Learning-Augmented Approximation for Unrelated-Machines Makespan Scheduling

Kaito Baba*
The University of Tokyo,
Tokyo, Japan
baba-kaito662@g.ecc.u-tokyo.ac.jp

Evripidis Bampis
Sorbonne Université, CNRS, LIP6,
F-75005 Paris, France
evripidis.bampis@lip6.fr

Giorgos Mitropoulos
Sorbonne Université, CNRS, LIP6,
F-75005 Paris, France
georgios.mitropoulos@lip6.fr

Abstract

Recently, Antoniadis et al. (ICLR 2025) proposed a framework for incorporating predictions to approximate NP-hard selection problems. Despite its simplicity, this approach tightly matches theoretical lower bounds, making its generalization highly compelling. We address an open question raised in the work of Antoniadis et al., concerning the extension of this approach to other important problems outside the class of selection problems, such as scheduling. We develop a learning-augmented algorithm for the makespan minimization problem on unrelated machines, denoted by $R\|C_{\max}$. By using predictions of heavy job assignments, we achieve a polynomial-time $(1+\varepsilon)$-approximation for accurate predictions that smoothly degrades to a worst-case 2-approximation as the error increases. We conclude our work with an empirical analysis of our method.[1]

1 Introduction

Scheduling is one of the central problems in combinatorial optimization (Pinedo, 2016; Brucker, 2007). A particularly fundamental model is scheduling on unrelated parallel machines, denoted by $R\|C_{\max}$ (Graham et al., 1979). Here, we are given a set of n jobs and a set of m machines, and each job j has a machine-dependent processing time p_{ij} on machine i. The goal is to assign each job to one of the machines so as to minimize the maximum load over all machines. This problem models settings in which processing times vary across machines. At the same time, it is a notoriously difficult NP-hard problem: the classical result of Lenstra et al. (1990) gives a polynomial-time 2-approximation, and they showed that improving the approximation factor below $3/2$ is NP-hard.

In recent years, there has been growing interest in algorithms with predictions, or *learning-augmented algorithms* (Purohit et al., 2018; Lykouris & Vassilvitskii, 2018). Motivated by the increasing availability of high-quality machine-learning predictions, this line of work studies how algorithms can exploit such predictions while retaining rigorous guarantees when they are inaccurate. Most early work focused on online algorithms, where the standard objectives are *consistency*, *robustness*, and *smoothness*: the algorithm should perform optimally when predictions are accurate, remain competitive when they are poor, and degrade smoothly as prediction error increases (Mitzenmacher & Vassilvitskii, 2022; Azar et al., 2023).

*This work was conducted during an internship at Sorbonne Université.

[1] Our code is available at https://github.com/kAIto47802/rcmax-with-predictions.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Table 1: Comparison with classical bounds for $R\|C_{\max}$. Here, R^-_{OPT}, $R^-_{\mathrm{OPT},K}$, and R^+_{OPT} are prediction error measures defined in Section 5.1: the first two capture false negatives, while R^+_{OPT} captures false-positive overload. $R^-_{\mathrm{OPT},K}$ monotonically decreases with the search budget K, which is chosen as a constant parameter. $P \coloneqq \max_{i,j} p_{ij}$ denotes the largest processing time of the input.

Algorithm	**Makespan guarantee**	**Running time**	**Assumption / Parameter**
LST (Lenstra et al., 1990)	$2\mathrm{OPT}$	$O(\mathrm{poly}(m,n,\log P))$	–
LST (Lenstra et al., 1990)	$(1+\varepsilon)\mathrm{OPT}$	$O((n+1)^{m/\varepsilon}\mathrm{poly}(m,n,\log P))$	fixed m
Mordechai (2015)	$(1+\varepsilon)T$	$O(2^k\mathrm{poly}(m,n,\log P,1/\varepsilon))$	$k \coloneqq \lvert\{(i,j) \mid p_{ij} > \varepsilon T\}\rvert$, given T
Ours (Theorem 5.2)	$\min\left\{\mathrm{OPT}+\max\{\varepsilon\mathrm{OPT}, R^-_{\mathrm{OPT},K}\}+R^+_{\mathrm{OPT}},\ 2\mathrm{OPT}\right\}$	$O\big((mn)^K\mathrm{poly}(m,n,\log P,1/\varepsilon)\big)$	–
Ours (Corollary 5.3)	$\min\left\{(1+\varepsilon)\mathrm{OPT}+R^+_{\mathrm{OPT}},\ 2\mathrm{OPT}\right\}$	$O\big((mn)^K\mathrm{poly}(m,n,\log P,1/\varepsilon)\big)$	$K \geq \lvert\mathrm{Miss}_{\mathrm{OPT}}(\widehat{\phi},\sigma^\star)\rvert$
Ours (Corollary 5.4)	$\min\left\{\mathrm{OPT}+\max\{\varepsilon\mathrm{OPT}, R^-_{\mathrm{OPT}}\}+R^+_{\mathrm{OPT}},\ 2\mathrm{OPT}\right\}$	$O(\mathrm{poly}(m,n,\log P,1/\varepsilon))$	–

More recently, learning-augmented ideas have also begun to appear in approximation algorithms for NP-hard problems (Cohen-Addad et al., 2024; Bampis et al., 2024; Antoniadis et al., 2025). In this offline setting, predictions may describe part of an optimal solution or some structural object that a classical approximation algorithm would otherwise need to guess or search over. The main goal is to obtain near-optimal performance when the prediction is accurate, together with a guarantee that degrades smoothly with the prediction error. Robustness can often be obtained by running a classical approximation algorithm in parallel, so the central challenge is to design algorithms that yield meaningful consistency and smoothness.

A recent work of Antoniadis et al. (2025) develops such algorithms for a broad class of selection problems with linear objectives, including Vertex Cover and Steiner Tree, and provides approximation guarantees that vary smoothly with false-positive and false-negative prediction errors. However, their framework does not cover problems beyond this linear-objective selection setting, and they explicitly identify constructing such prediction-dependent smooth guarantees for other optimization problems, such as scheduling, as an open direction.

In this work, we develop a learning-augmented approximation algorithm for $R\|C_{\max}$, addressing this open question posed by Antoniadis et al. (2025) in the context of scheduling. While motivated by their open question, our work is neither an instantiation of their black-box framework nor a direct extension of their algorithms; our algorithm, prediction-error measures, and analysis are tailored to the assignment structure and bottleneck objective of $R\|C_{\max}$. Our starting point is the classical heavy/short decomposition for unrelated-machine scheduling (Lenstra et al., 1990; Mordechai, 2015). For a target makespan T, the combinatorial difficulty lies in assignments with processing time greater than εT; once these assignments are fixed, the remaining short jobs can be handled by a linear program (LP) and an appropriate rounding procedure, such as LST rounding (Lenstra et al., 1990) and Shmoys–Tardos rounding (Shmoys & Tardos, 1993). Classical schemes search over the heavy assignments to obtain a $(1+\varepsilon)\mathrm{OPT}$ guarantee, but this requires non-polynomial running time unless m is fixed to be small. We use the prediction to localize this search.

Our learning-augmented algorithm achieves a $(1+\varepsilon)\mathrm{OPT}$ guarantee with accurate predictions while running in time comparable to the classical polynomial-time 2-approximation algorithm, thereby improving the makespan guarantee when m is large. The guarantee degrades smoothly with the prediction error and, even with poor predictions, never exceeds the classical $2\mathrm{OPT}$ baseline. At a high level, false-positive predictions contribute to the makespan bound, while false-negative predictions can either be recovered by local search or left in the bound, with the constant search budget K smoothly controlling this tradeoff. Table 1 summarizes the comparison with the classical guarantees. Experiments show that, consistently with our theory, accurate predictions allow our algorithm to outperform the classical 2-approximation baseline in makespan, while running significantly faster than the classical $(1+\varepsilon)$-approximation based on heavy-assignment search.

Our contributions can be summarized as follows:

- We introduce a learning-augmented framework for $R\|C_{\max}$, addressing the open question of Antoniadis et al. (2025) in the context of scheduling. Rather than searching globally over heavy assignments, our approach localizes the search around the prediction.
- Our guarantee exhibits the desired learning-augmented behavior. Accurate predictions yield a $(1+\varepsilon)\mathrm{OPT}$ makespan bound in polynomial time, and it degrades smoothly with the prediction error; even with poor predictions, it never exceeds the classical $2\mathrm{OPT}$ baseline.

- We also provide the smooth tradeoff between running time and makespan quality. Larger search budgets K allow the algorithm to repair more missed heavy assignments, while $K = 0$ yields a fast, purely prediction-based variant without local search.
- Numerical experiments show that, consistent with our theory, accurate predictions improve over the classical 2-approximation baseline, while our algorithm runs significantly faster than classical $(1 + \varepsilon)$-type heavy-assignment search.

We also extend our prediction-localized search approach to total weighted completion time on unrelated machines. This extension yields a guarantee that degrades smoothly with prediction error while preserving the approximation ratio of a classical baseline; see Appendix F.

2 Related work

Here we review the prior work most relevant to ours; further discussion is provided in Appendix A.

Learning-augmented online algorithms. Learning-augmented algorithms were first studied extensively in the online setting, where predictions are used to improve performance beyond worst-case guarantees while retaining robustness when the predictions are inaccurate (Mitzenmacher & Vassilvitskii, 2022). Starting with the works of Lykouris & Vassilvitskii (2018); Purohit et al. (2018) on caching, ski rental, and online scheduling, subsequent work developed sharper guarantees for caching (Rohatgi; Wei, 2020), primal-dual frameworks for online covering problems (Bamas et al., 2020), and more general online covering linear and semidefinite programs (Grigorescu et al., 2022).

Learning-augmented approximation algorithms. More recently, learning-augmented ideas have been applied to offline approximation algorithms for NP-hard problems. Recent results include learning-augmented approximation algorithms for Max-Cut and related CSPs (Cohen-Addad et al., 2024), dense NP-hard problems (Bampis et al., 2024), Maximum Independent Set (Braverman et al., 2024), and permutation problems (Bampis et al., 2025). Recently, Antoniadis et al. (2025) gave black-box algorithms for selection problems with linear objectives, including Vertex Cover, Steiner Tree, Min-Weight Perfect Matching, Knapsack, and Clique, with approximation guarantees that degrade smoothly with false-negative and false-positive prediction errors. Their framework, however, does not cover scheduling problems such as $R\|C_{\max}$, which are not selection problems and whose objective is makespan minimization rather than the total weight of a selected set. They leave scheduling as an open question for such smooth guarantees, which we address for $R\|C_{\max}$.

Scheduling and heavy-assignment search. In the context of classical approximation algorithms without prediction, Lenstra et al. (1990) gave a polynomial-time 2-approximation for $R\|C_{\max}$, as well as a $(1 + \varepsilon)\mathrm{OPT}$ approximation scheme when the number of machines m is fixed. The latter relies on exhaustive search over heavy assignments and LP rounding for the remaining short jobs. For a given threshold T, Mordechai (2015) later gave a parameterized approximation scheme whose parameter is $k \coloneqq \left|\{(i, j) \mid p_{ij} > \varepsilon T\}\right|$. However, these $(1 + \varepsilon)$-type guarantees are not polynomial-time unless m is fixed to be small. Our work uses predictions to go beyond these fixed-m regimes: when the predictions are accurate, we obtain a $(1 + \varepsilon)\mathrm{OPT}$ guarantee with running time comparable to that of the classical polynomial-time 2-approximation algorithm.

3 Problem setup

Problem definition. We consider the *makespan minimization* problem on *unrelated-machines*. There are m machines and n jobs. For every machine $i \in [m]$ and job $j \in [n]$[2], let $p_{ij} \in \mathbb{R}_{>0}$ denote the processing time of job j on machine i. A schedule is a function $\sigma\colon [n] \to [m]$, and its makespan is $C_{\max}(\sigma) \coloneqq \max_{i \in [m]} \sum_{j:\sigma(j)=i} p_{ij}$. The goal is to find a schedule σ minimizing $C_{\max}(\sigma)$.

Prediction model. The algorithm is given as input a predicted partial assignment $\widehat{\phi}\colon [n] \to [m] \cup \{\perp\}$, where $\widehat{\phi}(j) = i$ means that the prediction suggests assigning job j to machine i, and $\widehat{\phi}(j) = \perp$ means that no assignment is predicted for job j. Intuitively, $\widehat{\phi}$ is intended to capture part of the heavy-assignment structure of a good schedule. We let $\mathrm{dom}(\widehat{\phi}) \coloneqq \{j \in [n] \mid \widehat{\phi}(j) \neq \perp\}$ denote the set of jobs for which the prediction specifies a machine. See Remark 4.1 for further discussion. We discuss the motivation for this prediction model and compare it with alternatives in Appendix B.

[2]Here and throughout, for a positive integer r, we write $[r] \coloneqq \{1, 2, \ldots, r\}$.

Algorithm 1: Inner layer: PREDICTION-GUIDED-COMPLETION$(T, \varepsilon, K, \widehat{\phi})$

Input: target makespan T, parameter $\varepsilon \in (0, 1)$, search budget K, and predicted assignment $\widehat{\phi}$

// Step 1: Keep only the predicted assignments that are T-heavy.

Compute $\widehat{\phi}_T$ according to Eq. (1)

Initialize the candidate set $\mathcal{C} \leftarrow \emptyset$

// Step 2: Search only for missing heavy assignments.

foreach *augmentation A of $\widehat{\phi}_T$ with budget K satisfying Eq.* (2) **do**

  $\psi_A \leftarrow \widehat{\phi}_T \cup A$

  // Step 3: Fix the heavy assignments and solve the residual LP over relevant thresholds.

  foreach *threshold $\tau \in \Lambda(T)$ defined in Eq.* (3) **do**

    Solve the residual LP Eq. (4) with threshold τ for ψ_A

    if *the LP is feasible with optimum value L_{ψ_A}* **then**

      // Step 4: Round the solution of the residual LP.

      $S_A \leftarrow$ the schedule obtained by Shmoys–Tardos rounding (Shmoys & Tardos, 1993)

      $\mathcal{C} \leftarrow \mathcal{C} \cup \{S_A\}$

// Step 5: Return the best schedule among all candidates.

return *the schedule in $\mathcal{C}$ that minimizes makespan*

4 Algorithm

Our algorithm has the classical two-layer structure used in approximation schemes for $R\|C_{\max}$ (Lenstra et al., 1990; Mordechai, 2015): an *inner* layer that solves a completion problem for a fixed threshold makespan T, and an *outer* layer that ranges over the relevant threshold values. However, in our algorithm, both layers are prediction-aware, which we explain in the following sections.

4.1 Inner layer: prediction-localized heavy-assignment completion

We now describe the inner layer of our algorithm, which is the main algorithmic component where prediction enters the search procedure. The pseudocode is given in Algorithm 1.

Step 1: Keep only the predicted assignments that are T-heavy. The prediction $\widehat{\phi}$ may contain assignments that are irrelevant at scale T. Accordingly, the algorithm forms the T-heavy projection:

$$\widehat{\phi}_T(j) = \begin{cases} \widehat{\phi}(j), & \text{if } \widehat{\phi}(j) \neq \perp \text{ and } p_{\widehat{\phi}(j),j} > \varepsilon T, \\ \perp, & \text{otherwise.} \end{cases} \tag{1}$$

The intuition is that only jobs with processing time exceeding εT need special treatment. Assignments with processing time at most εT are left to the LP, since the rounding incurs only an additive εT loss.

Remark 4.1. A full-assignment prediction $\widehat{\phi}\colon [n] \to [m]$ is a special case of our framework: once T is fixed, one simply keeps the predicted assignments with $p_{\widehat{\phi}(j),j} > \varepsilon T$. Allowing abstention is more flexible, since uncertain heavy jobs can be left to the local search rather than fixed incorrectly.

Remark 4.2. In the present formulation, every predicted assignment with processing time larger than εT is trusted and fixed. This makes the distinction between false-positive and false-negative errors especially transparent. A variant that allows editing or deleting some predictions will be considered in Appendix E. Conceptually, such edits simply trade one type of prediction error for the other.

Step 2: Search only for missing heavy assignments. The projected prediction $\widehat{\phi}_T$ may miss some heavy assignments in a good schedule. The algorithm therefore enumerates augmentations $A\colon [n] \to [m] \cup \{\perp\}$ that add at most K extra heavy assignments, where

$$\operatorname{dom}(A) \cap \operatorname{dom}(\widehat{\phi}_T) = \emptyset, \text{ and } |\operatorname{dom}(A)| \leq K, \text{ and } p_{A(j),j} > \varepsilon T \text{ for every } j \in \operatorname{dom}(A). \tag{2}$$

Thus, the algorithm replaces the classical global heavy-assignment search (Lenstra et al., 1990; Mordechai, 2015) by a search for at most K missing heavy assignments. Even in the worst case,

Algorithm 2: Outer layer: PREDICTION-LOCALIZED-SEARCH$(\varepsilon, K, \widehat{\phi})$

Input: parameter $\varepsilon \in (0,1)$, search budget K, and predicted assignment $\widehat{\phi}$

// Step 1: Compute the baseline upper bound.

Compute a baseline schedule S_0 using the classical 2-approximation (Lenstra et al., 1990)

Set $B \leftarrow C_{\max}(S_0)$, and initialize the candidate set $\mathcal{C} \leftarrow \{S_0\}$

// Step 2: Run the inner procedure on all critical thresholds.

foreach $T \in \mathcal{T}$, *where* $\mathcal{T}$ *is defined in Eq.* (5) **do**

  $S_T \leftarrow$ PREDICTION-GUIDED-COMPLETION$(T, \varepsilon, K, \widehat{\phi})$

  $\mathcal{C} \leftarrow \mathcal{C} \cup \{S_T\}$

// Step 3: Return the best schedule among all candidates.

return *the schedule in* $\mathcal{C}$ *that minimizes makespan*

the necessary budget K never exceeds that of the classical heavy-assignment search. We also note that our algorithm still works if K is insufficient; the remaining missed heavy assignments are smoothly reflected in the makespan guarantees in Theorems 5.1 and 5.2. Thus, K is a freely chosen constant parameter, not a problem- or prediction-dependent quantity. This role is analogous to the trust parameter λ used in prior learning-augmented algorithms (Purohit et al., 2018; Bamas et al., 2020): it controls how much the algorithm corrects or trusts the prediction.

Step 3: Fix the heavy assignments and solve the residual LP over relevant thresholds. Once $\psi_A \coloneqq \widehat{\phi}_T \cup A$ is fixed, the algorithm computes the fixed machine loads from $\mathrm{dom}(\psi_A)$ and solves an LP on the remaining jobs. However, if the prediction has errors and K does not recover all missed heavy assignments, some residual jobs may still have processing time exceeding εT. Hence, unlike in the classical setting, the residual LP cannot be restricted to edges (i,j) with $p_{ij} \le \varepsilon T$ alone.

For this reason, the algorithm considers a threshold parameter $\tau \ge \varepsilon T$ and lets the residual LP use all edges (i,j) with $p_{ij} \le \tau$. Since τ affects the LP only through the admissible edge set, it changes only when τ crosses some processing time p_{ij}. Thus, it suffices to consider the discrete candidate set

$$\Lambda(T) \coloneqq \{\varepsilon T\} \cup \{p_{ij} \mid p_{ij} \ge \varepsilon T\}. \tag{3}$$

For each $\tau \in \Lambda(T)$, the algorithm solves an LP over edges with $p_{ij} \le \tau$ to minimize the load bound L:

$$\begin{aligned}
\text{minimize} \quad & L \\
\text{subject to} \quad & \sum_{i=1}^{m} x_{ij} = 1 && \forall j \notin \mathrm{dom}(\psi_A), \\
& \sum_{j \notin \mathrm{dom}(\psi_A)} p_{ij} x_{ij} \le L - \ell_i^{\mathrm{fix}}(\psi_A) && \forall i \in [m], \\
& x_{ij} = 0 && \forall i,j \text{ with } j \notin \mathrm{dom}(\psi_A) \text{ and } p_{ij} > \tau, \\
& x_{ij} \ge 0 && \forall i,j \text{ with } j \notin \mathrm{dom}(\psi_A),
\end{aligned} \tag{4}$$

where $\ell_i^{\mathrm{fix}}(\psi_A) \coloneqq \sum_{j:\psi_A(j)=i} p_{ij}$ is the fixed load on machine i induced by ψ_A. Note that the threshold τ only affects the residual LP; heavy assignments $\widehat{\phi}_T(j)$ are still defined by εT.

Step 4: Round the solution of the residual LP. If the residual LP is feasible, the algorithm applies the Shmoys–Tardos rounding (Shmoys & Tardos, 1993) to obtain an integral schedule.[3] Because every residual edge has processing time at most τ, the rounding increases the load of each machine by at most τ. Therefore, the final makespan of the schedule extending ψ_A is at most $L_{\psi_A} + \tau$, where L_{ψ_A} is the optimum value of the residual LP, if the LP is feasible. This is exactly why the heavy/short decomposition is useful and why the algorithm does not need predictions for short assignments.

Step 5: Return the best schedule among all candidates. The algorithm compares all schedules produced by feasible augmentations, and returns the one with minimum makespan.

[3]This rounding generalizes the LST rounding (Lenstra et al., 1990). In our setting, both yield the same guarantee.

4.2 The outer layer: synchronizing the threshold search with the prediction

The inner layer assumes that the target makespan T is fixed. The outer layer removes this assumption. Unlike the classical inner routine, our prediction-guided inner layer does not certify feasibility at a fixed threshold T, since failure may reflect an inaccurate prediction rather than true infeasibility; consequently, the classical binary search over T is no longer applicable. Nevertheless, we can still design the outer layer without increasing the overall running-time order, by enumerating the polynomially many critical threshold values at which the prediction-induced heavy-assignment structure changes. The pseudocode is given in Algorithm 2.

Step 1: Compute the baseline upper bound. The algorithm first computes the classical 2-approximation schedule S_0 (Lenstra et al., 1990) and sets $B \coloneqq C_{\max}(S_0)$. Since $B \le 2\mathrm{OPT}$, the unknown optimum lies in the interval $[B/2, B]$.

Step 2: Run the inner procedure on all critical thresholds. The heavy/short classification changes only when εT crosses some processing time p_{ij}, i.e., when T crosses some critical threshold $T = p_{ij}/\varepsilon$. Therefore, the outer algorithm constructs the critical threshold set

$$\mathcal{T} = \left\{ \frac{B}{2} \right\} \cup \left\{ \frac{p_{ij}}{\varepsilon} \;\middle|\; \frac{B}{2} \le \frac{p_{ij}}{\varepsilon} \le B \right\} \cup \{B\}, \tag{5}$$

and runs the inner procedure for every $T \in \mathcal{T}$: PREDICTION-GUIDED-COMPLETION$(T, \varepsilon, K, \widehat{\phi})$.

Step 3: Return the best schedule among all candidates. Finally, the algorithm returns the best schedule among: (i) the baseline schedule S_0, and (ii) all schedules produced at all critical thresholds.

5 Analysis

5.1 Prediction error measures

Following Antoniadis et al. (2025), we decompose prediction error into false-negative and false-positive components. Because their framework is designed for selection problems, we define the corresponding notions for $R\|C_{\max}$. For any schedule $\sigma\colon [n] \to [m]$, define the *false-negative threshold* and *false-positive overload*:

$$R_T^-(\widehat{\phi}, \sigma) \coloneqq \max_{j \in \mathrm{Miss}_T(\widehat{\phi}, \sigma)} p_{\sigma(j),j}, \qquad R_T^+(\widehat{\phi}, \sigma) \coloneqq \max_{i \in [m]} \sum_{\substack{j\colon \widehat{\phi}_T(j) = i \\ \widehat{\phi}_T(j) \ne \sigma(j)}} p_{ij}, \tag{6}$$

where $\mathrm{Miss}_T(\widehat{\phi}, \sigma)$ is the *missed heavy jobs* defined as $\mathrm{Miss}_T(\widehat{\phi}, \sigma) \coloneqq \{j \in [n] \mid p_{\sigma(j),j} > \varepsilon T \text{ and } \widehat{\phi}_T(j) = \perp\}$. Here, we treat $R_T^-(\widehat{\phi}, \sigma) = 0$ if $\mathrm{Miss}_T(\widehat{\phi}, \sigma) = \emptyset$. Intuitively, $R_T^-(\widehat{\phi}, \sigma)$ measures the largest heavy assignment missed by the prediction, while $R_T^+(\widehat{\phi}, \sigma)$ measures the maximum overload caused by predicted heavy assignments that disagree with σ. We further define the *K-residual false-negative threshold*:

$$R_{T,K}^-(\widehat{\phi}, \sigma) \coloneqq \min_{\substack{U \subseteq \mathrm{Miss}_T(\widehat{\phi}, \sigma) \\ |U| \le K}} \; \max_{j \in \mathrm{Miss}_T(\widehat{\phi}, \sigma) \setminus U} p_{\sigma(j),j}, \tag{7}$$

with the convention that the maximum over the empty set is 0. Equivalently, $R_{T,K}^-(\widehat{\phi}, \sigma)$ is the $(K+1)$-st largest processing time among the missed heavy jobs. This represents the largest processing time among the missed heavy jobs that remain after optimally recovering up to K of them. In particular, when $K = 0$, it is the same as the original false-negative threshold: $R_{T,0}^-(\widehat{\phi}, \sigma) = R_T^-(\widehat{\phi}, \sigma)$.

5.2 Guarantees for the inner layer

Theorem 5.1 (Inner-layer guarantee)**.** *Fix $\varepsilon \in (0, 1)$, a target value $T > 0$, a search budget $K \ge 0$, and a prediction $\widehat{\phi}$. Then for any schedule σ, Algorithm 1 returns a feasible schedule S satisfying*

$$C_{\max}(S) \le C_{\max}(\sigma) + \max\{\varepsilon T, R_{T,K}^-(\widehat{\phi}, \sigma)\} + R_T^+(\widehat{\phi}, \sigma). \tag{8}$$

The running time of the algorithm is $O((mn)^K \mathrm{poly}(m, n, \log P, 1/\varepsilon))$, where $P \coloneqq \max_{i,j} p_{ij}$ is the largest processing time.

Here we present a proof sketch of Theorem 5.1. The full proof is deferred to Appendix C.1.

Proof sketch. By the definition of $R^-_{T,K}(\widehat{\phi},\sigma)$, there exists a subset $U^\star \subseteq \mathrm{Miss}_T(\widehat{\phi},\sigma)$ with $|U^\star| \le K$ such that $\max_{j\in \mathrm{Miss}_T(\widehat{\phi},\sigma)\setminus U^\star} p_{\sigma(j),j} = R^-_{T,K}(\widehat{\phi},\sigma)$. Define an augmentation $A^\star(j)=\sigma(j)$ for $j\in U^\star$ and $A^\star(j)=\perp$ otherwise. Since $U^\star$ consists only of missed heavy assignments, we can confirm that $A^\star$ satisfies Eq. (2) and is one of the augmentations considered by the algorithm.

Let $\psi^\star \coloneqq \widehat{\phi}_T \cup A^\star$ and $\tau^\star \coloneqq \max\{\varepsilon T, R^-_{T,K}(\widehat{\phi},\sigma)\}$. By construction, we can confirm $\tau^\star \in \Lambda(T)$. Consider the residual LP for $(\psi^\star,\tau^\star)$, and assign every residual job j to $\sigma(j)$. Then we can confirm that every edge $(\sigma(j),j)$ is allowed in the residual LP with $\tau^\star$: if $p_{\sigma(j),j}\le \varepsilon T$ this is immediate, while otherwise j is a missed heavy job not recovered by $U^\star$, and hence $p_{\sigma(j),j}\le R^-_{T,K}(\widehat{\phi},\sigma)\le \tau^\star$.

Consider the assignments that follows $\psi^\star$ on fixed jobs and σ on residual jobs. We can confirm that this assignment is bounded by $C_{\max}(\sigma)+R^+_T(\widehat{\phi},\sigma)$. Thus, the residual LP for $(\psi^\star,\tau^\star)$ is feasible with load bound at most $C_{\max}(\sigma)+R^+_T(\widehat{\phi},\sigma)$.

Let $L^{\tau^\star}_{\psi^\star}$ be the returned optimum of the residual LP for $(\psi^\star,\tau^\star)$. By the classical rounding argument (Theorem 2.1 in Shmoys & Tardos (1993)), this yields an integral schedule of makespan at most $L^{\tau^\star}_{\psi^\star}+\tau^\star$. Hence, for $(A^\star,\tau^\star)$, the algorithm obtains a schedule of makespan at most

$$C_{\max}(\sigma)+R^+_T(\widehat{\phi},\sigma)+\max\{\varepsilon T, R^-_{T,K}(\widehat{\phi},\sigma)\}. \tag{9}$$

Since the algorithm returns the best candidate, the same bound holds for its output.

The running time follows from the enumeration bounds. The number of augmentations is at most $\sum_{s=0}^{K}\binom{n}{s}m^s \le (K+1)(mn)^K$, and $|\Lambda(T)|\le mn+1$. For each pair (A,τ), the LP and rounding take polynomial time: $\mathrm{poly}(m,n,\log P,1/\varepsilon)$. Hence, we obtain the claimed total running time. □

Interpretation. Theorem 5.1 exhibits a clean tradeoff, controlled by the budget K, between paying for false-negative heavy assignments in running time and paying for them in the makespan bound. The local search step (Step 2) runs in $(mn)^{O(K)}$ time, which increases monotonically with K. In contrast, the residual false-negative term $R^-_{T,K}(\widehat{\phi},\sigma)$ decreases monotonically with K. In particular, when $K=0$, the largest missed heavy job appears directly in the bound; when $K=1$, the algorithm can recover the largest missed heavy job by search, so the bound is determined by the second-largest missed heavy job; and so on. Hence increasing K makes the algorithm slower but improves the makespan guarantee, whereas decreasing K speeds up the algorithm at the cost of a weaker guarantee. In this sense, the parameter K provides a smooth interpolation between search effort and schedule quality. For further intuition, Corollaries D.1 and D.2 in Appendix D.1 discuss two special cases of Theorem 5.1: one where K is large enough to recover all missed heavy assignments, and one where $K=0$, so no local search is performed. The smoothness of the final makespan guarantee is discussed later in Section 5.3, where we analyze the outer-layer theorem.

5.3 Guarantees for the outer layer

Theorem 5.2 (Outer-layer guarantee)**.** *Fix $\varepsilon\in(0,1)$, a search budget $K\ge 0$, and a prediction $\widehat{\phi}$. Let $\sigma^\star$ be an optimal schedule, and write* $\mathrm{OPT}\coloneqq C_{\max}(\sigma^\star)$. *Then Algorithm 2 returns a feasible schedule S satisfying*

$$\begin{aligned} C_{\max}(S) &\le \min\left\{\mathrm{OPT}+\max\{\varepsilon\mathrm{OPT}, R^-_{\mathrm{OPT},K}(\widehat{\phi},\sigma^\star)\}+R^+_{\mathrm{OPT}}(\widehat{\phi},\sigma^\star),\ 2\mathrm{OPT}\right\} \\ &\le \min\left\{(1+\varepsilon)\mathrm{OPT}+R^-_{\mathrm{OPT},K}(\widehat{\phi},\sigma^\star)+R^+_{\mathrm{OPT}}(\widehat{\phi},\sigma^\star),\ 2\mathrm{OPT}\right\}. \end{aligned} \tag{10}$$

The running time of the algorithm is $O((mn)^K\mathrm{poly}(m,n,\log P,1/\varepsilon))$.

Proof. Fix an optimal schedule $\sigma^\star$ of makespan OPT. Recall that B is the baseline 2-approximation makespan. Since $\mathrm{OPT}\in[B/2,B]$, there exists a largest critical value $T^\star\in\mathcal{T}$ such that $T^\star\le\mathrm{OPT}$. By construction of $\mathcal{T}$, there is no value of the form p_{ij}/ε in the open interval $(T^\star,\mathrm{OPT}]$. Hence, for every machine–job pair (i,j), we have: $p_{ij}>\varepsilon T^\star \Leftrightarrow p_{ij}>\varepsilon\mathrm{OPT}$. Thus, the heavy/short decomposition is identical at thresholds $T^\star$ and OPT, and in particular we have $\widehat{\phi}_{T^\star}=\widehat{\phi}_{\mathrm{OPT}}$, $\mathrm{Miss}_{T^\star}(\widehat{\phi},\sigma^\star)=\mathrm{Miss}_{\mathrm{OPT}}(\widehat{\phi},\sigma^\star)$, $R^-_{T^\star,K}(\widehat{\phi},\sigma^\star)=R^-_{\mathrm{OPT},K}(\widehat{\phi},\sigma^\star)$ and $R^+_{T^\star}(\widehat{\phi},\sigma^\star)=R^+_{\mathrm{OPT}}(\widehat{\phi},\sigma^\star)$.

Hence we may apply Theorem 5.1 to the call PREDICTION-GUIDED-COMPLETION$(T^\star, \varepsilon, K, \widehat{\phi})$, taking $\sigma^\star$ as the comparison schedule. It follows that the returned schedule $S_{T^\star}$ satisfies

$$\begin{aligned} C_{\max}(S_{T^\star}) &\le C_{\max}(\sigma^\star) + \max\{\varepsilon T^\star, R^-_{T^\star,K}(\widehat{\phi}, \sigma^\star)\} + R^+_{T^\star}(\widehat{\phi}, \sigma^\star) \\ &\le \mathrm{OPT} + \max\{\varepsilon \mathrm{OPT}, R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)\} + R^+_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star). \end{aligned} \tag{11}$$

The outer algorithm returns the best schedule among the inner-layer candidates and the baseline schedule S_0. Since $C_{\max}(S_0) \le 2\mathrm{OPT}$, we obtain the claimed minimum bound.

It remains to bound the running time. The set $\mathcal{T}$ contains at most $mn + 2$ critical values. For each $T \in \mathcal{T}$, Algorithm 1 runs in time $O((mn)^K \cdot \mathrm{poly}(m, n, \log P, 1/\varepsilon))$. Multiplying by the polynomial number of outer iterations preserves the same asymptotic form. This completes the proof. □

Interpretation. As in Theorem 5.1, Theorem 5.2 provides a smooth tradeoff between running time and makespan quality. The running time grows monotonically with the budget K, while the makespan false-negative term $R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ decreases monotonically with K. Thus, larger budgets K let the algorithm spend more time correcting predictions and obtain stronger guarantees, whereas smaller budgets run faster at the cost of a weaker bound.

If the prediction is accurate enough that $R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star) = 0$ and $R^+_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star) = 0$, the theorem gives a $(1+\varepsilon)\mathrm{OPT}$ schedule, matching the classical heavy-assignment search guarantee (Lenstra et al., 1990; Mordechai, 2015). Unlike classical schemes, which are polynomial-time only when the number of machines m is fixed to be small, the exponent in our running time is governed by the search budget K. This budget is the user-chosen constant parameter that controls how much the algorithm corrects the prediction, analogous to trust parameters in prior learning-augmented algorithms (Purohit et al., 2018; Bamas et al., 2020). Thus, the running time remains polynomial independently of whether m or other problem-dependent quantities are fixed, and accurate predictions allow the algorithm to improve over the classical polynomial-time 2-approximation baseline.

As the prediction error increases, the error terms $R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ and $R^+_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$ increase, and the makespan bound degrades smoothly with them. Even in the worst case, the bound never exceeds the classical 2-approximation baseline. Thus, Theorem 5.2 exhibits the desired learning-augmented behavior: accurate predictions yield improved guarantees, while worse predictions lead to smooth degradation without exceeding the classical worst-case guarantee.

For further intuition, we present the following corollary for the sufficiently accurate prediction.

Corollary 5.3 (False-negative recovery)**.** *If $K \ge |\mathrm{Miss}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)|$, Algorithm 2 returns a feasible schedule S such that*

$$C_{\max}(S) \le \min\left\{(1+\varepsilon)\mathrm{OPT} + R^+_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star),\ 2\mathrm{OPT}\right\}. \tag{12}$$

The running time of the algorithm is $O((mn)^K \mathrm{poly}(m, n, \log P, 1/\varepsilon))$.

The proof follows immediately from Theorem 5.2; we provide the details in Appendix C.2.

Interpretation. When the budget K is sufficiently large, the false-negative term $R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ disappears entirely from the makespan bound. Thus, false negatives are paid for purely in running time, while false positives are paid for in the approximation bound through $R^+_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$, revealing a clear asymmetry between the two types of prediction error. The corresponding smoothness and robustness properties are exactly those discussed for Theorem 5.2.

Moreover, the budget needed for this condition has a prediction-sensitive behavior reminiscent of learning-augmented guarantees. If the prediction has no false negatives, then $K = 0$ already suffices to remove the false-negative term from the makespan bound. The budget needed to remove the false-negative term increases gradually with the prediction error and, even in the worst case, does not exceed the scale of the classical heavy-assignment search.

Note that this corollary is intended to isolate the intuition for the large-K case and to highlight the asymmetry between false negatives and false positives; it does not mean that our algorithm requires $K \ge |\mathrm{Miss}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)|$. As shown in Theorem 5.2, the algorithm also works for insufficient K, with the unrecovered false-negative term $R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ appearing smoothly in the makespan bound.

As the opposite extreme, we next state the corollary for the case with no local search, i.e., $K = 0$.

Corollary 5.4 (No-search bound)**.** *When $K = 0$, Algorithm 2 returns a feasible schedule S such that*

$$\begin{aligned} C_{\max}(S) &\le \min\left\{\mathrm{OPT} + \max\{\varepsilon\mathrm{OPT},\, R^{-}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})\} + R^{+}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star}),\, 2\mathrm{OPT}\right\} \\ &\le \min\left\{(1+\varepsilon)\mathrm{OPT} + R^{-}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star}) + R^{+}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star}),\, 2\mathrm{OPT}\right\}. \end{aligned} \tag{13}$$

The running time of the algorithm is $O(\mathrm{poly}(m, n, \log P, 1/\varepsilon))$.

The proof follows immediately from Theorem 5.2; we provide the details in Appendix C.2.

Interpretation. When $K = 0$, no local search is performed, so the K-dependent factor in the running time disappears. Even in this case, the prediction is still used effectively, and accurate predictions can lead to a significantly improved makespan bound. The corresponding smoothness and robustness properties are the same as those discussed for Theorem 5.2.

6 Numerical experiments

In this section, we empirically investigate how the proposed algorithm actually behaves as the prediction quality and the search budget K vary.

6.1 Experimental setup

Following prior work on $R\|C_{\max}$ (Fanjul-Peyro & Ruiz, 2011, 2010; Åblad et al., 2021), we use synthetic instances whose processing times are sampled independently and uniformly from the interval $[1, 100]$. Following prior work on learning-augmented algorithms (Purohit et al., 2018; Antoniadis et al., 2025), we generate synthetic predictions with controlled error rates to evaluate how the algorithm behaves as the prediction error varies. Specifically, we obtain predictions by randomly corrupting a prescribed fraction of the heavy assignments in an optimal schedule into false negatives or false positives. Throughout the experiments, we set $\varepsilon = 0.01$ and vary the search budget K from 0 to 2. Since Mordechai (2015) only gives a fixed-T routine, we combine it with the LST outer search as a baseline, which yields a running time of $O(2^{mn}\mathrm{poly}(m, n, \log P, 1/\varepsilon))$. All experiments are run with 30 different random seeds, and we report the mean and standard error. To ensure reproducibility, the implementation of our algorithm and all experimental code are publicly available.[4] Further details on the experimental setup and implementation are given in Appendix G.

6.2 Effect of false negatives and false positives on makespan

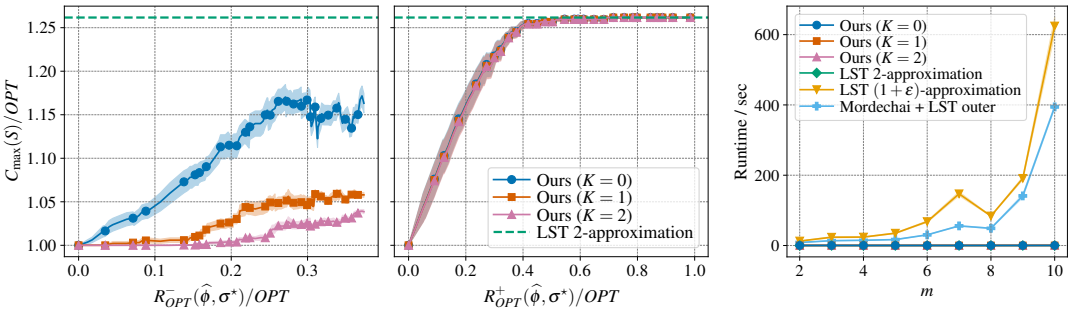
We use instances with $m = 10$ and $n = 100$, and vary either the false-negative rate or the false-positive rate while fixing the other to 0. The results are shown in Figs. 1a and 1b. Note that both classical $(1+\varepsilon)$-type baselines are not available in these experiments due to their excessive running time.

We first discuss the false-negative experiment in Fig. 1a. Across all choices of K, our algorithm recovers an optimal schedule when there are no false negatives, i.e., when $R^{-}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})/\mathrm{OPT} = 0.0$, and the makespan increases smoothly as the false-negative error grows. Even in the extreme case where all heavy predictions are removed, around $R^{-}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})/\mathrm{OPT} \simeq 0.38$, the makespan remains below the LST 2-approximation baseline (The fact that this baseline has $C_{\max}(S)/\mathrm{OPT} \simeq 1.26$ rather than 2 is consistent with the fact that the 2-approximation ratio is a worst-case guarantee). This behavior precisely matches the learning-augmented guarantee in Theorem 5.2.

Comparing $K = 0, 1, 2$, we observe that larger K consistently improves the makespan. This reflects the monotone decrease of $R^{-}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^{\star})$ in Theorem 5.2. In particular, for $K = 1$ when $R^{-}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})/\mathrm{OPT} \lesssim 0.1$, and for $K = 2$ when $R^{-}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})/\mathrm{OPT} \lesssim 0.14$, the algorithm still attains the optimal makespan despite the presence of false negatives. This is consistent with Corollary 5.3: in these regions, the condition $K \ge |\mathrm{Miss}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})|$ is satisfied, so a small amount of search around the prediction completely eliminates the effect of false negatives from the makespan.

We next discuss the false-positive experiment in Fig. 1b. As in the false-negative experiment, the algorithm attains the optimal makespan when there are no false positives, and the makespan degrades smoothly as the false-positive error increases. Even in the worst case, the makespan remains below

[4] `https://github.com/kAIto47802/rcmax-with-predictions`



(a) False negatives (b) False positives (c) Running time

Figure 1: Numerical results. (a) and (b) show the change in makespan as the false-negative and false-positive rates are varied, respectively. (c) shows the change in running time as the number of machines m is varied; the curves for Ours with $K = 0, 1, 2$ and the LST 2-approximation baseline overlap because they finish almost immediately. The lines denote the mean, and the shaded regions denote the standard error, both computed over 30 independent runs with different random seeds.

the classical 2-approximation baseline. Unlike the false-negative case, all choices of K yield nearly identical makespans. This reflects Theorem 5.2, where the false-positive term $R^{+}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^{\star})$ is independent of K. Thus, Figs. 1a and 1b illustrate the theoretical asymmetry: false negatives can be recovered by search, whereas false positives remain in the makespan bound.

Appendix H.1 reports additional results for other m, n settings and non-uniform instances with machine-dependent correlations in processing times, showing similar qualitative behavior consistently.

6.3 Running time comparison

We next examine the running time as the number of machines varies. In Fig. 1c, we fix $n = 10$ and vary m, with both the false-negative and false-positive rates set to 0.1.

Our algorithm and the classical LST 2-approximation baseline finish almost immediately even as m increases. In contrast, both classical $(1+\varepsilon)$-type baselines scale rapidly, reaching roughly 400 and 600 seconds even on these tiny m, n settings. This is consistent with Theorem 5.2: our algorithm, like the 2-approximation baseline, has polynomial running time with constant search budget K, whereas the classical $(1+\varepsilon)$-type baselines are not polynomial-time unless m is fixed to be small.

Appendix H.2 reports running time experiments with fixed m and varying n, where our algorithm remains comparable to the polynomial-time 2-approximation baseline and much faster than the classical $(1+\varepsilon)$-type baselines.

7 Conclusion

We developed a learning-augmented approximation algorithm for unrelated-machines makespan scheduling, addressing an open question of Antoniadis et al. (2025) on constructing smooth prediction-dependent guarantees to scheduling. The resulting algorithm achieves a $(1+\varepsilon)\mathrm{OPT}$ guarantee when the prediction is accurate, while its makespan bound degrades smoothly with false-negative and false-positive prediction errors and never exceeds the classical $2\mathrm{OPT}$ baseline. The search budget K also provides a tunable tradeoff between running time and solution quality. Our experiments support these theoretical properties, showing smooth degradation with prediction error, the predicted asymmetry between false negatives and false positives, and substantially better scalability than classical $(1+\varepsilon)$-type algorithms. We also extend the prediction-localized search approach to total weighted completion time on unrelated machines, obtaining a prediction-dependent guarantee while preserving the approximation ratio of a classical baseline (Appendix F). We hope that this prediction-localized perspective will help extend learning-augmented approximation guarantees to other optimization problems whose classical algorithms rely on expensive structural guessing.

References

Aamand, A., Chen, J. Y., Gollapudi, S., Silwal, S., and Wu, H. Improved approximations for hard graph problems using predictions. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 73–101. PMLR, 2025.

Antoniadis, A., Elias, M., Polak, A., and Venzin, M. Approximation algorithms for combinatorial optimization with predictions. In *The Thirteenth International Conference on Learning Representations*, 2025.

Azar, Y., Panigrahi, D., and Touitou, N. Discrete-smoothness in online algorithms with predictions. In *Advances in Neural Information Processing Systems*, volume 36, pp. 41932–41955. Curran Associates, Inc., 2023.

Bamas, E., Maggiori, A., and Svensson, O. The primal-dual method for learning augmented algorithms. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20083–20094. Curran Associates, Inc., 2020.

Bampis, E., Escoffier, B., and Xefteris, M. Parsimonious learning-augmented approximations for dense instances of $\mathcal{NP}$-hard problems. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 2700–2714. PMLR, 2024.

Bampis, E., Escoffier, B., Fotakis, D., Patsilinakos, P., and Xefteris, M. Polynomial time learning augmented algorithms for NP-hard permutation problems. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2790–2800. PMLR, 13–19 Jul 2025.

Bouška, M., Šůcha, P., Novák, A., and Hanzálek, Z. Deep learning-driven scheduling algorithm for a single machine problem minimizing the total tardiness. *European Journal of Operational Research*, 308(3):990–1006, 2023. ISSN 0377-2217. doi: 10.1016/j.ejor.2022.11.034.

Braverman, V., Dharangutte, P., Shah, V., and Wang, C. Learning-augmented maximum independent set. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 317 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 24:1–24:18. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. ISBN 978-3-95977-348-5. doi: 10.4230/LIPIcs.APPROX/RANDOM.2024.24.

Brucker, P. *Scheduling Algorithms*. Springer, 5 edition, 2007. ISBN 978-3-540-69516-5. doi: 10.1007/978-3-540-69516-5.

Cohen-Addad, V., d'Orsi, T., Gupta, A., Lee, E., and Panigrahi, D. Learning-augmented approximation algorithms for maximum cut and related problems. In *Advances in Neural Information Processing Systems*, volume 37, pp. 25961–25980. Curran Associates, Inc., 2024. doi: 10.52202/079017-0816.

Fanjul-Peyro, L. and Ruiz, R. Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, 207(1):55–69, 2010. ISSN 0377-2217. doi: 10.1016/j.ejor.2010.03.030.

Fanjul-Peyro, L. and Ruiz, R. Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research*, 38(1):301–309, 2011. ISSN 0305-0548. doi: 10.1016/j.cor.2010.05.005.

Graham, R., Lawler, E., Lenstra, J., and Kan, A. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pp. 287–326. Elsevier, 1979. doi: 10.1016/S0167-5060(08)70356-X.

Grigorescu, E., Lin, Y.-S., Silwal, S., Song, M., and Zhou, S. Learning-augmented algorithms for online linear and semidefinite programming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 38643–38654. Curran Associates, Inc., 2022.

Gupta, A., Panigrahi, D., Subercaseaux, B., and Sun, K. Augmenting online algorithms with ε-accurate predictions. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2115–2127. Curran Associates, Inc., 2022.

Hijazi, A., Ozaltin, O., and Uzsoy, R. All you need is an improving column: Enhancing column generation for parallel machine scheduling via transformers. *arXiv preprint arXiv:2410.15601*, 2024.

Huangfu, Q. and Hall, J. A. J. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 10(1):119–142, 2018. doi: 10.1007/s12532-017-0130-5.

Im, S., Kumar, R., Petety, A., and Purohit, M. Parsimonious learning-augmented caching. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9588–9601. PMLR, 17–23 Jul 2022.

Lenstra, J. K., Shmoys, D. B., and Tardos, É. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1):259–271, 1990. ISSN 1436-4646. doi: 10.1007/BF01585745.

Lykouris, T. and Vassilvitskii, S. Competitive caching with machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 3296–3305. PMLR, 2018.

Mitzenmacher, M. and Vassilvitskii, S. Algorithms with predictions. *Commun. ACM*, 65(7):33–35, 2022. ISSN 0001-0782. doi: 10.1145/3528087.

Mordechai, Y. Optimization and reoptimization in scheduling problems. *arXiv preprint arXiv:1509.01630*, 2015.

Perron, L. and Didier, F. Cp-sat. URL `https://developers.google.com/optimization/cp/cp_solver/`.

Pinedo, M. L. *Scheduling: Theory, Algorithms, and Systems*. Springer, 5 edition, 2016. ISBN 978-3-319-26580-3. doi: 10.1007/978-3-319-26580-3.

Purohit, M., Svitkina, Z., and Kumar, R. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Åblad, E., Strömberg, A.-B., and Spensieri, D. Exact makespan minimization of unrelated parallel machines. *Open Journal of Mathematical Optimization*, 2:2, 2021. doi: 10.5802/ojmo.4.

Rohatgi, D. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1834–1845. doi: 10.1137/1.9781611975994.112.

Shmoys, D. B. and Tardos, É. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62(1):461–474, Feb 1993. ISSN 1436-4646. doi: 10.1007/BF01585178.

Skutella, M. Convex quadratic and semidefinite programming relaxations in scheduling. *Journal of the ACM*, 48(2):206–242, 2001. ISSN 0004-5411. doi: 10.1145/375827.375840.

Smith, W. E. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3 (1–2):59–66, March 1956. doi: 10.1002/nav.3800030106.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Wei, A. Better and simpler learning-augmented online caching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, volume 176 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 60:1–60:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020. ISBN 978-3-95977-164-1. doi: 10.4230/LIPIcs.APPROX/RANDOM.2020.60.

A Extended related work

A.1 Learning-augmented online algorithms

The learning-augmented algorithms framework was developed primarily in the online setting (Mitzenmacher & Vassilvitskii, 2022). The central idea is to augment a worst-case algorithm with predictions, for example from a machine-learning model, and to derive guarantees that depend on the quality of these predictions. Early representative works include the learning-augmented caching algorithm of Lykouris & Vassilvitskii (2018) and the algorithms of Purohit et al. (2018) for ski rental and non-clairvoyant scheduling. These works introduced a paradigm in which predictions can improve performance when accurate, while the algorithm retains worst-case guarantees when predictions are inaccurate. This viewpoint is often summarized through the desiderata of consistency, robustness, and smoothness: the algorithm should perform well with accurate predictions, remain safe with poor predictions, and degrade gracefully as prediction error increases.

Subsequent work developed this framework in several directions. For caching, Rohatgi and Wei (2020) obtained sharper and simpler guarantees for learning-augmented caching, and later work studied parsimonious prediction models for the same problem (Im et al., 2022). Beyond caching, Bamas et al. (2020) introduced a primal-dual framework for learning-augmented online algorithms and applied it to a variety of online covering problems. This line was further generalized by Grigorescu et al. (2022), who studied learning-augmented online covering linear and semidefinite programs. Other representative works include learning-augmented algorithms with ε-accurate predictions for problems such as caching, online set cover, online facility location, and online network design (Gupta et al., 2022).

A.2 Learning-augmented approximation algorithms

A more recent line of work studies learning-augmented algorithms for offline NP-hard optimization problems. Here the prediction is no longer about future online inputs, but may describe part of an optimal solution, a structural property of an optimal solution, or an object that a classical approximation algorithm would otherwise need to guess.

Several recent works explore this direction beyond the online setting. Cohen-Addad et al. (2024) study Max-Cut and related max-CSPs with predictions, showing how learned information can help overcome classical approximation barriers. Bampis et al. (2024) consider parsimonious learning-augmented approximations for dense NP-hard problems. Braverman et al. (2024) study learning-augmented Maximum Independent Set using oracle access to predicted vertex membership in a fixed maximum independent set. Bampis et al. (2025) consider NP-hard permutation problems, including scheduling and network-design problems, with predictions about the relative order of elements in an optimal permutation. Aamand et al. (2025) design improved approximation algorithms for hard graph problems using edge-based predictions, including MaxCut, Vertex Cover, Set Cover, and Maximum Independent Set. These works highlight that the choice of prediction object is crucial: different problems require different forms of predictions.

A recent prominent work with broad applicability in offline learning-augmented approximation is Antoniadis et al. (2025). They give black-box algorithms for a broad class of selection problems with linear objectives, where a feasible subset of items is selected to minimize or maximize total weight. Their framework yields results for problems such as Vertex Cover, Steiner Tree, Min-Weight Perfect Matching, Knapsack, and Clique. However, scheduling problems, such as $R||C_{\max}$ lie outside this black-box setting, as they are assignment problems with a makespan objective rather than total-weight selection problems, and were explicitly identified as a natural open direction for future work. Our work takes a step in this direction by developing a learning-augmented algorithmic framework tailored to the heavy-assignment structure of unrelated-machine scheduling.

A.3 Classical approximation algorithms for $R||C_{\max}$

The unrelated-machines makespan problem $R||C_{\max}$ is a central problem in scheduling. The classical algorithm of Lenstra et al. (1990) gives a polynomial-time 2-approximation and shows that no polynomial-time approximation with ratio below $3/2$ is possible unless $P = NP$. The same work also gives a polynomial approximation scheme for the case of a fixed number of machines, which yields a $(1+\varepsilon)\mathrm{OPT}$ guarantee. The scheme uses a heavy/short decomposition: for a target

makespan T, assignments with processing time larger than εT are treated as large assignments, and there can be only a bounded number of such assignments on each machine in a schedule of makespan at most T. Thus, if the large assignments are guessed correctly, the remaining jobs are short and can be completed by solving an LP and applying the LST rounding procedure.

Mordechai (2015) later studied parameterized approximation schemes for scheduling, including unrelated-machines makespan. In this approach, the parameter is the number of large machine-job pairs $k(T) = |\{(i,j) \mid p_{ij} > \varepsilon T\}|$. The algorithm treats the variables corresponding to these large pairs integrally and rounds the remaining fractional solution. This yields a $(1+\varepsilon)\mathrm{OPT}$ guarantee with running time exponential in $k(T)$ and polynomial in the input size.

Our algorithm can be viewed as a prediction-localized refinement of this paradigm: instead of searching over the full heavy-assignment structure, it searches only for the heavy assignments that are missing from the prediction, yielding prediction-sensitive tradeoffs in both running time and approximation quality.

A.4 Learning and scheduling

There is also some work using machine learning to improve the practical performance of scheduling algorithms and solvers. Bouška et al. (2023) use a neural network within a decomposition-based algorithm for the single-machine total tardiness problem, where the model guides the decomposition by estimating the best split. Relatedly, Hijazi et al. (2024) use transformer-style models to enhance column generation for parallel-machine scheduling by predicting improving columns. These works show that ML predictions can be useful in scheduling practice, especially when they guide expensive search or decomposition steps. However, their focus is primarily empirical or solver-acceleration oriented. Also, neither of these works studies $R\|C_{\max}$ itself. In contrast, our work gives approximation-theoretic guarantees: the prediction determines a local neighborhood of the classical heavy-assignment search, and the resulting makespan and running time bounds depend explicitly on prediction errors.

B Choice of Prediction Model

Predicting which jobs should be assigned to which machines is a particularly natural prediction model for this problem, and it directly targets the main combinatorial difficulty in the classical approximation scheme.

For comparison, we discuss below several alternative prediction objects and their limitations.

- Full job-to-machine assignment: This is already covered as a special case of our model (as we wrote in Remark 4.1). Allowing partial predictions is more flexible, since the predictor can abstain on uncertain jobs and leave them to the algorithm rather than fixing them incorrectly.
- Set of heavy jobs: This is insufficient for unrelated machines because heaviness is a property of a machine-job pair (i,j), rather than of a job alone.
- Optimal makespan or the load of every machine in an optimal schedule: These predictions provide only aggregate information and do not identify the combinatorial heavy-assignment structure. Even if the exact value $T = \mathrm{OPT}$ were known, classical $(1+\varepsilon)$-approximation schemes would still need to search for the heavy assignments.
- Fractional LP solution for the full instance: A fractional solution does not directly provide the required integral heavy-assignment structure. In particular, rounding fractional mass on heavy edges can incur a large additive loss, so it cannot replace the heavy-assignment prediction without additional structural assumptions.
- Processing-time prediction: In our offline setting, every processing time p_{ij} is already given as part of the input. Predicting processing times therefore provides no additional information; this would instead correspond to a different problem in which processing times are unknown or stochastic.

C Proofs deferred from the main text

C.1 Full proof of Theorem 5.1

In Section 5.1, we presented the proof sketch of Theorem 5.1. Here we provide the full proof.

Proof. By the definition of $R^-_{T,K}(\widehat{\phi}, \sigma)$, there exists a subset $U^\star \subseteq \mathrm{Miss}_T(\widehat{\phi}, \sigma)$ with $|U^\star| \leq K$ such that every missed heavy job outside $U^\star$ has processing time at most $R^-_{T,K}(\widehat{\phi}, \sigma)$, i.e., $\max_{j \in \mathrm{Miss}_T(\widehat{\phi},\sigma) \setminus U^\star} p_{\sigma(j),j} = R^-_{T,K}(\widehat{\phi}, \sigma)$. Since $U^\star \subseteq \mathrm{Miss}_T(\widehat{\phi}, \sigma)$, every job in $U^\star$ is missed by $\widehat{\phi}_T$ and satisfies $p_{\sigma(j),j} > \varepsilon T$.

Define an augmentation

$$A^\star(j) \coloneqq \begin{cases} \sigma(j), & \text{if } j \in U^\star, \\ \perp, & \text{otherwise.} \end{cases} \tag{14}$$

Then we have $|\operatorname{dom}(A^\star)| = |U^\star| \leq K$ by construction. Thus, $A^\star$ satisfies Eq. (2) and is considered by the algorithm.

Let $\psi^\star \coloneqq \widehat{\phi}_T \cup A^\star$ and define $\tau^\star \coloneqq \max\{\varepsilon T,\, R^-_{T,K}(\widehat{\phi}, \sigma)\}$. By construction, $\tau^\star$ is one of the threshold candidate set $\Lambda(T)$ considered by the algorithm: if $R^-_{T,K}(\widehat{\phi}, \sigma) = 0$, then $\tau^\star = \varepsilon T$; otherwise, $R^-_{T,K}(\widehat{\phi}, \sigma)$ is the processing time $p_{\sigma(j),j}$ of some missed heavy job j.

Now we claim that the residual LP with threshold $\tau^\star$ for $\psi^\star$ is feasible with load bound $L \coloneqq C_{\max}(\sigma) + R^+_T(\widehat{\phi}, \sigma)$. Consider the fractional assignment that sends every residual job $j \notin \operatorname{dom}(\psi^\star)$ to machine $\sigma(j)$. We verify feasibility of this assignment for the residual LP with threshold $\tau^\star$.

Let $j \notin \operatorname{dom}(\psi^\star)$. If $p_{\sigma(j),j} \leq \varepsilon T$, then clearly $p_{\sigma(j),j} \leq \tau^\star$. Otherwise, $p_{\sigma(j),j} > \varepsilon T$. Since $j \notin \operatorname{dom}(\psi^\star)$, we have $\widehat{\phi}_T(j) = \perp$ and $j \notin U^\star$. Hence $j \in \mathrm{Miss}_T(\widehat{\phi}, \sigma) \setminus U^\star$, and therefore we have $p_{\sigma(j),j} \leq R^-_{T,K}(\widehat{\phi}, \sigma) \leq \tau^\star$. Thus, in all cases, the edge $(\sigma(j), j)$ is allowed in the residual LP with threshold $\tau^\star$.

We now verify the machine-capacity constraints. Fix a machine i. Under the assignment that follows $\psi^\star$ on fixed jobs and σ on residual jobs, the total load on machine i is

$$\begin{aligned} \ell_i^{\mathrm{fix}}(\psi^\star) + \sum_{\substack{j:\sigma(j)=i,\\ \psi^\star(j)=\perp}} p_{ij} &= \sum_{\substack{j:\sigma(j)=i\\ \psi^\star(j)\in\{\perp,i\}}} p_{ij} + \sum_{\substack{j:\psi^\star(j)=i\\ \sigma(j)\neq i}} p_{ij} = \sum_{\substack{j:\sigma(j)=i\\ \psi^\star(j)\in\{\perp,i\}}} p_{ij} + \sum_{\substack{j:\widehat{\phi}_T(j)=i\\ \widehat{\phi}_T(j)\neq\sigma(j)}} p_{ij} \\ &\leq \sum_{j:\sigma(j)=i} p_{ij} + \sum_{\substack{j:\widehat{\phi}_T(j)=i\\ \widehat{\phi}_T(j)\neq\sigma(j)}} p_{ij} \leq C_{\max}(\sigma) + R^+_T(\widehat{\phi}, \sigma) =: L. \end{aligned} \tag{15}$$

Here, the first equality uses

$$\ell_i^{\mathrm{fix}}(\psi^\star) = \sum_{j:\psi^\star(j)=i} p_{ij} = \sum_{\substack{j:\psi^\star(j)=i\\ \sigma(j)=i}} p_{ij} + \sum_{\substack{j:\psi^\star(j)=i\\ \sigma(j)\neq i}} p_{ij}, \tag{16}$$

and the second equality holds because every job in $A^\star$ is assigned according to σ by construction, so the only fixed jobs that may increase the load of machine i relative to σ are those coming from $\widehat{\phi}_T$.

Thus, from Eq. (15), if we define

$$x_{ij} = \begin{cases} 1, & \text{if } i = \sigma(j), \\ 0, & \text{otherwise,} \end{cases} \quad j \notin \operatorname{dom}(\psi^\star), \tag{17}$$

we have

$$\sum_{j \notin \operatorname{dom}(\psi^\star)} p_{ij} x_{ij} = \sum_{\substack{j:\sigma(j)=i,\\ \psi^\star(j)=\perp}} p_{ij} \leq L - \ell_i^{\mathrm{fix}}(\psi^\star). \tag{18}$$

Hence the residual jobs, when assigned according to σ, form a feasible solution to the residual LP with load bound L. In particular, the residual LP is feasible and its optimum value satisfies $L^{\tau^\star}_{\psi^\star} \leq L$.

By the classical rounding argument (Theorem 2.1 in Shmoys & Tardos (1993)), the algorithm can round this residual solution into an integral schedule of makespan at most $L^{\tau^\star}_{\psi^\star} + \tau^\star$. Therefore, for the candidate pair $(A^\star, \tau^\star)$, the algorithm obtains a schedule of makespan at most

$$C_{\max}(\sigma) + R^+_T(\widehat{\phi}, \sigma) + \max\{\varepsilon T,\ R^-_{T,K}(\widehat{\phi}, \sigma)\}. \tag{19}$$

Since Algorithm 1 returns the best schedule among all candidates, its output satisfies the same bound.

It remains to analyze the running time. The computation of $\widehat{\phi}_T$ takes polynomial time. The number of augmentations considered by the algorithm is at most

$$\sum_{s=0}^{K} \binom{n}{s} m^s \le (K+1)(mn)^K \le (n+1)(mn)^K. \tag{20}$$

For each augmentation A, the algorithm iterates over the threshold set $\Lambda(T)$, whose size is at most $mn + 1$. For each pair (A, τ), it forms the combined assignment, solves the residual LP, and, if feasible, applies LST rounding. Each of these steps takes time polynomial in m, n, $\log P$, and $1/\varepsilon$. Hence the total running time is $O((mn)^K \cdot \mathrm{poly}(m, n, \log P, 1/\varepsilon))$. □

C.2 Proofs of Corollaries 5.3 and 5.4

Proof of Corollary 5.3. Since $K \ge |\mathrm{Miss}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)|$, all missed $\varepsilon\mathrm{OPT}$-heavy assignments can be recovered by the augmentation step. Hence $R^-_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star) = 0$. Applying Theorem 5.2, we obtain the claimed bound immediately. The running-time bound is inherited directly from Theorem 5.2. □

Proof of Corollary 5.4. Set $K = 0$ in Theorem 5.2. Since we have $R^-_{\mathrm{OPT},0}(\widehat{\phi}, \sigma^\star) = R^-_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$ by definition, Theorem 5.2 gives the claimed bound immediately. The running-time bound also follows immediately by substituting $K = 0$ into the running time of Theorem 5.2. □

D Additional discussions

D.1 Additional discussions for the inner layer

The following corollary is obtained as the special case of Theorem 5.1 in which the search budget K is sufficiently large.

Corollary D.1. *Suppose there exists a schedule σ such that $K \ge |\mathrm{Miss}_T(\widehat{\phi}, \sigma)|$. Then, Algorithm 1 returns a feasible schedule S satisfying*

$$C_{\max}(S) \le C_{\max}(\sigma) + \varepsilon T + R^+_T(\widehat{\phi}, \sigma). \tag{21}$$

The running time of the algorithm is $O((mn)^K \mathrm{poly}(m, n, \log P, 1/\varepsilon))$.

Proof. Since $K \ge |\mathrm{Miss}_T(\widehat{\phi}, \sigma)|$, the algorithm can recover all missed heavy assignments. Hence, by the definition, we have $R^-_{T,K}(\widehat{\phi}, \sigma) = 0$. Applying Theorem 5.1, we obtain the claimed bound immediately. The running-time bound is inherited directly from Theorem 5.1. □

Interpretation. Corollary D.1 reveals a genuine asymmetry between the two types of prediction error. False-negative heavy assignments affect the running time through the augmentation budget K, since they must be recovered by additional local search. On the other hand, false-positive heavy assignments do not enlarge the search space, but instead consume machine capacity and therefore appear directly in the makespan bound through $R^+_T(\widehat{\phi}, \sigma)$.

The required running time to satisfy the condition degrades smoothly with the amount of false-negative heavy predictions, since the required augmentation budget K grows with the number of missed-heavy jobs $|\mathrm{Miss}_T(\widehat{\phi}, \sigma)|$. At the same time, even when the number of false negatives is large, the search remains in the same asymptotic regime as the classical algorithm based on exhaustive search over heavy assignments (Lenstra et al., 1990; Mordechai, 2015).

On the quality side, if the false-positive heavy assignments are negligible, then the algorithm achieves a $(1+\varepsilon)OPT$ makespan guarantee, matching the classical completion-based guarantee with exhaustive search (Lenstra et al., 1990; Mordechai, 2015). As the false-positive load increases, the makespan guarantee deteriorates smoothly through the additive overload term $R_T^+(\widehat{\phi}, \sigma)$.

As the opposite extreme, when no search is performed, i.e., $K = 0$, we obtain the following corollary.

Corollary D.2. *When $K = 0$, for any schedule σ, Algorithm 1 returns a feasible schedule S satisfying*

$$C_{\max}(S) \le C_{\max}(\sigma) + \max\{\varepsilon T,\ R_T^-(\widehat{\phi}, \sigma)\} + R_T^+(\widehat{\phi}, \sigma). \tag{22}$$

In particular, if σ has makespan at most T, then the returned schedule S satisfies

$$\begin{aligned} C_{\max}(S) &\le T + \max\{\varepsilon T,\ R_T^-(\widehat{\phi}, \sigma)\} + R_T^+(\widehat{\phi}, \sigma) \\ &\le (1+\varepsilon)T + R_T^-(\widehat{\phi}, \sigma) + R_T^+(\widehat{\phi}, \sigma). \end{aligned} \tag{23}$$

The running time of the algorithm is $O(\mathrm{poly}(m, n, \log P, 1/\varepsilon))$.

Proof. Apply Theorem 5.1 with $K = 0$. Since $R_{T,0}^-(\widehat{\phi}, \sigma) = R_T^-(\widehat{\phi}, \sigma)$ by definition, we obtain the claimed makespan bound immediately. The running-time bound also follows immediately from Theorem 5.1, which gives $O((mn)^0 \cdot \mathrm{poly}(m, n, \log P, 1/\varepsilon)) = O(\mathrm{poly}(m, n, \log P, 1/\varepsilon))$. □

Interpretation. Corollary D.2 eliminates the search requirement on K. Instead of recovering missed heavy assignments combinatorially, the algorithm absorbs them into the approximation bound through $R_T^-(\widehat{\phi}, \sigma)$, making the $(mn)^{O(K)}$ search disappear entirely. Here, the upper bound deteriorates smoothly with both false-negative term $R_T^-(\widehat{\phi}, \sigma)$ and false-positive term $R_T^+(\widehat{\phi}, \sigma)$.

E A variant with editable predictions

The formulation we presented in Section 4 trusts every predicted assignment that survives the T-heavy projection. We now describe a more flexible variant in which the algorithm is allowed to delete, insert, or edit a bounded number of heavy predicted assignments. This variant uses the same residual LP and rounding procedure as before; the only change is the local neighborhood of heavy partial assignments searched by the inner layer.

E.1 Editable heavy-assignment neighborhood

We first define the neighborhood of heavy partial assignments considered by the inner layer. Fix a threshold $T > 0$. Recall that $\widehat{\phi}_T$ denotes the T-heavy projection of the prediction. For a partial assignment $\psi : [n] \to [m] \cup \{\perp\}$, define its edit distance from the projected prediction at scale T by

$$d_T(\psi, \widehat{\phi}) := \left| \left\{ j \in [n] \;\middle|\; \psi(j) \neq \widehat{\phi}_T(j) \right\} \right|. \tag{24}$$

Thus, changing $\perp$ to a machine corresponds to an insertion, changing a predicted machine to $\perp$ corresponds to a deletion, and changing a predicted machine to another one corresponds to an edit.

For an edit budget $K \ge 0$, define the editable K-neighborhood of the prediction by

$$\mathcal{E}_{T,K}(\widehat{\phi}) := \left\{ \psi \colon [n] \to [m] \cup \{\perp\} \;\middle|\; d_T(\psi, \widehat{\phi}) \le K,\ \psi(j) \neq \perp \Rightarrow \varepsilon T < p_{\psi(j),j} \right\}. \tag{25}$$

The editable inner-layer algorithm enumerates all $\psi \in \mathcal{E}_{T,K}(\widehat{\phi})$ in Step 2, in place of all augmentations defined in Eq. (2). The rest of the algorithm is unchanged. For each such ψ, it fixes the jobs in $\mathrm{dom}(\psi)$, solves the same residual LP over thresholds $\tau \in \Lambda(T)$, applies Shmoys–Tardos rounding whenever the LP is feasible, and returns the best schedule among all candidates.

E.2 Edit-aware prediction error measures

We next define prediction-error measures for the editable variant. The key point is that deletions and edits may trade false positives against false negatives: deleting an incorrectly fixed heavy assignment

can reduce false-positive overload, but if the job is truly heavy in the comparison schedule, it may become a missed heavy assignment. Therefore, the two components must be defined with respect to the same edited prediction.

Accordingly, we use the same false-negative and false-positive error measures as in Eq. (6), but evaluate them on the edited partial assignment ψ instead of the projected prediction $\widehat{\phi}_T$. Concretely, fix an arbitrary minimizer

$$\psi^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) \in \operatorname*{arg\,min}_{\psi\in\mathcal{E}_{T,K}(\widehat{\phi})} \left[\max\{\varepsilon T, R^-_T(\psi,\sigma)\} + R^+_T(\psi,\sigma)\right], \tag{26}$$

where ties are broken deterministically. We then define the edit-aware false-negative and false-positive components by

$$R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) := R^-_T\left(\psi^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma),\sigma\right), \qquad R^{+,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) := R^+_T\left(\psi^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma),\sigma\right). \tag{27}$$

Equivalently, it is simpler to define the edit-aware combined prediction error directly as

$$\Delta^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) := \min_{\psi\in\mathcal{E}_{T,K}(\widehat{\phi})} \left[\max\{\varepsilon T, R^-_T(\psi,\sigma)\} + R^+_T(\psi,\sigma)\right]. \tag{28}$$

By definition, we have

$$\Delta^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) = \max\left\{\varepsilon T, R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma)\right\} + R^{+,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma). \tag{29}$$

The quantity $\Delta^{\mathrm{edit}}_{T,K}$ is monotone nonincreasing in the budget K, although $R^{-,\mathrm{edit}}_{T,K}$ and $R^{+,\mathrm{edit}}_{T,K}$ need not be individually monotone because an edit may trade one type of error for the other.

E.3 Inner-layer guarantee for the editable variant

Theorem E.1 (Inner-layer guarantee with editable predictions)**.** *Fix $\varepsilon \in (0,1)$, a target value $T > 0$, an edit budget $K \geq 0$, and a prediction $\widehat{\phi}$. Consider the editable inner-layer algorithm that enumerates all $\psi \in \mathcal{E}_{T,K}(\widehat{\phi})$ in place of all augmentations defined in Eq.* (2)*.*

Then, for any schedule σ, the algorithm returns a feasible schedule S satisfying

$$C_{\max}(S) \leq C_{\max}(\sigma) + \max\left\{\varepsilon T, R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma)\right\} + R^{+,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma). \tag{30}$$

Equivalently,

$$C_{\max}(S) \leq C_{\max}(\sigma) + \Delta^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma). \tag{31}$$

The running time of the algorithm is $O\big((mn)^K \operatorname{poly}(m,n,\log P, 1/\varepsilon)\big)$.

Proof. Let $\psi^\star := \psi^{\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma)$ be an edited partial assignment attaining the definition of the edit-aware error measures. Since $\psi^\star \in \mathcal{E}_{T,K}(\widehat{\phi})$, it is one of the candidates enumerated by the editable inner-layer algorithm.

Define $\tau^\star := \max\left\{\varepsilon T, R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma)\right\}$. We first note that $\tau^\star \in \Lambda(T)$. If $R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) = 0$, then $\tau^\star = \varepsilon T$. Otherwise, $R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma)$ is equal to $p_{\sigma(j),j}$ for some missed heavy job j, and hence appears in the discrete threshold set $\Lambda(T)$.

Consider the residual LP for the fixed assignment $\psi^\star$ and threshold $\tau^\star$. Assign every residual job $j \notin \mathrm{dom}(\psi^\star)$ to the machine $\sigma(j)$. We verify that each such edge is allowed. If $p_{\sigma(j),j} \leq \varepsilon T$, then $p_{\sigma(j),j} \leq \tau^\star$. Otherwise $p_{\sigma(j),j} > \varepsilon T$. Since j is residual under $\psi^\star$, we have $\psi^\star(j) = \perp$, and therefore $j \in \mathrm{Miss}_T(\psi^\star,\sigma)$. Hence $p_{\sigma(j),j} \leq R^-_T(\psi^\star,\sigma) = R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi},\sigma) \leq \tau^\star$. Thus the assignment of residual jobs according to σ uses only edges allowed by the residual LP.

We next bound the load. Fix a machine i. Under the assignment that follows $\psi^\star$ on fixed jobs and follows σ on residual jobs, the load on machine i is

$$\ell^{\mathrm{fix}}_i(\psi^\star) + \sum_{\substack{j:\sigma(j)=i\\ \psi^\star(j)=\perp}} p_{ij} = \sum_{\substack{j:\sigma(j)=i\\ \psi^\star(j)\in\{\perp,i\}}} p_{ij} + \sum_{\substack{j:\psi^\star(j)=i\\ \psi^\star(j)\neq\sigma(j)}} p_{ij}. \tag{32}$$

Therefore,

$$\ell_i^{\mathrm{fix}}(\psi^\star) + \sum_{\substack{j:\sigma(j)=i \\ \psi^\star(j)=\perp}} p_{ij} \le \sum_{j:\sigma(j)=i} p_{ij} + R_T^+(\psi^\star,\sigma) \le C_{\max}(\sigma) + R_{T,K}^{+,\mathrm{edit}}(\widehat{\phi},\sigma). \tag{33}$$

Hence the residual LP for $(\psi^\star,\tau^\star)$ is feasible with load bound at most $C_{\max}(\sigma) + R_{T,K}^{+,\mathrm{edit}}(\widehat{\phi},\sigma)$.

By the Shmoys–Tardos rounding guarantee, rounding the residual LP solution increases the load of every machine by at most $\tau^\star$. Thus, for the candidate $\psi^\star$, the algorithm obtains a schedule of makespan at most

$$C_{\max}(\sigma) + R_{T,K}^{+,\mathrm{edit}}(\widehat{\phi},\sigma) + \max\left\{\varepsilon T, R_{T,K}^{-,\mathrm{edit}}(\widehat{\phi},\sigma)\right\}. \tag{34}$$

Since the algorithm returns the best candidate, the same bound holds for its output.

It remains to bound the running time. The number of edited partial assignments in $\mathcal{E}_{T,K}(\widehat{\phi})$ is at most $\sum_{s=0}^{K}\binom{n}{s}(m+1)^s = O((mn)^K)$. For each candidate ψ, the algorithm considers at most $mn+1$ thresholds in $\Lambda(T)$, and for each pair (ψ,τ) it solves one LP and performs one rounding step, both in polynomial time. This gives the claimed running time. □

Interpretation. Theorem E.1 shows that allowing edits changes the role of the budget K from a false-negative recovery budget into a general correction budget for the predicted heavy-assignment structure. In the original augmentation-only variant, the algorithm can only insert missed heavy assignments; hence false positives remain fixed and appear directly in the makespan bound. In contrast, the editable variant can spend part of its budget to delete an incorrect prediction or to edit it to another machine. Thus, both false negatives and false positives can be repaired by local search.

The guarantee should be interpreted through the combined error $\Delta_{T,K}^{\mathrm{edit}}(\widehat{\phi},\sigma)$. This quantity is monotone nonincreasing in K, because a larger budget only enlarges the edit neighborhood $\mathcal{E}_{T,K}(\widehat{\phi})$. However, the two components $R_{T,K}^{-,\mathrm{edit}}(\widehat{\phi},\sigma)$ and $R_{T,K}^{+,\mathrm{edit}}(\widehat{\phi},\sigma)$ need not be individually monotone. For example, deleting a false-positive heavy assignment may reduce false-positive overload, but if that job is heavy in the comparison schedule, it may become a missed heavy assignment. This is why the theorem defines the edit-aware error by minimizing the sum

$$\max\{\varepsilon T, R_T^-(\psi,\sigma)\} + R_T^+(\psi,\sigma) \tag{35}$$

over a common edited partial assignment ψ.

Consequently, K provides a smooth tradeoff between running time and the quality of the edited prediction. Larger values of K allow the algorithm to search a larger neighborhood of the prediction and can only improve the combined prediction-dependent bound, while increasing the running time by the factor $(mn)^K$. When $K=0$, the editable variant reduces to trusting the projected prediction $\widehat{\phi}_T$ without any local correction; as K increases, the algorithm gradually interpolates toward the best completion obtainable after correcting up to K heavy predicted assignments.

E.4 Outer-layer guarantee for the editable variant

We next show that the outer-layer guarantee for the augmentation-only variant (Theorem 5.2) also extends to the editable variant with the same proof structure, by simply replacing the augmentation-based error measures with their edit-aware counterparts.

Theorem E.2 (Outer-layer guarantee with editable predictions)**.** *Fix $\varepsilon \in (0,1)$, an edit budget $K \ge 0$, and a prediction $\widehat{\phi}$. Let $\sigma^\star$ be an optimal schedule, and write $\mathrm{OPT} := C_{\max}(\sigma^\star)$. The editable outer-layer algorithm returns a feasible schedule S satisfying*

$$C_{\max}(S) \le \min\left\{\mathrm{OPT} + \max\left\{\varepsilon\mathrm{OPT}, R_{\mathrm{OPT},K}^{-,\mathrm{edit}}(\widehat{\phi},\sigma^\star)\right\} + R_{\mathrm{OPT},K}^{+,\mathrm{edit}}(\widehat{\phi},\sigma^\star),\ 2\mathrm{OPT}\right\}. \tag{36}$$

Equivalently,

$$C_{\max}(S) \le \min\left\{\mathrm{OPT} + \Delta_{\mathrm{OPT},K}^{\mathrm{edit}}(\widehat{\phi},\sigma^\star),\ 2\mathrm{OPT}\right\}. \tag{37}$$

In particular,

$$C_{\max}(S) \le \min\left\{(1+\varepsilon)\mathrm{OPT} + R_{\mathrm{OPT},K}^{-,\mathrm{edit}}(\widehat{\phi},\sigma^\star) + R_{\mathrm{OPT},K}^{+,\mathrm{edit}}(\widehat{\phi},\sigma^\star),\ 2\mathrm{OPT}\right\}. \tag{38}$$

The running time of the algorithm is $O\big((mn)^K \operatorname{poly}(m,n,\log P, 1/\varepsilon)\big)$.

Proof. Let S_0 be the baseline schedule computed by the classical 2-approximation, and let $B = C_{\max}(S_0)$. Then $\mathrm{OPT} \in [B/2, B]$. Let $T^\star$ be the largest critical threshold in the outer search satisfying $T^\star \le \mathrm{OPT}$.

By construction of the critical threshold set, there is no value of the form p_{ij}/ε in the open interval $(T^\star, \mathrm{OPT}]$. Therefore, for every machine-job pair (i, j), we have the equivalence $p_{ij} > \varepsilon T^\star \Leftrightarrow p_{ij} > \varepsilon\mathrm{OPT}$. Consequently, the heavy/short decomposition is identical at thresholds $T^\star$ and OPT. In particular, we have $\widehat{\phi}_{T^\star} = \widehat{\phi}_{\mathrm{OPT}}$, and $\mathcal{E}_{T^\star,K}(\widehat{\phi}) = \mathcal{E}_{\mathrm{OPT},K}(\widehat{\phi})$.

Let $\psi^\circ := \psi^{\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ be an edited partial assignment attaining the edit-aware error at scale OPT. Since $\mathcal{E}_{T^\star,K}(\widehat{\phi}) = \mathcal{E}_{\mathrm{OPT},K}(\widehat{\phi})$, the inner call at threshold $T^\star$ enumerates ψ°.

Moreover, because the heavy/short decomposition is identical at $T^\star$ and OPT, the primitive missed-heavy set and the primitive false-positive overload of ψ° with respect to $\sigma^\star$ are the same at the two thresholds:

$$R^-_{T^\star}(\psi^\circ, \sigma^\star) = R^-_{\mathrm{OPT}}(\psi^\circ, \sigma^\star), \qquad R^+_{T^\star}(\psi^\circ, \sigma^\star) = R^+_{\mathrm{OPT}}(\psi^\circ, \sigma^\star). \tag{39}$$

Applying the same inner-layer argument to the candidate ψ° at threshold $T^\star$ gives a schedule $S_{T^\star}$ satisfying

$$C_{\max}(S_{T^\star}) \le \mathrm{OPT} + \max\left\{\varepsilon T^\star, R^-_{\mathrm{OPT}}(\psi^\circ, \sigma^\star)\right\} + R^+_{\mathrm{OPT}}(\psi^\circ, \sigma^\star). \tag{40}$$

Since $T^\star \le \mathrm{OPT}$, we have $\max\left\{\varepsilon T^\star, R^-_{\mathrm{OPT}}(\psi^\circ, \sigma^\star)\right\} \le \max\left\{\varepsilon\mathrm{OPT}, R^-_{\mathrm{OPT}}(\psi^\circ, \sigma^\star)\right\}$. By the definition of ψ°, we have $R^-_{\mathrm{OPT}}(\psi^\circ, \sigma^\star) = R^{-,\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ and $R^+_{\mathrm{OPT}}(\psi^\circ, \sigma^\star) = R^{+,\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$. Therefore, it follows that

$$C_{\max}(S_{T^\star}) \le \mathrm{OPT} + \max\left\{\varepsilon\mathrm{OPT}, R^{-,\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)\right\} + R^{+,\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star). \tag{41}$$

The outer algorithm returns the best schedule among all inner-layer candidates and the baseline schedule S_0. Since $C_{\max}(S_0) \le 2\mathrm{OPT}$, the claimed bound follows.

Finally, the outer search considers at most $mn + 2$ critical thresholds. Combining this with the running-time bound from Theorem E.1 gives the claimed running time. □

Interpretation. Compared with the augmentation-only guarantee in Theorem 5.2, the editable guarantee is strictly more flexible. False-negative heavy assignments can be inserted, false-positive heavy assignments can be deleted, and incorrect heavy assignments can be edited to another machine. Therefore, the relevant error is not merely the number or size of missed heavy assignments, nor merely the overload caused by trusted false positives, but rather the smallest residual combination of these two effects after at most K edits. This residual combination is exactly captured by $\Delta^{\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$.

As the prediction becomes more accurate, the edit-aware error $\Delta^{\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star)$ decreases and the guarantee approaches $(1+\varepsilon)\mathrm{OPT}$. As the prediction becomes worse, the bound degrades smoothly through the same error term, but it never exceeds $2\mathrm{OPT}$ regardless of the prediction quality. Thus the editable variant preserves the same consistency–smoothness–robustness behavior as the main algorithm, while allowing the search budget to correct both types of prediction error.

Relation to the augmentation-only guarantee (Theorem 5.2). The editable guarantee generalizes the augmentation-only guarantee. Indeed, the augmentation-only search considered earlier corresponds to the restricted subfamily of $\mathcal{E}_{T,K}(\widehat{\phi})$ in which the only allowed changes are insertions of missed heavy assignments, i.e., changes of the form $\perp \to i$. For the candidate that inserts the best K missed heavy assignments according to the comparison schedule σ, the primitive false-negative threshold is exactly $R^-_{T,K}(\widehat{\phi}, \sigma)$ and the primitive false-positive overload remains $R^+_T(\widehat{\phi}, \sigma)$. Since the editable error minimizes over a larger family of corrected partial assignments, we have

$$\Delta^{\mathrm{edit}}_{T,K}(\widehat{\phi}, \sigma) \le \max\{\varepsilon T, R^-_{T,K}(\widehat{\phi}, \sigma)\} + R^+_T(\widehat{\phi}, \sigma). \tag{42}$$

Equivalently,

$$\max\left\{\varepsilon T, R^{-,\mathrm{edit}}_{T,K}(\widehat{\phi}, \sigma)\right\} + R^{+,\mathrm{edit}}_{T,K}(\widehat{\phi}, \sigma) \le \max\{\varepsilon T, R^-_{T,K}(\widehat{\phi}, \sigma)\} + R^+_T(\widehat{\phi}, \sigma). \tag{43}$$

Thus, allowing deletions and edits cannot worsen the prediction-dependent bound at the same asymptotic running time.

E.5 Corollary for consistency under small edit distance

For further intuition, we state the following consistency corollary for the case where the prediction is sufficiently accurate.

Corollary E.3 (Consistency under small edit distance)**.** *Let $\sigma^\star$ be an optimal schedule. Define its heavy projection at scale* OPT *by*

$$\sigma^\star_{\mathrm{OPT}}(j) := \begin{cases} \sigma^\star(j), & \text{if } p_{\sigma^\star(j),j} > \varepsilon \mathrm{OPT}, \\ \perp, & \text{otherwise}. \end{cases} \tag{44}$$

Define the heavy edit distance between the prediction and the optimum by

$$d^{\mathrm{edit}}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star) := \left| \left\{ j \in [n] : \widehat{\phi}_{\mathrm{OPT}}(j) \neq \sigma^\star_{\mathrm{OPT}}(j) \right\} \right|. \tag{45}$$

If $K \geq d^{\mathrm{edit}}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$, then the editable outer-layer algorithm returns a feasible schedule S satisfying

$$C_{\max}(S) \leq (1+\varepsilon)\mathrm{OPT}. \tag{46}$$

Proof. The condition $K \geq d^{\mathrm{edit}}_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$ implies that $\sigma^\star_{\mathrm{OPT}} \in \mathcal{E}_{\mathrm{OPT},K}(\widehat{\phi})$. For this edited partial assignment, there are no missed εOPT-heavy jobs and no false-positive fixed assignments with respect to $\sigma^\star$. Hence we have $R^-_{\mathrm{OPT}}(\sigma^\star_{\mathrm{OPT}}, \sigma^\star) = 0, R^+_{\mathrm{OPT}}(\sigma^\star_{\mathrm{OPT}}, \sigma^\star) = 0$, and therefore $\Delta^{\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star) \leq \varepsilon\mathrm{OPT}$. Applying Theorem E.2 gives $C_{\max}(S) \leq \min\{(1+\varepsilon)\mathrm{OPT}, 2\mathrm{OPT}\} = (1+\varepsilon)\mathrm{OPT}$, where the equality uses $\varepsilon \in (0,1)$. □

Interpretation. Corollary E.3 identifies the strongest consistency regime for the editable variant. If the predicted heavy partial assignment is within K edits of the heavy projection of an optimal schedule, the algorithm incurs only an additive εOPT loss.

This differs from the augmentation-only consistency statement in Corollary 5.3. In the augmentation-only variant, a sufficiently large budget can remove all false negatives, but any false positives that survive the T-heavy projection remain fixed and still contribute to the makespan bound. On the other hand, in the editable variant, false positives can also be corrected, either by deletion or by editing the predicted machine. Hence, when the entire heavy partial assignment is within the edit budget, both residual error terms vanish:

$$R^{-,\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star) = R^{+,\mathrm{edit}}_{\mathrm{OPT},K}(\widehat{\phi}, \sigma^\star) = 0. \tag{47}$$

The algorithm therefore achieves the clean $(1+\varepsilon)$OPT guarantee.

As in Corollary 5.3, the required budget here is also prediction-sensitive. If the prediction already matches the optimal heavy projection, then $K = 0$ is sufficient. If the prediction has a small number of incorrect, missing, or extra heavy assignments, the required budget increases by exactly the heavy edit distance. Even in the worst case, the required budget does not exceed the one needed in the classical $(1+\varepsilon)$-type baselines.

F Extension to total weighted completion time

The idea of using predictions to localize the assignment search also extends to the unrelated-machines total weighted completion-time problem $R\|\sum_j w_j C_j$, another NP-hard scheduling problem (Skutella, 2001). Following the K-edit approach in Appendix E, we search over assignments that differ from a full assignment prediction on at most K jobs. For each candidate assignment, the optimal processing order on each machine can be computed efficiently using Smith's rule (Smith, 1956). Combining this local search with a classical approximation algorithm yields a guarantee that improves with prediction accuracy while preserving the baseline worst-case approximation ratio.

F.1 Problem setting

We consider the unrelated-machines total weighted completion-time problem $R||\sum_j w_j C_j$.

Let $M = [m]$ be a set of machines and $J = [n]$ a set of jobs. Job j has processing time $p_{ij} > 0$ on machine i and weight $w_j > 0$. A feasible schedule assigns every job to one machine and specifies a

nonpreemptive processing order on each machine. Let C_j denote the completion time of job j. The objective is to minimize

$$\mathrm{WCT}(S) \coloneqq \sum_{j \in J} w_j C_j. \tag{48}$$

F.2 Prediction model

For simplicity, we consider a full assignment prediction $\widehat{\phi} : J \to M$, where $\widehat{\phi}(j) = i$ predicts that job j should be assigned to machine i.

F.3 Cost of a fixed assignment

For any fixed assignment $\psi : J \to M$, the optimal processing order on each machine is given by Smith's rule (Smith, 1956): jobs assigned to machine i are processed in nonincreasing order of w_j/p_{ij}.

For a set $S \subseteq J$, let $F_i(S)$ denote the weighted completion-time cost of processing exactly the jobs in S on machine i in Smith order. For distinct jobs j, k, define

$$q_i(j,k) \coloneqq \min\{p_{ij} w_k, p_{ik} w_j\}. \tag{49}$$

Then, expanding the completion times into individual processing costs and pairwise waiting costs gives

$$F_i(S) = \sum_{j \in S} w_j p_{ij} + \sum_{\{j,k\} \subseteq S} q_i(j,k). \tag{50}$$

For an assignment $\psi : J \to M$, let $J_i(\psi) \coloneqq \{j \in J \mid \psi(j) = i\}$ denote the set of jobs assigned to machine i. The minimum weighted completion-time cost among schedules consistent with ψ is

$$\mathrm{WCT}(\psi) = \sum_{i \in M} F_i(J_i(\psi)). \tag{51}$$

Thus, once an assignment is fixed, its optimal sequencing and objective value can be computed in polynomial time.

F.4 Algorithm

Fix an edit budget $K \geq 0$. For two assignments $\psi, \widehat{\phi} : J \to M$, define their assignment edit distance by

$$d_H(\psi, \widehat{\phi}) \coloneqq \left| \{j \in J \mid \psi(j) \neq \widehat{\phi}(j)\} \right|. \tag{52}$$

The K-edit neighborhood of the prediction is

$$\mathcal{E}_K(\widehat{\phi}) \coloneqq \left\{ \psi : J \to M \;\middle|\; d_H(\psi, \widehat{\phi}) \leq K \right\}. \tag{53}$$

The prediction-localized algorithm enumerates every assignment $\psi \in \mathcal{E}_K(\widehat{\phi})$. For each candidate assignment, it schedules the jobs on each machine according to Smith's rule and computes $\mathrm{WCT}(\psi)$. In parallel, it runs a polynomial-time α-approximation algorithm for $R\|\sum_j w_j C_j$. The algorithm returns the minimum-cost schedule among all enumerated candidates and the schedule returned by the α-approximation algorithm.

F.5 Prediction-error measure

Fix a reference assignment $\sigma : J \to M$. For each machine i, define $C_i(\psi, \sigma) \coloneqq J_i(\psi) \cap J_i(\sigma)$. Thus, $C_i(\psi, \sigma)$ contains the jobs assigned to machine i by both assignments.

Define the completion-time error of ψ relative to σ by

$$\Delta^{\mathrm{CT}}(\psi, \sigma) \coloneqq \sum_{i \in M} \left[F_i(J_i(\psi)) - F_i(C_i(\psi, \sigma)) \right]. \tag{54}$$

The residual error after at most K edits is

$$\Delta_K^{\mathrm{CT}}(\widehat{\phi}, \sigma) \coloneqq \min_{\psi \in \mathcal{E}_K(\widehat{\phi})} \Delta^{\mathrm{CT}}(\psi, \sigma). \tag{55}$$

Since $\mathcal{E}_K(\widehat{\phi}) \subseteq \mathcal{E}_{K+1}(\widehat{\phi})$, this quantity is monotone nonincreasing with respect to K.

F.6 Analysis

Theorem F.1 (Prediction-localized guarantee for weighted completion time)**.** *For any reference assignment σ, the prediction-localized algorithm returns a schedule S satisfying*

$$\mathrm{WCT}(S) \le \min\left\{\mathrm{WCT}(\sigma) + \Delta_K^{\mathrm{CT}}(\widehat{\phi}, \sigma),\ \alpha\,\mathrm{OPT}\right\}, \tag{56}$$

where α is the approximation ratio of the baseline algorithm.

In particular, for an optimal assignment $\sigma^\star$, the returned schedule satisfies

$$\mathrm{WCT}(S) \le \min\left\{\mathrm{OPT} + \Delta_K^{\mathrm{CT}}(\widehat{\phi}, \sigma^\star),\ \alpha\,\mathrm{OPT}\right\}. \tag{57}$$

The running time is

$$O\big((mn)^K\,\mathrm{poly}(m, n, \log P, \log W)\big). \tag{58}$$

where $P \coloneqq \max_{i \in M, j \in J} p_{ij}$ and $W \coloneqq \max_{j \in J} w_j$.

Proof. Let $\psi^\star \in \arg\min_{\psi \in \mathcal{E}_K(\widehat{\phi})} \Delta^{\mathrm{CT}}(\psi, \sigma)$. Since the algorithm enumerates all assignments in $\mathcal{E}_K(\widehat{\phi})$, the assignment $\psi^\star$ is one of its candidates.

For every machine i, we have

$$C_i(\psi^\star, \sigma) = J_i(\psi^\star) \cap J_i(\sigma) \subseteq J_i(\sigma). \tag{59}$$

The pairwise expression for F_i consists only of nonnegative terms. It follows that F_i is monotone with respect to set inclusion, and hence

$$F_i\big(C_i(\psi^\star, \sigma)\big) \le F_i\big(J_i(\sigma)\big). \tag{60}$$

Therefore,

$$\begin{aligned} \mathrm{WCT}(\psi^\star) &= \sum_{i \in M} F_i\big(J_i(\psi^\star)\big) \\ &= \sum_{i \in M} F_i\big(C_i(\psi^\star, \sigma)\big) + \Delta^{\mathrm{CT}}(\psi^\star, \sigma) \\ &\le \sum_{i \in M} F_i\big(J_i(\sigma)\big) + \Delta_K^{\mathrm{CT}}(\widehat{\phi}, \sigma) \\ &= \mathrm{WCT}(\sigma) + \Delta_K^{\mathrm{CT}}(\widehat{\phi}, \sigma). \end{aligned} \tag{61}$$

Since the algorithm returns the best schedule among the enumerated candidate and the baseline schedule, the same bound holds for the returned schedule S.

The number of assignments in the edit neighborhood is at most $\sum_{s=0}^{K} \binom{n}{s}(m-1)^s = O((mn)^K)$. For each candidate assignment, the Smith ordering and its objective value can be computed in time polynomial in $m, n, \log P, \log W$. The baseline algorithm also runs in polynomial time. Hence, the total running time is $O\big((mn)^K\,\mathrm{poly}(m, n, \log P, \log W)\big)$. □

Corollary F.2 (Consistency under small edit distance)**.** *If $d_H(\widehat{\phi}, \sigma^\star) \le K$ for some optimal assignment $\sigma^\star$, then the algorithm returns an optimal schedule.*

Proof. If $d_H(\widehat{\phi}, \sigma^\star) \le K$, then $\sigma^\star \in \mathcal{E}_K(\widehat{\phi})$. The algorithm therefore considers the optimal assignment $\sigma^\star$, and applying Smith's rule on each machine produces a schedule of cost OPT. □

Interpretation The theorem gives a prediction-dependent tradeoff between running time and schedule quality. Increasing the edit budget K enlarges the neighborhood $\mathcal{E}_K(\widehat{\phi})$, and therefore $\Delta_K^{\mathrm{CT}}(\widehat{\phi}, \sigma)$ is monotone nonincreasing in K, while the running time grows as $(mn)^K$.

If the prediction is within K edits of an optimal assignment, the optimal assignment is included in the search neighborhood and the algorithm returns an optimal schedule. As the residual prediction error increases, the guarantee degrades through Δ_K^{CT}, while the α-approximation baseline ensures that the resulting cost never exceeds the classical worst-case guarantee.

G Detailed experimental setup

Prediction generation. We generate synthetic predictions from an optimal schedule $\sigma^\star$. We first keep only the heavy assignments of $\sigma^\star$, namely assignments with processing time greater than $\varepsilon\mathrm{OPT}$, and set all other jobs to $\perp$. To introduce false negatives, we randomly select a ρ^- fraction of these heavy jobs and change their predicted assignment to $\perp$. To introduce false positives, we restrict attention to heavy jobs that have at least one alternative heavy machine $i \neq \sigma^\star(j)$ with $p_{ij} > \varepsilon\mathrm{OPT}$; we then randomly select a ρ^+ fraction of such jobs and reassign each of them to an alternative heavy machine. When varying the error level, we sweep ρ^- or ρ^+ over $\{0, 0.001, 0.002, \ldots, 1\}$, and keep the predictions that yield distinct values of $R^-_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$ or $R^+_{\mathrm{OPT}}(\widehat{\phi}, \sigma^\star)$.

Implementation details. All experiments are implemented in Python 3.13.11. We compute optimal schedules using the CP-SAT solver in OR-Tools (Perron & Didier). The LPs used in our algorithm and in the classical baseline are solved using the HiGHS solver (Huangfu & Hall, 2018) through SciPy (Virtanen et al., 2020). We use OR-Tools version 9.15.6755 and SciPy version 1.17.1.

Machine specifications. All experiments are run on a single machine with an Intel® Core™ Ultra 7 155H CPU (22 logical CPUs) and 16 GB RAM, running Arch Linux.

H Additional experimental results

H.1 Additional results on false negatives and false positives

In Section 6.2, we reported experiments for the setting with $m = 10$ and $n = 100$, where the processing times p_{ij} are generated independently and uniformly at random. In this appendix, we provide additional experiments under different choices of m and n, and under a non-uniform processing-time model.

H.1.1 Experimental settings

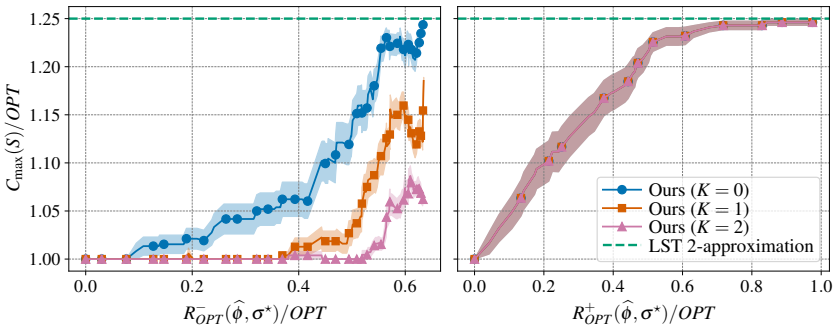
We consider the following three settings:

- $m = 10$ and $n = 50$, with processing times p_{ij} generated independently and uniformly at random as in Section 6.2.
- $m = 5$ and $n = 100$, with processing times p_{ij} generated independently and uniformly at random as in Section 6.2.
- $m = 10$ and $n = 100$, with processing times generated from a machine-correlated model.

For the machine-correlated setting, following prior work on $R\|C_{\max}$ (Fanjul-Peyro & Ruiz, 2011, 2010; Åblad et al., 2021), we sample $\alpha_i \sim \mathrm{Uniform}\{1, \ldots, 100\}$ and $\eta_{ij} \sim \mathrm{Uniform}\{1, \ldots, 20\}$ independently for all $i \in [m]$ and $j \in [n]$, and set

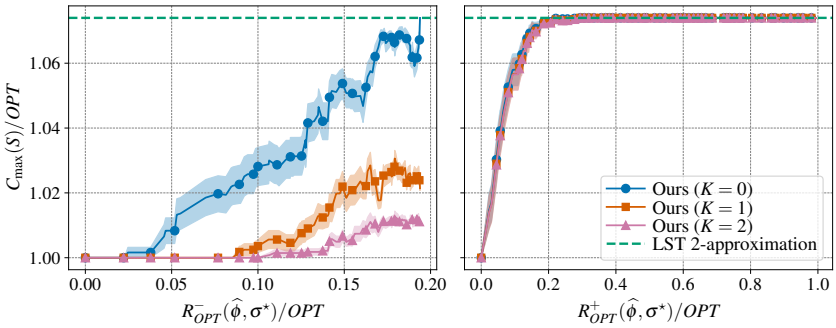
$$p_{ij} \coloneqq \alpha_i + \eta_{ij}. \tag{62}$$

This model induces correlations among processing times associated with the same machine through the shared machine-specific component α_i.



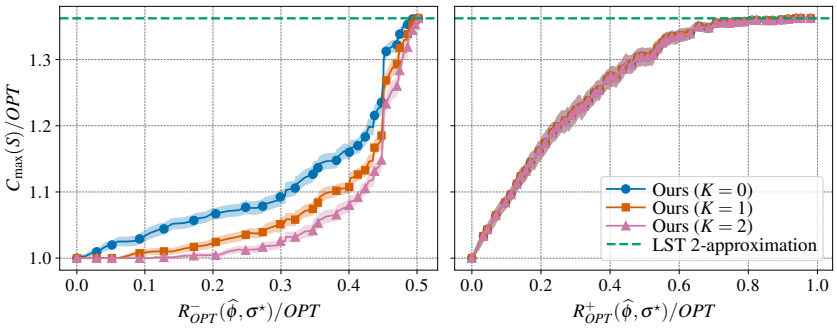
(a) $m = 10$, $n = 50$, i.i.d. uniform; false negatives

(b) $m = 10$, $n = 50$, i.i.d. uniform; false positives



(c) $m = 5$, $n = 100$, i.i.d. uniform; false negatives

(d) $m = 5$, $n = 100$, i.i.d. uniform; false positives



(e) $m = 10$, $n = 100$, machine-correlated; false negatives

(f) $m = 10$, $n = 100$, machine-correlated; false positives

Figure 2: Additional numerical results under alternative experimental settings. Panels (a) and (b) show the results for $m = 10$ and $n = 50$ with independently and uniformly generated processing times. Panels (c) and (d) show the results for $m = 5$ and $n = 100$ under the same generation model. Panels (e) and (f) show the results for $m = 10$ and $n = 100$ with machine-correlated processing times. In each row, the left panel varies the false-negative rate and the right panel varies the false-positive rate. The lines denote the mean, and the shaded regions denote the standard error, both computed over 30 independent runs with different random seeds.

H.1.2 Results and discussion

The results are shown in Fig. 2. As in Section 6.2, both classical $(1 + \varepsilon)$-type baselines were computationally infeasible in these experiments due to their excessive running time.

Across all instance classes, the results show the same qualitative behavior. For both false-negative and false-positive errors, our algorithm attains the optimal makespan when the prediction error is zero, and its makespan degrades smoothly as the error increases. The resulting schedules remain

no worse than the classical 2-approximation baseline. In the false-negative setting, increasing the search budget K further improves the makespan, reflecting the fact that additional local search can recover more missed heavy assignments. These trends are consistent with the results in Section 6.2 and provide additional empirical support for the effectiveness of our approach.

H.2 Additional running-time results with varying n

We also report additional running-time results in which the number of machines is fixed and the number of jobs is varied. Specifically, we fix $m = 5$ and vary $n \in \{5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15\}$. The results are shown in Fig. 3. Since the classical $(1+\varepsilon)$-type baselines become computationally expensive, we report their results only up to $n = 12$.

The results show that our algorithm remains substantially faster than the classical $(1+\varepsilon)$-type baselines even when m is fixed and n varies. This reflects the fact that the classical $(1+\varepsilon)$-type running times can still contain large factors such as $n^{m/\varepsilon}$ or 2^{mn}, even for constant m. In contrast, our algorithm runs almost as fast as the polynomial-time 2-approximation baseline throughout this range, which is consistent with our theoretical guarantee for constant search budget K.

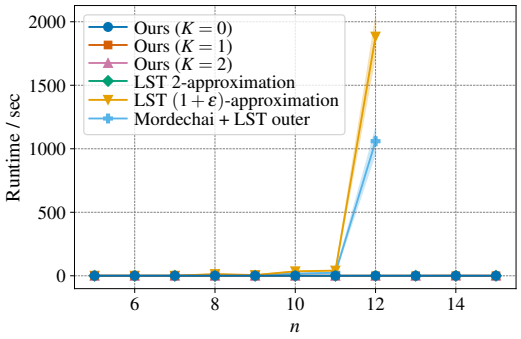


Figure 3: Additional running-time results with fixed $m = 5$ and varying n. The classical $(1+\varepsilon)$-type baselines are shown only up to $n = 12$ because of their large running time. The lines denote the mean, and the shaded regions denote the standard error, both computed over five independent runs with different random seeds.

I Limitations and broader impact

Limitations. Our results are developed for unrelated-machines makespan scheduling with predictions of heavy assignments. While this prediction model is well aligned with the heavy/short decomposition underlying classical approximation schemes, other scheduling settings may require different prediction objects. Our experiments use synthetic instances and controlled prediction errors, and evaluating the method with learned predictions on application-specific scheduling data remains an important direction for future work.

Broader Impact. The potential positive impact of this work is to provide a principled way to combine increasingly accurate machine-learning predictions with classical approximation guarantees for scheduling problems, a setting not covered by the framework of Antoniadis et al. (2025). As prediction quality continues to improve, such approaches enable scheduling algorithms to substantially improve practical performance beyond classical worst-case baselines, while retaining robustness when predictions are inaccurate.

We do not identify direct negative societal impacts specific to this theoretical work, since it does not introduce a deployed system, collect data, or make decisions about individuals. As with any scheduling system, however, deployment in application-specific settings should validate the quality of predictions and the consequences of scheduling decisions.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main theoretical guarantees, running-time bounds, and empirical claims, which are proved in Section 5 and evaluated in Section 6. The scope is limited to learning-augmented approximation for $R||C_{\max}$ with predictions of heavy assignments, as stated in the problem setup.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations explicitly in Appendix I, including the scope of the theoretical model and the use of synthetic experiments.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: All theoretical results state their assumptions explicitly, including the prediction model, threshold parameters, and search budget. Proof sketches are provided in the main text for the central guarantees, and complete proofs are deferred to the appendix and cross-referenced from the corresponding theorems and corollaries.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The experimental setup, instance generation procedures, prediction-generation procedure, parameter choices, baselines, and software used for optimization are described in Section 6 and Appendix G. The experiments use synthetic instances, and the exact code for reproducing all experiments is provided in the supplementary material.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide anonymized code in the supplementary material, including exact instructions for installing all required dependencies. A single command runs all experiments end-to-end, including generating synthetic instances and predictions, running the proposed scheduling algorithm and baselines, and producing the figures used in the paper. To preserve anonymity during review, the code is included in the supplementary material; we plan to include a public GitHub link after the review period.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper specifies the experimental settings needed to interpret the results, including the synthetic instance-generation procedures, prediction-generation procedure, parameter choices such as ε and K, baselines, number of random seeds, and implementation details. Additional details are provided in the appendix and the supplementary code.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The experimental figures report the mean over 30 independent runs with different random seeds, and the shaded regions denote the standard error. The captions specify how the error bars are computed and what source of variability they capture.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The appendix reports the machine specifications used for the experiments, including the CPU, memory, operating system, and software versions. The running-time experiment reports the execution time of the algorithms, and the supplementary code provides scripts to reproduce the reported experiments on comparable hardware.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research is theoretical and uses only synthetic experimental instances, with no human subjects, personal data, or sensitive datasets. We have reviewed the NeurIPS Code of Ethics and are not aware of any aspect of the work that deviates from it.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The broader impacts are discussed explicitly in Appendix I. The paper notes the potential positive impact of improving scheduling algorithms with accurate predictions, while also clarifying that the work is primarily theoretical, uses only synthetic experiments, and does not introduce a deployed system or new data collection.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not release pre-trained models, generative models, scraped datasets, or other assets with a high risk of misuse. The released code only reproduces theoretical algorithms and experiments on synthetic scheduling instances.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper uses standard open-source software packages, including OR-Tools and SciPy, which are cited with version information in the experimental details. The supplementary code lists the required dependencies, installation instructions, and relevant license information. No external datasets, pre-trained models, or scraped assets are used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The only new asset released with the paper is anonymized experimental code. It is documented in the supplementary material with dependency-installation instructions and commands for reproducing the full experimental pipeline. The paper does not introduce new datasets or models, and no human-subject consent is applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, human-subject experiments, or data collection from human participants. All experiments use synthetic scheduling instances.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, human-subject experiments, or data collection from human participants. Therefore, IRB approval or equivalent review is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs. We used an LLM only for language assistance, such as translation and English grammar/style editing. All scientific content, including the research ideas, methods, theoretical analysis, experimental design, implementations, results, analyses, and conclusions, was developed, verified, and approved by the authors.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.