A Switching Framework for Online Interval Scheduling with Predictions

Antonios Antoniadis[1], **Ali Shahheidar**[2], **Golnoosh Shahkarami**[3], **Abolfazl Soltani**[2]

[1]University of Twente
[2]Sharif University of Technology
[3]Max Planck Institut für Informatik, Universität des Saarlandes
a.antoniadis@utwente.nl, {alishahheidar98, absoltani02}@gmail.com, gshahkar@mpi-inf.mpg.de

Abstract

We study online interval scheduling in the irrevocable setting, where each interval must be immediately accepted or rejected upon arrival. The objective is to maximize the total length of accepted intervals while ensuring that no two accepted intervals overlap. We consider this problem in a learning-augmented setting, where the algorithm has access to (machine-learned) predictions. The goal is to design algorithms that leverage these predictions to improve performance while maintaining robust guarantees in the presence of prediction errors.

Our main contribution is the SemiTrust-and-Switch framework, which provides a unified approach for combining prediction-based and classical interval scheduling algorithms. This framework applies to both deterministic and randomized algorithms and captures the trade-off between consistency (performance under accurate predictions) and robustness (performance under adversarial inputs). Moreover, we provide lower bounds, proving the tightness of this framework in particular settings.

We further design a randomized algorithm that smoothly interpolates between prediction-based and robust algorithms. This algorithm achieves both robustness and smoothness–its performance degrades gracefully with the quality of the prediction.

Extended version — https://arxiv.org/abs/2511.16194

Introduction

The online interval scheduling problem is a well-studied model in algorithm design that captures the challenge of selecting a subset of non-overlapping time intervals with the goal of optimizing a given objective. Intervals arrive sequentially over time, and the algorithm must irrevocably decide, upon each arrival, whether to accept or reject the interval without knowledge of future inputs. Each interval typically represents a task, job, or request, and one natural objective is to maximize the total length of the accepted intervals. This model arises in a wide range of practical applications, including computing resource management, manufacturing systems, and real-time service scheduling. Additional domains include vehicle routing and satellite communication scheduling (Bar-Noy et al. 1999). Two surveys by (Kolen et al. 2007; Kovalyov, Ng, and Cheng 2007) provide comprehensive overviews of interval scheduling and its many application areas.

While classical online algorithms for interval scheduling, such as greedy approaches, offer strong worst-case guarantees, their performance can be overly pessimistic in practical settings. This is further highlighted by the fact that most lower-bound proofs rely on very carefully designed and fragile instances. In many real-world systems, partial information about future intervals is often available through historical trends or forecasting techniques, including machine-learned predictions. This motivates the study of *learning-augmented algorithms* (also known as algorithms with predictions), which aim to bridge the gap between worst-case online guarantees and the performance of offline algorithms by leveraging predicted information during the decision process.

As in the traditional online setting, the standard formulation of interval scheduling requires decisions, that is acceptance or rejection of intervals, to be made irrevocably. However, it is known that no online algorithm can achieve a constant competitive ratio for this problem in general (Lipton and Tomkins 1994; Long and Thakur 1993). To address this limitation, a relaxed model has been considered in which accepted intervals may be revoked prior to their deadlines, while rejections remain final. Several objective functions have been considered for the classic online interval scheduling problem, including maximizing the number of accepted intervals (unit-weight) and maximizing the total length of accepted intervals (proportional-weight). (Boyar et al. 2023) studies the unit-weight case, where the goal is to maximize the number of accepted intervals using predicted information. Moreover, (Karavasilis 2025) considers a proportional-weight objective, aiming to maximize the total length of accepted intervals in a revocable setting.

This leaves a natural gap: the irrevocable setting with proportional weights, which models many practical scenarios where job values depend on duration and decisions cannot be revoked. In this work, we fill this gap by providing a comprehensive analysis of learning-augmented interval scheduling under irrevocable decisions. We present both upper and lower bounds, characterizing the trade-offs between consistency and robustness in both deterministic and randomized

settings. Moreover, we propose a smooth randomized algorithm and provide both theoretical guarantees and experimental results demonstrating its effectiveness on real-world data.

Our Results and Techniques

A common design pattern in learning-augmented algorithms is to combine two strategies: one that follows the predictions and one that ignores them entirely. This allows the algorithm to adapt its behavior based on the quality of the prediction. The challenge, of course, is deciding how and when to rely on each component.

A natural approach is to switch between a predictive and a classic algorithm based on the reliability of the prediction. This technique has been used effectively in several minimization problems (Lykouris and Vassilvitskii 2021; Purohit, Svitkina, and Kumar 2018; Rohatgi 2020; Antoniadis et al. 2023; Wei 2020; Bamas et al. 2020; Bansal et al. 2020), where the algorithm initially relies on the prediction and transitions to a backup strategy when the prediction appears unreliable, potentially continuing to switch as new evidence emerges. While switching in maximization problems poses additional challenges, it has been explored in online bipartite matching with imperfect advice (Choo et al. 2024). In this paper, we apply the switching technique to irrevocable online interval scheduling, where early commitments may permanently block future high-value intervals. Therefore, the timing of the switch, as well as the behavior in the prediction-trusting phase, becomes particularly delicate.

While switching is a natural way to combine two strategies—one that trusts the prediction and one that ignores it—another powerful approach arises in the randomized setting: merging strategies probabilistically. We can define a randomized algorithm that selects between a predictive and a classic strategy based on a carefully chosen probability distribution. Leveraging this idea, we design a randomized algorithm that not only maintains robustness but also achieves *smoothness*—its performance degrades gracefully as the prediction quality deteriorates.

We now summarize our main results and techniques, which build on these merging principles.

The TRUST-AND-SWITCH Framework. We introduce a general mechanism that allows algorithms to balance predictive and non-predictive behavior. The TRUST-AND-SWITCH framework takes as input a prediction model and a classic algorithm, and it decides—based on an online evaluation of the prediction—when to switch from trusting the prediction to relying on the classic algorithm. The switch occurs at most once and is irrevocable. We show that this simple structure captures the trade-off between consistency and robustness. More specifically, it achieves 1-consistency, and if the algorithm used after the switch is θ-competitive, then the overall solution satisfies

$$|\textsc{Opt}| \leq \theta \cdot |\textsc{Alg}| + 2k,$$

where k is the maximum interval length.

The SEMITRUST-AND-SWITCH Framework. In addition to distinguishing between additive and multiplicative loss, our goal is to expose a tunable trade-off between consistency and robustness. This motivates a refined variant of the framework: instead of blindly following the prediction, the algorithm only follows the prediction for intervals whose length is below a threshold τ, and accepts longer intervals greedily. This hybrid approach improves robustness without sacrificing too much consistency—in effect, it interpolates between fully trusting and fully ignoring the prediction. More specifically, it achieves $(1 + \frac{k}{\tau})$-consistency, and if the algorithm used after the switch is θ-competitive, then the overall solution satisfies

$$|\textsc{Opt}| \leq \max(\theta, \Delta + 1) \cdot |\textsc{Alg}| + \tau,$$

where Δ is the ratio between the longest and shortest interval lengths.

Prediction Settings and Lower Bounds. Our frameworks work with a binary prediction model (denoted $\hat{\mathbf{O}}$), where predictions arrive online with the intervals and suggest whether to accept ($\hat{o}_i = 1$) or reject ($\hat{o}_i = 0$) interval I_i. We also establish lower bounds under a stronger, full-information setting (denoted $\hat{\mathbf{I}}$). Interestingly, our results show that having access to more detailed predictions, or receiving them offline, does not improve the achievable guarantees. In particular, for the TRUST-AND-SWITCH framework, we prove matching bounds for two-value instances: any $(1 + \alpha)$-consistent algorithm ALG$'$ with $\alpha < \frac{\lceil \Delta \rceil - 1}{\Delta}$ must incur a robustness loss of at least $\Delta \cdot |\textsc{Alg}'| + k$, even when given full predictions. This matches the guarantee of the framework for this class of instances, making the TRUST-AND-SWITCH framework optimal even when full predictions are available. Moreover, we provide a lower bound for our SEMITRUST-AND-SWITCH framework.

The SMOOTHMERGE Algorithm. We introduce a randomized algorithm, SMOOTHMERGE, that merges the behaviors of two strategies in a probabilistic manner. The algorithm simulates both strategies in parallel and independently. Upon the arrival of each interval, proceeds as follows: if the interval conflicts with previously accepted intervals, it is immediately rejected; otherwise, if both simulations agree on accepting (or rejecting), the algorithm follows that unanimous decision. When the simulations disagree, the algorithm accepts the interval with probabilities p_t and p_g, corresponding to the two strategies. This probabilistic merging yields both smoothness and robustness properties. Specifically, letting η denote the prediction error, we have:

$$\mathbb{E}[|\textsc{Alg}|] \geq \max\Big\{|\textsc{Opt}| \cdot (1-\eta) \cdot p_t \cdot (1-p_g), \frac{p_g - p_t p_g}{1 - p_t p_g} \cdot |\textsc{Greedy}|\Big\}.$$

Related Work

Online Interval Scheduling. The online interval scheduling problem was introduced by (Lipton and Tomkins 1994), who established the classical model in which intervals arrive sequentially and must be irrevocably accepted or rejected upon arrival. The goal is to maximize the total length of accepted, non-overlapping intervals. (Long and Thakur 1993) showed that there is no deterministic algorithm that can achieve a constant competitive ratio, and (Lipton and Tomkins 1994) showed that no randomized algorithm can achieve a competitive ratio better than $O(\log \Delta)$, where Δ is the ratio between the longest and shortest intervals, and they provided an $O((\log \Delta)^{1+\epsilon})$-competitive algorithm.

Several extensions of online interval scheduling have been studied. In the *revocable* setting, where accepted intervals can be later removed, improved competitive guarantees are possible (Borodin and Karavasilis 2023; Garay et al. 1997; Tomkins 1995). Another extension of online interval scheduling is the *any order* setting (Borodin and Karavasilis 2023), where the order of intervals can be arbitrary, and the *sorted order* setting (Lipton and Tomkins 1994), where the intervals arrive in order of their release time. In the *weighted* setting, where intervals have arbitrary weights, no deterministic or randomized algorithm can achieve a finite competitive ratio in general (Woeginger 1994; Canetti and Irani 1995). However, for weight functions tied to interval length, (Woeginger 1994) identified classes (e.g., benevolent functions) that admit finite guarantees, and (Seiden 2000) gave improved randomized bounds. For a comprehensive overview of the extensive literature on interval scheduling, we refer to the survey by (Kolen et al. 2007), which provides thorough coverage of algorithmic techniques and complexity results across various problem variants.

Learning-augmented Algorithms. Traditional worst-case analysis often fails to capture the practical performance of algorithms, motivating the study of beyond worst-case analysis (Roughgarden 2021). One prominent framework in this direction is that of learning-augmented algorithms (also known as algorithms with predictions), which incorporate machine-learned advice into decision-making (Mitzenmacher and Vassilvitskii 2022). (Lykouris and Vassilvitskii 2021) formalized this approach in the context of online caching, introducing the now-standard notions of consistency and robustness to measure how well an algorithm performs under both accurate and adversarial predictions. This framework has since been applied to a wide range of online problems, including caching (Lykouris and Vassilvitskii 2021; Antoniadis et al. 2023; Rohatgi 2020; Wei 2020), secretary and matching problems (Antoniadis et al. 2020; Dütting et al. 2021), online graph algorithms (Azar, Panigrahi, and Touitou 2022), and various scheduling problems (Purohit, Svitkina, and Kumar 2018; Bamas et al. 2020; Mitzenmacher 2020; Lattanzi et al. 2020; Antoniadis, Jabbarzade, and Shahkarami 2022). The scope of this area also extends beyond online problems. For an up-to-date list of papers in this field, we refer the reader to a curated online repository by (Lindermayr and Megow 2025).

Online Interval Scheduling with Predictions. Finally, the closest works to our own are (Boyar et al. 2023) and (Karavasilis 2025). (Boyar et al. 2023) study the unit-weight interval scheduling problem, where the objective is to maximize the number of accepted intervals, given predictions about the input sequence. They provide tight bounds on the competitive ratio achievable by deterministic algorithms. Their approach combines prediction-following with greedy acceptance when it is possible without a decrease in the profit of the planned solution. More recently, (Karavasilis 2025) considered proportional-weight interval scheduling in a revocable setting, where the objective is to maximize the total length of accepted intervals and decisions can be revoked before deadlines. This work focuses on binary predictions and develops algorithms that can adaptively modify their decisions based on observed prediction quality. Our work addresses a crucial gap by studying proportional-weight interval scheduling with irrevocable decisions.

Preliminaries

The input to the *online interval scheduling problem* consists of a set $\mathcal{I}$ of n intervals. Each interval $I_j \in \mathcal{I}$ is defined by a release time r_j and a deadline d_j, with length $l_j = d_j - r_j$. Intervals arrive online: the information about interval I_j, namely its release time, deadline, and length, becomes available to the algorithm only at time r_j. Upon arrival, the algorithm must make an irrevocable decision to either accept or reject the interval, without knowledge of future intervals. The accepted intervals must be non-overlapping, and the objective is to maximize the total length of the accepted subset.

Throughout this work, we assume that intervals arrive in non-decreasing order of their release times. Without loss of generality, we label the intervals $I_1, I_2, \ldots, I_n$ in order of arrival, so that $r_1 \leq r_2 \leq \cdots \leq r_n$.

We define the following key parameters:

- $k = \max_{I_j \in \mathcal{I}} l_j$: the maximum length of any interval.
- $\Delta = \frac{k}{\min_{I_j \in \mathcal{I}} l_j}$: the ratio between the maximum and minimum interval lengths.

Given an algorithm $\mathcal{A}$, we use $\mathcal{A}(\mathcal{I})$ to denote the outcome of $\mathcal{A}$ on instance $\mathcal{I}$, and $|\mathcal{A}(\mathcal{I})|$ to denote the total length of intervals selected by it.

We define notation for subsets of intervals based on their release times and deadlines. For timepoints $t_1, t_2 \in \mathbb{R}_{\geq 0}$, let $\mathcal{I}(t_1, t_2)$ denote the subset of intervals in $\mathcal{I}$ whose release times and deadlines satisfy constraints determined by t_1 and t_2, respectively. We use the symbol $\cdot$ to indicate that no constraint is applied to that component (release time or deadline). Superscripts $\leftarrow$ and $\rightarrow$ indicate the direction of the inequality: for a timepoint t, $t^{\leftarrow}$ denotes times at or before t (i.e., $\leq t$), and $t^{\rightarrow}$ denotes times strictly after t (i.e., $> t$). For example:

- $\mathcal{I}(t_1^{\leftarrow}, \cdot)$ is the set of intervals with release time $\leq t_1$ (no constraint on deadline),
- $\mathcal{I}(\cdot, t_2^{\leftarrow})$ is the set of intervals with deadline $\leq t_2$ (no constraint on release time),
- $\mathcal{I}(t_1^{\leftarrow}, t_2^{\rightarrow})$ is the set of intervals with release time $\leq t_1$ and deadline $> t_2$.

We denote the optimal offline algorithm by OPT. In particular, $|\textsc{Opt}(\mathcal{I})|$ denotes the total length of the optimal offline solution, and $|\textsc{Alg}(\mathcal{I})|$ denotes the total length achieved by the online algorithm under consideration.

The performance of an online algorithm is evaluated using the *competitive ratio*, which measures the worst-case performance relative to the optimal offline solution. The competitive ratio is defined as:

$$\text{competitive ratio} = \sup_{\mathcal{I}} \frac{|\textsc{Opt}(\mathcal{I})|}{|\textsc{Alg}(\mathcal{I})|},$$

where the supremum is taken over all possible input instances $\mathcal{I}$. An online algorithm is said to be *c-competitive*

if competitive ratio $\leq c$. In this case, the total length of intervals accepted by the algorithm is guaranteed to be at least $1/c$ of the total length of intervals accepted by the optimal offline solution for any input instance.

For randomized algorithms, the competitive ratio is defined based on the expected performance of the algorithm. Let $\mathbb{E}[\text{ALG}(\mathcal{I})]$ denote the expected total length of intervals accepted by the randomized online algorithm. The competitive ratio for randomized algorithms is defined as:

$$\text{competitive ratio (randomized)} = \sup_{\mathcal{I}} \frac{|\text{OPT}(\mathcal{I})|}{\mathbb{E}[\text{ALG}(\mathcal{I})]}.$$

A randomized online algorithm is said to be *c-competitive* if competitive ratio (randomized) $\leq c$, meaning that the expected total length of intervals accepted by the algorithm is at least $1/c$ of the total length accepted by the optimal offline solution, for any input instance $\mathcal{I}$.

Prediction Setup. Learning-augmented algorithms represent a class of algorithms that leverage pre-computed predictions to enhance decision-making in online settings. These predictions, often derived from sources such as statistical models or machine learning systems, provide guidance to the algorithm without requiring it to learn from real-time data. In the context of the online interval scheduling problem, we consider the following prediction settings:

- **Predictions $\hat{\mathbf{O}}$:** The predictions $\hat{\mathbf{O}} = \{\hat{o}_1, \hat{o}_2, \ldots, \hat{o}_n\}$ are a binary vector, where $\hat{o}_i \in \{0, 1\}$ for each interval $I_i \in \mathcal{I}$. The value $\hat{o}_i = 1$ indicates that the interval I_i should be accepted, while $\hat{o}_i = 0$ indicates that the interval should be rejected. This prediction represents a suggested optimal solution.
- **Predictions $\hat{\mathbf{I}}$:** The predictions $\hat{\mathbf{I}}$ aim to provide complete foresight into the characteristics of all intervals in the input. Specifically, $\hat{\mathbf{I}} = \{(\hat{r}_1, \hat{d}_1), (\hat{r}_2, \hat{d}_2), \ldots, (\hat{r}_n, \hat{d}_n)\}$, where each tuple $(\hat{r}_i, \hat{d}_i)$ contains the predicted release time $\hat{r}_i$ and deadline $\hat{d}_i$ of interval I_i. The length of each interval can be computed as $\hat{l}_i = \hat{d}_i - \hat{r}_i$. This prediction setting aims to provide the algorithm with more detailed information to guide its decisions.

Performance Metrics. We use $*$ to denote a generic prediction model, which can represent any of the specific prediction settings considered in this work (e.g., $\hat{\mathbf{O}}$ or $\hat{\mathbf{I}}$). If $\mathcal{I} \lhd *$ denotes all instances $\mathcal{I}$ for which the prediction $*$ is accurate, an algorithm is

- α-*consistent* if it is α-*competitive* when the prediction is correct, i.e.,

$$\max_{\mathcal{I}, *: \mathcal{I} \lhd *} \left\{ \frac{|\text{OPT}(\mathcal{I})|}{|\text{ALG}(\mathcal{I}, *)|} \right\} \leq \alpha,$$

  where $\text{ALG}(\mathcal{I}, *)$ denotes the solution returned by the algorithm using the prediction $*$, and $\text{OPT}(\mathcal{I})$ denotes the optimal offline solution.
- β-*robust* if it is β-*competitive* regardless of the quality of the prediction, i.e.,

$$\max_{\mathcal{I}, *} \left\{ \frac{|\text{OPT}(\mathcal{I})|}{|\text{ALG}(\mathcal{I}, *)|} \right\} \leq \beta,$$

  where $\mathcal{I}$ denotes all possible input instances, and the algorithm's performance is bounded even when the prediction $*$ is arbitrarily inaccurate.

We denote by TRUST the algorithm that follows the prediction blindly. Given a prediction $\hat{\mathbf{O}}$, for each interval I_j, it accepts I_j if and only if $\hat{o}_j = 1$.

Missing proofs and technical details are provided in the supplementary material.

TRUST-AND-SWITCH Framework

In this section, we introduce a general framework for addressing the online interval scheduling problem with predictions. Roughly speaking, the idea is to begin by following the predictions and fall back to a classic algorithm once it is certain that the predictions are not perfectly accurate. Using this framework, we design both deterministic and randomized algorithms that utilize predictions $\hat{\mathbf{O}}$ about the optimal set of intervals.

At time r_j, the algorithm has access to the set $\mathcal{I}(r_j^{\leftarrow}, \cdot)$, that is, all intervals released up to that point, alongside with the corresponding predictions for these intervals.

Framework TRUST-AND-SWITCH. The general framework receives intervals and predictions in an online manner, along with a classic algorithm as input, and returns a robust consistent online algorithm. It consists of two phases: the *trust phase* and the *independent phase*. The algorithm begins in the *trust phase*, where it follows the predictions blindly and may transition to the *independent phase* at a designated time point if the predictions are found to be inaccurate. In the *trust phase*, decisions on whether to accept or reject an interval are made solely based on the prediction. In contrast, the *independent phase* ignores the predictions entirely and makes decisions using the given classic algorithm, without relying on any prediction.

Upon the arrival of interval I_j at time r_j, the framework evaluates the predictions, and the moment it is certain that the predictions are inaccurate, it decides to switch to the classic algorithm. Although the framework re-evaluates the accuracy of the predictions at each release time r_j, it also takes into account the whole of $\mathcal{I}(r_j^{\leftarrow}, \cdot)$.

More formally, upon the arrival of interval I_j at time r_j, the framework computes the *evaluation point*:

$$t_j := \max\left(\{r_j\} \cup \{d_i \mid I_i \in \mathcal{I}(r_j^{\leftarrow}, \cdot),\ \hat{o}_i = 1\}\right).$$

This value ensures that if there is an accepted interval predicted to be in $\hat{\mathbf{O}}$ that is still active at time r_j, we evaluate the prediction up to the last deadline among predicted intervals in $\mathcal{I}(r_j^{\leftarrow}, \cdot)$. Otherwise, we simply consider r_j as the evaluation point.

Note that if $\hat{o}_j = 1$, then the corresponding evaluation point is $t_j = d_j$; however, if we are already certain at r_j about the inaccuracy of the predictions, the switch to the *independent phase* can take place earlier.

To evaluate the accuracy of the prediction up to the time t_j, the algorithm first verifies the consistency of $\hat{\mathbf{O}}$: in other words, whether all intervals in $\mathcal{I}(r_j^{\leftarrow}, \cdot)$ with $\hat{o}_i = 1$, including I_j, are non-overlapping. If not, the prediction $\hat{\mathbf{O}}$ is

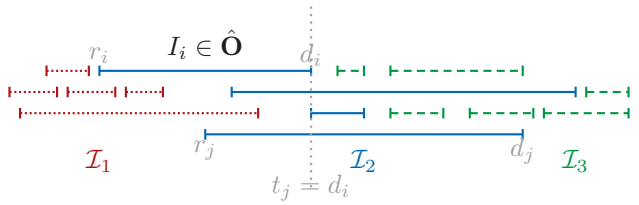

Figure 1: Classification of intervals when, at timepoint r_j, the framework decides to switch based on the evaluation up to $t_j = d_i$, where $I_i \in \hat{\mathbf{O}}$. The three classes $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$ are illustrated with distinct visual styles in the figure.

clearly invalid, and the algorithm switches as soon as possible to the *independent phase*.

Otherwise, it compares the quality of the prediction with the offline optimum up to time t_j. Specifically, it verifies whether

$$|\textsc{Opt}(\mathcal{I}(r_j^{\leftarrow}, t_j^{\leftarrow}))| > |\textsc{Trust}(\mathcal{I}(r_j^{\leftarrow}, t_j^{\leftarrow}))|.$$

If this inequality holds, the algorithm transitions to the *independent phase* and starts to run the classic algorithm as soon as possible – that is: at r_j if $I_j \in \hat{\mathbf{O}}$, and at t_j otherwise – on the instance $\mathcal{I}(r_j^{\rightarrow}, \cdot)$ (resp. $\mathcal{I}(t_j^{\rightarrow}, \cdot)$) consisting of the intervals released after this point. Otherwise, it continues in the *trust phase* and accepts I_j if and only if $\hat{o}_j = 1$.

Theorem 1. *The* Trust-and-Switch *framework* Alg *satisfies* 1*-consistency. Moreover, if the independent phase uses a* θ*-competitive algorithm, then the framework satisfies*

$$|\textsc{Opt}| \leq \theta \cdot |\textsc{Alg}| + 2k.$$

Proof Sketch. To prove 1-consistency, we show that if the predictions are accurate, the framework never switches to the *independent phase*. For the robustness guarantee, we partition the intervals into three classes based on the last evaluation point t_j: $\mathcal{I}_1$ (intervals ending before t_j), $\mathcal{I}_2$ (intervals spanning t_j), and $\mathcal{I}_3$ (intervals starting after t_j). Figure 1 illustrates this classification. We bound the optimal value on $\mathcal{I}_1$ and $\mathcal{I}_2$ by the algorithm's value plus an additive k each, accounting for prediction evaluation and transition cost. The contribution of $\mathcal{I}_3$ is bounded by the θ-competitive algorithm used in the *independent phase*. Summing the three parts yields the desired guarantee. □

We note that the framework allows any deterministic or randomized algorithm to be used in the *independent phase*.

Two-Value Instances We give a tighter bound for two-value instances. All intervals in these instances have one of two distinct lengths. We refer to these as *short* and *long* intervals. More specifically, we show that the Trust-and-Switch framework provides a better robustness guarantee in this special case, due to the restricted structure of the interval lengths.

Lemma 1. *Consider the* Trust-and-Switch *framework* Alg*, and assume that the independent phase uses a* θ*-competitive algorithm. Then, on two-value instances, the algorithm satisfies*

$$|\textsc{Opt}| \leq \max\{\theta, 2\} \cdot |\textsc{Alg}| + k.$$

Tightness

In this final subsection, we show that the guarantees of the Trust-and-Switch framework are tight for two-value instances. First, we prove that the Greedy algorithm achieves a robustness guarantee in this setting. Then, we show that even under the strongest prediction model $\hat{\mathbf{I}}$, no algorithm with consistency better than $1 + \frac{\lceil\Delta\rceil - 1}{\Delta}$ can achieve a stronger robustness guarantee. This establishes the tightness of the Trust-and-Switch framework for two-value instances. While our lower bound extends to general instances as well, there remains a gap between the upper and lower bounds in the general setting.

Greedy Algorithm We first bound the competitive ratio of the Greedy algorithm on two-value instances. This will allow us to instantiate the robustness guarantee in our framework. We briefly recall the definition of the Greedy algorithm below.

Greedy. The Greedy algorithm accepts an arriving interval if and only if it does not overlap with any previously accepted interval.

Lemma 2. *Given a two-value instance* $\mathcal{I}$*, let* Greedy$(\mathcal{I})$ *be the solution obtained by the* Greedy *algorithm. Then,*

$$|\textsc{Opt}(\mathcal{I})| \leq \max\{\Delta, 2\} \cdot |\textsc{Greedy}(\mathcal{I})|.$$

Now, using Theorem 1, and Lemmas 1, and 2, we can conclude the following proposition.

Proposition 1. *Given a two-value instance and predictions* $\hat{\mathbf{O}}$*,* Trust-and-Switch*(*Greedy*) satisfies* 1*-consistency. Moreover, it guarantees*

$$|\textsc{Opt}| \leq \max\{\Delta, 2\} \cdot |\textsc{Trust-and-Switch}(\textsc{Greedy})| + k,$$

independent of the quality of the predictions.

Lower Bound We now show that the consistency–robustness trade-off achieved by the Trust-and-Switch framework is tight for two-value instances. Even with access to full information through $\hat{\mathbf{I}}$, no algorithm with sufficiently strong consistency can guarantee a better robustness bound.

Lemma 3. *For any* $(1+\alpha)$*-consistent algorithm* Alg$'$ *for two-value instances, with* $\alpha < \frac{\lceil\Delta\rceil - 1}{\Delta}$*, there exists an instance* $\mathcal{I}$ *such that*

$$\Delta \cdot |\textsc{Alg}'(\mathcal{I})| + k \leq |\textsc{Opt}(\mathcal{I})|,$$

even when provided with full predictions $\hat{\mathbf{I}}$*.*

SemiTrust-and-Switch Framework

In this section, we turn to a more general variant of the Trust-and-Switch framework that relaxes its reliance on the predictions in the *trust phase*. The goal is to achieve a better consistency–robustness trade-off. The key idea is to define a length threshold and follow the prediction only for intervals whose lengths fall below this threshold. Although this approach sacrifices 1-consistency unless the threshold is set very high, it allows for improved robustness guarantees when predictions are unreliable.

Intuitively, this method ensures that the algorithm does not miss promising long intervals in the case of inaccurate predictions, thereby improving its worst-case performance.

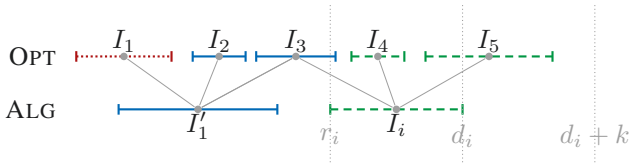

Figure 2: The last connected component of the conflict graph $G(\text{OPT}, \text{ALG})$ before switching. The styles of intervals represent their mapped counterparts. Note that I_1 is the only interval taken by OPT that is not mapped to any interval. All intervals I_4 and I_5, which are mapped to I_i, are contained within the range $[r_i, d_i + k]$.

Framework SEMITRUST-AND-SWITCH. In this framework, the algorithm uses a length threshold τ to determine whether to follow the prediction. More formally, in the *trust phase*, the algorithm accepts an interval if either (i) the prediction suggests it should be accepted, or (ii) the interval's length exceeds the threshold τ. As in TRUST-AND-SWITCH, the algorithm evaluates the predictions over time and switches to a classical algorithm in the *independent phase* using the same strategy.

Note that the framework uses the TRUST algorithm to evaluate the quality of the prediction. More specifically, it compares the value of the predicted solution with the offline optimum up to time t_j, and switches to the *independent phase* if

$$|\text{OPT}(\mathcal{I}(r_j^{\leftarrow}, t_j^{\leftarrow}))| > |\text{TRUST}(\mathcal{I}(r_j^{\leftarrow}, t_j^{\leftarrow}))|.$$

Since SEMITRUST-AND-SWITCH uses the same evaluation procedure and switching strategy as the TRUST-AND-SWITCH framework, we have the following observation.

Observation 1. SEMITRUST-AND-SWITCH *switches to the independent phase only if the predictions are inaccurate.*

We now analyze the performance of the SEMITRUST-AND-SWITCH framework as a function of the threshold parameter τ. Recall that in the *trust phase*, the algorithm accepts an interval if either the prediction suggests it should be accepted, or its length exceeds τ. If τ is set below the length of the shortest interval, the algorithm accepts all intervals greedily, behaving identically to GREEDY. On the other hand, if $\tau > k$, where k is the maximum interval length, then no interval is accepted based on length alone, and the algorithm behaves identically to TRUST-AND-SWITCH(GREEDY), relying fully on predictions in the *trust phase*.

The interesting regime occurs when the threshold τ lies strictly between the shortest and longest interval lengths. Specifically, we assume throughout this section that

$$\frac{k}{\Delta} < \tau < k,$$

so that the algorithm accepts long intervals (with length $> \tau$) greedily, while relying on predictions only for short intervals (with length $\leq \tau$). This setting improves robustness by ensuring that long, high-value intervals are not missed due to inaccurate predictions, while still maintaining reasonable consistency guarantees.

Theorem 2. *The* SEMITRUST-AND-SWITCH *framework* ALG *satisfies* $(1 + \frac{k}{\tau})$*-consistency. Moreover, if the independent phase uses a* θ*-competitive algorithm, then the framework satisfies*

$$|\text{OPT}| \leq \max(\theta, \Delta + 1) \cdot |\text{ALG}| + \tau.$$

Proof Sketch. For robustness, we analyze $\mathcal{I}_1 \cup \mathcal{I}_2$ via the conflict graph $G(\text{OPT}, \text{ALG})$. In the last connected component before switching (see Figure 2), each interval in OPT is mapped to one in ALG, contributing at most $l_i + k$ per $I_i \in$ ALG. At most one unmapped interval remains, with length at most τ. □

Lower Bound We show a lower bound on the robustness of algorithms with consistency better than Δ.

Lemma 4. *Given two-value instances, no algorithm with robustness better than* $\Delta \cdot |\text{ALG}| + \frac{k}{\Delta}$ *can achieve consistency better than* Δ*, even with full predictions* $\hat{\mathbf{I}}$.

Randomized Setting

In this section, we study randomized algorithms for online interval scheduling with predictions. Our goal is to understand how randomness can improve performance guarantees. We first show that if we plug a randomized algorithm into the *independent phase* of the TRUST-AND-SWITCH framework, we obtain meaningful consistency–robustness trade-offs. Our main result is the SMOOTHMERGE algorithm, whose performance degrades gracefully with the quality of the prediction. The key idea is to combine two algorithms, one that uses predictions and another that is prediction-agnostic, in a smooth and controlled manner.

The TRUST-AND-SWITCH Framework

The TRUST-AND-SWITCH framework naturally extends to this setting by allowing a randomized algorithm to be used in the *independent phase*.

VIRTUAL ALGORITHM, introduced by (Lipton and Tomkins 1994), achieves a competitive ratio of 2 on two-value instances. Therefore, using Theorem 1 and Lemma 1, we obtain the following proposition.

Proposition 2. *Given a two-value instance and predictions* $\hat{\mathbf{O}}$*,* TRUST-AND-SWITCH*(*VIRTUAL ALGORITHM*) satisfies* 1*-consistency. Moreover, it guarantees*

$$|\text{OPT}| \leq 2 \cdot |\text{TRUST-AND-SWITCH(VIRTUAL ALGORITHM)}| + k,$$

independent of the quality of the predictions.

For general instances, (Lipton and Tomkins 1994) introduced a randomized MARRIAGE ALGORITHM that achieves a competitive ratio of $O((\log \Delta)^{1+\epsilon})$, where $\epsilon > 0$ is an arbitrarily small constant. Therefore, using Theorem 1, we obtain the following proposition.

Proposition 3. *Given predictions* $\hat{\mathbf{O}}$*,* TRUST-AND-SWITCH*(*MARRIAGE ALGORITHM*) satisfies* 1*-consistency. Moreover, it guarantees*

$$|\text{OPT}| \leq \max\left\{2,\ O\left((\log \Delta)^{1+\epsilon}\right)\right\} \cdot |\text{ALG}| + 2k,$$

where ALG *denotes* TRUST-AND-SWITCH*(*MARRIAGE ALGORITHM*), independent of the quality of the predictions.*

Algorithm 1: SMOOTHMERGE

Input: Interval set $\mathcal{I}$, Prediction $\hat{\mathbf{O}}$, probabilities p_t, p_g
Output: Selected interval set S

$S \leftarrow \emptyset$
for each interval I arriving in online order **do**
  $A_{\text{TRUST}} \leftarrow$ **true** if TRUST accepts I
  $A_{\text{GREEDY}} \leftarrow$ **true** if GREEDY accepts I
  if I conflicts with any interval in S **then**
    reject I
  else if $A_{\text{TRUST}} =$ true **and** $A_{\text{GREEDY}} =$ true **then**
    $S \leftarrow S \cup \{I\}$
  else if $A_{\text{TRUST}} =$ true **and** $A_{\text{GREEDY}} =$ false **then**
    $S \leftarrow S \cup \{I\}$ with **probability** p_t
  else if $A_{\text{TRUST}} =$ false **and** $A_{\text{GREEDY}} =$ true **then**
    $S \leftarrow S \cup \{I\}$ with **probability** p_g
  else
    reject I
  end if
end for
return S

A Smooth Algorithm

In this section, we introduce the SMOOTHMERGE algorithm, which achieves smoothness while maintaining robustness. The key insight is to combine two complementary algorithms: one that follows the prediction and performs well when the predictions are good (i.e., low error), and another classic algorithm that maintains reasonable performance regardless of prediction quality. Specifically, we design a randomized algorithm by merging the TRUST and GREEDY algorithms.

SMOOTHMERGE. Upon the arrival of each interval, the algorithm proceeds as follows: if the interval conflicts with any previously accepted interval, it is immediately rejected. Otherwise, if both simulated strategies agree on accepting (or rejecting) the interval, the algorithm follows that unanimous decision. When the strategies disagree, the algorithm accepts the interval with probabilities p_t and p_g, corresponding to the TRUST and GREEDY strategies, respectively. The full procedure is given in Algorithm 1.

To analyze the performance of our algorithm based on the quality of the predictions, we need to formally define prediction error.

Definition 1. *For a given instance $\mathcal{I}$ and prediction $\hat{\mathbf{O}}$, the* prediction error *η is defined as:*

$$\eta(\mathcal{I}, \hat{\mathbf{O}}) = \frac{|\text{OPT}(\mathcal{I})| - |\text{TRUST}(\mathcal{I}, \hat{\mathbf{O}})|}{|\text{OPT}(\mathcal{I})|}.$$

The prediction error $\eta \in [0, 1]$ quantifies the reliability of predictions: $\eta = 0$ indicates perfect predictions (where TRUST achieves optimal performance), while larger values indicate less reliable predictions. When $\eta = 1$, the prediction-based algorithm performs arbitrarily poorly compared to the optimum.

p_t	p_g	Smoothness	Robustness
1	0	$(1-\eta) \cdot \lvert\text{OPT}(\mathcal{I})\rvert$	0
0	1	0	$\lvert\text{GREEDY}(\mathcal{I})\rvert$
0.50	0.50	$0.25 \cdot (1-\eta) \cdot \lvert\text{OPT}(\mathcal{I})\rvert$	$0.33 \cdot \lvert\text{GREEDY}(\mathcal{I})\rvert$
0.75	0.33	$0.50 \cdot (1-\eta) \cdot \lvert\text{OPT}(\mathcal{I})\rvert$	$0.11 \cdot \lvert\text{GREEDY}(\mathcal{I})\rvert$
0.50	0.75	$0.12 \cdot (1-\eta) \cdot \lvert\text{OPT}(\mathcal{I})\rvert$	$0.60 \cdot \lvert\text{GREEDY}(\mathcal{I})\rvert$

Table 1: Smoothness and robustness coefficients for different values of p_t and p_g.

Theorem 3. SMOOTHMERGE *achieves both smoothness and robustness guarantees. Specifically,*

$$\mathbb{E}[|\text{ALG}|] \geq \max\Big\{|\text{OPT}| \cdot (1-\eta) \cdot p_t \cdot (1-p_g), \frac{p_g - p_t p_g}{1 - p_t p_g} \cdot |\text{GREEDY}|\Big\},$$

where p_t and p_g are the chosen probabilities for the TRUST *and* GREEDY *algorithms, respectively.*

When $p_t = 1$ and $p_g = 0$, the algorithm purely follows predictions, achieving perfect smoothness but no robustness. Conversely, when $p_t = 0$ and $p_g = 1$, the algorithm reduces to the standard GREEDY approach, providing full robustness but no smoothness benefit. Other balanced choices of p_t and p_g offer both properties simultaneously, though with reduced coefficients. Table 1 illustrates the trade-offs between smoothness and robustness for different parameter choices.

This approach can be generalized to combine any pair of algorithms beyond TRUST and GREEDY. The key insight is to merge a prediction-dependent algorithm with a robust classical baseline. While it is natural to use a state-of-the-art classical algorithm in this role, our choice of the GREEDY algorithm as the robust component is motivated by its simplicity and the tractability of its competitive analysis.

Experimental Analysis We provide an experimental evaluation comparing SMOOTHMERGE with GREEDY, TRUST, and OPT, included in the supplementary material. The goal is to offer intuition and support our theoretical findings; the experiments are intended as a complement to the analysis, not as a standalone contribution.

Conclusion

We presented a systematic study of online interval scheduling with predictions in the irrevocable setting, focusing on maximizing the total length of accepted intervals. Our techniques show how to effectively combine predictive and classic algorithms, with provable trade-offs between consistency, robustness, and smoothness. Beyond the specific results presented here, our frameworks offer a modular approach that may extend to other variants of online interval scheduling. While maximization problems pose unique challenges for switching strategies, our findings suggest that such approaches may have broader applicability in the design of learning-augmented algorithms across a wider range of settings.

References

Antoniadis, A.; Coester, C.; Eliás, M.; Polak, A.; and Simon, B. 2023. Online Metric Algorithms with Untrusted Predictions. *ACM Trans. Algorithms*, 19(2): 19:1–19:34.

Antoniadis, A.; Gouleakis, T.; Kleer, P.; and Kolev, P. 2020. Secretary and online matching problems with machine learned advice. *NeurIPS*.

Antoniadis, A.; Jabbarzade, P.; and Shahkarami, G. 2022. A Novel Prediction Setup for Online Speed-Scaling. In *SWAT*.

Azar, Y.; Panigrahi, D.; and Touitou, N. 2022. Online Graph Algorithms with Predictions. *Proceedings of the Thirty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*.

Bamas, E.; Maggiori, A.; Rohwedder, L.; and Svensson, O. 2020. Learning Augmented Energy Minimization via Speed Scaling. In *NeurIPS*.

Bansal, N.; Coester, C.; Kumar, R.; Purohit, M.; and Vee, E. 2020. Scale-Free Allocation, Amortized Convexity, and Myopic Weighted Paging. *CoRR*, abs/2011.09076.

Bar-Noy, A.; Canetti, R.; Kutten, S.; Mansour, Y.; and Schieber, B. 1999. Bandwidth Allocation with Preemption. *SIAM Journal on Computing*, 28(5): 1806–1828.

Borodin, A.; and Karavasilis, C. 2023. Any-Order Online Interval Selection. In *WAOA*.

Boyar, J.; Favrholdt, L. M.; Kamali, S.; and Larsen, K. S. 2023. Online Interval Scheduling with Predictions. In *Algorithms and Data Structures, WADS*.

Canetti, R.; and Irani, S. 1995. Bounding the power of preemption in randomized scheduling. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*.

Choo, D.; Gouleakis, T.; Ling, C. K.; and Bhattacharyya, A. 2024. Online bipartite matching with imperfect advice. arXiv:2405.09784.

Dütting, P.; Lattanzi, S.; Paes Leme, R.; and Vassilvitskii, S. 2021. Secretaries with advice. In *Proceedings of the 22nd ACM Conference on Economics and Computation*.

Garay, J. A.; Gopal, I. S.; Kutten, S.; Mansour, Y.; and Yung, M. 1997. Efficient On-Line Call Control Algorithms. *Journal of Algorithms*, 23(1): 180–194.

Karavasilis, C. 2025. Interval Selection with Binary Predictions. *To appear in IJCAI*.

Kolen, A. W.; Lenstra, J. K.; Papadimitriou, C. H.; and Spieksma, F. C. 2007. Interval Scheduling: A Survey. *Naval Research Logistics*, 54(5): 530–543.

Kovalyov, M. Y.; Ng, C.; and Cheng, T. E. 2007. Fixed interval scheduling: Models, applications, computational complexity and algorithms. *European Journal of Operational Research*, 178(2): 331–342.

Lattanzi, S.; Lavastida, T.; Moseley, B.; and Vassilvitskii, S. 2020. Online scheduling via learned weights. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*.

Lindermayr, A.; and Megow, N. 2025. ALPS. https://algorithms-with-predictions.github.io/.

Lipton, R. J.; and Tomkins, A. 1994. Online interval scheduling. In *ACM-SIAM Symposium on Discrete Algorithms*.

Long, D. D. E.; and Thakur, M. N. 1993. Scheduling Real-Time Disk Transfers for Continuous Media Applications. In *Proceedings of the Twelfth Symposium on Mass Storage Systems*.

Lykouris, T.; and Vassilvitskii, S. 2021. Competitive Caching with Machine Learned Advice. *J. ACM*, 68(4): 24:1–24:25.

Mitzenmacher, M. 2020. Scheduling with Predictions and the Price of Misprediction. In *11th Innovations in Theoretical Computer Science Conference (ITCS)*.

Mitzenmacher, M.; and Vassilvitskii, S. 2022. Algorithms with Predictions. *Commun. ACM*, 65(7): 33–35.

Purohit, M.; Svitkina, Z.; and Kumar, R. 2018. Improving Online Algorithms via ML Predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS*.

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

Roughgarden, T. 2021. *Beyond the Worst-Case Analysis of Algorithms*. Cambridge University Press.

Seiden, S. S. 2000. Randomized algorithms for that ancient scheduling problem. *Journal of Algorithms*, 36(1): 51–77.

Tomkins, A. 1995. Lower bounds for two call control problems. *Information Processing Letters*, 56(3): 173–178.

Wei, A. 2020. Better and Simpler Learning-Augmented Online Caching. In *APPROX/RANDOM*.

Woeginger, G. J. 1994. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1): 5–16.