglisi, and Grzegorz Herman, editors, *31st Annual European Symposium on Algorithms, ESA 2023, September 4-6, 2023, Amsterdam, The Netherlands*, volume 274 of *LIPIcs*, pages 12:1–12:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.12.
9. Evripidis Bampis, Bruno Escoffier, and Michalis Xefteris. Canadian traveller problem with predictions. In Parinya Chalermsook and Bundit Laekhanukit, editors, *Approximation and Online Algorithms*, pages 116–133. Springer International Publishing, 2022.
10. Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
11. Bo Chen and Arjen P.A. Vestjens. Scheduling on identical machines: How good is lpt in an on-line setting? *Operations Research Letters*, 21(4):165–169, 1997.
12. Franziska Eberle, Alexander Lindermayr, Nicole Megow, Lukas Nölke, and Jens Schlöter. Robustification of online graph exploration methods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(9):9732–9740, June 2022.
13. Donald K. Friesen. Tighter bounds for the multifit processor scheduling algorithm. *SIAM J. Comput.*, 13(1):170–181, 1984.
14. Donald K. Friesen and Michael A. Langston. Evaluation of a multifit-based scheduling algorithm. *J. Algorithms*, 7(1):35–59, 1986.
15. Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *ICML*, 2019.
16. Themis Gouleakis, Konstantinos Lakis, and Golnoosh Shahkarami. Learning-Augmented Algorithms for Online TSP on the Line. *To appear in AAAI*, 2023. CoRR abs/2206.00655.
17. Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technical J.*, 45(9):1563–1581, 1966.
18. Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM Journal of Applied Mathematics*, 17(2):416–429, 1969.
19. Dorit S. Hochbaum. *Approximation Algorithms for NP-hard Problems*. PWS Publishing, 1997.
20. Dorit S. Hochbaum and David B. Shmoys. A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. *SIAM J. Comput.*, 17(3):539–551, 1988.

21 Sungjin Im, Ravi Kumar, Mahshid Montazer Qaem, and Manish Purohit. Non-clairvoyant scheduling with predictions. In *SPAA*, pages 285–294. ACM, 2021.

22 Edward G. Coffman Jr., M. R. Garey, and David S. Johnson. An application of bin-packing to multiprocessor scheduling. *SIAM J. Comput.*, 7(1):1–17, 1978.

23 Murali Kodialam and T. V. Lakshman. Prediction augmented segment routing. In *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, pages 1–6, 2021.

24 M.A. Langston. *A. Processors cheduling with improved heuristic algorithms. Doctoral dissertation,*. PhD thesis, Texas A&M Univ., College Station, Tex., 1981.

25 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *SODA*, pages 1859–1877. SIAM, 2020.

26 Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In Petra Mutzel, Rasmus Pagh, and Grzegorz Herman, editors, *29th Annual European Symposium on Algorithms, ESA 2021, September 6-8, 2021, Lisbon, Portugal (Virtual Conference)*, volume 204 of *LIPIcs*, pages 59:1–59:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

27 Joseph Leung, Laurie Kelly, and James H. Anderson. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, Inc., USA, 2004.

28 Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. In Kunal Agrawal and I-Ting Angelina Lee, editors, *SPAA '22: 34th ACM Symposium on Parallelism in Algorithms and Architectures, Philadelphia, PA, USA, July 11 - 14, 2022*, pages 357–368. ACM, 2022.

29 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 3302–3311. PMLR, 2018.

30 Robert McNaughton. Scheduling with deadlines and loss functions. *Management Science*, 6(1):1–12, 1959.

31 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. In *Beyond Worst Case Analysis*, pages 646–662. Cambridge University Press, 2021.

32 Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022.

33 Narges Peyravi and Ali Moeini. Estimating runtime of a job in hadoop mapreduce. *Journal of Big Data*, 7(: 44), 2020.

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

35 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. In *NeurIPS*, 2018.

36 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *SODA*, 2020.

37 Sartaj Sahni. Algorithms for scheduling independent tasks. *J. ACM*, 23(1):116–127, 1976.

38 Jirí Sgall. On-line scheduling. In A. Fiat and G. J. Woeginger, editors, *Online Algorithms: The State of the Art*, pages 196–231. Springer, 1998.

39 David B. Shmoys, Joel Wein, and David P. Williamson. Scheduling parallel machines on-line. *SIAM J. Comput.*, 24(6):1313–1331, 1995.

40 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

41 Shufan Wang and Jian Li. Online algorithms for multi-shop ski rental with machine learned predictions. In *AAMAS*, 2020.

42 Alexander Wei. Better and simpler learning-augmented online caching. In *APPROX/RANDOM*, 2020.

43 Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *NeurIPS*, 2020.

44 Hirochika Yamashiro and Hirofumi Nonaka. Estimation of processing time using machine learning and real factory data for optimization of parallel machine scheduling problem. *Operations Research Perspectives*, 8(: 100196), 2021.

45 Tianming Zhao, Wei Li, and Albert Y. Zomaya. Uniform machine scheduling with predictions. In Akshat Kumar, Sylvie Thiébaux, Pradeep Varakantham, and William Yeoh, editors, *Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling, ICAPS 2022, Singapore (virtual), June 13-24, 2022*, pages 413–422. AAAI Press, 2022.