Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Uniform Machine Scheduling with Predictions

Tianming Zhao, Wei Li, Albert Y. Zomaya

School of Computer Science, The University of Sydney, Australia
tzha2101@uni.sydney.edu.au, weiwilson.li@sydney.edu.au, albert.zomaya@sydney.edu.au

Abstract

The revival in learning theory has provided us with improved capabilities for accurate predictions. This work contributes to an emerging research agenda of online scheduling with predictions by studying the makespan minimization in uniformly related machine non-clairvoyant scheduling with job size predictions. Our task is to design online algorithms that effectively use predictions and have performance guarantees with varying prediction quality. We first propose a simple algorithm-independent prediction error measurement to quantify prediction quality. To effectively use the predicted job sizes, we design an offline improved 2-relaxed decision procedure approximating the optimal schedule. With this decision procedure, we propose an online $O(\min\{\log\eta, \log m\})$-competitive algorithm that assumes a known prediction error. Finally, we extend this algorithm to construct a robust $O(\min\{\log\eta, \log m\})$-competitive algorithm that does not assume a known error. Both algorithms require only moderate predictions to improve the well-known $\Omega(\log m)$ lower bound, showing the potential of using predictions in managing uncertainty.

Introduction

Managing uncertainty is the focus of online optimization. Traditional methods strive to bound the worst-case performance against the uncertainties from problem input (Borodin and El-Yaniv 1998; Leung, Kelly, and Anderson 2004). However, dealing with uncertainty incurs a high cost compared to when such information is known (Karp 1992; Awerbuch, Kutten, and Peleg 1992). Thus, researchers recently studied augmenting algorithms with advice on unknown input (Boyar et al. 2016; Antoniadis et al. 2020). Several works (Gollapudi and Panigrahi 2019; Mitzenmacher 2020; Dütting et al. 2021) have shown that additional information can boost the overall algorithm performance and reduce the cost incurred in managing uncertainties. Meanwhile, recent development in learning theory has made accurate predictions in many fields (Frye et al. 2019; Anand, Ge, and Panigrahi 2020). Combining techniques from the advice model and learning theory, the new online optimization with predictions framework emerges.

One of the most significant fields boosted by the above approach is online scheduling. Multiple works have shown that using predictions improves the theoretical performance bounds. (Purohit, Svitkina, and Kumar 2018) developed a preferential round-robin algorithm using job size predictions for single machine scheduling to minimize total completion time. (Lattanzi et al. 2020) developed an online rounding algorithm using machine weight predictions for parallel machine scheduling under restricted assignment to minimize makespan. (Azar, Leonardi, and Touitou 2021) developed greedy and binning algorithms using job size predictions for single machine scheduling to minimize weighted flow time. (Im et al. 2021) revisited the problem of single machine scheduling to minimize total completion time, and developed a new prediction error measurement and an improved round-robin algorithm using job size predictions. All these works reduced the attainable competitive ratios via predictions. This work extends scheduling with predictions to another online scheduling problem: makespan minimization in uniformly related machine non-clairvoyant scheduling. We study how to use the imperfect predictions to improve online optimization and present the first such algorithm for uniformly related machine non-clairvoyant scheduling.

We develop algorithms with job size predictions, yielding a significantly improved performance bound. The development begins with defining a simple algorithm-independent prediction error measurement η. With this error measurement and the predicted job sizes, the algorithm simultaneously schedules jobs and computes underestimations on actual job sizes and the error η on the fly. We propose an (improved) offline approximation compared to (Shmoys, Wein, and Williamson 1995) and two online algorithms: one assumes a known prediction error, and the other one does not. We prove that our proposed online algorithms achieve $O(\min\{\log\eta, \log m\})$ competitive ratio with m machines, improving the $\Omega(\log m)$ lower bound. Our contributions are summarized below.

1. An improved offline 2-relaxed decision procedure for approximating the optimal schedule. (Theorem 1)
2. An online $O(\min\{\log\eta, \log m\})$-competitive algorithm with the known prediction error. (Theorem 2)
3. A robust online $O(\min\{\log\eta, \log m\})$-competitive algorithm with unknown prediction error. (Theorem 3)
4. Conduct experiments to evaluate the theoretical results and the practicality of the proposed algorithms.

Preliminaries

Problem Definition

We study the makespan minimization problem in uniformly-related machine non-clairvoyant scheduling. There are m uniformly related parallel machines and n independent jobs. A machine is denoted by M_i, $1 \leq i \leq m$, and a job is denoted by J_i, $1 \leq i \leq n$. Job J_i has size p_i^*, but this value remains unknown until J_i is completed. This setting is known as *non-clairvoyant* in literature. Machines are heterogeneous in processing power, and machine M_i has a processing power of s_i. For easy understanding, we let $s_1 \geq s_2 \geq ... \geq s_m$. If job J_j is assigned to machine M_i, the processing time for this job is $\frac{p_j^*}{s_i}$. The jobs are priority-free, preemptive-restart, and ready at time 0. By preemptive-restart, we mean those jobs are non-preemptive but can be restarted later on any machine. Our objective is to minimize the *makespan*, $C_{\max}$, the time of the last job completes. Our scheduling problem can be defined as $Qm \mid \textit{online-time-nclv}, \textit{pmtn-restart} \mid C_{\max}$ (Graham et al. 1979). In addition, there is a Machine Learning (ML) oracle accessible to the system. The scheduler is allowed to make decisions using the predictions of some parameters to manage uncertainty. However, the prediction is imperfect. The performance of the scheduler could be adversely affected by the quality of predictions. We will discuss the measurements of algorithm performance and prediction quality below.

ML Oracle and Prediction Error

When integrating ML oracle into a scheduler, one must consider what to predict and how the prediction quality is measured. In principle, we expect the predictions to improve the competitive ratio. The scheduler should guarantee a bounded performance under poor predictions and a near-optimal solution with accurate predictions. Last, the learning problem itself must be actionable. A counterexample is to predict the optimal schedule, which simplifies the scheduling algorithm but makes the learning problem extremely hard.

This work considers using ML oracle to predict job sizes. Recent works (Amiri and Mohammad-Khanli 2017; Peyravi and Moeini 2020; Yamashiro and Nonaka 2021) have shown that job sizes are highly predictable in many scenarios, e.g., cloud, clusters, and factories. In addition, when the prediction is accurate, we have the 2-relaxed decision procedure guaranteeing a near-optimal makespan, and when the prediction goes arbitrarily bad, the existing $O(\log m)$-competitive algorithm can bound the performance.

Our online algorithms use the ML oracle to predict the actual job size p_j^* as p_j. Similar to the work (Lattanzi et al. 2020) that measures the error by ratios, we define the error for job J_j as $\eta_j = \max\{\frac{p_j}{p_j^*}, \frac{p_j^*}{p_j}\}$, and the overall prediction error η as

$$\eta = \max_{1 \leq j \leq n} \eta_j = \max_{1 \leq j \leq n} \max\{\frac{p_j}{p_j^*}, \frac{p_j^*}{p_j}\}$$

Observe that $\eta_j \geq 1$ for any j, so that $\eta \geq 1$. The prediction is perfect if and only if $\eta = 1$. Please note that the prediction error measurement is simple and algorithm-independent.

Performance Evaluation

We evaluate the performance of the algorithms by the competitive framework and the metrics of γ-robustness and β-consistency (Purohit, Svitkina, and Kumar 2018).

Competitive Framework We study the performance of online algorithms under the competitive framework (Borodin and El-Yaniv 1998). An online algorithm A will compare against an optimal offline algorithm A^*. Algorithm A^* knows the exact job sizes, and it produces the schedule with the minimal makespan. Let $\text{cost}(I, A)$ denote the makespan obtained by A on problem instance I. We say an online algorithm A is c-competitive if for any problem instance I, it satisfies $\text{cost}(I, A) \leq c \cdot \text{cost}(I, A^*)$, for some function c of problem input and prediction error η.

γ-robustness and β-consistency The work (Purohit, Svitkina, and Kumar 2018) proposed two metrics to measure how well an online algorithm performs against the changes in prediction quality. The robustness measures the performance with the worst prediction, while the consistency measures it with the perfect prediction. Specifically, we say an algorithm is γ-robust if its competitive ratio $c \leq \gamma$ for any η, and is β-consistent if $c = \beta$ for $\eta = 1$.

Related Results

Let us review some important related results: (1) the problem is NP-complete in the strong sense for m arbitrary, (2) the offline version of the problem has a polynomial-time 2-relaxed decision procedure, (3) the online version has a lower bound $\Omega(\log m)$ on the competitive ratio, and (4) there exists an $O(\log m)$-competitive algorithm matching this lower bound.

For the computational complexity, consider the simpler offline problem with identical machines where the job sizes are known. This problem is NP-complete in the strong sense for m arbitrary (Garey and Johnson 2009). A simple reduction to our problem shows that the offline version of the uniformly related machine scheduling is NP-complete in the strong sense for m arbitrary.

There exists a 2-relaxed decision procedure for the offline version (Shmoys, Wein, and Williamson 1995). An algorithm is a 2-relaxed decision procedure if given a makespan d, the algorithm either determines that no schedule of d makespan exists or produces a schedule of $2d$ makespan. With this procedure, the optimal makespan can be found using a bisection method, as well as a 2-approximation solution. The online algorithms will use this 2-relaxed decision procedure to approximate a near-optimal schedule.

The best result that resolved this problem for decades was (Shmoys, Wein, and Williamson 1995). It was shown that any online deterministic algorithm has a competitive ratio $\Omega(\log m)$, and the authors gave an $O(\log m)$-competitive algorithm matching this lower bound. This algorithm is therefore asymptotically optimal in non-clairvoyant scheduling. Recent advances in learning theory lead to better prediction accuracy, thus enabling new ways to minimize makespan.

Robust Online Scheduling Algorithms with Job Size Predictions

Algorithm Overview

The development of our algorithms involves three stages. The first stage assumes that all job size p_j^* and the optimal makespan d are known. There is a 2-relaxed decision procedure for producing a schedule of length at most $2d$ (Shmoys, Wein, and Williamson 1995). We will use this procedure, with minor improvement, as our base algorithm. The second stage solves the problem in an online manner by no longer using the actual job sizes and the optimal makespan d. Instead, we use the predicted job sizes p_j and the known prediction error η. Note that accessing the prediction error η is a reasonable assumption as the ML oracle is often trained offline on large labelled datasets. The prediction error in the training stage will likely reflect the prediction error in run time. We propose an $O(\min\{\log \eta, \log m\})$-competitive algorithm for this case. The algorithm estimates the actual job sizes in at most $O(\min\{\log \eta, \log m\})$ rounds. The 2-relaxed decision procedure is executed each round while the optimal makespan is searched online using a standard doubling method. Estimating the job sizes incurs a cost of $O(\min\{\log \eta, \log m\})$ factor in makespan, while the doubling method incurs at most a cost of 2. In the last stage, we further remove the assumption of knowing the prediction error η in the general case. The prediction error is then searched online by using a doubling method. The search will impose an extra $\log \eta$ term in the competitive ratio. To provide an $O(\log m)$ strong worst-case guarantee, we switch to the guaranteed $O(\log m)$-competitive algorithm once we believe the prediction error is large enough. This will give us a general $O(\min\{\log \eta, \log m\})$-competitive algorithm.

An Improved 2-Relaxed Procedure

Consider the offline schedule for all job sizes p_j^* are known and the makespan d is given. Our proposed 2-relaxed procedure ensures to produce a schedule of length at most $2d$, or otherwise have no d-length schedule exist. Our procedure works as follows. When a machine M_i is idle, it starts processing either the largest unfinished job that can be completed in period d or the smallest job not yet started. If multiple machines are idle, we order them in the non-increasing order of their speed. Specifically, machine M_i will first consider the largest unfinished job J_j with $p_j^* \leq s_i \cdot d$ that is either not started or terminated and rerun the currently processed job on another machine M_k with $p_j^* > s_k \cdot d$. If there is no such job, the machine will process the smallest job not started. Otherwise, machine M_i stays idle when all above conditions are not met. The entire procedure lasts until all jobs are processed or until time $2d$. If at time $2d$, there are still unfinished jobs, the procedure outputs **no**. Otherwise, it has constructed a schedule of length at most $2d$.

We outline the improvement of our 2-relaxed procedure over the original one (Shmoys, Wein, and Williamson 1995). First, our procedure uses the largest job first strategy to assign those large jobs to fast machines for processing. It provides slow machines opportunities to complete relatively small jobs that could be allocated to fast machines. The workload reduction in fast machines contributes to small makespan as using fast machines to process small jobs increases the makespan. Second, our procedure can invoke idle machines to complete large jobs processing in $2d$ but not d time. In contrast, the procedure (Shmoys, Wein, and Williamson 1995) leaves the machine idle. If the makespan of a schedule is at most $2d$, the schedule generated by our procedure is strictly shorter than that generated by (Shmoys, Wein, and Williamson 1995). The detailed arguments are omitted due to the page limit. The pseudo-code is shown in Algorithm 1. Our first result is the correctness of the decision procedure stated as follows.

Algorithm 1: Improved 2-relaxed procedure

Require: Actual job sizes p_j^* and a makespan d
Ensure: A schedule of length at most $2d$, or output **no** indicating that it is impossible to construct a schedule of length d.
Define Function *getJob*(i) {subroutine that returns the next job for machine M_i}
$J_{\text{next1}} \leftarrow J_j$ with $p_j^* \leq s_i \cdot d$, J_j not begin, or is being processed on M_k with $p_j^* > s_k \cdot d$, p_j^* maximal
$J_{\text{next2}} \leftarrow J_j$ with $p_j^* > s_i \cdot d$, J_j not begin, p_j^* minimal
return the first nonempty $J_{\text{next}p}$ (smallest p) or null
EndFunction
while Time $\leq 2d$ **do**
  if all jobs have been completed **then**
    return {the whole schedule finishes}
  end if
  for all idle machine M_i ($1 \leq i \leq n$) **do**
    $J_j \leftarrow$ *getJob*(i)
    M_i starts processing J_j or stays idle if J_j is null
  end for
  increase Time to the next event
end while
return no

Theorem 1 (Theorem 4 restated)**.** *The improved 2-relaxed procedure can produce a schedule of length at most $2d$, or otherwise ensures no d-length schedule exist.*

Online Scheduling with Job Size Predictions and a Known η

We now consider the online problem for all job sizes and the optimal makespan are unknown. Instead, the scheduler knows the predicted job sizes p_j and the prediction error η. The pseudo-code is shown in Algorithm 2. Initially, the estimated size of job J_j is as $p_j^e = \frac{p_j}{\eta}$. The estimation of p_j^e will last k rounds of doubling until $p_j^* \leq 2^k \cdot \frac{p_j}{\eta} < 2 \cdot p_j^*$. The number of rounds can be bounded and we will show that $k \leq [2 \log \eta + 2]$. Meanwhile, the algorithm also estimates the optimal makespan $C_{\max}^*$ in rounds. Initially, we set the estimated makespan $d = \frac{\max_{1 \leq j \leq n} p_j}{\eta \cdot s_1}$, where $d \leq C_{\max}^*$. Then we run a similar procedure as in Algorithm 1 to test if the estimated makespan is achievable. This estimated makespan will undergo rounds of doubling until $C_{\max}^* \leq 2^{k'} \cdot \frac{\max_{1 \leq j \leq n} p_j}{\eta \cdot s_1} < 2 \cdot C_{\max}^*$ for some k'. Finally, to bound the number of doubling rounds for job size estimation, we refine *getJob* function in Algorithm 1. This refinement induces some fast machines to carry out more jobs.

Algorithm 2: Online scheduling with known error

Require: Predicted job sizes p_j and the prediction error η
Ensure: A schedule with makespan $O(\min\{\log\eta, \log m\})C^*_{\max}$
1: sup $\leftarrow \min\{k \mid \sum_{i=1}^{k} s_i \geq \frac{1}{2}\sum_{i=1}^{m} s_i, 1 \leq k \leq m\}$
Define Function *getJob*(i) {subroutine that returns the next job for machine M_i}
2: **if** $i \leq$ sup **then**
3: $J_{\text{next1}} \leftarrow J_j$ with $p_j^e \leq s_i \cdot d$, J_j not begin, or is being processed on M_k with $p_j^e > s_k \cdot d$ or $k >$ sup, p_j^e maximal
4: **else**
5: $J_{\text{next1}} \leftarrow J_j$ with $p_j^e \leq s_i \cdot d$, J_j not begin or is being processed on M_k with $p_j^e > s_k \cdot d$, p_j^e maximal
6: **end if**
7: $J_{\text{next2}} \leftarrow J_j$ with $p_j^e > s_i \cdot d$, J_j not begin, p_j^e minimal
8: **return** the first nonempty $J_{\text{next}p}$ (smallest p) or null
EndFunction
9: $p_j^e \leftarrow \max\{p_j^e, \frac{p_j}{\eta}\}$, $\forall 1 \leq j \leq n$ {job size estimates}
10: $d \leftarrow \frac{\max_{1\leq j\leq n} p_j}{\eta \cdot s_1}$ {optimal makespan estimate}
11: **while** there are unfinished jobs **do**
12: **while** *True* **do**
13: Time $\leftarrow 0$
14: **while** Time $\leq 2d$ **do**
15: **if** all jobs have been completed **then**
16: **return** {the whole schedule finishes}
17: **end if**
18: **for all** machine M_i ($1 \leq i \leq n$) **do**
19: $J_j \leftarrow$ *getJob*(i)
20: **if** J_j not null and M_i is not processing J_j **then**
21: M_i starts processing J_j
22: **end if**
23: **end for**
24: increase Time to the next event
25: **end while**
26: **if** there is an unfinished job not started by any machine M_i with $i \leq$ sup and $p_j^e \leq s_i \cdot d$ before time d **or** machine M_1 processes any job for more than d time **then**
27: $d \leftarrow 2d$ {double the makespan estimate}
28: **break**
29: **else**
30: **for all** job J_j been processing on M_i for time t **do**
31: $p_j^e \leftarrow \max\{2p_j^e, 2s_i \cdot t\}$
32: **end for**
33: **end if**
34: **end while**
35: **end while**

Define sup $= \min\{k \mid \sum_{i=1}^{k} s_i \geq \frac{1}{2}\sum_{i=1}^{m} s_i, 1 \leq k \leq m\}$ (according to (Shmoys, Wein, and Williamson 1995)). For machine M_i with $i >$ sup, the definition of *getJob* remains the same. Otherwise, the definition of J_{next1} changes to $J_{\text{next1}} \leftarrow J_j$ with the maximal p_j^* such that $p_j^* \leq s_i \cdot d$, where J_j is not started or currently processed on another $M_{k''}$ with $p_j^* > s_k \cdot d$ or $k'' >$ sup. It means that a fast machine M_i ($i \leq$ sup) can cancel a running job on a slow machine $M_{k''}$ ($k'' >$ sup) regardless its completion time. The benefit of this refinement is to let the number of doubling rounds for job size estimation bounded by $O(\min\{\log\eta, \log m\})$.

For easy discussion, we call the loop from Line 14 in Algorithm 2 the *inner procedure*. This inner procedure will run multiple rounds until all jobs are completed. After each round, the estimated job sizes are updated, or the estimated optimal makespan is updated. These updates could affect the output of *getJob* in the next round. The inner procedure continues checking on every machine M_i if its processing job returned by the current *getJob*. If not, the running job is cancelled, and the machine will process the new one returned by *getJob*. We will show that the doubling strategy for estimating the actual job sizes incurs a cost of $O(\min\{\log\eta, \log m\})$ factor in the competitive ratio, and the doubling strategy for estimating the optimal makespan incurs only a cost of 2 factors. Combining these, we obtain the following significant result.

Algorithm 3: Online scheduling with unknown error

Require: Predicted job sizes p_j
Ensure: A schedule with makespan $O(\min\{\log\eta, \log m\})C^*_{\max}$
1: $\eta^e \leftarrow 1$
2: **while** $(\eta^e)^2 < m$ **do**
3: run Algorithm 2 with p_j and prediction error η^e, and stops by confirming an underestimation of prediction error when finding a job J_j with $\frac{p_j}{\eta} > p_j^*$.
4: **if** all jobs have been completed **then**
5: **return** {the whole schedule finishes}
6: **end if**
7: $\eta^e \leftarrow 2\eta^e$
8: **end while**
9: **if** there are unfinished jobs **then**
10: run Algorithm 2 with $p_j = 0$ for all j and $\eta = \infty$.
11: **end if**
12: **return** {the whole schedule finishes}

Theorem 2 (Theorem 10 restated)**.** *Given the predicted job sizes p_j and the prediction error $\eta = \max_{1\leq j\leq n} \max\{\frac{p_j}{p_j^*}, \frac{p_j^*}{p_j}\}$, Algorithm 2 achieves $O(\min\{\log\eta, \log m\})$ competitive ratio in online makespan minimization for uniformly related machine scheduling.*

Online Scheduling with Job Size Predictions and Unknown η

We now consider a general situation where the scheduler has no (little) knowledge of the prediction error. The pseudocode is given in Algorithm 3. We will first assume the prediction is perfect, i.e., $\eta^e = 1$. Then run Algorithm 2 with η^e until we find that $\eta^e = 1$ is an underestimation. To confirm this finding, we need to identify if any job J_j can complete on machine M_i within time t by $s_i \cdot t < \frac{p_j}{\eta^e}$. Once confirmed, Algorithm 2 stops, η^e is doubled, and it runs again with the updated η^e. These activities will repeat at most $O(\log\eta)$ times before obtaining the actual prediction error. To also bound the performance for arbitrarily bad prediction error, Algorithm 3 assumes $\eta = \infty$ when it finds $(\eta^e)^2 \geq m$. With $\eta = \infty$, Algorithm 2 is $O(\log m)$-competitive. Finally, we obtain a general online algorithm with job size predictions.

Theorem 3 (Theorem 14 restated)**.** *Given only the predicted job sizes p_j, Algorithm 3 achieves $O(\min\{\log\eta, \log m\})$ competitive ratio in online makespan minimization for uniformly related machine scheduling, where η is the prediction error and $\eta = \max_{1\leq j\leq n} \max\{\frac{p_j}{p_j^*}, \frac{p_j^*}{p_j}\}$.*

Analysis

In this section, we shall prove the above theorems.

Theorem 4 (Correctness of the improved 2-relaxed procedure)**.** *The improved 2-relaxed procedure either produces a schedule of length at most $2d$, or otherwise ensures no d-length schedule exist.*

Proof. The whole schedule finishes only while Time $\leq 2d$. Thus it suffices to show that no valid d-length schedule exists if the procedure outputs **no**. For an unfinished job J_j, no valid d-length schedule exists if $p_j^* > s_1 \cdot d$. We thus assume $p_j^* \leq s_1 \cdot d$. Let k be the largest machine index that satisfies $p_j^* \leq s_k \cdot d$. For this statement to hold, machines $M_1, ..., M_k$ must be fully occupied by jobs with size at least p_j^* before time d, since otherwise the function *getJob* will make one of these k machines to process J_j before time d, completed by time $2d$. To construct a d-length schedule, any job with size at least p_j^* must be processed on one of the $M_1, ..., M_k$. Observe that no job is cancelled on the first k machines before time d by the existence of J_j and the definition of *getJob*. Then the total size for those jobs that must be processed by machines M_i ($i \leq k$) are already more than $\sum_{i=1}^{k} s_i \cdot d$, so no valid d-length schedule exists. □

Next, we show the competitive ratio of Algorithm 2. Recall that we refer the *inner procedure* to the while loop begins with '**while** Time $\leq 2d$'. There are a set of bounds to be proved. (1) For a given makespan d, the number of rounds of the inner procedure is bounded by either $O(\log \eta)$ or $O(\log m)$. (2) Each inner procedure incurs a cost of $O(1)C^*_{\max}$ on the makespan. (3) The doubling rounds for makespan d incurs a constant-factor cost in the makespan. Combining these results will establish the proof for the $O(\min\{\log \eta, \log m\})$ competitive ratio.

Lemma 5 (Bound on the number of inner procedure rounds by $\log \eta$)**.** *For a given makespan d, the number of inner procedure rounds is at most* $[2 \log \eta + 2]$.

Proof. Let the inner procedure execute once. We double the estimated makespan d when there is an unfinished job J_j not started on any machine M_i, with $i \leq$ sup and $p_j^e \leq s_i \cdot d$ before time d or machine M_1 completes any job after time d. Otherwise the inner procedure continues the next round. This happens only if there are at most sup unfinished jobs. The job size estimates will undergo k doubling rounds until either the whole schedule finishes or the makespan estimate doubles. Consider the inner procedure after the $(k-1)$-th round of doubling. For any unfinished job J_j, we claim that $p_j^e < p_j^*$. Since otherwise, $p_j^e \geq p_j^*$. Observe that, because the algorithm continues the next round without a break, J_j must start processing on some machine M_i with $p_j^e \leq s_i \cdot d$ before time d. Then at time $2d$, it will be completed since $p_j^* \leq p_j^e \leq s_i \cdot d$. This contradicts the definition of J_j. Finally, observe that each round at least double the estimated job size. We end up with

$$2^{k-1} \cdot \frac{p_j}{\eta} \leq p_j^e < p_j^*$$

Therefore we have

$$2^{k-1} < \eta \cdot \frac{p_j^*}{p_j} = \eta \cdot \eta_j \leq \eta^2 \implies k \leq [2 \log \eta + 1]$$

and the number of inner procedure rounds is at most $k+1 \leq [2 \log \eta + 2]$ □

Lemma 6 (Bound on the number of inner procedure rounds by $\log m$)**.** *For a given makespan d, the number of inner procedure rounds is at most* $[\log m + 2]$.

Proof. Let the inner procedure execute once. With similar arguments used in Lemma 5, the inner procedure continues the next round only if there are at most sup unfinished jobs running on machines $M_1, ..., M_{\text{sup}}$ for more than d time. Without loss of generality, let these unfinished jobs be $J_1, ..., J_r$ ($r \leq$ sup). After updating the job size estimates for the first time, it follows that $p_j^e \geq 2s_{\text{sup}} \cdot d$, $j \leq r$. The job size estimates will undergo k doubling rounds before either the whole schedule finishes or the makespan estimate doubles. Consider the inner procedure after the $(k-1)$-th round of doubling. Any unfinished job J_j must start processing on some machine M_i with $p_j^e \leq s_i \cdot d$. We let p_{j1}^e and $p_{j(k-1)}^e$ denote the job size estimate for J_j after the first and the $(k-1)$-th doubling respectively. We end up with

$$2^{k-2} \cdot 2 \cdot s_{\text{sup}} \cdot d \leq 2^{k-2} \cdot p_{j1}^e \leq p_{j(k-1)}^e \leq s_i \cdot d \leq s_1 \cdot d$$

Therefore we have

$$2^{k-1} \leq \frac{s_1}{s_{\text{sup}}} \implies k \leq [\log \frac{s_1}{s_{\text{sup}}} + 1]$$

Finally, by definition of sup, $\frac{s_1}{s_{\text{sup}}} \leq m$. Recall that sup $\leftarrow \min\{k \mid \sum_{i=1}^{k} s_i \geq \frac{1}{2}\sum_{i=1}^{m} s_i, 1 \leq k \leq m\}$. If $s_1 \geq \frac{1}{2}\sum_{i=1}^{m} s_i$, then sup $= 1$ and apparently $\frac{s_1}{s_{\text{sup}}} = 1 \leq m$. From now on, we assume $s_1 < \frac{1}{2}\sum_{i=1}^{m} s_i$. Suppose, for contradiction, that $\frac{s_1}{s_{\text{sup}}} > m$ or equivalently, $s_{\text{sup}} < \frac{s_1}{m}$. Then $s_{\text{sup}+1}, ..., s_m \leq s_{\text{sup}} < \frac{s_1}{m}$, so $\sum_{i=1}^{\text{sup}-1} s_i = \sum_{i=1}^{m} s_i - \sum_{i=\text{sup}}^{m} s_i > \sum_{i=1}^{m} s_i - m \cdot \frac{s_1}{m} > \frac{1}{2}\sum_{i=1}^{m} s_i$. This contradicts with the definition of sup, and it follows that $\frac{s_1}{s_{\text{sup}}} \leq m$. The number of inner procedure rounds, therefore, is at most $k + 1 \leq [\log \frac{s_1}{s_{\text{sup}}} + 2] \leq [\log m + 2]$. □

Lemma 7 (Bound on the number of inner procedure rounds)**.** *Fixing a makespan d, the number of inner procedure rounds is bounded by* $\min\{[2 \log \eta + 2], [\log m + 2]\}$.

Proof. It immediately follows from Lemma 5 and 6. □

Below we will prove that each inner procedure incurs a cost of $O(1)C^*_{\max}$ on the makespan. We consider the worst case where all jobs running on machines M_i ($i >$ sup) are cancelled by some machine M_k ($k \leq$ sup), as if only machines $M_1, ..., M_{\text{sup}}$ are processing. We will first show that if the optimal makespan is $C^*_{\max}$, then the optimal makespan is at most $3C^*_{\max}$ if only given machines $M_1, ..., M_{\text{sup}}$. It then follows that Algorithm 2 finishes the whole schedule when $d \geq 6C^*_{\max}$ and an inner procedure spends $O(1)C^*_{\max}$ time.

Lemma 8 (Bound on the optimal makespan with only fast machines)**.** *Let the optimal makespan be $C^*_{\max}$. Then the optimal makespan using only fast machines $M_1, ..., M_{sup}$ is at most $3C^*_{\max}$.*

Proof. To prove the statement, it is sufficient to show that there exists a schedule with makespan at most $3C^*_{\max}$. The schedule is constructed as follows. For the first $2C^*_{\max}$ time, run all the jobs arbitrarily on machines $M_1, ..., M_{\text{sup}}$. Specifically, if any machine becomes idle, it selects a job not started to process. If all jobs are being processed, the machine stays idle. No jobs are cancelled before time $2C^*_{\max}$. At time $2C^*_{\max}$, there must be at least one idle machine. Since otherwise, if all fast machine are busy at time $2C^*_{\max}$, the total job size must be more than $\sum_{i=1}^{\text{sup}} 2s_i \cdot C^*_{\max} > \sum_{i=1}^{m} s_i \cdot C^*_{\max}$. This contradicts with the optimal makespan $C^*_{\max}$. If there is at least one idle machine at time $2C^*_{\max}$, the number of unfinished jobs is at most sup $- 1$. Without loss of generality, we denote these unfinished jobs as $J_1, ..., J_r$ ($r <$ sup). Now, cancel these jobs. Consider the optimal schedule with all the machines $M_1, ..., M_m$. In this optimal schedule, jobs $J_1, ..., J_r$ must be assigned on some machines $M_{i1}, M_{i2}, ..., M_{il}$ with $i1 < i2 < ... < il$ and $il \leq r <$ sup. With only fast machines, we process these remaining jobs that are assigned to machine M_{i1} on machine M_1, jobs to machine M_{i2} on machine M_2, and so on. Formally, process the remaining jobs that are assigned to machine M_{ip} on machine M_p. Since $s_{ip} \geq s_p$, the time for processing these remaining jobs on fast machines with this assignment will be no more than $C^*_{\max}$. This constructs a at most $3C^*_{\max}$-length schedule using only the fast machines $M_1, ..., M_{\text{sup}}$. □

Lemma 9 (Bound on the estimated makespan)**.** *If $d \geq 6C^*_{\max}$, the whole schedule will finish after several rounds of inner procedure.*

Proof. The proof involves three main observations. (1) Given a problem instance I that has optimal makespan C, if all job sizes double, the new problem will have optimal makespan at most $2C$. (2) After the inner procedure terminates, any job size estimate follows $p^e_j < 2p^*_j$. (3) The makespan estimate doubles only when the algorithm confirms that it is impossible to construct a schedule of length d using only the fast machines $M_1, ..., M_{\text{sup}}$.

Observation (1) is trivial. If all job sizes double, the optimal schedule will also double. Thus, the optimal makespan for the new problem is at most $2C$. Observation (2) follows from the proof of Lemma 5. With a fixed makespan d, the estimated sizes will undergo k doubling rounds. We have shown that after the $(k-1)$-th round, any unfinished job J_j has $p^e_j < p^*_j$. Therefore, after the last round of doubling, it follows that $p^e_j < 2p^*_j$ for any unfinished job J_j. Observation (3) is supported by similar arguments in the proof of Theorem 4. If machine M_1 processes any job for more than d time, there is no valid schedule of length d. If there is an unfinished job J_j not started by any machine M_i with $i \leq$ sup and $p^e_j \leq s_i \cdot d$ before time d, then all the machines $M_1, ..., M_q$ (q is the largest index such that $p^e_j \leq s_q \cdot d$ and $q \leq$ sup) are busy processing jobs that cannot complete on slower machines before time d. Then it follows that a valid schedule of length d using only machines $M_1, ..., M_{\text{sup}}$ does not exist, and observation (3) holds.

Now we combine these observations. First, by Lemma 8, there exists a schedule of length $3C^*_{\max}$ using only the fast machines $M_1, ..., M_{\text{sup}}$. Since at any time, we have $p^e_j < 2p^*_j$ for all jobs (observation (2)), there exists a schedule of length at most $6C^*_{\max}$ using only the fast machines $M_1, ..., M_{\text{sup}}$ (observation (1)). Therefore, if $d \geq 6C^*_{\max}$, the algorithm will not double d (observation (3)). With a bounded number of inner procedure rounds by Lemma 7, the algorithm will return with the whole schedule finished. □

It immediately follows that, since each inner procedure lasts at most $2d$ time, each call spends $12C^*_{\max}$ time at most. Finally, putting all above lemmas together, we obtain the performance bound for Algorithm 2.

Theorem 10 (Performance bound for Algorithm 2)**.** *Given the predicted job sizes p_j and the prediction error $\eta = \max_{1\leq j\leq n} \max\{\frac{p_j}{p^*_j}, \frac{p^*_j}{p_j}\}$, Algorithm 2 computes a schedule of makespan $C_{\max}$ at most $48 \cdot \min\{[2\log\eta + 2], [\log m + 2]\} \cdot C^*_{\max}$, where $C^*_{\max}$ denotes the optimal makespan.*

Proof. Initially, the algorithm sets optimal makespan estimate $d \leftarrow d_0 := \frac{\max_{1\leq j\leq n} p_j}{\eta \cdot s_1}$ and executes the inner procedure. Clearly $d_0 \leq C^*_{\max}$. The algorithm will undergo x rounds of doubling in makespan with $d = 2d_0, 2^2 d_0, ..., 2^x d_0$. Fixing the makespan d, the number of inner procedure calls is at most $\min\{[2\log\eta + 2], [\log m + 2]\}$ by Lemma 7, with each spending at most $2d$ time. Observe that $2^{x-1}d_0 < 6C^*_{\max}$, otherwise it would have computed the whole schedule after the $(x-1)$-th round of doubling by Lemma 9. Thus, the makespan of the entire schedule is bounded by

$$\begin{aligned} C_{\max} &\leq 2 \cdot (d_0 + ... + 2^x d_0) \cdot \\ &\qquad \min\{[2\log\eta + 2], [\log m + 2]\} \\ &< 2 \cdot (2^{x+1} d_0) \cdot \min\{[2\log\eta + 2], [\log m + 2]\} \\ &< 48 \cdot \min\{[2\log\eta + 2], [\log m + 2]\} \cdot C^*_{\max} \end{aligned}$$

□

Remark 11. *It immediately follows from Theorem 10 that Algorithm 2 has competitive ratio $O(\min\{\log\eta, \log m\})$. Let $C_{\max}$ be the makespan of the computed schedule. With $\eta = \infty$, we have $C_{\max} < 48 \cdot [\log m + 2] \cdot C^*_{\max}$. Thus the algorithm is $48 \cdot [\log m + 2]$-robustness or $O(\log m)$-robustness. With $\eta = 1$, we have $C_{\max} < 96 \cdot C^*_{\max}$. Thus the algorithm is 96-consistent or $O(1)$-consistent.*

Note that, even in the worst case of $\eta = \infty$, our algorithm returns a schedule of makespan at most $48 \cdot [\log m + 2] \cdot C^*_{\max}$. This outperforms the algorithm proposed in (Shmoys, Wein, and Williamson 1995) with $48 \cdot [\log m + 6] \cdot C^*_{\max}$ bound.

Finally, we prove the performance bound of Algorithm 3. With the unknown prediction error, the algorithm estimates η in rounds. The proof builds on the below core observation. Fixed the estimated prediction error η^e, the time spent by an inner procedure is bounded by $O(1)C^*_{\max}$ if $\frac{p_j}{\eta^e} > p^*_j$ for some job J_j, and is bounded by $O(\min\{\log\eta, \log m\})C^*_{\max}$

otherwise. To bound the overall performance, we let $(\eta^e)^2 < m$ and switch to the $O(\log m)$-competitive algorithm if $(\eta^e)^2 \geq m$. With such strategy, the makespan can be bounded by $O(\log m)C^*_{\max}$ if η is arbitrarily large.

Lemma 12 (Bound on the time spent fixing η^e for the case of $\frac{p_j}{\eta^e} > p^*_j$ for some j)**.** *Fix η^e with $\frac{p_j}{\eta^e} > p^*_j$ for some j. The time spent for running Algorithm 2 is at most $24C^*_{\max}$.*

Proof. Let $J_{j'}$ denote the largest overestimated job, i.e., $\frac{p_{j'}}{\eta^e} > p^*_{j'}$ with $\frac{p_{j'}}{\eta^e}$ maximal. We will show that with a fixed η^e and a makespan estimate d, an inner procedure either detects an underestimation of η^e or doubles d until $d \geq 3C^*_{\max}$. Then, since each inner procedure call spends at most $2d$ time, the total time spent will be bounded by $O(1)C^*_{\max}$.

First consider a special case where $\frac{p_{j'}}{\eta^e} = \frac{\max_{1\leq j\leq n} p_j}{\eta^e} \geq s_1 \cdot C^*_{\max}$. In such case, the first makespan estimate $d := d_0 = \frac{p_{j'}}{s_1\eta^e} \geq C^*_{\max}$. By Algorithm 2 machine M_1 will first select an overestimated job to process, and then confirm an underestimation of η^e within at most $\frac{\max_{1\leq j\leq n} p^*_j}{s_1} \leq C^*_{\max}$ time. η^e will double and the time spent is apparently no more than $24C^*_{\max}$. From now on, we will assume $\frac{p_{j'}}{\eta^e} \leq s_1 \cdot C^*_{\max}$ and the first makespan estimate $d := d_0 \leq C^*_{\max}$.

With a fixed η^e, Algorithm 2 will experience several rounds of doubling makespan estimate d. We will show that when $d < 3C^*_{\max}$, an inner procedure either detects an underestimation of η^e or doubles d at the end, so the job size estimates never get updated. We prove this by contradiction. Suppose the algorithm decides to update job size estimates after the inner procedure. This happens only if $J_{j'}$ runs on some machine M_i with $i \leq$ sup and $p^e_{j'} \leq s_i \cdot d$ before time d. But $p^*_{j'} < \frac{p_{j'}}{\eta^e} \leq p^e_{j'} \leq s_i \cdot d$, indicating job $J_{j'}$ should have been completed. An underestimation of η^e is confirmed at the first time this happens, which is a contradiction.

Next we show that when $d \geq 3C^*_{\max}$, Algorithm 2 will detect an underestimation of η^e by showing that at least one overestimated job $J_{k'}$ ($\frac{p_{k'}}{\eta^e} > p^*_{k'}$) will start processing before time $d + C^*_{\max}$ on some machine M_i with $i \leq$ sup and $p^e_{k'} \leq s_i \cdot d$. We prove this by contradiction. Suppose the opposite. Consider the fast machines $M_1, ..., M_{\text{sup}}$. There is a schedule to process all jobs in $3C^*_{\max}$ using only the fast machines by Lemma 8. So all the non-overestimated jobs can be processed using only the fast machines in d time. The procedure ensures no non-overestimated job starts after time d. So the procedure for processing non-overestimated jobs uses at most $d + C^*_{\max}$ time. All machine will become idle before $d + C^*_{\max}$, including the fastest machine M_1. Recall that $p^e_{j'} = \frac{p_{j'}}{\eta^e} \leq s_1 \cdot C^*_{\max} < s_1 \cdot d$ (the first equality is due to that p^e_j is never updated). It follows that either job $J_{j'}$ is processing on another machine M_i with $p^e_{j'} \leq s_i \cdot d$ or it begins processing on M_1 before $d + C^*_{\max}$. This is a contradiction. Then within at most $d + 2C^*_{\max} < 2d$ time, some overestimated job completes processing.

Finally, let Algorithm 2 returns after x rounds of doubling d with $d = 2d_0, 2^2d_0, ..., 2^xd_0$, where $d_0 = \frac{\max_{1\leq j\leq n} p_j}{s_1}$. Fixing a makespan d, the number of inner procedure calls is at most 1. It spends at most $2d$ time at each round. Observe that $2^{x-1}d_0 < 3C^*_{\max}$. The total time spent in Algorithm 2, before confirming an underestimation of η^e, is at most

$$2 \cdot (d_0 + ... + 2^x d_0) < 24C^*_{\max}$$

□

Lemma 13 (Bound on the time spent fixing η^e for the case of $\frac{p_j}{\eta^e} \leq p^*_j$ for all j)**.** *Fix η^e with $\frac{p_j}{\eta^e} \leq p^*_j$ for all j. Algorithm 2 will finish the whole schedule with makespan at most $48 \cdot \min\{[2\log\eta^e + 2], [\log m + 2]\} \cdot C^*_{\max}$.*

Proof. The lemma holds, by the same arguments for Theorem 10. All bounds apply. □

Theorem 14 (Performance bound for Algorithm 3)**.** *Given only the predicted job sizes p_j, Algorithm 3 computes a schedule of makespan $C_{\max}$ at most $\min\{[120\log\eta + 216], [60\log m + 216]\} \cdot C^*_{\max}$, where $C^*_{\max}$ denotes the optimal makespan and η denotes the prediction error and $\eta = \max_{1\leq j\leq n} \max\{\frac{p_j}{p^*_j}, \frac{p^*_j}{p_j}\}$.*

Proof. First assume the algorithm returns the entire schedule while $(\eta^e)^2 < m$. The algorithm will undergo x rounds of doubling η^e with $\eta^e = 1, 2, ..., 2^x$. When $\eta^e = 1, 2, ..., 2^{x-1}$, there must be some job J_j with $\frac{p_j}{\eta^e} > p^*_j$, since otherwise it will have less rounds of doubling η^e. Each round spends at most $24C^*_{\max}$ time by Lemma 12. In total, this sums to $24xC^*_{\max}$. After setting $\eta^e = 2^x$, the algorithm returns within time $48 \cdot \min\{[2\log 2^x + 2], [\log m + 2]\} \cdot C^*_{\max} = 48 \cdot (2x + 2) \cdot C^*_{\max}$ (since $(\eta^e)^2 < m$) by Lemma 13. Observe that $2^{x-1} < \eta$. Therefore, the makespan of the whole schedule, if returned when $(\eta^e)^2 < m$, is at most

$$24xC^*_{\max} + 48(2x + 2)C^*_{\max} \leq [120\log\eta + 216] \cdot C^*_{\max}$$

Next assume the algorithm returns the whole schedule by running Algorithm 2 with $p_j = 0$ and $\eta = \infty$. The algorithm will experience $x - 1$ rounds of doubling η^e with $\eta^e = 1, 2, ..., 2^{x-1}$ before breaking the while loop. Each round spends at most $24C^*_{\max}$ time. This sums to $24xC^*_{\max}$. With $(2^{x-1})^2 < m$, it follows that $24xC^*_{\max} \leq (12\log m + 24)C^*_{\max}$. After breaking the while loop, Algorithm 2 with $p_j = 0$ and $\eta = \infty$ will cost at most $48 \cdot [\log m + 2] \cdot C^*_{\max}$. Therefore, the makespan of the whole schedule, if returned when $(\eta^e)^2 \geq m$, is at most

$$24xC^*_{\max} + 48 \cdot [\log m + 2]C^*_{\max} \leq [60\log m + 120] \cdot C^*_{\max}$$

The entire schedule, therefore, has makespan $C_{\max}$ at most $\min\{[120\log\eta + 216], [60\log m + 216]\} \cdot C^*_{\max}$. □

Remark 15. *It immediately follows from Theorem 14 that Algorithm 3 has competitive ratio $O(\min\{\log\eta, \log m\})$. The algorithm is $[60\log m + 120]$-robustness or $O(\log m)$-robustness, and is 216-consistent or $O(1)$-consistent.*

Finally, we state the time complexity results and brief the correctness for Algorithms 2 and 3. By maintaining jobs in the order of estimated size, an operation in the inner procedure takes $O(\log n)$ time. At the end of an inner procedure, doubling makespan takes $O(1)$ time, and doubling job sizes $O(\text{sup})$ time. This complexity is identical to the existing

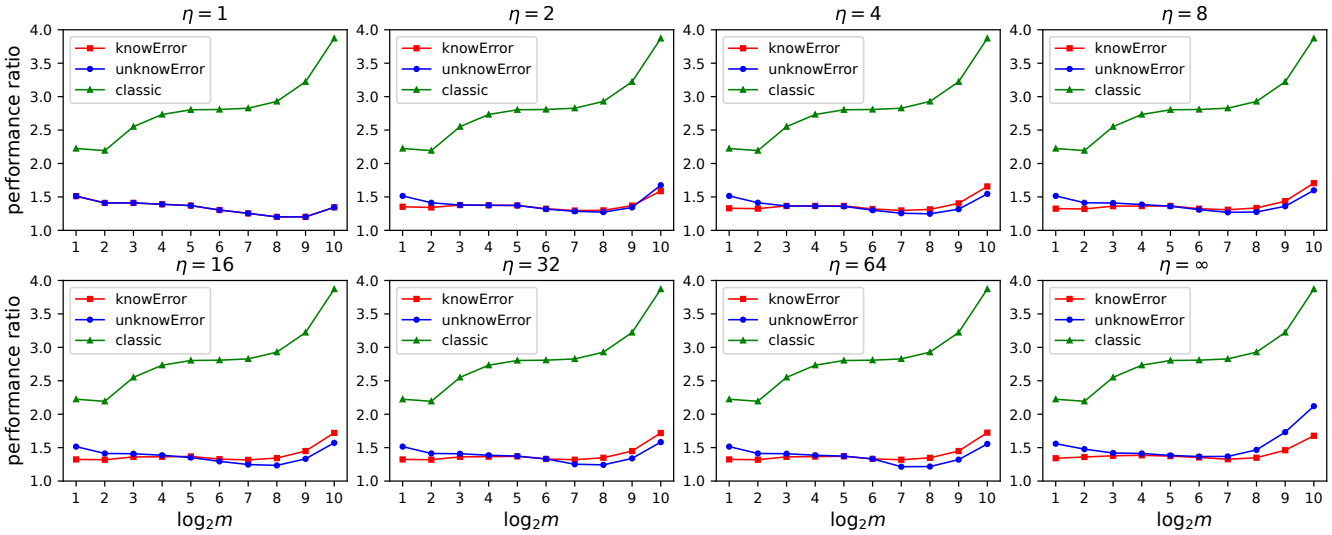

Figure 1: Performance ratios comparison by varying m and η

non-clairvoyant algorithm (Shmoys, Wein, and Williamson 1995). Therefore, using predictions does not introduce overhead in Algorithm 2. Algorithm 3 requires an additional cost when doubling η^e, which takes $O(m)$ time. The correctness implicitly follows the established bounds. The algorithm terminates in a bounded number of iterations. By examining the termination conditions, a valid schedule is constructed at the termination of the algorithm. The detailed arguments are omitted due to the page limit.

Experimental Results

Experimental Setup

This section provides numerical studies to show the competitiveness of our proposed algorithms. We implemented four algorithms: (1) an offline near-optimal algorithm based on Algorithm 1, (2) the $O(\log m)$-competitive algorithm (Shmoys, Wein, and Williamson 1995), (3) Algorithm 2 with a known prediction error, and (4) Algorithm 3 without a known error. We denote these algorithms as A_{opt}, A_{classic}, $A_{\text{knowError}}$, and $A_{\text{unknowError}}$. Finding the optimal solutions is computationally infeasible due to NP-hardness. We instead implemented A_{opt} with a bisection method on Algorithm 1 to find a 2-approximation solution, giving a baseline for other algorithms. Thus, the reported performance ratios are approximations to the analytical ones. The approximations are within twice the analytical competitive ratios.

We generated 2 million independent problem instances to evaluate the performance ratios. The number of jobs ranges from 1000 to 10000, and each job size randomized from $[1, 1000]$. The number of machines is from $2^1, ..., 2^{10}$, with machine speed varies between $[1, 1000]$. The prediction errors were set to $2^0, 2^1, ..., 2^6$, and ∞. With a given prediction error η, we randomly generated a predicted size for every job J_j (with actual size p_j^*) from $[\max\{1, \frac{p_j^*}{\eta}\}, \eta \cdot p_j^*]$. To minimize measurement deviation caused by randomly generated predictions, each problem instance was executed 50 times. All reported data points are an average of 2500 executions.

Results

Figure 1 shows the performance comparisons between A_{classic}, $A_{\text{knowError}}$, and $A_{\text{unknowError}}$ with increasing machines m and prediction error η. As shown, $A_{\text{knowError}}$ and $A_{\text{unknowError}}$ consistently outperform A_{classic}, even with arbitrarily bad predictions. This confirms that our improved 2-relaxed procedure ensures the worst case performance of our proposed algorithms is better than A_{classic}. The performance ratios of $A_{\text{knowError}}$ and $A_{\text{unknowError}}$ increase sublinearly as $\log m$ increases, verifying the theoretical result of $O(\min\{\log \eta, \log m\})$-competitiveness. Our proposed algorithms have most performance ratios around 1.5 and a slight increase for unreasonably large η ($\eta \to \infty$). Thus, our algorithms bound the performance ratios within small constants.

Next, we discuss the performance difference between $A_{\text{knowError}}$ and $A_{\text{unknowError}}$. They have the same performance when $\eta = 1$. When $\eta = \infty$, $A_{\text{knowError}}$ outperforms $A_{\text{unknowError}}$ since $A_{\text{unknowError}}$ spends extra time in estimating η before realizing that it is too large and it is better to switch to the $O(\log m)$-competitive algorithm by setting $\eta^e = \infty$. We also observe that, knowing the error is favourable if η is relatively large compared to m. For relatively large η compared to m, $A_{\text{knowError}}$ implicitly bounds the performance by $\log m$ term. However and interestingly, if η is small, $A_{\text{unknowError}}$ slightly outperforms $A_{\text{knowError}}$. Both algorithms have $\log \eta$ term to bound the performance, but $A_{\text{unknowError}}$ works better. Here, we brief the subtle reason. $A_{\text{unknowError}}$ starts with many iterations with optimistic error estimates $\eta^e = 1, 2$, etc. Though these η^e underestimate η, they make many jobs with accurate predictions to be near-optimally scheduled. In contrast, $A_{\text{knowError}}$ acts conservatively at the beginning to underestimate every job size, which abandons the benefits of those high-quality predictions.

Conclusion

We study online optimization with predictions and show how job size predictions improve the competitive ratio for minimizing makespan in uniformly related machine scheduling. We first design an offline improved 2-relaxed decision procedure approximating the optimal schedule using job sizes. The online algorithms use this procedure as a base. With a simple algorithm-independent prediction error measurement η and the decision procedure, we give an $O(\min\{\log \eta, \log m\})$-competitive algorithm assuming a known prediction error. Built upon this algorithm, we finally propose a robust $O(\min\{\log \eta, \log m\})$-competitive algorithm that does not assume a known error. Both algorithms improve the known $\Omega(\log m)$ lower bound via the predictions. We prove the performance bounds and conduct numerical simulations to verify these results.

Many interesting problems remain open in online optimization with predictions. An immediate direction is to extend our work to other online scheduling or optimization problems. It is worth studying how the proposed error measurement and the algorithm design technique lead to improved competitiveness. Finally, since predictions expose rich information to algorithms, exploring the theoretical performance lower bounds in such settings is valuable. The new lower bounds will give us a better understanding of the potential of predictions in managing uncertainty.

Acknowledgments

Dr. Wei Li acknowledges the support of the Australian Research Council (ARC) through the Discovery Early Career Researcher Award (DE210100263). Professor Zomaya and Dr. Wei Li acknowledge the support of an ARC Discovery Project (DP200103494).

References

Amiri, M.; and Mohammad-Khanli, L. 2017. Survey on Prediction Models of Applications for Resources Provisioning in Cloud. *J. Netw. Comput. Appl.*, 82(C): 93–113.

Anand, K.; Ge, R.; and Panigrahi, D. 2020. Customizing ML Predictions for Online Algorithms. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 303–313. PMLR.

Antoniadis, A.; Coester, C.; Elias, M.; Polak, A.; and Simon, B. 2020. Online metric algorithms with untrusted predictions. In III, H. D.; and Singh, A., eds., *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, 345–355. PMLR.

Awerbuch, B.; Kutten, S.; and Peleg, D. 1992. Competitive Distributed Job Scheduling (Extended Abstract). In *Proceedings of the Twenty-Fourth Annual ACM Symposium on Theory of Computing*, STOC '92, 571–580. New York, NY, USA: Association for Computing Machinery. ISBN 0897915119.

Azar, Y.; Leonardi, S.; and Touitou, N. 2021. Flow Time Scheduling with Uncertain Processing Time. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, 1070–1080. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380539.

Borodin, A.; and El-Yaniv, R. 1998. *Online Computation and Competitive Analysis*. USA: Cambridge University Press. ISBN 0521563925.

Boyar, J.; Favrholdt, L. M.; Kudahl, C.; Larsen, K. S.; and Mikkelsen, J. W. 2016. Online Algorithms with Advice: A Survey. *SIGACT News*, 47(3): 93–129.

Dütting, P.; Lattanzi, S.; Paes Leme, R.; and Vassilvitskii, S. 2021. Secretaries with Advice. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, EC '21, 409–429. New York, NY, USA: Association for Computing Machinery. ISBN 9781450385541.

Frye, M.; Gyulai, D.; Bergmann, J.; and Schmitt, R. H. 2019. Adaptive scheduling through machine learning-based process parameter prediction. *MM Science journal*, 2019(November): HSM2019–023.

Garey, M. R.; and Johnson, D. S. 2009. *Computers and intractability: a guide to the theory of NP-completeness*. W.H. Freeman and Company.

Gollapudi, S.; and Panigrahi, D. 2019. Online Algorithms for Rent-Or-Buy with Expert Advice. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, 2319–2327. PMLR.

Graham, R.; Lawler, E.; Lenstra, J.; and Kan, A. 1979. Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey. In Hammer, P.; Johnson, E.; and Korte, B., eds., *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, 287–326. Elsevier.

Im, S.; Kumar, R.; Montazer Qaem, M.; and Purohit, M. 2021. Non-Clairvoyant Scheduling with Predictions. In *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '21, 285–294. New York, NY, USA: Association for Computing Machinery. ISBN 9781450380706.

Karp, R. M. 1992. On-Line Algorithms Versus Off-Line Algorithms: How Much is It Worth to Know the Future? In *Proceedings of the IFIP 12th World Computer Congress on Algorithms, Software, Architecture - Information Processing '92, Volume 1 - Volume I*, 416–429. NLD: North-Holland Publishing Co. ISBN 044489747X.

Lattanzi, S.; Lavastida, T.; Moseley, B.; and Vassilvitskii, S. 2020. Online Scheduling via Learned Weights. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '20, 1859–1877. USA: Society for Industrial and Applied Mathematics.

Leung, J.; Kelly, L.; and Anderson, J. H. 2004. *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. USA: CRC Press, Inc. ISBN 1584883979.

Mitzenmacher, M. 2020. Scheduling with Predictions and the Price of Misprediction. In *ITCS*.

Peyravi, N.; and Moeini, A. 2020. Estimating runtime of a job in Hadoop MapReduce. *Journal of Big Data*, 7(1): 44.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving Online Algorithms via ML Predictions. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Shmoys, D. B.; Wein, J.; and Williamson, D. P. 1995. Scheduling Parallel Machines On-Line. *SIAM Journal on Computing*, 24(6): 1313–1331.

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