Scheduling with Speed Predictions

Eric Balkanski[1], Tingting Ou[1], Clifford Stein[1], and Hao-Ting Wei[1]

Department of Industrial Engineering and Operations Research, Columbia University
eb3224@columbia.edu
to2372@columbia.edu
cliff@ieor.columbia.edu
hw2738@columbia.edu

Abstract. Algorithms with predictions is a recent framework that has been used to overcome pessimistic worst-case bounds in incomplete information settings. In the context of scheduling, very recent work has leveraged machine-learned predictions to design algorithms that achieve improved approximation ratios in settings where the processing times of the jobs are initially unknown. In this paper, we study the speed-robust scheduling problem where the speeds of the machines, instead of the processing times of the jobs, are unknown and augment this problem with predictions.
Our main result is an algorithm that achieves a $\min\{\eta^2(1+\alpha), (2+2/\alpha)\}$ approximation, for any $\alpha \in (0,1)$, where $\eta \geq 1$ is the prediction error. When the predictions are accurate, this approximation outperforms the best known approximation for speed-robust scheduling without predictions of $2-1/m$, where m is the number of machines, while simultaneously maintaining a worst-case approximation of $2+2/\alpha$ even when the predictions are arbitrarily wrong. In addition, we obtain improved approximations for three special cases: equal job sizes, infinitesimal job sizes, and binary machine speeds. We also complement our algorithmic results with lower bounds. Finally, we empirically evaluate our algorithm against existing algorithms for speed-robust scheduling. The full version of the paper can be referred to the following link https://arxiv.org/abs/2205.01247.

Keywords: Algorithms with prediction · Scheduling · Approximation algorithm

1 Introduction

In many optimization problems, the decision maker faces crucial information limitations due to the input not being completely known in advance. A natural goal in such settings is to find solutions that have a good worst-case performance over all potential input instances. However, even though worst-case analysis provides a useful measure for the robustness of an algorithm, it is also known to be a measure that often leads to needlessly pessimistic results.

A recent, yet extensive, line of work on *algorithms with predictions* models the partial information that is often available to the decision maker and overcomes

worst-case bounds by leveraging machine-learned predictions about the inputs (see [22] for a survey of the early work in this area). In this line of work, the algorithm is given some type of prediction about the input, but the predictions are not necessarily accurate. The goal is to design algorithms that achieve stronger bounds when the provided predictions are accurate, which are called *consistency* bounds, but also maintain worst-case *robustness* bounds that hold even when the predictions are inaccurate. Optimization problems that have been studied under this framework include online paging [20], scheduling [23], secretary [10], covering [6], matching [8,9,17], knapsack [16], facility location [13], Nash social welfare [7], and graph [4] problems. Most of the work on scheduling in this model has considered predictions about the processing times of the jobs [23,21,18,5,15,2,3].

There is a large body of work considering uncertainty in the input to scheduling problems, including whole fields like stochastic scheduling. Most of it studies uncertainty in the jobs. A recent line of work considers scheduling problems where there is uncertainty surrounding the available machines (e.g. [1,12,24,11]). In particular, we emphasize *scheduling with an unknown number of parallel machines*, introduced in [24] where, given a set of jobs, there is a first *partitioning* stage where they must be partitioned into *bags* without knowing the number of machines available and then, in a second *scheduling* stage, the algorithm learns the number of machines and the bags must be scheduled on the machines without being split up. This problem was generalized to *speed-robust scheduling* [11] where there are m machines, but speeds of the machines are unknown in the partitioning stage and are revealed in the scheduling stage [1]. We will use the speed robust scheduling model in the rest of this paper, as it captures applications where partial packing decisions have to be made with only partial information about the machines. As discussed in [24], such applications include MapReduce computations in shared data centers where data is partitioned into groups by a mapping function that is designed without full information about the machines that will be available in the data center, or in a warehouse where items are grouped into boxes without full information about the trucks that will be available to ship the items.

In this paper, we introduce and study the problem of scheduling with machine-learned predictions about the speeds of the machines. In the two applications mentioned above, MapReduce computations and package shipping, it is natural to have some relevant historical data about the computing resources or the trucks that will be available, which can be used to obtain machine-learned predictions about these quantities. In the *scheduling with speed predictions* problem, we are given jobs and predictions about the speeds of the m machines. In the first, *partitioning stage*, jobs are partitioned into m bags, using only the predictions about the speeds of the machines. Then, in the second, *scheduling stage*, the true speeds of the machines are revealed, and the bags must be scheduled on the machines without being split up. The goal is to use the predictions to design algorithms that achieve improved guarantees for speed-robust scheduling. The fundamental question we ask is:

[1] This problem strictly generalizes the first problem by setting speed to 1 for actual machines, and speed to 0 for the other (non)machines.

Job sizes	Speeds	Upper bound	Lower bound
General	General	$2 + 2/\alpha$ (Theorem 3)	$1 + (1-\alpha)/2\alpha - O(1/m)$ (Theorem 1)
Equal-size	General	$2 + 1/\alpha$ (Theorem 4)	$1 + (1-\alpha)/2\alpha - O(1/m)$ (Theorem 1)
Infinitesimal	General	$1 + 1/\alpha$ (Theorem 5)	$1 + (1-\alpha)^2/4\alpha - O(1/m)$ (Theorem 1)
General	{0,1}	2 (Theorem 6)	$(4-2\alpha)/3$ (Theorem 7)

Table 1. Robustness of deterministic $1+\alpha$ consistent algorithms, where $\alpha \in (0,1)$ except for the $(4-2\alpha)/3$ lower bound, for which $\alpha \in (0,1/2)$.

Can speed predictions be used to obtain both improved guarantees when the predictions are accurate and bounded guarantees when the prediction errors are arbitrarily large?

We focus on the classical makespan (completion time of the last completed job) minimization objective. Two main evaluation metrics for our problem, or for any algorithms with predictions problem, are robustness and consistency. The consistency of an algorithm is the approximation ratio it achieves when the speed predictions are equal to the true speeds of the machines, and its robustness is its worst-case approximation ratio over all possible machine speeds, i.e., when the predictions are arbitrarily wrong. The main focus of this paper is on general job processing times and machine speeds, but we also consider multiple special cases.

Without predictions, [11] achieves a $(2-1/m)$-approximation. Thus, if we do not trust the predictions, we can ignore them and use this algorithm to achieve a $2-1/m$ consistent and $2-1/m$ robust algorithm. On the other hand, if we fully trust the predictions, we can pretend that the predictions are correct and use a polynomial time approximation scheme (PTAS) for makespan minimization on related machines to obtain a $1+\epsilon$ consistent algorithm, for any constant $\epsilon > 0$. However, as we show in Section 3, this approach would have unbounded robustness. Thus, the main challenge is to develop an algorithm that leverages predictions to improve over the best known $2-1/m$ approximation when the predictions are accurate, while maintaining bounded robustness guarantees even when the predictions are arbitrarily wrong.

1.1 Our results

Our main result is a deterministic algorithm for minimizing makespan in the scheduling with speed predictions (SSP) model that is $1+\alpha$ consistent and $2+2/\alpha$ robust, for any $\alpha \in (0,1)$ (Theorem 2). When the predictions are accurate, the $1+\alpha$ consistency outperforms the best-known approximation for speed-robust scheduling without predictions of $2-1/m$ [11], while maintaining a $2+2/\alpha$ robustness guarantee that holds even when the predictions are arbitrarily wrong. To obtain a polynomial time algorithm, the consistency and robustness both increase by a $1+\epsilon$ factor, for any constant $\epsilon \in (0,1)$, due to the PTAS for makespan minimization on related machines that we use as a subroutine [14].

We extend this result to obtain an approximation ratio that interpolates between $1+\alpha$ and $2+2/\alpha$ as a function of the prediction error. More precisely,

for any $\alpha \in (0,1)$, our algorithm achieves an approximation of $\min\{\eta^2(1+\alpha), (2+2/\alpha)\}$ (Theorem 3), where the prediction error $\eta := \max_{i\in[m]} \frac{\max\{\hat{s}_i, s_i\}}{\min\{\hat{s}_i, s_i\}}$ is the maximum ratio between the predicted speed $\hat{s}_i$ and the true speed s_i of the m machines. The following hardness result motivates this choice for the prediction error: for any $\alpha \in (0,1)$, any deterministic $1+\alpha$ consistent algorithm has robustness at least $1 + \frac{1-\alpha}{2\alpha} - O(\frac{1}{m})$, even when a single machine speed is incorrectly predicted (Theorem 1). Thus, a single incorrectly predicted machine speed can cause a strong lower bound on the approximation ratio. We also note that the maximum ratio over all the predictions is a common definition for the prediction error in scheduling with predictions (see, e.g., [18,19]). Additionally, we obtain the following results (summarized in Table 1):

- When the job processing times are equal or infinitesimal, the best-known approximations without predictions are 1.8 and $e/(e-1) \approx 1.58$ [11], respectively. For these cases, our $1+\alpha$ consistent algorithm achieves a robustness of $2+1/\alpha$ (Theorem 4) and $1+1/\alpha$ (Theorem 5), respectively.
- When the machine speeds are either 0 or 1, which corresponds to the scenario where the number of machines is unknown, the best-known approximation without predictions is 5/3 [24]. We develop an algorithm that is 1 consistent and 2 robust (Theorem 6). We also show that, for any $\alpha \in [0, 1/2)$, any deterministic $1+\alpha$ consistent algorithm has robustness at least $(4-2\alpha)/3$ (Theorem 7).
- Even when the prediction error is relatively large, our algorithm often empirically outperforms existing speed-robust algorithms that do not use predictions.

We note that, subsequent to our work, a scheduling with predictions problem where the machine speeds are unknown was also studied in [19], but in an incomparable online setting where the speeds can be job-dependent.

1.2 Technical overview

We give an overview of the main technical ideas used to obtain our main result (Theorem 3). The second stage of the SSP problem corresponds to a standard makespan minimization problem in the full information setting, so the main problem is the first stage where jobs must be partitioned into bags given predictions about the speeds of the machines. At a high level, our partitioning algorithm initially creates a partition of the jobs in bags, and a tentative assignment of the bags to machines, assuming that the predictions are the true speeds of the machines. This tentative solution is optimal if the predictions are perfect, but as we discuss in Section 3, if the predictions are wrong, its makespan may be far from optimal. To address this concern, the algorithm iteratively moves away from the initial partition in order to obtain a more robust partitioning, while also maintaining that the bags can be scheduled to give a $(1+\alpha)$-approximation of the makespan if the predictions are correct. The parameter $\alpha \in (0,1)$ is an input to the algorithm that controls the consistency-robustness trade-off, i.e., it controls how much the predictions should be trusted. Starting from a consistent solution

and then robustifying has been used in some other algorithms with predictions. Our main technical contribution is in designing such a robustification algorithm for the SSP problem.

More concretely, let the total processing time of a bag be the sum of the processing time of the jobs in that bag. The partitioning algorithm always maintains a tentative assignment of bags to the machines. To robustify this assignment, the algorithm iteratively reassigns the bag with minimum total processing time to the machine that is assigned the bag with maximum total processing time. If there are now ℓ bags assigned to this machine, we break open these ℓ bags, and reassign the jobs to ℓ new bags using the Longest Processing Time first algorithm, which will roughly balance the size of the ℓ bags assigned to this machine. Thus, at every iteration, the bags that had the maximum and minimum total processing times at the beginning of that iteration end up with approximately equal total processing times, which improves the robustness of the partition. The algorithm terminates when the updated partition would not achieve a $1 + \alpha$ consistency anymore.

The analysis of the $2 + 2/\alpha$ robustness consists of three main lemmas. The algorithm and analysis use a parameter β, which is the ratio of the maximum total processing time of a bag that contains at least two jobs to the minimum total processing time of a bag. We use this particular parameter partly to handle the case of very large jobs. Informally, both the algorithm and the adversary will need to put that one job in its own bag and on its own machine, so we can just "ignore" such jobs. We first show that if we can solve the second-stage scheduling problem optimally, then the robustness achieved by any partition is at most $\max\{2, \beta\}$. Then, we show that at each iteration, the minimum total processing time of a bag is non-decreasing. Finally, we use this monotonicity property to show that, for the partition returned by the algorithm, $\beta \leq 2 + 2/\alpha$. Together with the first lemma, this implies that the algorithm achieves a $2+2/\alpha$ robustness. The last lemma requires a careful argument to show that, if $\beta > 2 + 2/\alpha$, then an additional iteration of the algorithm does not break the $1 + \alpha$ consistency achieved by the current partition. To obtain a polynomial-time algorithm, we pay an extra factor of $1 + \epsilon$ in the scheduling stage by using the PTAS of [14].

Finally, we provide an empirical evaluation of our algorithm that shows that, even when the prediction error is relatively large, it often outperforms existing speed-robust algorithms that do not use predictions.

2 Preliminaries

We first describe the speed-robust scheduling problem, which was introduced by [11] and builds on the scheduling with an unknown number of machines problem from [24]. There are n jobs with processing times $\mathbf{p} = (p_1, \ldots, p_n) \geq \mathbf{0}$ and m machines with speeds $\mathbf{s} = (s_1, \ldots, s_m) > \mathbf{0}$ such that the time needed to

process job j on machine i is p_j/s_i.[2] The problem consists of the following two stages. In the first stage, called the partitioning stage, the speeds of the machines are unknown and the jobs must be partitioned into m (possibly empty) bags $B_1, \ldots, B_m$ such that $\cup_{i\in[m]} B_i = [n]$ (where $[n] = \{1, \ldots, n\}$) and $B_{i_1} \cap B_{i_2} = \emptyset$ for all $i_1, i_2 \in [m]$, $i_1 \neq i_2$. In the second stage, called the scheduling stage, the speeds $\mathbf{s}$ are revealed to the algorithm and each bag B_i created in the partitioning stage must be assigned, i.e., scheduled, on a machine without being split up.

The paper on speed-robust scheduling, [11], considers the classical makespan minimization objective. Let $\mathcal{M}_i$ be the bags assigned to machine i; the goal is to minimize $\max_{i\in[m]}(\sum_{B\in\mathcal{M}_i}\sum_{j\in B} p_j)/s_i$. An algorithm for speed-robust scheduling is β-robust if it achieves an approximation ratio of β compared to the optimal schedule that knows the speeds in advance, i.e., $\max_{\mathbf{p},\mathbf{s}} alg(\mathbf{p},\mathbf{s})/opt(\mathbf{p},\mathbf{s}) \leq \beta$ where $alg(\mathbf{p},\mathbf{s})$ and $opt(\mathbf{p},\mathbf{s})$ are the makespans of the schedule returned by the algorithm (that learns $\mathbf{s}$ in the second stage) and the optimal schedule (that knows $\mathbf{s}$ in the first stage).

We augment the speed-robust scheduling problem with predictions about the speeds of the machines and call this problem Scheduling with Speed Predictions (SSP). The difference between SSP and speed-robust scheduling is that, during the partitioning stage, the algorithm is now given access to, potentially incorrect, predictions $\hat{\mathbf{s}} = (\hat{s}_1, \ldots, \hat{s}_m) \geq 0$ about the speeds of the machines (see Appendix A.1 of the full version of the paper for additional discussion about how we learn the machine speeds and obtain $\hat{s}$). The true speeds of the machines $\mathbf{s}$ are revealed during the scheduling stage, as in the speed-robust scheduling problem. We also want to minimize the makespan.

Consistency and robustness are two standard measures in algorithms with predictions [20]. An algorithm is c-consistent if it achieves a c approximation ratio when the predictions are correct, i.e., if $\max_{\mathbf{p},\mathbf{s}} alg(\mathbf{p},\mathbf{s},\mathbf{s})/opt(\mathbf{p},\mathbf{s}) \leq c$ where $alg(\mathbf{p},\hat{\mathbf{s}},\mathbf{s})$ is the makespan of the schedule returned by the algorithm when it is given predictions $\hat{\mathbf{s}}$ in the first stage and speeds $\mathbf{s}$ in the second stage. An algorithm is β-robust if it achieves a β approximation ratio when the predictions can be arbitrarily wrong, i.e., if $\max_{\mathbf{p},\hat{\mathbf{s}},\mathbf{s}} alg(\mathbf{p},\hat{\mathbf{s}},\mathbf{s})/opt(\mathbf{p},\mathbf{s}) \leq \beta$. We note that a β-robust algorithm for speed-robust scheduling is also a β-robust (and β-consistent) algorithm for SSP which ignores the speed predictions.

The main challenge in algorithms with predictions problems is to simultaneously achieve "good" consistency and robustness, which requires partially trusting the predictions (for consistency), but not trusting them too much (for robustness). In particular, the goal is to obtain an algorithm that achieves a consistency that improves over the best known approximation without predictions ($2 - 1/m$ for speed-robust scheduling), ideally close to the best known approximation in the full information setting ($1 + \epsilon$, for any constant $\epsilon > 0$, for makespan minimization on related machines), while also achieving bounded robustness.

[2] The non-zero speed assumption is for ease of notation. Having a machine with speed $s_i = 0$ is equivalent to $s_i = \epsilon$ for ϵ arbitrarily small since in both cases no schedule can assign a job to i without the completion time of this job being arbitrarily large.

Even though consistency and robustness capture the main trade-off in SSP, we are also interested in giving approximation ratios as a function of the prediction error. It is important, in any algorithms with predictions problem, to define the prediction error appropriately, so that it actually captures the proper notion of error in the objective. It might seem that, for example, L_1 distance between the predictions and data is natural, but for many problems, including this one, such a definition would mainly give vacuous results. We define the prediction error $\eta \geq 1$ to be the maximum ratio[3] between the true speeds $\mathbf{s}$ and the predicted speeds $\hat{\mathbf{s}}$, or vice versa, i.e., $\eta(\hat{\mathbf{s}}, \mathbf{s}) = \max_{i \in [m]} \frac{\max\{\hat{s}_i, s_i\}}{\min\{\hat{s}_i, s_i\}}$ (see Appendix A.2 of the full version for further discussion on the choice of error measure). Given a bound η on the prediction error, an algorithm achieves a $\gamma(\eta)$ approximation if $\max_{\mathbf{p}, \hat{\mathbf{s}}, \mathbf{s}: \eta(\hat{\mathbf{s}}, \mathbf{s}) \leq \eta} alg(\mathbf{p}, \hat{\mathbf{s}}, \mathbf{s}) / opt(\mathbf{p}, \mathbf{s}) \leq \gamma(\eta)$.

Given arbitrary bags $B_1, \ldots, B_m$, the scheduling stage corresponds to a standard makespan minimization problem in the full information setting, for which polynomial-time approximation schemes (PTAS) are known [14]. Thus, the main challenge is the partitioning stage. We define the consistency and robustness of a partitioning algorithm $\mathcal{A}_P$ to be the consistency and robustness achieved by the two-stage algorithm that first runs $\mathcal{A}_P$ and then solves the scheduling stage optimally. If we want to require that algorithms be polynomial time, we may simply run the PTAS for makespan minimization in the scheduling stage, and the bounds increase by a $(1 + \epsilon)$ factor. We will not explicitly mention this in the remainder of the paper.

3 Consistent Algorithms are not Robust

A natural first question is whether there is an algorithm with optimal consistency that also achieves a good robustness. We answer this question negatively and show that there exists an instance for which any 1-consistent algorithm cannot be $o(n)$-robust. This impossibility result is information-theoretic and is not due to computational constraints. The proofs in this section can be found in Appendix B of the full version.

Proposition 1. *For any $n > m$, there is no algorithm that is 1-consistent and $\frac{n-m+1}{\lceil n/m \rceil}$-robust, even in the case of equal-size jobs. In particular, for $m = n/2$, there is no algorithm that is 1-consistent and $o(n)$-robust.*

More generally, we show that there is a necessary non-trivial trade-off between consistency and robustness for the SSP problem. In particular, the robustness of any deterministic algorithm for SSP must grow inversely proportional as a function of the consistency.

Theorem 1. *For any $\alpha \in (0, 1)$, if a deterministic algorithm for SSP is $(1+\alpha)$-consistent, then its robustness is at least $1 + \frac{1-\alpha}{2\alpha} - O(\frac{1}{m})$, even in the case where*

[3] We scale $\mathbf{s}, \hat{\mathbf{s}}$ such that $\max_i s_i = \max_i \hat{s}_i$ before computing η, to make sure the speeds are on the same scale.

the jobs have equal processing times. In the special case where the processing times are infinitesimal, the robustness of a deterministic $(1+\alpha)$*-consistent algorithm is at least* $1 + \frac{(1-\alpha)^2}{4\alpha} - O(\frac{1}{m})$.

Recall that in the setting without predictions, the best known algorithm is $(2-1/m)$-robust (and thus also $(2-1/m)$-consistent) [11]. Since we have shown that algorithms with near-optimal consistency must have unbounded robustness, a main question is thus whether it is even possible to achieve a consistency that improves over $(2-1/m)$ while also obtaining bounded robustness. We note that the natural idea of randomly choosing to run the $(2-1/m)$-robust algorithm or an algorithm with near-optimal consistency (with unbounded robustness), aiming to hedge between robustness and consistency, does not work since the resulting algorithm would still have unbounded robustness due to SSP being a minimization problem.

4 The Algorithm

In this section, we give an algorithm for scheduling with speed predictions with arbitrary-sized jobs that achieves a $\min\{\eta^2(1+\epsilon)(1+\alpha), (1+\epsilon)(2+2/\alpha)\}$ approximation for any constant $\epsilon \in (0,1)$ and any $\alpha \in (0,1)$. The proofs in this section can be found in Appendix C of the full version.

4.1 Description of the algorithm

Our algorithm, called IPR and formally described in Algorithm 1, takes as input the processing times of the jobs $\mathbf{p}$, the predicted speeds of the machines $\hat{\mathbf{s}}$, an accuracy parameter ϵ, a consistency goal $1+\alpha$, and a parameter ρ that influences the ratio between the size of the smallest and largest bags. For general job processing times and machine speeds, we use $\rho = 4$. For some special cases in Section 5, we use $\rho = 2$. IPR first uses the PTAS for makespan minimization [14] to construct a partition of the jobs into bags $B_1, \ldots, B_m$ such that scheduling the jobs in B_i on machine i achieves a $1+\epsilon$ approximation when the predictions are correct. In other words, it initially assumes that the predictions are correct and creates a $(1+\epsilon)$-consistent partition of the jobs into bags. In addition, it also creates a tentative assignment $\mathcal{M}_1 = \{B_1\}, \ldots, \mathcal{M}_m = \{B_m\}$ of the bags $B_1, \ldots, B_m$ on the machines.

Even though this tentative assignment achieves a good consistency, its robustness is arbitrarily poor. To improve the robustness, IPR iteratively rebalances this partition while maintaining a $(1+\epsilon)(1+\alpha)$ bound on its consistency. The design and analysis of such an iterative rebalancing procedure is the main challenge.

At each iteration, the subroutine LPT-Rebalance rebalances the bags and modifies $\mathcal{M}_1, \ldots, \mathcal{M}_m$. We define the processing time $p(B)$ of a bag to the total processing time of the jobs in that bag, i.e., $p(B) = \sum_{j \in B} p_j$. The algorithm terminates either when scheduling the bags in each $\mathcal{M}_i$ on machine i violates the desired $(1+\epsilon)(1+\alpha)$ consistency bound or when the ratio of the largest processing

time of a bag containing at least two jobs to the smallest processing time of a bag is at most ρ. To verify the consistency bound, the algorithm compares the makespan of the new tentative assignment to the makespan $\overline{\texttt{OPT}}_C$ of the initial assignment, assuming that the speed predictions are correct.

Algorithm 1 ITERATIVE-PARTIAL-REBALANCING (IPR)

Input: predicted machine speeds $\hat{s}_1 \geq \cdots \geq \hat{s}_m$, job processing times $p_1, \ldots, p_n$, consistency $1+\alpha$, accuracy $\epsilon \in (0,1)$, maximum bag size ratio $\rho \geq 1$.

$\{B_1, \ldots, B_m\} \leftarrow$ a $(1+\epsilon)$-consistent partition such that $p(B_1) \geq \ldots \geq p(B_m)$
$\overline{\texttt{OPT}}_C \leftarrow \max_{i\in[m]} p(B_i)/\hat{s}_i$
$\mathcal{M}_1, \ldots, \mathcal{M}_m \leftarrow \{B_1\}, \ldots, \{B_m\}$
while $\max_{B\in\cup_i \mathcal{M}_i, |B|\geq 2} p(B) > \rho \min_{B\in\cup_i \mathcal{M}_i} p(B)$:
  $\mathcal{M}'_1, ..., \mathcal{M}'_m \leftarrow$ LPT-REBALANCE$(\mathcal{M}_1, \ldots, \mathcal{M}_m)$
  if $\max_{i\in[m]} \sum_{B\in\mathcal{M}'_i} p(B)/\hat{s}_i > (1+\alpha)\overline{\texttt{OPT}}_C$:
    $\{B_1, \ldots, B_m\} \leftarrow \cup_{i\in[m]} \mathcal{M}_i$
    return $\{B_1, \ldots, B_m\}$
  $\mathcal{M}_1, \ldots, \mathcal{M}_m \leftarrow \mathcal{M}'_1, \ldots, \mathcal{M}'_m$
$\{B_1, \ldots, B_m\} \leftarrow \cup_{i\in[m]} \mathcal{M}_i$
return $\{B_1, \ldots, B_m\}$

The LPT-Rebalance subroutine. This subroutine first moves the bag $B_{\min}$ with the smallest processing time to the collection of bags $\mathcal{M}_{\max}$ that contains the bag with the largest processing time among the bags that contain at least two jobs. Let ℓ be the number of bags in $\mathcal{M}_{\max}$, including $B_{\min}$. The subroutine then balances the processing time of the bags in $\mathcal{M}_{\max}$ by running the Longest Processing Time first (LPT) algorithm over all jobs in bags in $\mathcal{M}_{\max}$, i.e. jobs in $\cup_{B\in\mathcal{M}_{\max}} B$, to create ℓ new, balanced, bags that are placed in $\mathcal{M}_{\max}$. LPT-REBALANCE finally returns the updated assignment of bags to machines $\mathcal{M}_1, \ldots, \mathcal{M}_m$. We note that among these m collections of bags, only two, $\mathcal{M}_{\min}$ and $\mathcal{M}_{\max}$, are modified. We use Figure 1 to illustrate this rebalancing procedure.

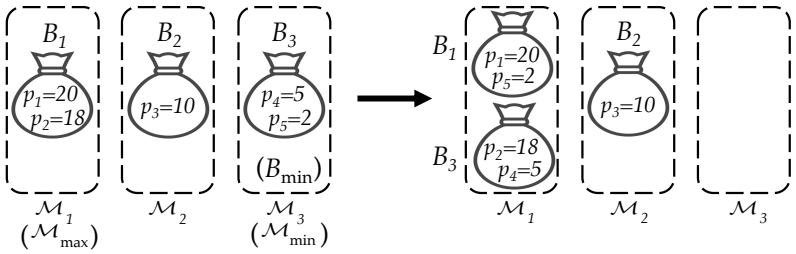

Fig. 1. Illustration of one iteration of the IPR algorithm on an example with $m = 3$ bags and machines and $n = 5$ jobs.

Algorithm 2 LPT-REBALANCE

Input: assignments of bags $\mathcal{M}_1, \ldots, \mathcal{M}_m$

1: $B_{\min} \leftarrow \text{argmin}_{B \in \cup_i \mathcal{M}_i} p(B)$
2: $\mathcal{M}_{\min} \leftarrow$ the collection of bags $\mathcal{M}$ such that $B_{\min} \in \mathcal{M}$
3: $\mathcal{M}_{\max} \leftarrow \text{argmax}_{\mathcal{M}_i : i \in [m]} \max_{B \in \mathcal{M}_i : |B| \geq 2} p(B)$
4: $\mathcal{M}_{\max} \leftarrow \mathcal{M}_{\max} \cup \{B_{\min}\}, \mathcal{M}_{\min} \leftarrow \mathcal{M}_{\min} \setminus \{B_{\min}\}$
5: $J_{\max} \leftarrow \cup_{B \in \mathcal{M}_{\max}} B, \ell \leftarrow |\mathcal{M}_{\max}|$
6: $B'_1, \ldots, B'_\ell \leftarrow \{\}, \ldots, \{\}$
7: **while** $|J_{\max}| > 0$ **do**
8: $\quad j' \leftarrow \text{argmax}_{j \in J_{\max}} p_j$
9: $\quad B' \leftarrow \text{argmin}_{B \in \{B'_1, \ldots, B'_\ell\}} p(B)$
10: $\quad B' \leftarrow B' \cup \{j'\}, J_{\max} \leftarrow J_{\max} \setminus \{j'\}$
11: $\mathcal{M}_{\max} \leftarrow \{B'_1, \ldots, B'_\ell\}$
12: **return** $\mathcal{M}_1, \ldots, \mathcal{M}_m$

4.2 Analysis of the algorithm

We first show that IPR with parameter $\rho = 4$ in the general case is a $(1+\epsilon)(1+\alpha)$-consistent and $(2+2/\alpha)$-robust partitioning algorithm (Lemma 1 and Theorem 2). Then, we use these consistency and robustness guarantees to obtain the $\min\{\eta^2(1+\epsilon)(1+\alpha), (1+\epsilon)(2+2/\alpha)\}$ approximation as a function of the prediction error η (Theorem 3). Finally, we analyze the running time (Lemma 5). The main challenge is to analyze IPR's robustness.

The consistency The consistency almost comes from the definition of IPR.

Lemma 1. *For any constants $\alpha, \epsilon \in (0,1)$,* IPR *is a $(1+\epsilon)(1+\alpha)$-consistent partitioning algorithm.*

Proof. To prove the consistency, we consider the final tentative assignment of the bags on the machines $\mathcal{M}_1, \ldots, \mathcal{M}_m$ when IPR terminates. With true speeds $\mathbf{s}$, the makespan of this schedule is $\max_{i \in [m]} \sum_{B \in \mathcal{M}_i} p(B)/s_i$. When the speed predictions are correct, i.e., $\mathbf{s} = \hat{\mathbf{s}}$, we have

$$\begin{aligned}\max_{i \in [m]} \frac{\sum_{B \in \mathcal{M}_i} p(B)}{s_i} &= \max_{i \in [m]} \frac{\sum_{B \in \mathcal{M}_i} p(B)}{\hat{s}_i} \\ &\leq (1+\alpha)\overline{\texttt{OPT}}_C \\ &\leq (1+\alpha)(1+\epsilon) opt(\mathbf{p}, \mathbf{s}).\end{aligned}$$

Line 6 of IPR enforces the first inequality. For the second inequality, observe that when $\mathbf{s} = \hat{\mathbf{s}}$, $\overline{\texttt{OPT}}_C$ is the makespan of the initial assignment, which is a $1+\epsilon$ approximation to the optimal makespan $opt(\mathbf{p}, \mathbf{s})$. Since there exists an assignment of the bags returned by IPR that achieves a $(1+\epsilon)(1+\alpha)$ approximation when $\mathbf{s} = \hat{\mathbf{s}}$, IPR is a $(1+\epsilon)(1+\alpha)$-consistent partitioning algorithm.

The robustness The main part of the analysis is to bound the algorithm's robustness. First, we show that the ratio $\beta(\mathcal{B}) = \frac{\max_{B\in\mathcal{B},|B|\geq 2} p(B)}{\min_{B\in\mathcal{B}} p(B)}$ of the maximum total processing time of a bag containing at least two jobs to the minimum total processing time of a bag can be used to bound the robustness of any partition $\mathcal{B}$.

Lemma 2. *Let $\mathcal{B} = \{B_1, \cdots, B_m\}$ be a partition of n jobs with processing times $p_1, \ldots p_n$ into m bags. Then $\mathcal{B}$ is a $\max\{2, \beta(\mathcal{B})\}$-robust partition, where $\beta(\mathcal{B}) = \frac{\max_{B\in\mathcal{B},|B|\geq 2} p(B)}{\min_{B\in\mathcal{B}} p(B)}$.*

By Lemma 2, it remains to bound the ratio β of the bags $\mathcal{B}$ returned by IPR. Let $\mathcal{B}^{(i)}$ denote the collection of all bags B at iteration i of the algorithm and define $b^{(i)}_{\min} = \min_{B\in\mathcal{B}^{(i)}} p(B)$ to be the minimum processing time of a bag at each iteration i. To bound the ratio β, we first show in Lemma 3 that $b^{(i)}_{\min}$ is non-decreasing in i.

Lemma 3. *At each iteration i of* IPR *with $\rho = 4$, $b^{(i+1)}_{\min} \geq b^{(i)}_{\min}$.*

Using Lemma 3, we bound the size ratio β needed for Lemma 2.

Lemma 4. *Let $\mathcal{B}_{\text{IPR}} = \{B_1, \ldots, B_m\}$ be the partition of the n jobs returned by* IPR *with $\rho = 4$. Then, we have that $\beta(\mathcal{B}_{\text{IPR}}) \leq 2 + 2/\alpha$.*

We are now ready to show the algorithm's robustness.

Theorem 2. *For any constants $\alpha, \epsilon \in (0, 1)$,* IPR *with $\rho = 4$ is a $(2+2/\alpha)$-robust partitioning algorithm.*

Proof. Let $\mathcal{B}_{\text{IPR}} = \{B_1, \ldots, B_m\}$ be the partition of the n jobs returned by IPR with $\rho = 4$. By Lemma 4, we have that $\beta(\mathcal{B}_{\text{IPR}}) \leq 2 + 2/\alpha$. Thus, by Lemma 2, the robustness of IPR with $\rho = 4$ is $2 + 2/\alpha$.

The approximation as a function of the prediction error We extend the consistency and robustness results for IPR to obtain our main result. We show that for the SSP problem, the algorithm that runs IPR in the partitioning stage and then a PTAS in the scheduling stage achieves an approximation ratio that gracefully degrades as a function of the prediction error η from $(1 + \epsilon)(1 + \alpha)$ to $(1 + \epsilon)(2 + 2/\alpha)$.

Theorem 3. *Consider the algorithm that runs* IPR *with $\rho = 4$ in partitioning stage and a PTAS for makespan minimization in scheduling stage. For any constant $\epsilon \in (0, 1)$ and any $\alpha \in (0, 1)$, this algorithm achieves a $\min\{\eta^2(1+\epsilon)(1+\alpha), (1+\epsilon)(2+2/\alpha)\}$ approximation for SSP where $\eta = \max_{i\in[m]} \frac{\max\{\hat{s}_i, s_i\}}{\min\{\hat{s}_i, s_i\}}$ is the prediction error.*

If we do not care about the computation runtime; that is, we can solve each scheduling problem optimally including the initial step of IPR in the partition stage and the scheduling stage, then our result improves to a $\min\{\eta^2(1 + \alpha), (2 + 2/\alpha)\}$ approximation.

The running time of IPR We show that the main algorithm performs $O(m^2)$ iterations, which implies that its running time is polynomial in n and m.

Lemma 5. *At most $O(m^2)$ iterations are needed for* IPR *with $\rho = 4$ to terminate.*

5 Improved Trade-offs for Special Cases

When all job processing times are either equal or infinitesimal, t the IPR algorithm with $\rho = 2$ achieves an improved robustness. The proofs of this section can be found in Section 5 of the full version.

Theorem 4. *If $p_j = 1$ for all $j \in [n]$, then, for any constant $\epsilon \in (0, 1)$ and any $\alpha \in (0, 1)$,* IPR *with $\rho = 2$ is $(1 + \epsilon)(1 + \alpha)$-consistent and $(2 + 1/\alpha)$-robust.*

Theorem 5. *If all jobs are infinitesimal, then, for any constant $\epsilon \in (0, 1)$ and any $\alpha \in (0, 1)$,* IPR *with $\rho = 2$ is $(1 + \epsilon)(1 + \alpha)$-consistent and $(1 + 1/\alpha)$-robust.*

When the machine speeds are in $\{0, 1\}$, we propose a different partitioning algorithm that is $(1 + \epsilon)$-consistent and $2(1 + \epsilon)$-robust for this special case.

Theorem 6. *For any constant $\epsilon > 0$, there is a $(1 + \epsilon)$-consistent and $2(1 + \epsilon)$-robust partitioning algorithm for the $\{0, 1\}$-speed SSP problem.*

We also provide a robustness lower bound for $\{0, 1\}$ speeds.

Theorem 7. *For any $\alpha \in [0, 1/2)$, if a deterministic algorithm for the $\{0, 1\}$-speed SSP problem is $(1 + \alpha)$-consistent, then its robustness is at least $(4 - 2\alpha)/3$.*

6 Experiments

We empirically evaluate the performance of IPR on synthetic data against benchmarks that achieve either the best-known consistency or the best-known robustness for SSP.

6.1 Experiment settings

Benchmarks We compare three algorithms. **IPR** is Algorithm 1 with $\rho = 4$ and $\alpha = 0.5$. The Largest Processing Time first partitioning algorithm, which we call **LPT-Partition**, creates m bags by adding each job, in decreasing order of their processing time, to the bag with minimum total processing time. LPT-Partition is 2-robust (and 2-consistent since it ignores the predictions) [11]. The **1-consistent** algorithm completely trusts the prediction and generates a partition that is 1-consistent (but has arbitrarily poor robustness due to our lower bound in Proposition 1). In practice, PTAS algorithms for scheduling are extremely slow. Instead of using a PTAS for the scheduling stage, we give an advantage to the two benchmarks by solving their scheduling stage via integer programming (IP). However, since we want to ensure that our algorithm has a polynomial running time, we use the LPT algorithm to compute a schedule during both the partitioning and scheduling stage of IPR, instead of a PTAS or an IP and we use IP to compute the optimal solution.

Data sets In the first set of experiments, we generate synthetic datasets with $n = 50$ jobs and $m = 10$ machines and evaluate the performance of the different algorithms as a function of the standard deviation of the the prediction error distribution. The job processing times p_j are generated i.i.d. either from $\mathcal{U}(0, 100)$, the uniform distribution in the interval $(0, 100)$, or $\mathcal{N}(50, 5)$, the normal distribution with mean $\mu_p = 50$ and standard deviation $\sigma_p = 5$. The machine speeds s_i are also generated i.i.d., either from $\mathcal{U}(0, 40)$ or $\mathcal{N}(20, 4)$. We evaluate the performance of the algorithms over each of the 4 possible combinations of job processing time and machine speed distributions. The prediction error $err(i) = \hat{s}_i - s_i$ of each machine is sampled i.i.d. from $\mathcal{N}(0, x)$ and we vary x from $x = 0$ to $x = \mu_s$ (the mean of machine speeds).

In the second set of experiments, we fix the distributions of the processing times, machine speeds, and prediction errors to be $\mathcal{N}(50, \sigma_p)$, $\mathcal{N}(20, \sigma_s)$, and $\mathcal{N}(0, 4)$ respectively, with default values of $\sigma_p = 5$ and $\sigma_s = 4$. We evaluate the algorithms' performance as a function of (1) the number n of jobs, (2) the number m of machines, (3) σ_p, and (4) σ_s. For each figure, the approximation ratio achieved by the different algorithms are averaged over 100 instances generated i.i.d. as described above. Additional details of the experiment setup are provided in Appendix D of the full version.

6.2 Experiment results

Experiment set 1 From the first row of Figure 2, we observe that, in all four settings, when we vary the magnitude of the prediction error, IPR outperforms LPT-Partition when the error is small and outperforms 1-consistent when the error is large. Since LPT-Partition does not use the predictions, its performance remains constant as a function of the prediction errors. Since 1-consistent completely trusts the predictions, it is optimal when the predictions are exactly correct but its performance deteriorates quickly as the prediction errors increase.

IPR combines the advantages of LPT-Partition and 1-consistent: when the predictions are relatively accurate, it is able to take advantage of the predictions and outperform LPT-Partition. When the predictions are increasingly inaccurate, IPR has a slower deterioration rate compared to 1-consistent. It is noteworthy that, in some settings, IPR simultaneously outperforms both benchmarks for a wide range of values of the standard deviation σ_{err} of the prediction error distribution. When the distributions of job processing times and machine speeds are $\mathcal{N}(50, 5)$ and $\mathcal{N}(20, 4)$ respectively, IPR achieves the best performance when $\sigma_{err}/\mu_s \geq 0.2$. When they are $\mathcal{N}(50, 5)$ and $\mathcal{U}(0, 40)$, IPR outperforms both benchmarks when $\sigma_{err}/\mu_s \geq 0.4$.

Experiment set 2 The number of jobs has almost no impact on the performance of any of the algorithms. However, the approximations achieved by the algorithms do improve as the number of machines m increases, especially for LPT-Partition. The reason is that m is also the number of bags, so when the number of bags increases, there is more flexibility in the scheduling stage, especially when the total processing times of the bags are balanced.

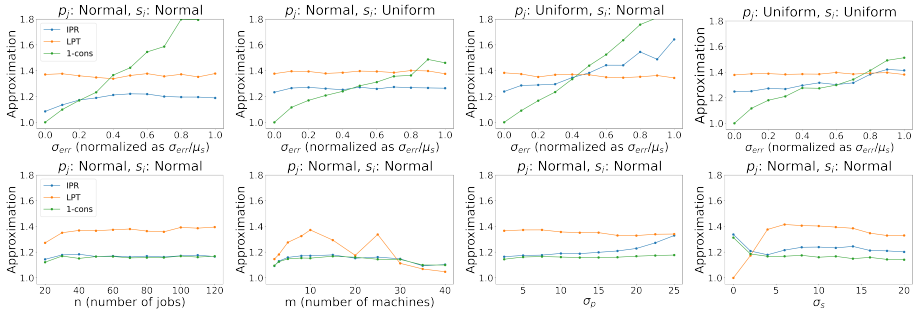

Fig. 2. The approximation ratio achieved by our algorithm, IPR, and the two benchmarks as a function of the standard deviation of the prediction error σ_{err} for different job processing time and true speed distributions (row 1) and as a function of the number of jobs n, the number of machines m, the standard deviation σ_p of the job processing time distribution, and the standard deviation σ_s of the true speed distribution (row 2).

IPR is the algorithm most sensitive to the standard deviation σ_p of the job processing times. It has performance close to that of 1-CONSISTENT when σ_p is small, and similar to LPT-PARTITION when σ_p is large. The approximation ratio of LPT-PARTITION increases as σ_s increases, while our algorithm and the 1-CONSISTENT partitioning algorithm are relatively insensitive to the change in σ_s. Since the LPT-PARTITION algorithm generates balanced bags of similar total processing times, it performs well when the machine speeds are all almost equal, but its performance then quickly degrades as σ_s increases. An additional set of experiments that studies the impact of the α parameter in the performance of the IPR algorithm can be found in Section 6 of the full version.

References

1. Albers, S., Schmidt, G.: Scheduling with unexpected machine breakdowns. Discrete Applied Mathematics **110**(2-3), 85–99 (2001)
2. Azar, Y., Leonardi, S., Touitou, N.: Flow time scheduling with uncertain processing time. In: Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing. pp. 1070–1080 (2021)
3. Azar, Y., Leonardi, S., Touitou, N.: Distortion-oblivious algorithms for minimizing flow time. In: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 252–274. SIAM (2022)
4. Azar, Y., Panigrahi, D., Touitou, N.: Online graph algorithms with predictions. Proceedings of the Thirty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (2022)
5. Bamas, E., Maggiori, A., Rohwedder, L., Svensson, O.: Learning augmented energy minimization via speed scaling. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 15350–15359. Curran Associates, Inc. (2020)
6. Bamas, E., Maggiori, A., Svensson, O.: The primal-dual method for learning augmented algorithms. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F.,

Lin, H. (eds.) Advances in Neural Information Processing Systems. pp. 20083–20094 (2020)
7. Banerjee, S., Gkatzelis, V., Gorokh, A., Jin, B.: Online nash social welfare maximization with predictions. In: Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022. SIAM (2022)
8. Dinitz, M., Im, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Faster matchings via learned duals. Advances in neural information processing systems **34**, 10393–10406 (2021)
9. Dinitz, M., Im, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Algorithms with prediction portfolios. arXiv preprint arXiv:2210.12438 (2022)
10. Dütting, P., Lattanzi, S., Paes Leme, R., Vassilvitskii, S.: Secretaries with advice. In: Proceedings of the 22nd ACM Conference on Economics and Computation. pp. 409–429 (2021)
11. Eberle, F., Hoeksma, R., Megow, N., Nölke, L., Schewior, K., Simon, B.: Speed-robust scheduling - sand, bricks, and rocks. In: Integer Programming and Combinatorial Optimization - 22nd International Conference, IPCO 2021, Atlanta, GA, USA, May 19-21, 2021, Proceedings. pp. 283–296 (2021)
12. Epstein, L., Levin, A., Marchetti-Spaccamela, A., Megow, N., Mestre, J., Skutella, M., Stougie, L.: Universal sequencing on an unreliable machine. SIAM Journal on Computing **41**(3), 565–586 (2012)
13. Fotakis, D., Gergatsouli, E., Gouleakis, T., Patris, N.: Learning augmented online facility location. CoRR **abs/2107.08277** (2021), `https://arxiv.org/abs/2107.08277`
14. Hochbaum, D.S., Shmoys, D.B.: A polynomial approximation scheme for scheduling on uniform processors: Using the dual approximation approach. SIAM Journal on Computing **17**(3), 539–551 (1988)
15. Im, S., Kumar, R., Montazer Qaem, M., Purohit, M.: Non-clairvoyant scheduling with predictions. In: Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures. pp. 285–294 (2021)
16. Im, S., Kumar, R., Montazer Qaem, M., Purohit, M.: Online knapsack with frequency predictions. Advances in Neural Information Processing Systems **34** (2021)
17. Jin, B., Ma, W.: Online bipartite matching with advice: Tight robustness-consistency tradeoffs for the two-stage model. arXiv preprint arXiv:2206.11397 (2022)
18. Lattanzi, S., Lavastida, T., Moseley, B., Vassilvitskii, S.: Online scheduling via learned weights. In: Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1859–1877 (2020)
19. Lindermayr, A., Megow, N., Rapp, M.: Speed-oblivious online scheduling: Knowing (precise) speeds is not necessary. arXiv preprint arXiv:2302.00985 (2023)
20. Lykouris, T., Vassilvtiskii, S.: Competitive caching with machine learned advice. In: International Conference on Machine Learning. pp. 3296–3305. PMLR (2018)
21. Mitzenmacher, M.: Scheduling with Predictions and the Price of Misprediction. In: 11th Innovations in Theoretical Computer Science Conference (ITCS 2020). Leibniz International Proceedings in Informatics (LIPIcs), vol. 151, pp. 14:1–14:18 (2020)
22. Mitzenmacher, M., Vassilvitskii, S.: Algorithms with predictions. arXiv preprint arXiv:2006.09123 (2020)
23. Purohit, M., Svitkina, Z., Kumar, R.: Improving online algorithms via ml predictions. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. Curran Associates, Inc. (2018)
24. Stein, C., Zhong, M.: Scheduling when you do not know the number of machines. ACM Trans. Algorithms (2019)