NEAR-OPTIMAL LEARNING-AUGMENTED ONLINE SCHEDULING WITH AND WITHOUT REJECTION

Anonymous authors
Paper under double-blind review

ABSTRACT

We study learning-augmented online scheduling on m identical parallel machines, both with and without rejection. Each arriving job reveals its processing time and a predicted machine assignment, which may instead recommend rejection when allowed. Decisions are irrevocable, and the objective is makespan plus any rejection penalties. Our algorithms use two load caps in the no-rejection setting and cost comparisons with a prediction-free reference scheduler when rejection is allowed. Both are $(1 + \varepsilon)$-consistent, degrade smoothly with prediction error, and retain worst-case guarantees under arbitrary predictions. We further establish robustness lower bounds for all deterministic online algorithms satisfying the same consistency requirement, showing that both algorithms attain optimal robustness for sufficiently small ε and remain near-optimal over wider parameter ranges. We conclude our work with an empirical analysis of our method.

1 INTRODUCTION

Online scheduling requires assigning jobs as they arrive, without knowing future arrivals. A fundamental problem in this setting is makespan minimization on m identical parallel machines, where the objective is to minimize the maximum machine load (Graham, 1966). We study this problem both with and without job rejection, with processing times revealed on arrival and decisions made immediately and irrevocably. The model with rejection was introduced in the seminal work of Bartal et al. (1996): jobs may be rejected at a penalty, and the objective is makespan plus total rejection penalties. This captures real-world settings such as manufacturing, where accepting all orders may overload production, while rejecting them sacrifices revenue (Engels et al., 1998; Shabtay & Gasper, 2012). In both settings, an initially promising decision may become costly as further jobs arrive.

In recent years, algorithms with predictions, also known as *learning-augmented algorithms*, have attracted growing interest. Motivated by advances in machine learning, these algorithms use predictions to guide online decisions while retaining guarantees when the predictions are inaccurate (Purohit et al., 2018; Lykouris & Vassilvitskii, 2018). Three desirable properties are *consistency*, performance close to the offline optimum under perfect predictions; *robustness*, a worst-case guarantee under arbitrary predictions; and *smoothness*, gradual degradation of the guarantee as prediction error increases (Mitzenmacher & Vassilvitskii, 2022; Azar et al., 2023).

In this work, we study learning-augmented scheduling on identical parallel machines, both with and without rejection, using assignment predictions. Each arriving job comes with a suggested machine, or a suggestion to reject it when rejection is allowed, which the algorithm may follow or override. Without rejection, we follow the predicted assignment whenever it satisfies two load caps, one controlling the worst-case makespan and the other limiting deviation from the predicted loads. With rejection, load control alone is insufficient because rejecting jobs incurs penalties. We instead compare costs to choose between a prediction-based action and one proposed by a prediction-free reference scheduler. A direct switching approach is nontrivial because previously assigned jobs cannot be moved, so selecting only a subset of reference actions does not reproduce the reference schedule. Our selection rule limits the cost of departing from the predicted solution to preserve consistency, while leveraging the reference schedule for robustness.

Both algorithms are $(1 + \varepsilon)$-consistent, allowing them to approach the offline optimum arbitrarily closely under perfect predictions. This goes beyond what deterministic online algorithms can guarantee without predictions. They also admit smooth performance bounds in terms of the excess

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

cost of the predicted solution over the offline optimum. We further ask how close their robustness is to optimal under the same consistency requirement, deriving lower bounds for all deterministic online algorithms satisfying it. Without rejection, our algorithm attains optimal robustness when $\varepsilon \leq 1/m$. Throughout $0 < \varepsilon < 1/2$, its robustness is within a factor of $(1+\varepsilon/2)^2$ of the optimum, with an additive gap of at most $1+\varepsilon/4$. The multiplicative factor tends to one as $\varepsilon \to 0$, uniformly in the number of machines. With rejection, our algorithm attains optimal robustness when $\varepsilon \leq g_m(m) = \sum_{s=1}^{m-1} m^{-s}$. Using RTP of Bartal et al. (1996) as the reference schedule, its robustness is within a uniform factor of $\frac{3}{2}(1+\varphi) \approx 3.927$ of the optimum throughout $0 < \varepsilon < 1$, where $\varphi = (1+\sqrt{5})/2$. Both algorithms achieve these guarantees using only $O(m)$ time per job, while our lower bounds apply even to algorithms with unrestricted computation. Numerical experiments illustrate the consistency–robustness trade-off and smooth degradation with prediction error, consistent with our theory. Our contributions are summarized as follows:

- We develop learning-augmented algorithms for scheduling on identical parallel machines, both with and without rejection. Our algorithms use predicted machine assignments to guide their online decisions. Predictions may also recommend rejection when allowed.
- Our guarantee exhibits the desired learning-augmented behavior. In both settings, the algorithms are $(1+\varepsilon)$-consistent, retain worst-case guarantees under arbitrary predictions, and admit bounds that degrade smoothly with prediction error (Sections 3.4 and 4.3).
- We prove general lower bounds on robustness for all $(1+\varepsilon)$-consistent deterministic online algorithms in both settings. These bounds establish that our algorithms achieve optimal robustness for sufficiently small ε and remain near-optimal throughout the wider parameter ranges (Sections 3.5 and 4.4).

2 Related Work

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

Classical online scheduling. Online makespan minimization on identical machines dates back to Graham (1966), whose List Scheduling algorithm is $(2-1/m)$-competitive. Fleischer & Wahl (2000) later improved the asymptotic competitive ratio to below 1.9201. For makespan plus rejection penalties, Bartal et al. (1996) established a tight deterministic competitive ratio of $(3+\sqrt{5})/2$ when the number of machines is unrestricted. More recently, Yang & Tan (2026) obtained improved guarantees for small numbers of machines. However, for deterministic online algorithms, the best known asymptotic guarantees in these two settings have remained unchanged for several decades. Our algorithms improve on these guarantees by exploiting accurate assignment predictions.

Scheduling with predictions. Learning-augmented scheduling has been studied with several forms of predictions. Processing-time predictions guide non-clairvoyant algorithms for completion-time objectives (Purohit et al., 2018) and makespan minimization (Bampis et al., 2023). For load balancing, predicted machine weights guide fractional assignments and online rounding (Lattanzi et al., 2020; Li & Xian, 2021). Closer to our model, Guyot & Lassota (2025) study predicted machine assignments for online makespan minimization with class constraints, obtaining guarantees based on the number of prediction errors. Predicted assignments also appear alongside job-order predictions in non-clairvoyant scheduling (Lindermayr & Megow, 2022). We consider immediate, irrevocable assignments without class constraints, with processing times revealed on arrival.

Consistency–robustness trade-offs. Our focus is the robustness achievable under a prescribed consistency requirement, both with and without rejection. Wei & Zhang (2020) establish consistency–robustness lower bounds for ski rental and non-clairvoyant scheduling with processing-time predictions. We derive lower bounds for all deterministic algorithms in our assignment-prediction model, establishing optimality for sufficiently small ε and near-optimality over wider ranges.

3 Near-Optimal Scheduling without Rejection

3.1 Problem Setup

We consider online makespan minimization on $m \geq 2$ identical parallel machines, denoted by $[m] \coloneqq \{1, \ldots, m\}$. Jobs arrive sequentially, and the processing time $p_j \geq 0$ of job j is revealed upon its arrival. The algorithm must immediately and irrevocably assign each job to a machine, without rejection, splitting, or migration. Neither future jobs nor the total number of jobs is known

Algorithm 1 Prediction-First Dual-Cap

Input: Number of machines m and parameter $\varepsilon > 0$

$\ell_i \leftarrow 0, S_i \leftarrow 0$ for all $i \in [m]$
$P \leftarrow 0, p_{\max} \leftarrow 0, H \leftarrow 0$
for each arriving job t with (p_t, h_t) **do**
  $P \leftarrow P + p_t, p_{\max} \leftarrow \max\{p_{\max}, p_t\}$
  $S_{h_t} \leftarrow S_{h_t} + p_t, H \leftarrow \max\{H, S_{h_t}\}$
  $B \leftarrow \max\{p_{\max}, P/m\}$
  $\mathcal{A} \leftarrow \{i \in [m] : \ell_i + p_t \le \beta B,\ \ell_i + p_t \le S_i + \varepsilon H\}$
  if $h_t \in \mathcal{A}$ **then**
    $q \leftarrow h_t$
  else
    $q \leftarrow \arg\min_{i \in \mathcal{A}} \ell_i$
  Assign the job to machine q and set $\ell_q \leftarrow \ell_q + p_t$

in advance. For assignments $a_j \in [m]$, the objective is to minimize $C_{\max} \coloneqq \max_{i \in [m]} \sum_{j:a_j=i} p_j$. We write $\mathrm{OPT}(I)$ for the minimum makespan achievable offline with full knowledge of I.

Upon arrival, each job j also reveals a predicted machine $h_j \in [m]$, which the algorithm may follow or override. Predictions are *perfect* if assigning every job to its predicted machine yields a makespan of $\mathrm{OPT}(I)$ on the full input. The predicted assignment need not be optimal on every prefix.

3.2 The Prediction-First Dual-Cap Algorithm

Our algorithm follows the predicted assignment whenever it satisfies two load caps: one relative to a lower bound on the optimal makespan, and the other relative to the predicted loads. For $\varepsilon > 0$, set the cap coefficient before processing any jobs as

$$\beta \coloneqq \min\left\{m, \frac{(2 - 1/m + \varepsilon)^2}{4\varepsilon}\right\}. \tag{1}$$

The parameter ε controls the permitted deviation from the predicted loads, while β controls the worst-case guarantee. The analysis in Section 3.3 shows that this choice of β ensures that every arriving job can be assigned while respecting both load caps.

Let $\ell_i(t)$ denote the actual load of machine i after the first t jobs. Alongside the actual schedule, we maintain a virtual schedule that assigns every job to its predicted machine, including jobs whose predictions are overridden by the algorithm. Its loads and makespan are $S_i(t) \coloneqq \sum_{j \le t : h_j = i} p_j$ and $H_t \coloneqq \max_{i \in [m]} S_i(t)$, respectively. These virtual loads provide a reference for controlling how far the actual schedule may deviate from the predicted assignment.

We also maintain the total processing volume $P_t \coloneqq \sum_{j \le t} p_j$ and largest processing time $p_{\max}(t) \coloneqq \max_{j \le t} p_j$, and define $B_t \coloneqq \max\{p_{\max}(t), P_t/m\}$. For the prefix I_t, we have $B_t \le \mathrm{OPT}(I_t)$, so B_t provides an online lower bound without computing an optimal schedule.

When job t arrives, we first update S_{h_t}, H_t, and B_t to include the current job, while leaving the actual loads unchanged. The admissible set $\mathcal{A}_t$ consists of machines $i \in [m]$ satisfying both

$$\ell_i(t-1) + p_t \le \beta B_t, \qquad \ell_i(t-1) + p_t \le S_i(t) + \varepsilon H_t. \tag{2}$$

The first inequality is the *robustness cap*, and the second is the *consistency cap*. If $h_t \in \mathcal{A}_t$, we assign the job to its predicted machine. Otherwise, we choose a machine of minimum actual load in $\mathcal{A}_t$, breaking ties by machine index. Algorithm 1 gives the procedure; state variables are updated in place, with time indices omitted. The algorithm uses $O(m)$ time per job and $O(m)$ memory.

3.3 Feasibility of Dual-Cap

We show that every arriving job can be assigned to a machine without violating either cap, ensuring that the algorithm is well-defined on every input. The proof uses the fact that the actual and virtual schedules account for the same total processing volume.

Theorem 1 (Feasibility)**.** *For every $\varepsilon > 0$, the choice of β in Eq.* (1) *ensures that $\mathcal{A}_t$ is nonempty at every arrival. Consequently, Algorithm 1 maintains*

$$\ell_i(t) \le \beta B_t, \qquad \ell_i(t) \le S_i(t) + \varepsilon H_t \quad \textit{for all } i \in [m] \textit{ and } t. \tag{3}$$

Proof sketch. We prove feasibility and both load bounds by induction. If $\beta = m$, following every prediction satisfies both caps because $S_i(t) \le P_t \le mB_t$. Otherwise, let $c_m := 2 - 1/m$, so $\beta = (c_m + \varepsilon)^2/(4\varepsilon) \ge c_m$. Assume both bounds hold after $t-1$ jobs. The predicted machine satisfies the consistency cap because its virtual load increases by p_t and H_t does not decrease. A least-loaded machine satisfies the robustness cap because its load after assignment is at most $c_m B_t$.

Suppose no machine satisfies both caps: each violates the robustness cap or, otherwise, the consistency cap. Since $\sum_i(\ell_i(t-1) - S_i(t)) = -p_t$ and $\sum_i \ell_i(t-1) = P_t - p_t \le mB_t - p_t$, summing the bounds over these two groups, eliminating the first group's size, and using $p_t \le B_t$ yields

$$0 > \beta B_t^2 - (c_m + \varepsilon) H_t B_t + \varepsilon H_t^2 = \varepsilon \left(H_t - \frac{c_m + \varepsilon}{2\varepsilon} B_t \right)^2 \ge 0, \tag{4}$$

a contradiction. Thus, an admissible machine exists and preserves both bounds, completing the induction. The full proof is given in Appendix B.1. □

3.4 Performance Guarantees

For a deterministic online algorithm A, let $C^A_{\max}(I)$ denote its makespan on input I and define

$$\mathrm{Cons}(A) := \sup_{I:\, H_n = \mathrm{OPT}(I) > 0} \frac{C^A_{\max}(I)}{\mathrm{OPT}(I)}, \quad \mathrm{Rob}(A) := \sup_{I:\, \mathrm{OPT}(I) > 0} \frac{C^A_{\max}(I)}{\mathrm{OPT}(I)}. \tag{5}$$

The suprema range over all finite job sequences and their predictions, with n denoting the input length. We call A α-consistent and ρ-robust if these quantities are at most α and ρ, respectively. These are purely multiplicative guarantees, with no additive constant.

Theorem 2 (Consistency and robustness)**.** *For every $\varepsilon > 0$ and β defined in Eq.* (1)*, Algorithm 1 satisfies*

$$C^{\mathrm{DualCap}}_{\max}(I_t) \le \min\{(1+\varepsilon)H_t, \beta B_t\} \tag{6}$$

on every prefix, and its consistency and robustness satisfy

$$\mathrm{Cons}(\mathrm{DualCap}) \le 1 + \varepsilon, \qquad \mathrm{Rob}(\mathrm{DualCap}) = \beta. \tag{7}$$

Proof sketch. The two load caps give Eq. (6). Since $B_n \le \mathrm{OPT}(I)$ and $H_n = \mathrm{OPT}(I)$ under perfect predictions, this yields the consistency and robustness upper bounds. For tightness, a unit job followed by $m-1$ equal-sized jobs of total size $\beta - 1$, all predicted to the same machine, yields makespan β while $\mathrm{OPT}(I) = 1$. The full proof is given in Appendix B.2. □

We next show how the performance guarantee varies with prediction accuracy. To this end, define the prediction error as $E := H_n - \mathrm{OPT}(I) \ge 0$. This error measures the excess makespan of the predicted assignment over the optimum. It is therefore a solution-quality error rather than a job-wise prediction error; in particular, $E = 0$ whenever the predicted assignment is optimal, even if it differs from another optimal assignment.

Corollary 3 (Smoothness)**.** *For every input with $\mathrm{OPT}(I) > 0$ and arbitrary predictions, Algorithm 1 satisfies*

$$\frac{C^{\mathrm{DualCap}}_{\max}(I)}{\mathrm{OPT}(I)} \le \min\left\{ \beta, (1+\varepsilon)\left(1 + \frac{E}{\mathrm{OPT}(I)}\right) \right\}. \tag{8}$$

Proof. Substitute $H_n = \mathrm{OPT}(I) + E$ into Eq. (6) at $t = n$ and use $B_n \le \mathrm{OPT}(I)$. □

Interpretation. Theorem 2 and Corollary 3 show that our algorithm exhibits the desired behavior of a learning-augmented algorithm. With perfect predictions, it approaches the offline optimum arbitrarily closely. As the prediction error $E/\mathrm{OPT}(I)$ increases, the performance guarantee degrades gradually, while the competitive ratio remains bounded by β regardless of how inaccurate the predictions are. Neither the error nor $\mathrm{OPT}(I)$ needs to be known by the algorithm.

3.5 Lower Bounds and Near-Optimality

We now compare Dual-Cap with the best robustness achievable under the same consistency. Define

$$R^*(m, \varepsilon) := \inf_{A:\, \mathrm{Cons}(A) \le 1+\varepsilon} \mathrm{Rob}(A), \tag{9}$$

where the infimum ranges over all deterministic online algorithms in our model, without restrictions on their computational complexity or assignment rules.

Although an algorithm may override every prediction, its consistency guarantee must hold on every input with perfect predictions. We exploit this requirement by extending a given prefix to such an input, thereby constraining the assignments the algorithm can make on the prefix.

Lemma 4 (Consistency limits load spreading). *Consider a prefix of total processing volume $V > 0$ in which every job is predicted to the same machine. Any deterministic $(1+\varepsilon)$-consistent algorithm has at most one machine with load greater than εV after processing this prefix.*

Proof. If $\varepsilon \geq 1$, the claim is immediate because no machine has load greater than the total volume V, so assume $0 < \varepsilon < 1$. Extend this prefix by appending $m-1$ jobs of size V, each predicted to a different machine among the remaining $m-1$ machines. The predicted load on every machine is now V, and the total processing volume is mV, so the predictions are perfect and $\mathrm{OPT} = V$. To achieve makespan at most $(1+\varepsilon)\mathrm{OPT} = (1+\varepsilon)V$ on the full extended input, no two appended jobs can share a machine, since $\varepsilon < 1$. Moreover, no appended job can be assigned to a machine whose load already exceeds εV before the appended jobs arrive. Accommodating the $m-1$ appended jobs therefore requires at least $m-1$ machines whose loads after the prefix are at most εV, leaving at most one machine with load greater than εV. Since the input length is unknown, the algorithm makes the same prefix decisions whether or not these jobs are appended. □

Interpretation. The offline optimum can choose its assignments knowing the entire input, whereas an online algorithm cannot tell whether the input will end at the current prefix or continue with the appended jobs. Consequently, the algorithm must leave enough machines with low loads to satisfy consistency on the possible extension, even if the input actually ends at the prefix. This restriction yields the following robustness lower bound for any deterministic $(1+\varepsilon)$-consistent algorithm.

Theorem 5 (General robustness lower bound). *For $0 < \varepsilon < 1/2$, the optimal robustness under $(1+\varepsilon)$-consistency satisfies*

$$R^*(m,\varepsilon) \geq \min\{m, 1/\varepsilon\}. \tag{10}$$

Proof sketch. Predict every job to the same machine and use Lemma 4 to force all jobs onto one machine. For $\varepsilon < 1/m$, m unit jobs give ratio m. Otherwise, suitably chosen equal-sized jobs followed by a unit job give a ratio arbitrarily close to $1/\varepsilon$, with $\mathrm{OPT} = 1$. The full construction and proof are given in Appendix B.3. □

Combining this lower bound with Theorem 2 gives $\min\{m, 1/\varepsilon\} \leq R^*(m,\varepsilon) \leq \mathrm{Rob}(\mathrm{DualCap}) = \beta$, yielding the following near-optimality result.

Theorem 6 (Optimality and near-optimality). *For $0 < \varepsilon < 1/2$, Algorithm 1 with the coefficient in Eq.* (1) *satisfies*

$$1 \leq \frac{\mathrm{Rob}(\mathrm{DualCap})}{R^*(m,\varepsilon)} \leq \left(1+\frac{\varepsilon}{2}\right)^2, \quad 0 \leq \mathrm{Rob}(\mathrm{DualCap}) - R^*(m,\varepsilon) \leq 1+\frac{\varepsilon}{4}. \tag{11}$$

Moreover, if $\varepsilon \leq 1/m$, it is optimal under $1+\varepsilon$ consistency, with $\mathrm{Rob}(\mathrm{DualCap}) = R^*(m,\varepsilon) = m$.

Proof sketch. The result follows immediately by comparing the exact robustness β from Theorem 2 with the lower bound in Theorem 5. The full proof is given in Appendix B.4. □

Interpretation. When the consistency requirement is sufficiently close to 1, namely $\varepsilon \leq 1/m$, the algorithm attains exactly optimal robustness. Furthermore, its robustness is within a factor of $(1+\varepsilon/2)^2$ of the optimum throughout $0 < \varepsilon < 1/2$. Its additive gap is at most $1+\varepsilon/4$, independently of the number of machines. These comparisons hold against all deterministic online algorithms satisfying the same consistency requirement, without any restriction on their computational complexity. Our algorithm achieves this near-optimal robustness using $O(m)$ time per job and $O(m)$ memory.

Refined Guarantees. We can further sharpen the robustness guarantee in two parameter ranges by choosing a smaller β without changing the assignment rule. For $m \geq 3$ and $1/m \leq \varepsilon \leq 1/(m-1)$ with $\varepsilon < 1/2$, we show that choosing $\beta := 2m - 1 - m(m-1)\varepsilon$ keeps $\mathcal{A}_t$ nonempty at every arrival, so the algorithm remains well-defined. This choice gives a multiplicative factor of at most $1+\varepsilon^2/4$ and an additive gap of at most $\varepsilon/4$ relative to $R^*(m,\varepsilon)$. Compared with Theorem 6, this removes the first-order term in ε from the multiplicative bound and the constant term 1 from the additive bound, yielding a second-order multiplicative gap and a first-order additive gap in ε.

For $m \geq 5$ and $1/(m-1) < \varepsilon \leq 2/(m+2)$, we use load invariants that the algorithm maintains automatically to establish feasibility for a smaller β. This gives a multiplicative factor of at most $1+\varepsilon/6+\varepsilon^2/(3(1-\varepsilon))$ and an additive gap of at most $1/6+\varepsilon/(3(1-\varepsilon))$ relative to $R^*(m,\varepsilon)$. In this range, both the coefficient of the first-order term in the multiplicative bound and the constant term in the additive bound improve from 1 in Theorem 6 to $1/6$.

Both refinements preserve $(1+\varepsilon)$-consistency and require $O(m)$ time per job and $O(m)$ memory. The full details and proofs are given in Appendix C.

4 NEAR-OPTIMAL SCHEDULING WITH REJECTION

4.1 PROBLEM SETUP

We extend the setting in Section 3.1 by allowing job rejection. Each job j also has a rejection penalty $r_j \geq 0$, revealed upon arrival with its processing time p_j. The algorithm must immediately and irrevocably either assign the job to a machine or reject it and pay r_j. We encode this decision by $a_j \in \{0\} \cup [m]$, where $a_j = 0$ denotes rejection. The objective is to minimize the makespan of accepted jobs plus the total rejection penalty: $C(I) \coloneqq \max_{i\in[m]} \sum_{j:a_j=i} p_j + \sum_{j:a_j=0} r_j$. Here, $\mathrm{OPT}(I)$ denotes the minimum offline cost when rejection is allowed.

The prediction for job j is now $h_j \in \{0\} \cup [m]$, where $h_j = 0$ predicts rejection and $h_j = i \in [m]$ predicts acceptance on machine i. As before, predictions are revealed upon arrival and may be followed or overridden. They are *perfect* if following all predicted decisions achieves total cost $\mathrm{OPT}(I)$ on the full input. Consistency and robustness are defined as in Section 3.4, using total cost in place of makespan and this notion of perfect predictions.

4.2 THE COST-BALANCED SCHEDULING ALGORITHM

In Algorithm 1, the robustness cap controls worst-case performance, while the consistency cap preserves consistency. With rejection, however, keeping machine loads small by rejecting jobs can incur large penalties, so load caps alone do not control the objective. For each job, we therefore choose between a prediction-based action and one proposed by a prediction-free reference scheduler.

For $0 < \varepsilon < 1$, choose $K > 1$ as the unique solution of $g_m(K) \coloneqq \sum_{s=1}^{m-1} K^{-s} = \varepsilon$. Here, $g_m(K)$ is a finite geometric sum with common ratio $1/K$. The analysis below relates this choice to the consistency–robustness tradeoff.

Virtual reference schedule. To maintain robustness when predictions are inaccurate, we obtain alternative actions from a virtual reference schedule. Let $\mathcal{B}$ be a deterministic, prediction-free, ρ_m-competitive online scheduler for the same problem that rejects only if $r_t \leq p_t$ and accepts only if $p_t \leq m r_t$. One such reference is the Reject-Total-Penalty algorithm (RTP) of Bartal et al. (1996) with parameter $\alpha = (\sqrt{5}-1)/2$. We run $\mathcal{B}$ on every job using its own virtual state, independently of our actual decisions, and denote its current action by $b_t \in \{0\} \cup [m]$.

Candidate actions. The two candidates for job t are the reference action b_t and a prediction-based action defined as follows. For $h_t \neq 0$, the prediction-based action assigns the job to h_t. For $h_t = 0$, it rejects if $r_t \leq p_t$ and otherwise accepts on a machine of minimum actual load.

Maintained costs. Let $\ell_i(t)$ be the actual loads. We group jobs by the selected rule, even when both candidate actions coincide. Among jobs for which the prediction-based action is selected, let $L_i^{\mathrm{P}}(t)$ be the acceptance-predicted load on machine i and $M_t^{\mathrm{P}} \coloneqq \max_i L_i^{\mathrm{P}}(t)$. Let A_t^{P} be the sum of $w_j \coloneqq \min\{p_j, r_j\}$ over rejection-predicted jobs in this group. For jobs using the reference action, let $L_i^{\mathrm{B}}(t)$ and R_t^{B} be their machine loads and total rejection penalty, with cost $Q_t^{\mathrm{B}} \coloneqq \max_{i\in[m]} L_i^{\mathrm{B}}(t) + R_t^{\mathrm{B}}$. The actual cost is at most $M_t^{\mathrm{P}} + A_t^{\mathrm{P}} + Q_t^{\mathrm{B}}$. For a rejection-predicted job handled by the prediction rule, its actual contribution is at most w_j: it is $r_j = w_j$ when rejected and at most $p_j = w_j$ when accepted. The values $q_i(t)$ record cumulative increases in Q^{B}, starting from zero. Each increase is attributed entirely to q_{h_t} for an acceptance prediction, and equally to all q_i for a rejection prediction. Thus, $q_i(t) \geq 0$ and $\sum_i q_i(t) = Q_t^{\mathrm{B}}$.

Choosing the action. To preserve consistency, we select a reference action only when the accumulated reference cost Q^{B} and its increment fit a budget determined by the prediction-side state. The parameter K weights these costs, with larger values making the reference action harder to select. Before choosing an action, we compute c_t, the increase in Q^{B} that would result from selecting the reference action b_t. For an acceptance prediction $h_t \in [m]$, select the reference action if and only if

$$(K-1)Q_{t-1}^{\mathrm{B}} + K c_t \leq L_{h_t}^{\mathrm{P}}(t-1) + A_{t-1}^{\mathrm{P}} + p_t. \tag{12}$$

For a rejection prediction $h_t = 0$, select the reference action if and only if

$$(K-1)Q_{t-1}^{\mathrm{B}} + K c_t + \max_i q_i(t-1) \leq M_{t-1}^{\mathrm{P}} + A_{t-1}^{\mathrm{P}} + w_t. \tag{13}$$

In either case, execute the prediction-based action if the condition fails. For a rejection prediction, selecting the reference action distributes its cost increment across all q_i. The $\max_i q_i(t-1)$ term therefore accounts for the largest accumulated cost among the q_i when selecting a reference action.

Algorithm 2 Cost-Balanced Scheduling (CBS) with Rejection

Input: Number of machines m, parameter $0 < \varepsilon < 1$, and a reference scheduler $\mathcal{B}$ satisfying the stated conditions

Initialize a virtual instance of $\mathcal{B}$ and set all state variables to zero
for each arriving job t with (p_t, r_t, h_t) **do**
  Process (p_t, r_t) in $\mathcal{B}$ to obtain b_t
  If $b_t = 0$, set $c_t \leftarrow r_t$; else $c_t \leftarrow \max\{0, L^{\mathrm{B}}_{b_t} + p_t - \max_i L^{\mathrm{B}}_i\}$
  if $h_t \in [m]$ **and** $(K-1)Q^{\mathrm{B}} + Kc_t \le L^{\mathrm{P}}_{h_t} + A^{\mathrm{P}} + p_t$ **then**
    $a_t \leftarrow b_t$, $Q^{\mathrm{B}} \leftarrow Q^{\mathrm{B}} + c_t$
    If $b_t = 0$, set $R^{\mathrm{B}} \leftarrow R^{\mathrm{B}} + r_t$; else $L^{\mathrm{B}}_{b_t} \leftarrow L^{\mathrm{B}}_{b_t} + p_t$
    $q_{h_t} \leftarrow q_{h_t} + c_t$
  else if $h_t = 0$ **and** $(K-1)Q^{\mathrm{B}} + Kc_t + \max_i q_i \le M^{\mathrm{P}} + A^{\mathrm{P}} + w_t$ **then**
    $a_t \leftarrow b_t$, $Q^{\mathrm{B}} \leftarrow Q^{\mathrm{B}} + c_t$
    If $b_t = 0$, set $R^{\mathrm{B}} \leftarrow R^{\mathrm{B}} + r_t$; else $L^{\mathrm{B}}_{b_t} \leftarrow L^{\mathrm{B}}_{b_t} + p_t$
    $q_i \leftarrow q_i + c_t/m$ for all $i \in [m]$
  else if $h_t \in [m]$ **then**
    $a_t \leftarrow h_t$, $L^{\mathrm{P}}_{h_t} \leftarrow L^{\mathrm{P}}_{h_t} + p_t$, $M^{\mathrm{P}} \leftarrow \max\{M^{\mathrm{P}}, L^{\mathrm{P}}_{h_t}\}$
  else
    $A^{\mathrm{P}} \leftarrow A^{\mathrm{P}} + w_t$, $a_t \leftarrow 0$
    If $r_t > p_t$, set $a_t \leftarrow \arg\min_{i \in [m]} \ell_i$
  Execute a_t: pay r_t if $a_t = 0$; otherwise assign to a_t and increase ℓ_{a_t} by p_t

When the reference action accepts a job, it uses the same machine b_t as the virtual reference. Equality favors the reference action, and minimum-load ties are broken by machine index. Algorithm 2 gives the procedure with state updated in place and time indices omitted. It uses $O(m)$ additional time per job and $O(m)$ additional memory beyond the reference scheduler. With RTP as the reference, the total time per job and memory are both $O(m)$. The same algorithm can be implemented without computing the root K; see Appendix E.1.

4.3 Performance Guarantees

Let CBS denote Algorithm 2, and let $\widehat{C}_t$ be the cost of following every prediction on the prefix I_t: $\widehat{C}_t := \max_{i \in [m]} \sum_{j \le t : h_j = i} p_j + \sum_{j \le t : h_j = 0} r_j$. This quantity is used only in the analysis.

Theorem 7 (Consistency and robustness)**.** *Let* $0 < \varepsilon < 1$*, and define* $\beta_{\mathrm{rej}} := \min\{\max\{m, K\}, \rho_m(K+1)\}$*. For every input and every prefix,* CBS *satisfies*

$$C^{\mathrm{CBS}}(I_t) \le \min\{(1+\varepsilon)\widehat{C}_t, \beta_{\mathrm{rej}}\mathrm{OPT}(I_t)\}. \tag{14}$$

Consequently, $\mathrm{Cons}(\mathrm{CBS}) \le 1 + \varepsilon$ *and* $\mathrm{Rob}(\mathrm{CBS}) \le \beta_{\mathrm{rej}}$.

Proof sketch. For consistency, order the positive values $q_i(t)$ by their last increase times. The selection rules imply that any $m-1$ values among $q_1(t), \dots, q_m(t)$ sum to at most $g_m(K)\widehat{C}_t$. Combining this bound with the prediction-side load on a machine attaining M^{P}_t gives $C^{\mathrm{CBS}}(I_t) \le (1 + g_m(K))\widehat{C}_t = (1+\varepsilon)\widehat{C}_t$.

For robustness, consider the potential $\Phi_t := \max\{M^{\mathrm{P}}_t + A^{\mathrm{P}}_t + Q^{\mathrm{B}}_t, KQ^{\mathrm{B}}_t\}$. Each job increases it by at most $\min\{p_t, \max\{m, K\}r_t\}$. Summing the processing-time bound over jobs accepted by an optimal solution and the penalty bound over its rejected jobs gives $C^{\mathrm{CBS}}(I_t) \le \max\{m, K\}\mathrm{OPT}(I_t)$.

A second bound uses the full reference cost $C^{\mathcal{B}}(I_t)$. Whenever $M^{\mathrm{P}}_t + A^{\mathrm{P}}_t$ increases, the reference selection condition fails, bounding its new value by $KC^{\mathcal{B}}(I_t)$. Monotonicity of the reference cost and $Q^{\mathrm{B}}_t \le C^{\mathcal{B}}(I_t)$ then give $C^{\mathrm{CBS}}(I_t) \le (K+1)C^{\mathcal{B}}(I_t) \le \rho_m(K+1)\mathrm{OPT}(I_t)$. Both robustness bounds hold for the same execution. The full proof is given in Appendix D.1. □

Interpretation. For the RTP reference specified above, $\rho_m = 1 + (1 - 1/m)(1 + \sqrt{5})/2$ (Bartal et al., 1996). Since $1/K \le g_m(K) < 1/(K-1)$, the parameter satisfies $1/\varepsilon \le K < 1 + 1/\varepsilon$. Thus, the algorithm is $(1+\varepsilon)$-consistent and $O(1/\varepsilon)$-robust uniformly in m.

The same prefix guarantee also controls performance under imperfect predictions. Define the prediction error as $E := \widehat{C}_n - \mathrm{OPT}(I) \ge 0$. This measures the excess cost of the predicted solution over

the offline optimum. As in the setting without rejection, this is a solution-quality error rather than a job-wise prediction error; in particular, $E = 0$ for any predicted solution that is offline optimal.

Corollary 8 (Smoothness). *For every input with* $\mathrm{OPT}(I) > 0$ *and arbitrary predictions,*

$$\frac{C^{\mathrm{CBS}}(I)}{\mathrm{OPT}(I)} \leq \min\left\{\beta_{\mathrm{rej}}, (1+\varepsilon)\left(1+\frac{E}{\mathrm{OPT}(I)}\right)\right\}. \tag{15}$$

Proof. Substitute $\widehat{C}_n = \mathrm{OPT}(I) + E$ into Eq. (14) at $t = n$ and divide by $\mathrm{OPT}(I)$. □

Interpretation. Together, Theorem 7 and Corollary 8 establish the three desirable properties of a learning-augmented algorithm: consistency, robustness, and smoothness. The algorithm approaches the offline optimum arbitrarily closely with perfect predictions, retains a worst-case guarantee under arbitrary predictions, and has a performance bound that degrades continuously with prediction error.

4.4 Lower Bounds and Near-Optimality

We compare the algorithm with the best robustness achievable under the same consistency requirement. Let $R^*_{\mathrm{rej}}(m,\varepsilon) \coloneqq \inf_{A:\,\mathrm{Cons}(A)\leq 1+\varepsilon} \mathrm{Rob}(A)$ be the best achievable robustness under $(1+\varepsilon)$-consistency, where the infimum ranges over all deterministic online algorithms in our model, without restrictions on computational complexity or decision rules. The same geometric sum that controls consistency also yields a lower bound on this optimum.

Theorem 9 (General robustness lower bound). *For* $0 < \varepsilon < 1$,

$$R^*_{\mathrm{rej}}(m,\varepsilon) \geq K. \tag{16}$$

Proof sketch. Suppose an algorithm has robustness $\rho < K$. Choose b with $\max\{1,\rho\} < b < K$ and $\varepsilon < g_m(b) < 1$. Present $m-1$ unit jobs with penalties $b^{-1},\ldots,b^{-(m-1)}$, each predicted to a different machine. At each prefix, accepting the current job after rejecting the preceding ones would violate ρ-robustness, so every job must be rejected. Append one more unit job, with a large penalty and a prediction to the remaining machine. The predictions are now perfect with $\mathrm{OPT} = 1$, but the algorithm's cost exceeds $1+\varepsilon$, contradicting consistency. See Appendix D.2 for the full proof. □

Theorem 10 (Optimality and near-optimality). *Let* $0 < \varepsilon < 1$. *The algorithm satisfies*

$$1 \leq \frac{\mathrm{Rob(CBS)}}{R^*_{\mathrm{rej}}(m,\varepsilon)} \leq \frac{\beta_{\mathrm{rej}}}{K} = \min\left\{\max\left\{\frac{m}{K},1\right\}, \rho_m\left(1+\frac{1}{K}\right)\right\}. \tag{17}$$

If $0 < \varepsilon \leq g_m(m) = (1-m^{-(m-1)})/(m-1)$, *then the algorithm achieves optimal robustness:*

$$\mathrm{Rob(CBS)} = R^*_{\mathrm{rej}}(m,\varepsilon) = K. \tag{18}$$

Proof sketch. Combining Theorems 7 and 9 gives $K \leq R^*_{\mathrm{rej}}(m,\varepsilon) \leq \mathrm{Rob(CBS)} \leq \beta_{\mathrm{rej}}$, proving Eq. (17). Since g_m is strictly decreasing, $\varepsilon \leq g_m(m)$ is equivalent to $K \geq m$, in which case $\beta_{\mathrm{rej}} \leq K$ and all bounds coincide. The full proof is given in Appendix D.3. □

Interpretation. For each fixed m, the algorithm therefore attains the optimal robustness when the required consistency is sufficiently close to one. With the RTP reference specified in Section 4.2, the upper bound in Eq. (17) yields

$$\frac{\mathrm{Rob(CBS)}}{R^*_{\mathrm{rej}}(m,\varepsilon)} \leq (1+\varphi)\left(1+\frac{\varepsilon}{1+\varepsilon}\right) < \frac{3}{2}(1+\varphi), \tag{19}$$

where $\varphi \coloneqq (1+\sqrt{5})/2$; the derivation is given in Appendix D.3. Thus, with RTP, the algorithm remains within a uniform factor of $\frac{3}{2}(1+\varphi) \approx 3.927$ of the optimum throughout $0 < \varepsilon < 1$. With this reference, our algorithm achieves these guarantees using only $O(m)$ time per job, even though the optimum is taken over all deterministic online algorithms satisfying the same consistency requirement, without computational restrictions. The simpler choice $K_0 = 1 + 1/\varepsilon$ also preserves $(1+\varepsilon)$-consistency and gives near-optimal robustness when $\varepsilon \leq 1/(m-1)$; see Appendix E.2.

5 Numerical Experiments

5.1 Experimental Setup

Following prior work on learning-augmented algorithms (Purohit et al., 2018; Bampis et al., 2022; Antoniadis et al., 2025; Baba et al., 2026), we generate synthetic predictions with controlled perturbation levels to evaluate how the algorithm behaves as the prediction error varies. We consider

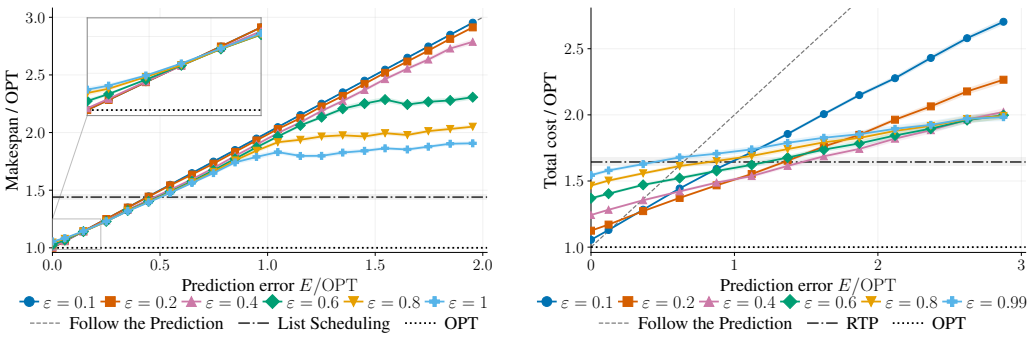

Figure 1: Numerical results of Algorithm 1 without rejection (**Left**) and Algorithm 2 with rejection (**Right**) as prediction error increases. Both prediction error and objective values are normalized by the offline optimum. The lines denote the mean, and the shaded regions denote the standard error, both computed over 50 independent input instances with different random seeds. On the left, Follow the Prediction nearly coincides with the $\varepsilon = 0.1$ curve over the plotted range.

$n = 100$ jobs on $m = 10$ machines in both settings, using 50 input seeds and 200 prediction seeds per input. For each instance, we compute an offline optimal schedule and generate predictions by randomly modifying its assignments, varying the amount of modification from perfect predictions to increasingly inaccurate ones. We evaluate Algorithms 1 and 2 with $\varepsilon \in \{0.1, 0.2, 0.4, 0.6, 0.8, 1\}$ (replacing 1 with 0.99 for the rejection setting, since our analysis assumes $\varepsilon < 1$), using the same inputs and predictions for all parameter values within each setting.

We compare against following the predictions directly (*Follow the Prediction*) and List Scheduling (Graham, 1966) without rejection or RTP (Bartal et al., 1996) with rejection. We plot the objective value and prediction error, both normalized by $\mathrm{OPT}(I)$. Further details are provided in Appendix F. We provide our code in the supplementary materials for reproducibility.

5.2 Results and Discussion

Figure 1 illustrates the consistency–robustness trade-off predicted by our analysis in Sections 3.4 and 4.3. When the prediction error is small, our algorithms remain close to Follow the Prediction and achieve objective values near the offline optimum, reflecting their consistency under accurate predictions. As the prediction error increases, Follow the Prediction continues to deteriorate, whereas our algorithms gradually saturate at bounded objective values, illustrating robustness to inaccurate predictions. Between these two regimes, performance degrades smoothly with the prediction error, consistent with the smoothness guarantees. Thus, the experiments illustrate the three desirable properties of learning-augmented algorithms: consistency, robustness, and smoothness.

The effect of ε is also consistent with the theoretical trade-off. Smaller values of ε keep the algorithms closer to the predicted solution, yielding better performance when the predictions are accurate, but leading to a higher saturation level as the prediction error grows. Larger values of ε allow greater deviation from the predictions, slightly sacrificing performance in the low-error regime while reaching a lower plateau earlier. The additional results in Appendix G show that these qualitative trends persist across different experimental settings.

6 Conclusion

We studied learning-augmented online scheduling on identical parallel machines, both with and without rejection, using predicted assignments. Our algorithms are $(1 + \varepsilon)$-consistent, robust to arbitrary predictions, and admit smooth performance guarantees as prediction error increases. We also established general robustness lower bounds for deterministic algorithms under the same consistency requirement, showing that our algorithms achieve optimal robustness for sufficiently small ε and remain near-optimal over broader parameter ranges. Our numerical experiments further illustrate the resulting consistency–robustness trade-off. Limitations of our study include the use of synthetic problems and prediction errors in the experiments and the restriction of our theoretical optimality results to deterministic algorithms and identical machines. Extending the theory to randomized algorithms and heterogeneous machines, and evaluating the algorithms on large-scale real-world workload traces, are natural directions for future work.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, formulate mathematical claims, and assist with translation. We have not used generative AI tools to help develop theoretical models or conceptual frameworks, provide critical ingredients for proving mathematical claims, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, and generating synthetic data sets. Additionally, we used generative AI tools to improve readability, create or edit software code (documentation and minor refactoring). We have reviewed all AI-assisted work. All AI-assisted text, code, and documentation, including English-language editing and translation, were thoroughly reviewed by the authors. The authors verified their correctness and appropriateness and substantially revised them as needed before incorporating them into the final work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This section is not applicable to this work.

REPRODUCIBILITY STATEMENT

For the theoretical results, complete proofs, together with the assumptions and supporting derivations, are fully provided in the appendix. Implementation details are also provided in the appendix, and the source code is included in the supplementary materials.

REFERENCES

Antonios Antoniadis, Christian Coester, Marek Eliáš, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *ACM Transactions on Algorithms*, 19(2):1–34, 2023. doi: 10.1145/3582689.

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

Antonios Antoniadis, Ali Shahheidar, Golnoosh Shahkarami, and Abolfazl Soltani. A switching framework for online interval scheduling with predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 36129–36136, 2026. doi: 10.1609/aaai.v40i43.40930.

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

Kaito Baba, Evripidis Bampis, and Giorgos Mitropoulos. Learning-augmented approximation for unrelated-machines makespan scheduling. *arXiv preprint arXiv:2606.13133*, 2026.

Evripidis Bampis, Konstantinos Dogeas, Alexander Kononov, Giorgio Lucarelli, and Fanny Pascual. Scheduling with untrusted predictions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pp. 4581–4587. International Joint Conferences on Artificial Intelligence Organization, 7 2022. doi: 10.24963/ijcai.2022/636.

Evripidis Bampis, Alexander Kononov, Giorgio Lucarelli, and Fanny Pascual. Non-clairvoyant makespan minimization scheduling with predictions. In *34th International Symposium on Algorithms and Computation*, volume 283 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 9:1–9:15, 2023. ISBN 978-3-95977-289-1. doi: 10.4230/LIPIcs.ISAAC.2023.9.

Yair Bartal, Stefano Leonardi, Alberto Marchetti-Spaccamela, Jiří Sgall, and Leen Stougie. Multiprocessor scheduling with rejection. In *Proceedings of the Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 95–103. Society for Industrial and Applied Mathematics, 1996. ISBN 0898713668.

Daniel W. Engels, David R. Karger, Stavros G. Kolliopoulos, Sudipta Sengupta, R. N. Uma, and Joel Wein. Techniques for scheduling with rejection. In *Proceedings of the 6th Annual European Symposium on Algorithms*, pp. 490–501. Springer-Verlag, 1998. ISBN 3540648488.

Rudolf Fleischer and Michaela Wahl. Online scheduling revisited. In *Proceedings of the 8th Annual European Symposium on Algorithms*, pp. 202–210. Springer-Verlag, 2000. ISBN 354041004X.

R. L. Graham. Bounds for certain multiprocessing anomalies. *The Bell System Technical Journal*, 45(9):1563–1581, 1966. doi: 10.1002/j.1538-7305.1966.tb01709.x.

Dorian Guyot and Alexandra Lassota. Minimalistic predictions for online class constraint scheduling. In *International Conference on Learning Representations*, volume 2025, pp. 12742–12754, 2025.

Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 1859–1877. Society for Industrial and Applied Mathematics, 2020.

Thomas Lavastida, Benjamin Moseley, R. Ravi, and Chenyang Xu. Learnable and instance-robust predictions for online matching, flows and load balancing. In *29th Annual European Symposium on Algorithms*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 59:1–59:17, 2021. ISBN 978-3-95977-204-4. doi: 10.4230/LIPIcs.ESA.2021.59.

Shi Li and Jiayi Xian. Online unrelated machine load balancing with predictions revisited. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 6523–6532. PMLR, 2021.

Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, pp. 357–368. Association for Computing Machinery, 2022. ISBN 9781450391467. doi: 10.1145/3490148.3538579.

Thodoris Lykouris and Sergei Vassilvitskii. 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.

Michael Mitzenmacher and Sergei Vassilvitskii. Algorithms with predictions. *Communications of the ACM*, 65(7):33–35, 2022. ISSN 0001-0782. doi: 10.1145/3528087.

Laurent Perron and Frédéric Didier. Cp-sat. URL `https://developers.google.com/optimization/cp/cp_solver/`.

Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Dvir Shabtay and Nufar Gasper. Two-machine flow-shop scheduling with rejection. *Computers & Operations Research*, 39(5):1087–1096, 2012. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2011.05.023.

Alexander Wei and Fred Zhang. Optimal robustness-consistency trade-offs for learning-augmented online algorithms. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*. Curran Associates Inc., 2020. ISBN 9781713829546.

Michalis Xefteris. From estimates to schedules: Learning-augmented restricted assignment. *arXiv preprint arXiv:2606.08377*, 2026.

Jiaming Yang and Zhiyi Tan. Online scheduling with rejection revisited. *Information and Computation*, 310:105442, 2026. ISSN 0890-5401. doi: https://doi.org/10.1016/j.ic.2026.105442.

Tianming Zhao and Albert Zomaya. On smoothness bounds for non-clairvoyant scheduling with predictions. In *The Fourteenth International Conference on Learning Representations*, 2026.

Tianming Zhao, Wei Li, and Albert Y. Zomaya. Uniform machine scheduling with predictions. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 32, pp. 413–422, 2022. doi: 10.1609/icaps.v32i1.19827.

A Extended Related Work

A.1 Classical Online Scheduling and Rejection

In the classical list model, a job's processing time is revealed on arrival and its machine must be chosen immediately. List Scheduling assigns each job to a least-loaded machine and achieves competitive ratio $2 - 1/m$ (Graham, 1966). Subsequent algorithms improve this worst-case guarantee; for example, Fleischer & Wahl (2000) obtain an asymptotic ratio below 1.9201.

Bartal et al. (1996) introduced the objective of accepted-job makespan plus total rejection penalties. They give a $(3 + \sqrt{5})/2$-competitive online algorithm and a matching deterministic lower bound when the number of machines is unrestricted, as well as sharper results for some fixed machine counts. More recently, Yang & Tan (2026) obtain a competitive ratio of approximately 1.83929 for three machines and improve the guarantees for fixed $m \geq 4$; however, their algorithm retains the asymptotic ratio $(3 + \sqrt{5})/2$ as $m \to \infty$.

Our algorithms use assignment predictions to go beyond these classical worst-case guarantees when the predictions are accurate, while retaining robustness under arbitrary predictions.

A.2 Processing-Time Predictions and Learned Weights

A substantial line of work uses predictions to estimate unknown processing times. Purohit et al. (2018) study non-clairvoyant single-machine scheduling for total completion time, combining predictions with protection against inaccurate estimates. For identical-machine makespan minimization, Bampis et al. (2023) analyze algorithms and error-dependent lower bounds in both preemptive and non-preemptive settings. Zhao et al. (2022) study processing-time predictions on uniformly related machines. More recently, Zhao & Zomaya (2026) refine the analysis of smoothness and obtain bounds for non-clairvoyant completion-time and makespan objectives. These works address uncertainty about processing requirements. In our model, each arriving job's processing time is known exactly, while its predicted machine provides guidance about how to accommodate unknown future jobs.

A different approach predicts a compact description of a fractional assignment. Lattanzi et al. (2020) use machine weights for online restricted-assignment scheduling, where each job has a set of eligible machines. Accurate weights yield near-optimal fractional assignments, which are converted into integral schedules by online rounding. Li & Xian (2021) extend this approach to unrelated machines and improve the rounding guarantees. Our consistency guarantee concerns the resulting integral schedule, and our predictions directly specify integral actions.

Lavastida et al. (2021) study the learnability of compact predictions and their performance under changes in the input instance, including applications to load balancing. Such guarantees concern how predictions can be obtained from samples and reused across related instances. Our analysis takes assignment predictions as given and establishes worst-case guarantees over their possible values; it does not assume a distribution on instances or a particular learning procedure.

A.3 Predictions of Actions and Solutions

Guyot & Lassota (2025) provide a particularly close comparison: their online model has identical machines, known processing times on arrival, and a limit on the number of job classes assigned to each machine. Among several prediction models, they consider a suggested machine for each job, supplied either in advance or on arrival. Their action-prediction algorithm has competitive ratio at most $\ell + 1$, where ℓ counts prediction errors, and they establish lower bounds using the class constraints. We study the unrestricted assignment model and the robustness attainable under a fixed consistency requirement. Without class constraints, their action-prediction algorithm simply follows every prediction and can incur a competitive ratio of m, while their lower-bound construction relies on these constraints. Our algorithms selectively override predictions to control robustness while preserving $(1 + \varepsilon)$-consistency, both with and without rejection. We also establish robustness lower bounds for all deterministic $(1 + \varepsilon)$-consistent algorithms without class constraints.

Lindermayr & Megow (2022) predict relative job orders for non-clairvoyant scheduling with a total weighted completion-time objective. On unrelated machines, their predictions additionally specify a

machine assignment. Their algorithms and error measures exploit scheduling order and the ability to share processing capacity over time. Our decision is the irrevocable placement of an entire arriving job, and only the machine loads affect the makespan objective.

Assignment predictions also appear in recent work on offline makespan minimization. Xefteris (2026) studies restricted assignment and develops approximation and repair algorithms, measuring error by the minimum processing volume that must be moved to obtain an optimal feasible schedule. Baba et al. (2026) use predictions of heavy job assignments on unrelated machines to obtain a polynomial-time $(1+\varepsilon)$-approximation under accurate predictions, with smooth degradation to a worst-case factor of 2. These approaches use predictions to guide computation on a fully known instance, whereas our algorithms receive jobs and predictions sequentially and must commit before seeing future jobs.

A.4 Trade-offs, Smoothness, and Combining Algorithms

Wei & Zhang (2020) establish lower bounds on consistency–robustness trade-offs for ski rental and non-clairvoyant scheduling. Their scheduling setting is substantially different from ours: they consider a preemptive single-machine problem with processing-time predictions and a total completion-time objective, including a tight characterization for two jobs. In contrast, we study immediate and irrevocable assignments on identical parallel machines with assignment predictions, for makespan objectives with and without rejection. Our lower bounds apply to all deterministic $(1+\varepsilon)$-consistent algorithms in these models, without restrictions on computation or decision rules, and quantify how close our algorithms are to the best achievable robustness.

Combining a prediction-based algorithm with a robust algorithm is a recurring technique. Antoniadis et al. (2023) develop methods for combining online algorithms in metric settings, with applications to metrical task systems and other online problems. In non-clairvoyant scheduling, Lindermayr & Megow (2022) use preferential time sharing between algorithms. Antoniadis et al. (2026) study the combination of prediction-based and classical algorithms for online interval scheduling with irrevocable acceptance decisions, proving trade-off bounds. Their objective is to maximize the total length of accepted, nonoverlapping intervals, whereas our rejection model minimizes makespan plus rejection penalties. In our setting, previously assigned jobs cannot be moved to reproduce the reference schedule, and selecting some of its actions produces a different load profile from following it throughout.

B Deferred Proofs for Scheduling without Rejection (Section 3)

B.1 Feasibility of Dual-Cap

We give the full proof of Theorem 1.

Proof. If $\beta = m$, the algorithm always follows the prediction: inductively, assigning the current job to its predicted machine gives load $S_{h_t}(t) \le P_t \le mB_t$ and also satisfies the consistency cap. Thus the algorithm is well-defined and maintains both caps in this case.

Suppose $\beta < m$, and write $c_m := 2 - 1/m$. Then $\beta = (c_m+\varepsilon)^2/(4\varepsilon) \ge c_m$. We prove feasibility and Eq. (3) by induction, starting from the empty schedules. Assume that the two load bounds in Eq. (3) hold after the first $t-1$ jobs, and consider the arrival of job t. A zero-size job leaves all loads and scales unchanged and is immediately feasible, so assume $p_t > 0$. Because the virtual schedule already includes this job while the actual schedule does not, we have

$$\sum_i \big(\ell_i(t-1) - S_i(t)\big) = -p_t, \qquad \sum_i \ell_i(t-1) = P_t - p_t \le mB_t - p_t. \tag{20}$$

The predicted machine satisfies the consistency cap, since

$$\ell_{h_t}(t-1) + p_t \le S_{h_t}(t-1) + \varepsilon H_{t-1} + p_t = S_{h_t}(t) + \varepsilon H_{t-1} \le S_{h_t}(t) + \varepsilon H_t. \tag{21}$$

Here, the first inequality uses the induction hypothesis, the equality uses $S_{h_t}(t) = S_{h_t}(t-1) + p_t$, and the last inequality uses $H_{t-1} \le H_t$.

Moreover, a least-loaded machine satisfies the robustness cap, as

$$\min_i \ell_i(t-1) + p_t \le \frac{P_t - p_t}{m} + p_t \le c_m B_t \le \beta B_t. \tag{22}$$

The first inequality holds because the minimum load is at most the average load, and the second follows by writing $(P_t - p_t)/m + p_t = P_t/m + (1 - 1/m)p_t$ and using $P_t/m \le B_t$ and $p_t \le B_t$ by the definition of B_t.

Suppose, for a contradiction, that no machine satisfies both caps. Define the machines blocked by the robustness cap as

$$K \coloneqq \{i \in [m] : \ell_i(t-1) + p_t > \beta B_t\}, \qquad k \coloneqq |K|. \tag{23}$$

By Eq. (21), the predicted machine h_t satisfies the consistency cap, so under our contradiction assumption it must violate the robustness cap; hence $h_t \in K$. Combining $h_t \in K$ with Eq. (21) and $S_{h_t}(t) \le H_t$ gives $\beta B_t < \ell_{h_t}(t-1) + p_t \le S_{h_t}(t) + \varepsilon H_t \le (1+\varepsilon)H_t$. Thus, we have $U \coloneqq (1+\varepsilon)H_t - \beta B_t > 0$.

By Eq. (22), a least-loaded machine i satisfies the robustness cap and therefore lies outside K. This machine satisfies the robustness cap but must violate the consistency cap under our assumption, giving $\beta B_t \ge \ell_i(t-1) + p_t > S_i(t) + \varepsilon H_t \ge \varepsilon H_t$. Thus, we have $V \coloneqq \beta B_t - \varepsilon H_t > 0$.

For $i \in K$, the robustness cap is violated, so $\ell_i(t-1) > \beta B_t - p_t$. For $i \notin K$, the robustness cap is satisfied, so our contradiction assumption implies that the consistency cap is violated, giving $\ell_i(t-1) > S_i(t) + \varepsilon H_t - p_t$. Using $S_i(t) \le H_t$ for $i \in K$ and Eq. (20), we obtain

$$\begin{aligned} -p_t &= \sum_i \big(\ell_i(t-1) - S_i(t)\big) \\ &> k(\beta B_t - p_t - H_t) + (m-k)(\varepsilon H_t - p_t) \\ &= m\varepsilon H_t - mp_t - kU. \end{aligned} \tag{24}$$

Here, the strict inequality applies the corresponding lower bounds separately to the k machines in K and the $m-k$ machines outside K. Similarly, using $S_i(t) \ge 0$ for $i \notin K$ and Eq. (20), we obtain

$$\begin{aligned} mB_t - p_t &\ge \sum_i \ell_i(t-1) \\ &> k(\beta B_t - p_t) + (m-k)(\varepsilon H_t - p_t) \\ &= m\varepsilon H_t - mp_t + kV. \end{aligned} \tag{25}$$

Rearranging these two bounds gives

$$kU > m\varepsilon H_t - (m-1)p_t, \tag{26}$$

$$kV < mB_t + (m-1)p_t - m\varepsilon H_t. \tag{27}$$

Multiplying the first inequality by $V > 0$ and the second by $U > 0$ gives a lower and an upper bound on kUV; comparing these bounds eliminates k and yields

$$V\big(m\varepsilon H_t - (m-1)p_t\big) < U\big(mB_t + (m-1)p_t - m\varepsilon H_t\big) \tag{28}$$

$$\iff \quad (U+V)\big(m\varepsilon H_t - (m-1)p_t\big) < mB_t U \tag{29}$$

$$\iff \quad H_t\big(m\varepsilon H_t - (m-1)p_t\big) < mB_t\big((1+\varepsilon)H_t - \beta B_t\big) \tag{30}$$

$$\iff \quad 0 > \beta B_t^2 - (1+\varepsilon)H_t B_t + \varepsilon H_t^2 - \left(1 - \frac{1}{m}\right) H_t p_t \tag{31}$$

$$\ge \beta B_t^2 - (c_m + \varepsilon)H_t B_t + \varepsilon H_t^2 \tag{32}$$

$$= \varepsilon \left(H_t - \frac{c_m + \varepsilon}{2\varepsilon} B_t\right)^2 \ge 0. \tag{33}$$

Here, Eq. (30) substitutes $U + V = H_t$ and $U = (1+\varepsilon)H_t - \beta B_t$. Then Eq. (32) uses $p_t \le B_t$ from the definition of B_t, and Eq. (33) uses $\beta = (c_m + \varepsilon)^2/(4\varepsilon)$ to complete the square. This is a contradiction. Therefore, $\mathcal{A}_t$ is nonempty.

The chosen machine satisfies both load bounds after assignment by the definition of $\mathcal{A}_t$. On every other machine i, the actual load remains unchanged, while the caps βB_t and $S_i(t) + \varepsilon H_t$ do not decrease, so the load bounds continue to hold by the induction hypothesis. Thus, both bounds in Eq. (3) hold after job t, completing the induction. □

B.2 Consistency and Robustness of Dual-Cap

We give the full proof of Theorem 2.

Proof. By Theorem 1, both load caps hold after every assignment. Taking the maximum over machines in Eq. (3) gives Eq. (6), since $\max_i S_i(t) = H_t$. At the final time, $B_n \le \mathrm{OPT}(I)$ gives the robustness upper bound β. Under perfect predictions, $H_n = \mathrm{OPT}(I)$ additionally gives the consistency bound $1 + \varepsilon$.

To show that the robustness bound is exact, consider m jobs, all predicted to the same machine. The first job has processing time 1, and each of the remaining $m - 1$ jobs has processing time $s := (\beta - 1)/(m - 1)$. Since $1 \le \beta \le m$, we have $0 \le s \le 1$, and the total processing volume is $\beta \le m$, so $B_t = 1$ at every nonempty prefix. Following the prediction keeps the actual and virtual loads equal, and their common maximum never exceeds β. Thus the predicted machine always satisfies both caps, and the algorithm places all jobs on that machine, attaining makespan β. Assigning each job to a different machine gives $\mathrm{OPT}(I) = 1$, proving $\mathrm{Rob}(\mathrm{DualCap}) \ge \beta$. □

B.3 General Robustness Lower Bound

We give the full proof of Theorem 5.

Proof. Fix any deterministic $(1 + \varepsilon)$-consistent algorithm A. If $\varepsilon < 1/m$, present m unit jobs, all predicted to the same machine. Using two machines would give two loads of at least $1 > \varepsilon m$, violating Lemma 4. Thus all jobs must be assigned to one machine, yielding makespan m while $\mathrm{OPT} = 1$.

Now suppose $1/m \le \varepsilon < 1/2$, and define $\rho := 1/\varepsilon$, $k := \lceil \rho \rceil - 1$. Then $\rho > 2$, $2 \le k \le m - 1$, and $\varepsilon k < 1$. Choose $0 < \tau < \min\{1, (\rho - 2)/(1 - \varepsilon)\}$ and set $a := (\rho - 1 - \tau)/k$. Present k jobs of size a, all predicted to the same machine. Here $0 < a < 1$, and distributing these jobs over two machines would give two loads of at least $a > \varepsilon k a$. By Lemma 4, all k jobs must therefore be assigned to one machine.

Next, present a unit job with the same predicted machine, bringing the total volume to $V := ka + 1 = \rho - \tau$. The choice of τ ensures $1 > \varepsilon V$ and $ka - \varepsilon V = \rho - 2 - (1 - \varepsilon)\tau > 0$. Assigning this job to a different machine would create two loads exceeding εV, again contradicting Lemma 4. Hence A incurs makespan $\rho - \tau$. The input has at most m jobs, all of size at most 1 and including a unit job, so $\mathrm{OPT} = 1$. Ending the input here and taking $\tau \to 0^+$ gives $\mathrm{Rob}(A) \ge \rho = 1/\varepsilon$. □

B.4 Near-Optimality of Dual-Cap

We give the full proof of Theorem 6.

Proof. Dual-Cap is $(1 + \varepsilon)$-consistent, so $R^*(m, \varepsilon) \le \mathrm{Rob}(\mathrm{DualCap}) = \beta$ by Theorem 2. If $\varepsilon \le 1/m$, Eq. (1) gives $\beta = m$, and Theorem 5 forces $R^*(m, \varepsilon) = m$. Both gap bounds follow immediately in this case.

If $\varepsilon > 1/m$, the lower bound is $R^*(m, \varepsilon) \ge 1/\varepsilon$. Hence

$$\frac{\mathrm{Rob}(\mathrm{DualCap})}{R^*(m, \varepsilon)} \le \varepsilon\beta \le \frac{(2 - 1/m + \varepsilon)^2}{4} \le \left(1 + \frac{\varepsilon}{2}\right)^2. \tag{34}$$

Similarly, using $2 - 1/m \le 2$, we obtain

$$\mathrm{Rob}(\mathrm{DualCap}) - R^*(m, \varepsilon) \le \beta - \frac{1}{\varepsilon} \le \frac{(2 + \varepsilon)^2 - 4}{4\varepsilon} = 1 + \frac{\varepsilon}{4}. \tag{35}$$

□

C Refined Guarantees for Scheduling without Rejection

We can improve the robustness in some parameter ranges by choosing a smaller coefficient β while keeping the assignment rule in Algorithm 1 unchanged. We first sharpen the feasibility argument near the saturation threshold, then exploit additional load bounds maintained implicitly by the algorithm. Both refinements preserve the $(1+\varepsilon)$-consistency guarantee.

C.1 Refinement near the Saturation Threshold

Theorem 11 (Refinement near the saturation threshold)**.** *Suppose $m \geq 3$, $1/m \leq \varepsilon \leq 1/(m-1)$, and $\varepsilon < 1/2$. The coefficient*

$$\beta \coloneqq 2m - 1 - m(m-1)\varepsilon \tag{36}$$

guarantees feasibility on every input, and Algorithm 1 using this coefficient satisfies the robustness bounds

$$\frac{\mathrm{Rob(DualCap)}}{R^*(m,\varepsilon)} \leq 1 + \frac{\varepsilon^2}{4}, \qquad \mathrm{Rob(DualCap)} - R^*(m,\varepsilon) \leq \frac{\varepsilon}{4}. \tag{37}$$

Proof. We have $m-1 \leq \beta \leq m$, and $\beta = m$ simply follows the prediction. For $\beta < m$, use the induction setup in Appendix B.1 and suppose job t cannot be placed. Let K be the machines violating the robustness cap and $k \coloneqq |K|$. Since $\beta \geq m - 1 \geq 2 - 1/m$, we again have $h_t \in K$ and $1 \leq k \leq m-1$. Each machine $i \in K$ has load $\ell_i(t-1) > \beta B_t - p_t$, while the total load over all machines is $P_t - p_t \leq mB_t - p_t$. Summing over the k machines in K therefore gives

$$k(\beta B_t - p_t) < \sum_{i \in K} \ell_i(t-1) \leq \sum_{i \in [m]} \ell_i(t-1) = P_t - p_t \leq mB_t - p_t. \tag{38}$$

Rearranging and using $k \geq 1$ and $p_t \leq B_t$, we obtain

$$k\beta B_t < mB_t + (k-1)p_t \leq (m+k-1)B_t. \tag{39}$$

Dividing by $kB_t > 0$ yields

$$\beta < 1 + \frac{m-1}{k}. \tag{40}$$

Thus $k \geq 2$ would imply $\beta < (m+1)/2 \leq m-1$, a contradiction, so $K = \{h_t\}$. Define $a \coloneqq 1 - (m-1)\varepsilon \geq 0$. The deviation and load bounds in Eqs. (26) and (27) from the feasibility proof also hold here. Substituting $k = 1$, $U = (1+\varepsilon)H_t - \beta B_t$, and $V = \beta B_t - \varepsilon H_t$ into these inequalities and rearranging yields

$$aH_t > \beta B_t - (m-1)p_t, \qquad \beta B_t + (m-1)\varepsilon H_t < mB_t + (m-1)p_t. \tag{41}$$

If $a > 0$, the first inequality gives $H_t > (\beta B_t - (m-1)p_t)/a$. Substituting this lower bound into the second inequality and using $(m-1)\varepsilon = 1 - a > 0$ yields

$$\beta B_t + \frac{1-a}{a}\big(\beta B_t - (m-1)p_t\big) < mB_t + (m-1)p_t. \tag{42}$$

Rearranging and using $p_t \leq B_t$ gives $\beta B_t < maB_t + (m-1)p_t \leq (ma + m - 1)B_t$, hence $\beta < ma + m - 1 = 2m - 1 - m(m-1)\varepsilon$, contradicting its definition. If $a = 0$, the first inequality gives $\beta B_t < (m-1)p_t \leq (m-1)B_t$, hence $\beta < m-1$. However, $a = 0$ implies $\varepsilon = 1/(m-1)$, so the definition of β gives $\beta = m-1$, a contradiction. Therefore the algorithm can place the job, and the two caps are preserved by induction.

Using $R^*(m,\varepsilon) \geq 1/\varepsilon$ and $\mathrm{Rob(DualCap)} = \beta$, we obtain

$$\begin{aligned} \frac{\mathrm{Rob(DualCap)}}{R^*(m,\varepsilon)} &\leq \varepsilon\beta = 1 + (m\varepsilon - 1)(1 - (m-1)\varepsilon) \\ &= 1 + \frac{\varepsilon^2}{4} - \Big(m\varepsilon - 1 - \frac{\varepsilon}{2}\Big)^2 \leq 1 + \frac{\varepsilon^2}{4}. \end{aligned} \tag{43}$$

The additive bound follows from the same estimate:

$$\mathrm{Rob(DualCap)} - R^*(m,\varepsilon) \leq \beta - \frac{1}{\varepsilon} = \frac{\varepsilon\beta - 1}{\varepsilon} \leq \frac{\varepsilon}{4}. \tag{44}$$

□

Interpretation. In this parameter range, the refinement removes the first-order term in ε from the multiplicative bound in Theorem 6, and reduces the additive bound from $1 + \varepsilon/4$ to $\varepsilon/4$. Thus, the multiplicative gap is second-order in ε, while the additive gap is first-order in ε, with the assignment rule, consistency guarantee, and computational complexity unchanged.

C.2 Implicit Load Invariants

Beyond this band, a sharper analysis uses two additional load invariants implicitly maintained by Algorithm 1. These invariants hold on every prefix that the algorithm completes and require no additional tests in the algorithm.

C.2.1 The Leave-One-Out Load Bound

For each machine, define the largest predicted load on the other machines as

$$H_{-i}(t) \coloneqq \max_{j \neq i} S_j(t). \tag{45}$$

Lemma 12 (Leave-one-out load bound)**.** *Let $0 < \varepsilon \leq 1$ and $\beta \in [1, m]$. On every prefix that Dual-Cap completes, it maintains*

$$\ell_i(t) \leq S_i(t) + \varepsilon H_{-i}(t) \quad \textit{for all } i \in [m]. \tag{46}$$

Proof. We prove this bound by induction over completed assignments. The bound holds initially; assume it holds after the first $t-1$ jobs and consider job t. Let i_t denote the machine chosen for job t.

Case 1: The algorithm follows the prediction. Following the prediction increases $\ell_{h_t}(t-1)$ and $S_{h_t}(t-1)$ by the same amount p_t and leaves $H_{-h_t}(t) = H_{-h_t}(t-1)$, preserving Eq. (46) for $i = h_t$.

Case 2: The algorithm assigns the job elsewhere. The induction hypothesis gives $\ell_{h_t}(t-1) + p_t \leq S_{h_t}(t) + \varepsilon H_{-h_t}(t) \leq S_{h_t}(t) + \varepsilon H_t$, so machine h_t satisfies the consistency cap. Since the algorithm always chooses the predicted machine when it satisfies both caps, assigning the job elsewhere implies that h_t violates the robustness cap. Thus, writing $T_i(t) \coloneqq S_i(t) + \varepsilon H_{-i}(t)$, we have $\beta B_t < \ell_{h_t}(t-1) + p_t \leq T_{h_t}(t)$.

(a) If the predicted maximum is attained by multiple machines. Excluding any one machine still leaves a machine attaining H_t, so $H_{-i}(t) = H_t$ for every i. Hence $T_i(t) = S_i(t) + \varepsilon H_t$, and the consistency cap ensures $\ell_i(t-1) + p_t \leq T_i(t)$ for every admissible destination i. Thus we have $\ell_{i_t}(t) = \ell_{i_t}(t-1) + p_t \leq T_{i_t}(t)$, and Eq. (46) holds on the chosen machine after the assignment.

(b) If the predicted maximum is attained by a unique machine. Let $i^\star$ be the unique maximizer of $S_i(t)$ and $s_t \coloneqq \max_{i \neq i^\star} S_i(t)$. Only machine $i^\star$ has $T_{i^\star}(t) < S_{i^\star}(t) + \varepsilon H_t$, and $\varepsilon \leq 1$ implies

$$T_i(t) = S_i(t) + \varepsilon H_t \leq s_t + \varepsilon H_t \leq H_t + \varepsilon s_t = T_{i^\star}(t) \quad \text{for } i \neq i^\star. \tag{47}$$

For every admissible destination $i \neq i^\star$, the consistency cap already ensures $\ell_i(t-1) + p_t \leq T_i(t)$, so only machine $i^\star$ requires an additional check. Whenever $i^\star$ is an admissible destination, the robustness cap gives $\ell_{i^\star}(t-1) + p_t \leq \beta B_t < T_{h_t}(t) \leq T_{i^\star}(t)$. Every admissible destination i therefore satisfies $\ell_i(t-1) + p_t \leq T_i(t)$. Thus $\ell_{i_t}(t) = \ell_{i_t}(t-1) + p_t \leq T_{i_t}(t)$, establishing Eq. (46) on the chosen machine after the assignment.

In both Cases 1 and 2, every machine $i \neq i_t$ retains its previous actual load, while $S_i(t) + \varepsilon H_{-i}(t)$ is nondecreasing, giving $\ell_i(t) = \ell_i(t-1) \leq S_i(t-1) + \varepsilon H_{-i}(t-1) \leq S_i(t) + \varepsilon H_{-i}(t)$. Therefore, Eq. (46) holds on every machine after job t, completing the induction. □

Interpretation. Lemma 12 shows that the excess of a machine's actual load over its predicted load is bounded by ε times the largest predicted load on the other machines. Even if we strengthen the algorithm's consistency cap by replacing H_t with $H_{-i}(t)$, every job is assigned to the same machine for fixed β. The original algorithm therefore satisfies this stronger bound automatically, without any additional check or change to its assignment rule.

C.2.2 THE SUBSET LOAD BOUND

Define $\theta \coloneqq (\beta-1)/m$. For a machine set J, write $\ell(J,t) \coloneqq \sum_{i\in J} \ell_i(t)$ and $S(J,t) \coloneqq \sum_{i\in J} S_i(t)$.

Lemma 13 (Subset load bound)**.** *Let $0 < \varepsilon \le 1$ and $\beta \in [1,m]$ with $\theta \ge 1/2$. On every prefix that Algorithm 1 completes, it maintains*

$$\ell(J,t) \ge S(J,t) - (1-\theta)\max_{i\in J} S_i(t) \quad \textit{for every nonempty } J \subseteq [m]. \tag{48}$$

Proof. For a nonnegative vector $S = (S_1,\dots,S_m)$, write $S(J) \coloneqq \sum_{i\in J} S_i$. Define $\Phi(J,S) \coloneqq S(J) - (1-\theta)\max_{i\in J} S_i$ for nonempty $J \subseteq [m]$, and set $\Phi(\varnothing, S) \coloneqq 0$. Increasing one coordinate of S by $p \ge 0$ increases $\Phi(J,S)$ by an amount between 0 and p.

We next bound the decrease in Φ when a machine $h \in J$ is removed from J. If S_h is a maximum, set $a \coloneqq S_h$ and let b be the largest remaining coordinate, or zero if none remains. Then

$$\begin{aligned} \Phi(J,S) - \Phi(J\setminus\{h\},S) &= a - (1-\theta)(a-b) \\ &= \theta a + (1-\theta)b \\ &\le \theta(a+b) \\ &\le \theta S(J). \end{aligned} \tag{49}$$

The first inequality uses $\theta \ge 1/2$, and the second uses $a + b \le S(J)$. If S_h is not a maximum, removing h leaves the maximum unchanged, so $\Phi(J,S) - \Phi(J\setminus\{h\},S) = S_h \le S(J)/2 \le \theta S(J)$. In either case, $\Phi(J,S) - \Phi(J\setminus\{h\},S) \le \theta S(J)$, hence

$$\Phi(J,S) \le \theta S(J) + \Phi(J\setminus\{h\},S). \tag{50}$$

We now induct on completed assignments. Write $S(t) \coloneqq (S_1(t),\dots,S_m(t))$. Eq. (48) holds initially; assume it holds after the first $t-1$ jobs and consider job t. Let i_t denote the machine chosen for job t. If $i_t = h_t$, every set J containing h_t receives job t, so its actual load increases by p_t, while $\Phi(J,S(t)) - \Phi(J,S(t-1)) \le p_t$. The induction hypothesis therefore gives

$$\ell(J,t) = \ell(J,t-1) + p_t \ge \Phi(J,S(t-1)) + p_t \ge \Phi(J,S(t)). \tag{51}$$

For sets J not containing h_t, neither the actual loads nor the predicted loads within J change, so

$$\ell(J,t) = \ell(J,t-1) \ge \Phi(J,S(t-1)) = \Phi(J,S(t)). \tag{52}$$

Thus Eq. (48) is preserved for every set J when the algorithm follows the prediction. If $i_t \ne h_t$, the predicted machine h_t violates the robustness cap, giving

$$\ell_{h_t}(t-1) > \beta B_t - p_t \ge (\beta-1)B_t = \theta m B_t \ge \theta P_t. \tag{53}$$

Only the predicted load on h_t and the actual load on i_t increase. Thus, the sets requiring further consideration are those containing h_t but not i_t: for these sets, $\Phi(J,S(t))$ may increase while $\ell(J,t)$ remains unchanged. For such a set J, the predicted loads on $J\setminus\{h_t\}$ have not changed, so $\Phi(J\setminus\{h_t\},S(t-1)) = \Phi(J\setminus\{h_t\},S(t))$. Combining this equality with the induction hypothesis and Eqs. (50) and (53) gives

$$\begin{aligned} \ell(J,t) = \ell(J,t-1) &= \ell_{h_t}(t-1) + \ell(J\setminus\{h_t\},t-1) \\ &> \theta P_t + \Phi(J\setminus\{h_t\},S(t-1)) \\ &= \theta P_t + \Phi(J\setminus\{h_t\},S(t)) \\ &\ge \theta S(J,t) + \Phi(J\setminus\{h_t\},S(t)) \ge \Phi(J,S(t)). \end{aligned} \tag{54}$$

For every other set J, either $h_t \notin J$, so its predicted loads are unchanged and its actual load cannot decrease, or $h_t, i_t \in J$, so its actual load increases by p_t while $\Phi(J,S(t)) - \Phi(J,S(t-1)) \le p_t$. Consequently,

$$\begin{aligned} &\ell(J,t) \ge \ell(J,t-1) \ge \Phi(J,S(t-1)) = \Phi(J,S(t)) && \text{if } h_t \notin J, \\ &\ell(J,t) = \ell(J,t-1) + p_t \ge \Phi(J,S(t-1)) + p_t \ge \Phi(J,S(t)) && \text{if } h_t, i_t \in J. \end{aligned} \tag{55}$$

Together, Eqs. (51), (52), (54) and (55) cover all cases and establish $\ell(J,t) \ge \Phi(J,S(t))$ for every nonempty $J \subseteq [m]$, which is exactly Eq. (48), completing the induction. □

Interpretation. Lemma 13 shows that, for any set of machines, the total actual load cannot fall below the total predicted load by more than $(1-\theta)$ times the largest predicted load in that set.

C.3 Refinement Using Implicit Invariants

We now combine the load bounds in Lemmas 12 and 13 to establish feasibility for a smaller β, improving the robustness guarantee without changing the assignment rule in Algorithm 1.

C.3.1 Refined Robustness Guarantees

Theorem 14 (Refinement using implicit invariants)**.** *Suppose $m \geq 5$ and $1/(m-1) < \varepsilon \leq 2/(m+2)$. Define the function $\kappa(c) \coloneqq 2(c-1)^3/(c(c^2+2c-2))$ and choose*

$$\beta \coloneqq \frac{1}{\varepsilon} + \frac{m}{m-2}\kappa(m\varepsilon). \tag{56}$$

This coefficient guarantees feasibility on every input, and its robustness satisfies

$$\frac{\mathrm{Rob(DualCap)}}{R^*(m,\varepsilon)} \leq 1 + \frac{\varepsilon}{6} + \frac{\varepsilon^2}{3(1-\varepsilon)}, \quad 0 \leq \mathrm{Rob(DualCap)} - R^*(m,\varepsilon) \leq \frac{1}{6} + \frac{\varepsilon}{3(1-\varepsilon)}. \tag{57}$$

Proof. Feasibility follows from Lemma 16, whose proof uses the sufficient condition in Lemma 15; we briefly outline these two steps here. Lemma 15 derives a sufficient condition on β using Lemmas 12 and 13 to show that a job with no admissible destination would force the actual loads to exceed the total processing volume. Lemma 16 verifies that the coefficient in Eq. (56) satisfies this condition throughout the stated parameter range, including all finite-m terms.

The same argument as in the proof of Theorem 2 gives $\mathrm{Rob(DualCap)} = \beta$ once feasibility is established. In this band, $R^*(m,\varepsilon) \geq 1/\varepsilon$, so

$$\begin{aligned} \frac{\mathrm{Rob(DualCap)}}{R^*(m,\varepsilon)} &\leq 1 + \frac{m\varepsilon}{m-2}\kappa(m\varepsilon), \\ \mathrm{Rob(DualCap)} - R^*(m,\varepsilon) &\leq \frac{m}{m-2}\kappa(m\varepsilon). \end{aligned} \tag{58}$$

For $1 \leq c \leq 2$, we have

$$0 \leq \kappa(c) \leq \frac{c-1}{3c} \leq \frac{1}{6}, \tag{59}$$

since $c^2+2c-2-6(c-1)^2 = (2-c)(5c-4) \geq 0$. Finally, $\varepsilon > 1/(m-1)$ implies $1/(m-2) < \varepsilon/(1-\varepsilon)$, giving

$$\begin{aligned} \frac{m\varepsilon}{6(m-2)} &= \frac{\varepsilon}{6} + \frac{\varepsilon}{3(m-2)} \leq \frac{\varepsilon}{6} + \frac{\varepsilon^2}{3(1-\varepsilon)}, \\ \frac{m}{6(m-2)} &= \frac{1}{6} + \frac{1}{3(m-2)} \leq \frac{1}{6} + \frac{\varepsilon}{3(1-\varepsilon)}. \end{aligned} \tag{60}$$

Combining Eqs. (58) to (60) proves Eq. (57). □

Interpretation. Theorem 14 gives a multiplicative factor of $1 + \varepsilon/6 + O(\varepsilon^2)$ and an additive gap of at most $1/6 + \varepsilon/(3(1-\varepsilon)) = 1/6 + O(\varepsilon)$ in this parameter range. Compared with Theorem 6, both the coefficient of the first-order term in the multiplicative bound and the constant term in the additive bound improve from 1 to $1/6$.

Before processing the input, we may choose the smallest applicable coefficient among Eqs. (1), (36) and (56). Each candidate has its own feasibility guarantee, so this choice retains the original general guarantee and the refinements in their respective bands, with $O(m)$ time per job and $O(m)$ memory.

C.3.2 A Sufficient Condition for Feasibility

We derive a sufficient condition for feasibility using the load bounds in Lemmas 12 and 13.

Lemma 15 (Sufficient condition for feasibility)**.** *Suppose $m \geq 4$, $0 < \varepsilon < 1$, $D \coloneqq (m-1)\varepsilon - 1 > 0$, and $(m+2)/2 \leq \beta < m$. With $\theta \coloneqq (\beta-1)/m$, set $A \coloneqq 1-(1-\theta)/(m-2)$ and $C \coloneqq (m-3)/2$, and define*

$$G \coloneqq 1 - (1-\varepsilon)A - \varepsilon^2(1+C), \tag{61}$$

$$\Gamma_{m,\varepsilon}(\beta) \coloneqq D(\beta(1+\varepsilon-A) - m\varepsilon(1-A)) + G\beta - D\varepsilon(1+C) - (m-1)G. \tag{62}$$

If $G > 0$ and $\Gamma_{m,\varepsilon}(\beta) \geq 0$, then Algorithm 1 is feasible on every input.

Proof. We prove feasibility by induction on the number of processed jobs. Assume that the first $t-1$ jobs have been assigned. Both caps hold after these assignments by the assignment rule, and the inequalities in Lemmas 12 and 13 hold by these lemmas. Zero-size jobs are feasible, so consider job t with $p_t > 0$ and predicted machine h_t. Suppose, for a contradiction, that no machine satisfies both caps for job t. Let $K \coloneqq \{i : \ell_i(t-1) + p_t > \beta B_t\}$ and $k \coloneqq |K|$. The predicted machine satisfies its consistency cap, and a least-loaded machine satisfies the robustness cap, so $h_t \in K$ and $1 \le k \le m-1$. The counting argument in Eq. (40) gives $\beta < 1+(m-1)/k$. Since $\beta \ge (m+2)/2$, this forces $K = \{h_t\}$.

If some $j \neq h_t$ had $S_j(t) = H_t$, then

$$\ell_j(t-1) + p_t \le \beta B_t < \ell_{h_t}(t-1) + p_t \le (1+\varepsilon)H_t = S_j(t) + \varepsilon H_t. \tag{63}$$

Thus, machine j satisfies both caps, contradicting our assumption that no machine does. Therefore, h_t is the unique machine attaining the maximum predicted load H_t. Choose $j \neq h_t$ with $S_j(t) = s_t \coloneqq \max_{i \neq h_t} S_i(t)$. Applying Eq. (46) in Lemma 12 at time $t-1$, using $S_{h_t}(t-1) + p_t = H_t$ and $H_{-h_t}(t-1) = H_{-h_t}(t) = s_t$, implies

$$\beta B_t < \ell_{h_t}(t-1) + p_t \le S_{h_t}(t-1) + p_t + \varepsilon H_{-h_t}(t-1) = H_t + \varepsilon s_t. \tag{64}$$

Rearranging gives

$$s_t > \frac{\beta B_t - H_t}{\varepsilon}. \tag{65}$$

Machine h_t violates the robustness cap and has $S_{h_t}(t) = H_t$, so $\ell_{h_t}(t-1) - S_{h_t}(t) > \beta B_t - p_t - H_t$. Every other machine satisfies the robustness cap and must therefore violate the consistency cap under our contradiction assumption, giving $\ell_i(t-1) - S_i(t) > \varepsilon H_t - p_t$ for $i \neq h_t$. Summing the lower bound for h_t and those for the other $m-1$ machines therefore gives

$$\begin{aligned} -p_t &= \sum_{i \in [m]} \left(\ell_i(t-1) - S_i(t)\right) \\ &> (\beta B_t - p_t - H_t) + (m-1)(\varepsilon H_t - p_t) \\ &= \beta B_t - m p_t + D H_t. \end{aligned} \tag{66}$$

Since $D > 0$, rearranging yields

$$H_t < \frac{(m-1)p_t - \beta B_t}{D}. \tag{67}$$

Let $U \coloneqq [m] \setminus \{h_t, j\}$. Since $S_i(t) = S_i(t-1)$ for every $i \in U$, Eq. (48) in Lemma 13 at time $t-1$ bounds the actual loads on every nonempty subset of U in terms of its predicted loads at time t as well. Each $i \in U$ satisfies $\ell_i(t-1) > S_i(t) + \varepsilon H_t - p_t$. Order the machines in U as $u_1, \dots, u_{m-2}$ so that $S_{u_1}(t) \ge \cdots \ge S_{u_{m-2}}(t)$. For each $q = 0, \dots, m-3$, summing the individual lower bounds over the first q machines gives $\sum_{v=1}^{q} \ell_{u_v}(t-1) \ge \sum_{v=1}^{q} S_{u_v}(t) - q(p_t - \varepsilon H_t)$, including $q = 0$ with empty sums. The remaining machines form a nonempty set whose largest predicted load is $S_{u_{q+1}}(t)$. Thus, Eq. (48) in Lemma 13 at time $t-1$ shows that their total actual load satisfies $\sum_{v=q+1}^{m-2} \ell_{u_v}(t-1) \ge \sum_{v=q+1}^{m-2} S_{u_v}(t) - (1-\theta) S_{u_{q+1}}(t)$. Adding these two bounds gives

$$\begin{aligned} \ell(U, t-1) &= \sum_{v=1}^{q} \ell_{u_v}(t-1) + \sum_{v=q+1}^{m-2} \ell_{u_v}(t-1) \\ &\ge \sum_{v=1}^{q} S_{u_v}(t) - q(p_t - \varepsilon H_t) + \sum_{v=q+1}^{m-2} S_{u_v}(t) - (1-\theta) S_{u_{q+1}}(t) \\ &= S(U, t) - (1-\theta) S_{u_{q+1}}(t) - q(p_t - \varepsilon H_t). \end{aligned} \tag{68}$$

Averaging Eq. (68) over $q = 0, \dots, m-3$, using $\sum_{q=0}^{m-3} S_{u_{q+1}}(t) = S(U, t)$ and $\sum_{q=0}^{m-3} q = (m-2)(m-3)/2$, gives

$$\begin{aligned} \ell(U, t-1) &\ge \frac{1}{m-2} \sum_{q=0}^{m-3} \left(S(U, t) - (1-\theta) S_{u_{q+1}}(t) - q(p_t - \varepsilon H_t) \right) \\ &= \left(1 - \frac{1-\theta}{m-2}\right) S(U, t) - \frac{m-3}{2}(p_t - \varepsilon H_t) \\ &= A(P_t - H_t - s_t) - C(p_t - \varepsilon H_t). \end{aligned} \tag{69}$$

The last equality uses the definitions of A, C and $S(U,t) = P_t - H_t - s_t$. Summing the actual loads on h_t, j, and U gives

$$\begin{aligned} P_t - p_t &= \ell_{h_t}(t-1) + \ell_j(t-1) + \ell(U, t-1) \\ &> (\beta B_t - p_t) + (s_t + \varepsilon H_t - p_t) + A(P_t - H_t - s_t) - C(p_t - \varepsilon H_t). \end{aligned} \tag{70}$$

Here, the inequality uses $\ell_{h_t}(t-1) > \beta B_t - p_t$, $\ell_j(t-1) > s_t + \varepsilon H_t - p_t$, and Eq. (69). Rearranging and using $1 - A > 0$ and $P_t \le mB_t$, we obtain

$$\begin{aligned} (m(1-A) - \beta)B_t + (1+C)p_t &\ge (1-A)P_t - \beta B_t + (1+C)p_t \\ &> (1-A)s_t + (\varepsilon(1+C) - A)H_t. \end{aligned} \tag{71}$$

Since $1 - A > 0$, substituting Eq. (65) and multiplying by ε gives

$$(\beta(1 + \varepsilon - A) - m\varepsilon(1-A))B_t - \varepsilon(1+C)p_t < GH_t. \tag{72}$$

Substituting Eq. (67) into Eq. (72), using $G > 0$, gives

$$\begin{aligned} & (\beta(1+\varepsilon - A) - m\varepsilon(1-A))B_t - \varepsilon(1+C)p_t < \frac{G}{D}((m-1)p_t - \beta B_t) \\ \iff \quad & (D(\beta(1+\varepsilon - A) - m\varepsilon(1-A)) + G\beta)B_t < (D\varepsilon(1+C) + (m-1)G)p_t \\ \iff \quad & \Gamma_{m,\varepsilon}(\beta)B_t < (D\varepsilon(1+C) + (m-1)G)(p_t - B_t) \le 0. \end{aligned} \tag{73}$$

The equivalences use $D > 0$ and the definition of $\Gamma_{m,\varepsilon}(\beta)$, while the final inequality uses $D\varepsilon(1 + C) + (m-1)G > 0$ and $p_t \le B_t$. Since $B_t \ge p_t > 0$, this implies $\Gamma_{m,\varepsilon}(\beta) < 0$, contradicting the assumption $\Gamma_{m,\varepsilon}(\beta) \ge 0$. Therefore, some machine satisfies both caps for job t, so the algorithm can assign it. Starting from the empty prefix, this completes the induction and proves feasibility on every input. □

C.3.3 Feasibility of the Refined Coefficient

It remains to verify that the coefficient in Eq. (56) satisfies the sufficient condition in Lemma 15.

Lemma 16 (Feasibility of the refined coefficient)**.** *Suppose $m \ge 5$ and $1/(m-1) < \varepsilon \le 2/(m+2)$. With β chosen as in Eq. (56), Algorithm 1 satisfies $\mathcal{A}_t \neq \varnothing$ at every arrival.*

Proof. By Lemma 15, it suffices to verify its parameter assumptions and the inequalities $G > 0$ and $\Gamma_{m,\varepsilon}(\beta) \ge 0$ for the chosen coefficient. Set $c \coloneqq m\varepsilon$, $L \coloneqq 1/\varepsilon = m/c$, and $d \coloneqq m\kappa(c)/(m-2)$, so $\beta = L + d$. The assumed parameter range implies $1 < c < 2$. By Eq. (59), we have $0 < d \le m/(6(m-2)) < 1$. Since $L < m - 1$ and $L \ge (m+2)/2$, it follows that $(m+2)/2 \le \beta < m$ and $\theta \ge 1/2$. Also, $D = (m-1)\varepsilon - 1 > 0$.

Define

$$A_0(c) \coloneqq \frac{c^2}{2} + c - 1, \qquad N_0(c) \coloneqq \frac{(c-1)^3}{c}, \tag{74}$$

$$a_m(c) \coloneqq A_0(c) + \frac{(c-1)^2 + 1 - 2c/m}{2(m-2)}, \tag{75}$$

$$N_m(c) \coloneqq \frac{m}{m-2}\frac{(c-1)((c-1)^2 - (c/m)^2)}{c}. \tag{76}$$

We have $\kappa(c) = N_0(c)/A_0(c)$, and substitution into Eq. (62) gives

$$\begin{aligned} m\Gamma_{m,\varepsilon}(L+d) &= -N_m(c) + a_m(c)d - \varepsilon d^2 \\ &\ge d(a_m(c) - A_0(c) - \varepsilon d) \\ &= \frac{d}{m-2}\left(\frac{(c-1)^2 + 1 - 2c/m}{2} - c\kappa(c)\right). \end{aligned} \tag{77}$$

Here, the inequality uses $N_m(c) \le mN_0(c)/(m-2) = A_0(c)d$. Writing $u \coloneqq c - 1 \in (0,1)$ and using $m \ge 5$ and $c\kappa(c) \le (c-1)/3$, we obtain

$$\begin{aligned} \frac{(c-1)^2 + 1 - 2c/m}{2} - c\kappa(c) &\ge \frac{u^2+1}{2} - \frac{u+1}{5} - \frac{u}{3} \\ &= \frac{1}{2}\left(u - \frac{8}{15}\right)^2 + \frac{71}{450} > 0. \end{aligned} \tag{78}$$

By Eqs. (77) and (78) and $d > 0$, we have $\Gamma_{m,\varepsilon}(\beta) > 0$.

It remains to verify $G > 0$. Regarding G in Eq. (61) as a function of β, it decreases as d increases. Define $g(c) \coloneqq -c^2/2 + c + 1 - 1/c$. Direct calculation gives

$$mG(L+1) - g(c) = \frac{(m-2)c^3 - 2m(c-1)(c-2)}{2cm(m-2)} \geq 0. \tag{79}$$

The function g is concave on $[1, 2]$ and equals $1/2$ at both endpoints, so $g(c) \geq 1/2$. Since $d < 1$, we conclude

$$mG(L+d) \geq mG(L+1) \geq g(c) \geq \frac{1}{2}. \tag{80}$$

All conditions of Lemma 15 are satisfied, so $\mathcal{A}_t \neq \varnothing$ at every arrival. □

D Deferred Proofs for Scheduling with Rejection (Section 4)

D.1 Consistency and Robustness

We prove Theorem 7 for any reference scheduler satisfying the stated conditions. Write $B_t \coloneqq C^{\mathcal{B}}(I_t)$ for its full-input virtual cost, which is nondecreasing and satisfies $B_t \leq \rho_m \mathrm{OPT}(I_t)$. For brevity, set $X_t \coloneqq M_t^{\mathrm{P}} + A_t^{\mathrm{P}}$ and $D \coloneqq \max\{m, K\}$.

Proof. We first record the properties of the candidate increment c_t. If the reference rejects, then $c_t = r_t \leq p_t$. If it accepts, then $0 \leq c_t \leq p_t \leq m r_t$. Moreover, the previously selected reference actions together with the current candidate form a subset of the full reference schedule, so

$$0 \leq c_t \leq p_t, \qquad c_t/m \leq r_t, \qquad Q_{t-1}^{\mathrm{B}} + c_t \leq B_t. \tag{81}$$

Comparison with the predictions. Let $S_i(t) \coloneqq \sum_{j \leq t : h_j = i} p_j$ and $\widehat{R}_t \coloneqq \sum_{j \leq t : h_j = 0} r_j$, so $\widehat{C}_t = \max_i S_i(t) + \widehat{R}_t$. We first show that, at every prefix,

$$L_i^{\mathrm{P}}(t) + A_t^{\mathrm{P}} + q_i(t) \leq S_i(t) + \widehat{R}_t \quad \text{for all } i \in [m]. \tag{82}$$

For an acceptance prediction, the prediction-based action increases $L_{h_t}^{\mathrm{P}}$ by p_t, whereas the reference action increases q_{h_t} by $c_t \leq p_t$. In either case, S_{h_t} increases by p_t. For a rejection prediction, the prediction-based action increases A^{P} by $w_t \leq r_t$, whereas the reference action increases every q_i by $c_t/m \leq r_t$. These increases are covered by the increase r_t in $\widehat{R}_t$. Together, the two cases establish Eq. (82) by induction.

Whenever q_i increases positively, the selection rules also ensure

$$(K-1)Q_t^{\mathrm{B}} + q_i(t) \leq \widehat{C}_t. \tag{83}$$

For an acceptance prediction, add $q_{h_t}(t-1)$ to Eq. (12) and use Eq. (82) to obtain

$$\begin{aligned} (K-1)Q_t^{\mathrm{B}} + q_{h_t}(t) &= (K-1)Q_{t-1}^{\mathrm{B}} + Kc_t + q_{h_t}(t-1) \\ &\leq L_{h_t}^{\mathrm{P}}(t-1) + A_{t-1}^{\mathrm{P}} + p_t + q_{h_t}(t-1) \\ &\leq S_{h_t}(t) + \widehat{R}_t \leq \widehat{C}_t. \end{aligned} \tag{84}$$

For a rejection prediction, every coordinate satisfies

$$\begin{aligned} (K-1)Q_t^{\mathrm{B}} + q_i(t) &= (K-1)(Q_{t-1}^{\mathrm{B}} + c_t) + q_i(t-1) + c_t/m \\ &\leq (K-1)Q_{t-1}^{\mathrm{B}} + Kc_t + \max_j q_j(t-1) \\ &\leq X_{t-1} + w_t \leq \widehat{C}_t. \end{aligned} \tag{85}$$

The last inequality follows from Eq. (82) at time $t-1$ and $w_t \leq r_t$.

Fix a prefix I_t and any s values among $q_1(t), \ldots, q_m(t)$. Order the positive values among them by their last increase times as $q_{i_1}(t), \ldots, q_{i_u}(t)$, where $u \leq s$, and break simultaneous ties arbitrarily.

Write $P_0 = 0$ and $P_j \coloneqq \sum_{a=1}^{j} q_{i_a}(t)$. At the last increase of q_{i_j}, the total Q^{B} already includes all these first j final values, including those updated simultaneously. Using Eq. (83) and monotonicity of $\widehat{C}_t$ gives

$$(K-1)P_{j-1} + Kq_{i_j}(t) \le \widehat{C}_t, \qquad P_j \le \frac{\widehat{C}_t + P_{j-1}}{K}. \tag{86}$$

By induction, the sum of the chosen s values is $P_u \le \widehat{C}_t \sum_{a=1}^{s} K^{-a}$. Choose i^* attaining M_t^{P} and apply this bound to the $m-1$ values $q_i(t)$ with $i \ne i^*$. Since $\sum_i q_i(t) = Q_t^{\mathrm{B}}$, Eq. (82) implies

$$\begin{aligned} C^{\mathrm{CBS}}(I_t) &\le X_t + Q_t^{\mathrm{B}} \\ &\le S_{i^*}(t) + \widehat{R}_t + \sum_{i \ne i^*} q_i(t) \\ &\le (1 + g_m(K))\widehat{C}_t = (1+\varepsilon)\widehat{C}_t. \end{aligned} \tag{87}$$

Robustness from the potential. Consider the potential $\Phi_t \coloneqq \max\{X_t + Q_t^{\mathrm{B}}, KQ_t^{\mathrm{B}}\}$. We show that each job satisfies

$$0 \le \Phi_t - \Phi_{t-1} \le \min\{p_t, Dr_t\}. \tag{88}$$

Nonnegativity follows from the monotonicity of X_t and Q_t^{B}. In the following cases, write X, Q for their values before the current arrival, and X', Q' for their values after processing it.

For an acceptance prediction assigned to its predicted machine, $Q' = Q$ and $X' - X \le p_t$. If $X' = X$, the potential is unchanged. Otherwise, failure of Eq. (12) gives $X' = L_{h_t}^{\mathrm{P}}(t-1) + A_{t-1}^{\mathrm{P}} + p_t < (K-1)Q + Kc_t$. Since the old potential is at least KQ, its increase is also at most Kc_t.

If the reference action is selected, then $X' = X$ and $Q' = Q + c_t$, so the potential increases by at most Kc_t. Its new branch $X + Q + c_t$ is at most the old potential plus p_t, by Eq. (81). For an acceptance prediction, the selection rule bounds the other branch by

$$K(Q + c_t) \le Q + L_{h_t}^{\mathrm{P}}(t-1) + A_{t-1}^{\mathrm{P}} + p_t \le X + Q + p_t \le \Phi_{t-1} + p_t. \tag{89}$$

For a rejection prediction, the corresponding bound is

$$K(Q + c_t) \le X + Q + w_t - \max_i q_i(t-1) \le \Phi_{t-1} + p_t. \tag{90}$$

Thus the potential increase is at most both p_t and Kc_t in these cases. If the reference rejects, $c_t = r_t$ gives the bound Dr_t; if it accepts, $p_t \le mr_t \le Dr_t$ gives that bound.

Finally, for a rejection prediction handled by the prediction-based action, only X increases, by $w_t \le \min\{p_t, r_t\}$. This proves Eq. (88) in all cases. For any offline solution on I_t with accepted makespan C^* and rejection cost R^*, summing the increments over its accepted and rejected jobs gives

$$C^{\mathrm{CBS}}(I_t) \le \Phi_t \le mC^* + DR^* \le D(C^* + R^*). \tag{91}$$

Taking an optimal solution establishes the robustness bound $D = \max\{m, K\}$.

Robustness from the reference schedule. The quantity X_t increases only when the prediction-based action is selected. For an acceptance prediction that increases M^{P}, failure of the reference condition gives

$$X_t + Q_{t-1}^{\mathrm{B}} < K(Q_{t-1}^{\mathrm{B}} + c_t) \le KB_t. \tag{92}$$

For a rejection prediction that increases A^{P}, failure of its condition gives

$$X_t < (K-1)Q_{t-1}^{\mathrm{B}} + Kc_t + \max_i q_i(t-1) \le K(Q_{t-1}^{\mathrm{B}} + c_t) \le KB_t, \tag{93}$$

where $\max_i q_i(t-1) \le Q_{t-1}^{\mathrm{B}}$. Apply these bounds at the last increase of X and use monotonicity of B_t to obtain $X_t \le KB_t$ at every prefix. If X has never increased, this bound is immediate from $X_t = 0$. Since the selected reference actions form a subset of the full reference schedule, $Q_t^{\mathrm{B}} \le B_t$, and hence

$$C^{\mathrm{CBS}}(I_t) \le X_t + Q_t^{\mathrm{B}} \le (K+1)B_t \le \rho_m(K+1)\mathrm{OPT}(I_t). \tag{94}$$

Combining this bound with Eqs. (87) and (91) proves Eq. (14). Perfect predictions satisfy $\widehat{C}_n = \mathrm{OPT}(I)$, giving the stated consistency bound; arbitrary predictions satisfy both robustness bounds. □

D.2 General Robustness Lower Bound

We give the full proof of Theorem 9.

Proof. Fix a deterministic $(1+\varepsilon)$-consistent algorithm A and suppose its robustness is $\rho < K$. Since g_m is continuous and strictly decreasing on $(1,\infty)$, with $g_m(K) = \varepsilon < 1$, we can choose b sufficiently close to K from below so that

$$\max\{1,\rho\} < b < K, \qquad \varepsilon < g_m(b) < 1. \tag{95}$$

For $j = 1,\dots,m-1$, present a job with processing time $p_j = 1$, penalty $r_j = b^{-j}$, and prediction $h_j = j$. Set $\Pi_0 = 0$ and $\Pi_j \coloneqq \sum_{a=1}^{j} b^{-a}$. On each prefix I_j, rejecting all jobs costs $\Pi_j \le g_m(b) < 1$, whereas accepting any job incurs makespan at least one. Hence $\mathrm{OPT}(I_j) = \Pi_j$.

Suppose the first $j-1$ jobs have all been rejected. Accepting job j would give cost at least $1+\Pi_{j-1}$, and the geometric sum satisfies

$$1 + \Pi_{j-1} = b\Pi_j > \rho\,\mathrm{OPT}(I_j). \tag{96}$$

Because the input may end at this prefix, ρ-robustness forces the algorithm to reject job j as well. The input length is unknown, so the decision on this prefix is unchanged when more jobs follow. Induction therefore forces rejection of all $m-1$ jobs, with total penalty $\Pi_{m-1} = g_m(b)$.

Finally, append a job with $p_m = 1$, $r_m = m+2$, and $h_m = m$. Following all predictions places one unit job on each machine and costs one. Every feasible solution either accepts the final job and has makespan at least one, or rejects it and pays $m+2$, so $\mathrm{OPT}(I_m) = 1$ and the predictions are perfect. If A accepts the final job, its cost is at least $1 + g_m(b) > 1+\varepsilon$. If it rejects the job, its cost is at least $m+2 > 1+\varepsilon$. Both choices contradict $(1+\varepsilon)$-consistency. Thus every such algorithm has robustness at least K, proving the lower bound on $R^*_{\mathrm{rej}}(m,\varepsilon)$. □

D.3 Optimality and Near-Optimality

We prove Theorem 10 and derive the RTP bound in Eq. (19).

Proof. By Theorem 7, CBS satisfies the consistency requirement defining $R^*_{\mathrm{rej}}(m,\varepsilon)$. Together with Theorem 9, this gives

$$K \le R^*_{\mathrm{rej}}(m,\varepsilon) \le \mathrm{Rob}(\mathrm{CBS}) \le \beta_{\mathrm{rej}}. \tag{97}$$

Dividing by $R^*_{\mathrm{rej}}(m,\varepsilon) \ge K$ proves Eq. (17). Since g_m is strictly decreasing, $\varepsilon = g_m(K) \le g_m(m)$ if and only if $K \ge m$. In this case, $\beta_{\mathrm{rej}} \le \max\{m,K\} = K$, so every inequality in Eq. (97) is an equality. The finite geometric sum gives $g_m(m) = (1-m^{-(m-1)})/(m-1)$, establishing the stated threshold.

It remains to prove the uniform comparison for RTP. Write $a \coloneqq 1 - 1/m$ and $\tau \coloneqq \varepsilon/(1+\varepsilon)$, so $a \ge 1/2$, $0 < \tau < 1/2$, and $\rho_m = 1 + a\varphi$. The identity $\varepsilon(K-1) = 1 - K^{-(m-1)}$ yields

$$\frac{1+1/\varepsilon}{K} = 1 + \frac{1}{\sum_{j=1}^{m-1} K^j} \le 1 + \frac{1}{m-1} = \frac{1}{a}. \tag{98}$$

Consequently, $K \ge a(1+1/\varepsilon)$ and $1/K \le \tau/a$. Using the reference-based robustness bound, we obtain

$$\frac{\mathrm{Rob}(\mathrm{CBS})}{R^*_{\mathrm{rej}}(m,\varepsilon)} \le \frac{\beta_{\mathrm{rej}}}{K} \le \rho_m\left(1+\frac{1}{K}\right) \le (1+a\varphi)\left(1+\frac{\tau}{a}\right). \tag{99}$$

Finally,

$$(1+\varphi)(1+\tau) - (1+a\varphi)\left(1+\frac{\tau}{a}\right) = (1-a)\left(\varphi - \frac{\tau}{a}\right) \ge 0, \tag{100}$$

because $\tau/a < 1 < \varphi$. Substituting $\tau = \varepsilon/(1+\varepsilon) < 1/2$ into Eqs. (99) and (100) proves Eq. (19). □

E Parameter Choice and Implementation for Scheduling with Rejection

E.1 Exact Decisions without Computing the Root

The parameter K need not be computed explicitly to execute Algorithm 2. Both selection conditions can be written as $K(Q + c_t) \le T_t$, where $Q \coloneqq Q^{\mathrm{B}}_{t-1}$ and

$$T_t \coloneqq \begin{cases} L^{\mathrm{P}}_{h_t}(t-1) + A^{\mathrm{P}}_{t-1} + p_t + Q, & h_t \in [m], \\ M^{\mathrm{P}}_{t-1} + A^{\mathrm{P}}_{t-1} + w_t + Q - \max_i q_i(t-1), & h_t = 0. \end{cases} \tag{101}$$

Here $T_t \ge 0$, since $\max_i q_i(t-1) \le Q$. If $Q + c_t = 0$, the reference condition holds. Otherwise, set $u_t \coloneqq T_t/(Q + c_t)$. For $u_t \le 1$, the condition fails because $K > 1$. For $u_t > 1$, strict monotonicity of g_m gives

$$K(Q + c_t) \le T_t \quad \Longleftrightarrow \quad K \le u_t \quad \Longleftrightarrow \quad g_m(u_t) \le \varepsilon. \tag{102}$$

Thus, evaluating a finite geometric sum produces exactly the same decisions, including the rule that equality favors the reference action. All guarantees for the original algorithm remain unchanged.

For rational input data and rational ε, this selection test uses only rational arithmetic, even if K is irrational. The sum $g_m(u_t)$ can be evaluated in $O(m)$ arithmetic operations by successively multiplying by $1/u_t$. Together with state maintenance, this requires $O(m)$ arithmetic operations per job and $O(m)$ stored values beyond the reference scheduler, and the same total bounds hold with RTP. These are arithmetic-operation and storage counts, rather than bit-complexity bounds.

E.2 Near-Optimality with a Simple Parameter

Alternatively, one can replace the root by the explicit coefficient $K_0 \coloneqq 1 + 1/\varepsilon$. Let CBS_{K_0} denote the algorithm with K_0 in place of K and the same reference scheduler. The proof in Appendix D.1 applies to any coefficient greater than one, giving consistency at most $1 + g_m(K_0)$ and robustness at most $\min\{\max\{m, K_0\}, \rho_m(K_0 + 1)\}$. The geometric sum satisfies

$$g_m(K_0) = \varepsilon \left(1 - \left(\frac{\varepsilon}{1+\varepsilon} \right)^{m-1} \right) < \varepsilon, \tag{103}$$

so CBS_{K_0} preserves $(1+\varepsilon)$-consistency throughout $0 < \varepsilon < 1$.

Corollary 17 (Near-optimality with a simple parameter)**.** *Suppose* $0 < \varepsilon < 1$ *and* $\varepsilon \le 1/(m-1)$. *Then* CBS_{K_0} *is* $(1+\varepsilon)$*-consistent and satisfies*

$$0 \le \mathrm{Rob}(\mathrm{CBS}_{K_0}) - R^*_{\mathrm{rej}}(m, \varepsilon) \le \varepsilon^{m-2}, \tag{104}$$

$$1 \le \frac{\mathrm{Rob}(\mathrm{CBS}_{K_0})}{R^*_{\mathrm{rej}}(m, \varepsilon)} \le 1 + \varepsilon^{m-1}. \tag{105}$$

Proof. Consistency follows from Eq. (103). Since $\varepsilon \le 1/(m-1)$, we have $K_0 \ge m$ and hence $\mathrm{Rob}(\mathrm{CBS}_{K_0}) \le K_0$. The root identity $\varepsilon(K-1) = 1 - K^{-(m-1)}$, together with $K \ge 1/\varepsilon$, yields

$$0 < K_0 - K = \frac{K^{-(m-1)}}{\varepsilon} \le \varepsilon^{m-2}. \tag{106}$$

Because CBS_{K_0} satisfies the required consistency and Theorem 9 gives $R^*_{\mathrm{rej}}(m, \varepsilon) \ge K$, we obtain

$$0 \le \mathrm{Rob}(\mathrm{CBS}_{K_0}) - R^*_{\mathrm{rej}}(m, \varepsilon) \le K_0 - K \le \varepsilon^{m-2}. \tag{107}$$

Dividing by $R^*_{\mathrm{rej}}(m, \varepsilon) \ge K \ge 1/\varepsilon$ proves Eq. (105). □

For each fixed m, the multiplicative bound tends to one as $\varepsilon \to 0^+$. For $m \ge 3$, the additive bound also tends to zero. These refined comparisons require $\varepsilon \le 1/(m-1)$, whereas the consistency guarantee for K_0 holds throughout $0 < \varepsilon < 1$.

F Detailed Experimental Setup

This section specifies the settings used in Section 5.

Input instances. Each instance has $m = 10$ machines and $n = 100$ jobs, consisting of 95 small jobs and five large jobs. In both settings, small processing times are sampled independently and uniformly from $\{1, \dots, 12\}$ and large processing times from $\{50, \dots, 100\}$. All jobs are then uniformly shuffled to determine their arrival order. The two settings use identical processing times and arrival orders for each input seed. For the rejection setting, each job independently receives a factor $f_j \sim \mathrm{Unif}(0.25, 4)$ and penalty $r_j = \max\{1, \mathrm{round}(f_j p_j / m)\}$, where rounding is to the nearest integer.

Prediction generation We start from an offline optimal schedule. We partition the arrival sequence into consecutive blocks by independently ending the current block after each job (except the last) with probability 0.1. Each block is selected for modification independently with probability q. For each selected block, we sample a target machine z uniformly from $[m]$. Without rejection, every job in a selected block is predicted to be assigned to z. With rejection, each job accepted by the optimal schedule is independently predicted to be rejected with probability $1/2$ and assigned to z otherwise. Each job rejected by the optimal schedule is predicted to be assigned to z, because predicting rejection would leave its optimal action unchanged and hence would not perturb that job. We use $q \in \{0, 0.01, 0.025, 0.05, 0.075, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.6, 0.8, 1\}$. Random draws are shared across values of q, and the same prediction is used for all ε.

Aggregation. We plot mean normalized cost $C/\mathrm{OPT}(I)$ in bins of normalized prediction error $E/\mathrm{OPT}(I)$, using bin widths 0.1 without rejection and 0.25 with rejection. Shaded regions show one standard error, clustered by input instance to account for multiple prediction seeds per instance. The horizontal baselines show the mean normalized costs of List Scheduling and RTP.

Implementation details. All experiments are implemented in Python 3.14.6. We compute an offline optimal schedule for each instance using the OR-Tools CP-SAT solver (Perron & Didier), running the solver until optimality is certified. We use OR-Tools version 9.15.6755 and NumPy version 2.5.3.

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.

G Additional Experiments

We examine the effects of problem size, the proportion of large jobs, rejection penalties, and prediction generation. Each experiment changes one aspect of the main setting in Appendix F. All other settings are unchanged, including the processing-time distributions, random arrival order, 50 input seeds, 200 prediction seeds per input, and the values of ε used in Section 5. We use the same normalization, binning, and standard-error calculation as in Fig. 1.

G.1 Problem Size

We increase the number of machines and jobs to $m = 20$ and $n = 200$, using 190 small jobs and ten large jobs. This preserves both $n/m = 10$ and the proportion of large jobs at 5%. Fig. 2 shows the results without and with rejection.

G.2 Proportion of Large Jobs

We keep $m = 10$ and $n = 100$ and reduce the number of large jobs from five to three, leaving 97 small jobs. Fig. 3 shows the results without and with rejection.

G.3 Rejection Penalties and Prediction Generation

We consider two further variants with rejection, both using $m = 10$, $n = 100$, and five large jobs. In the first variant, we retain the penalty definition in Appendix F but sample $f_j \sim \mathrm{Unif}(0.25, 8)$ instead of $\mathrm{Unif}(0.25, 4)$, increasing the range of rejection penalties. In the second variant, we retain the original penalty distribution but remove the random rejection predictions for jobs accepted by the optimal schedule. Within selected blocks, every job is predicted to be assigned to the randomly chosen target machine z, regardless of whether the optimal schedule accepts or rejects it. Thus,

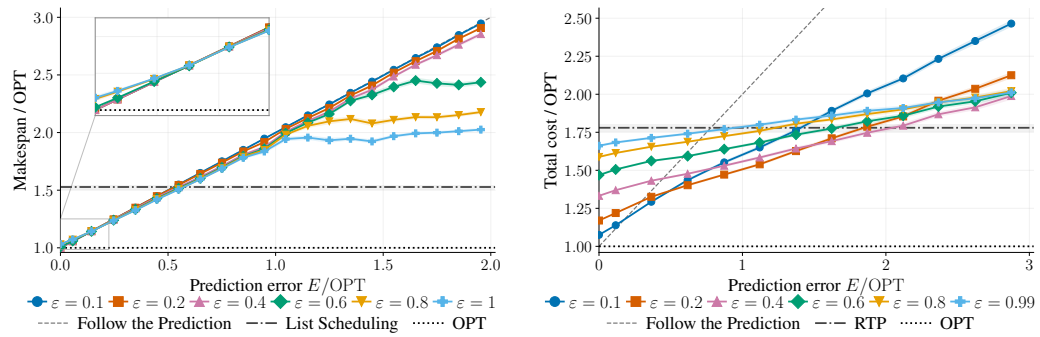

Figure 2: Results for $m = 20$, $n = 200$, and ten large jobs, without rejection (**Left**) and with rejection (**Right**). Shaded regions show one standard error over 50 input instances.

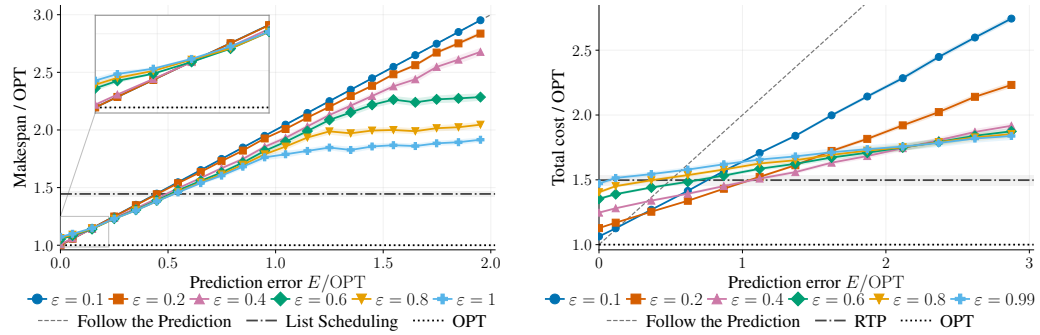

Figure 3: Results for $m = 10$, $n = 100$, and three large jobs, without rejection (**Left**) and with rejection (**Right**). Shaded regions show one standard error over 50 input instances. On the left, Follow the Prediction nearly coincides with the $\varepsilon = 0.1$ curve over the plotted range.

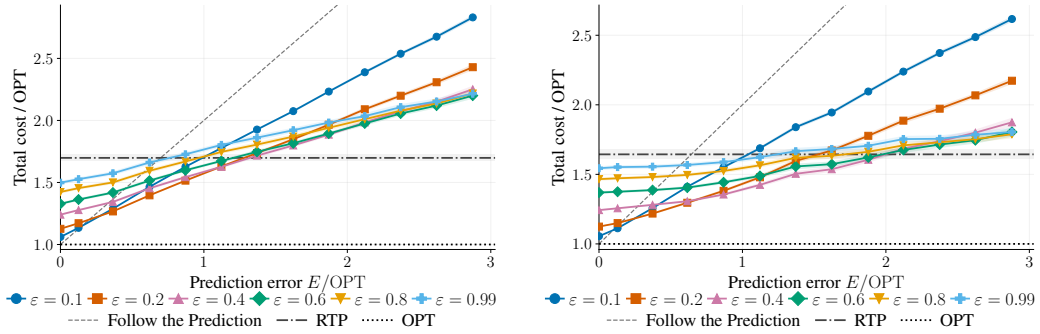

Figure 4: Results with rejection for $m = 10$, $n = 100$, and five large jobs: penalty factors from $\mathrm{Unif}(0.25, 8)$ (**Left**) and predictions assigning every job in selected blocks to z (**Right**). Shaded regions show one standard error over 50 input instances.

this variant never predicts rejection for jobs accepted by the optimal schedule. Predictions outside selected blocks remain unchanged, and the block construction and sampling of z are the same as in the main setting. Fig. 4 shows both variants.

G.4 Discussion

The additional results illustrate consistency: with accurate predictions, small values of ε yield costs close to the offline optimum. As prediction error increases, moderate and large values of ε limit the degradation relative to Follow the Prediction, reflecting robustness. The gradual change in performance with prediction error is also consistent with smoothness.

Smaller values of ε favor accurate predictions, whereas larger values generally improve performance with inaccurate predictions at the expense of low-error performance. These qualitative trends are consistent with the main results in Section 5.